IBM

Technical Reference

JX

IBMパーソナルコンピューター

JX TECHNICAL REFERENCE MANUAL



SOFTWARE PLANNING & DEVELOPMENT

WSBU

DECEMBER 1984



1st edition November, 1984

© Copyright International Business Machines Corporation 1984, 1985

Preface

This manual contains necessary information for the development of I/O devices and software. To understand and utilize this manual in a proper manner, basic knowledge of operation of the IBM 5510 Personal Computer is required. The CPU of this system uses an INTEL 8088 microprocessor and related knowledge of it is also required.
This manual consists of the following:

Chapter 1	Introduction to the System
Chapter 2	Base System
Chapter 3	Video Subsystem
Chapter 4	System Options
Chapter 5	Software
Chapter 6	Compatibility
Appendix A	BIOS Listing
Appendix B	Logic Diagrams
Appendix C	Character Code Table/Character Font

First Edition (December 1984)

International Business Machines Corporation provides this manual "as is" without warranty of any kind, either express or implied, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose. IBM may make improvements and or changes in the product(s) and or the program(s) described in this manual at any time and without notice.

This publication could contain technical inaccuracies or typographical errors. Changes will be incorporated in new editions of this manual.

Table of Contents

1. Introduction to the System.....1-1

2. Base System.....2-1
   2.1. System Board.....2-3
   2.2. Fundamental System Function.....2-8
      2.2.1. System Clock.....2-8
      2.2.2. Interrupt Controller.....2-8
      2.2.3. Parallel Port.....2-10
      2.2.4. Port A0 Output Description.....2-13
      2.2.5. Memory.....2-14
      2.2.6. Memory Space and I/O Address Setting.....2-16
      2.2.7. Expansion Channel.....2-22
      2.2.8. Cassette Interface.....2-27
      2.2.9. Sound Subsystem.....2-31
      2.2.10. Beep Subsystem.....2-38
      2.2.11. Keyboard Interface.....2-39
      2.2.12. Joystick Interface.....2-44
      2.2.13. ROM Cartridge.....2-47
      2.2.14. Printer Interface.....2-55
   2.3. Keyboard.....2-59
   2.4. Power Unit.....2-62

3. Video Subsystem.....3-1
   3.1. Introduction.....3-2
   3.2. Video Subsystem Memory Usage.....3-4
   3.3. Character Generator.....3-8
      3.3.1. Charactor Generator 1 (CG1).....3-8
      3.3.2. Character Generator 2 (CG2).....3-8
   3.4. Display Function of VP1 and VP2.....3-10
      3.4.1. VP1 Text Display.....3-11
      3.4.2. VP2 Text Display.....3-12
      3.4.3. VP1 & VP2 Graphics Display.....3-13
      3.4.4. Video Gate Array.....3-18
      3.4.5. Superimpose.....3-26
      3.4.6. Video RAM and Display Screen Relationships.....3-27
   3.5. VP3 Display Function.....3-29
      3.5.1. VP3 Text Display.....3-29
      3.5.2. VP3 Graphics Display.....3-29
      3.5.3. Gate Array.....3-31
      3.5.4. Video RAM and Display Screen Relationships.....3-38
   3.6. CRT Controller.....3-39
   3.7. Light Pen Interface.....3-40
   3.8. Video Subsystem I/O Addresses.....3-42
   3.9. Hardware Interface.....3-43

4. System Options.....4-1
   4.1. 64KB RAM Card.....4-2
   4.2. 128KB RAM Card.....4-6
   4.3. Extension Video Card.....4-7
   4.4. Diskette Drive Adapter.....4-12
   4.5. Diskette Drive.....4-23
   4.6. TV Adapter.....4-24

4.7. Keyboard Cable....4-25
4.8. RS-232C Card....4-26
4.9. RS-232C Cable....4-32
4.10. Display....4-33
4.11. CMT Cable....4-34
4.12. Joystick....4-35
4.13. Expansion Board....4-36
4.14. Expansion Unit....4-37

5. Software....5-1
5.1. Software Structure....5-2
5.2. System Software....5-3
5.3. BIOS Usage....5-5
5.3.1. Native Mode BIOS Interrupts....5-8
5.3.2. Extension Video Mode BIOS Interrupts....5-35
5.3.3. Interrupt Routines For Special Use....5-43
5.4. Keyboard Scan Codes....5-45
5.5. Memory Map....5-46
5.6. I/O Map....5-49

6. Compatibility....6-1
6.1. Unequal Configurations....6-2
6.2. Hardware Differences....6-3
6.3. Diskette Compatibility....6-7

Appendices

A. BIOS Listing....A-1

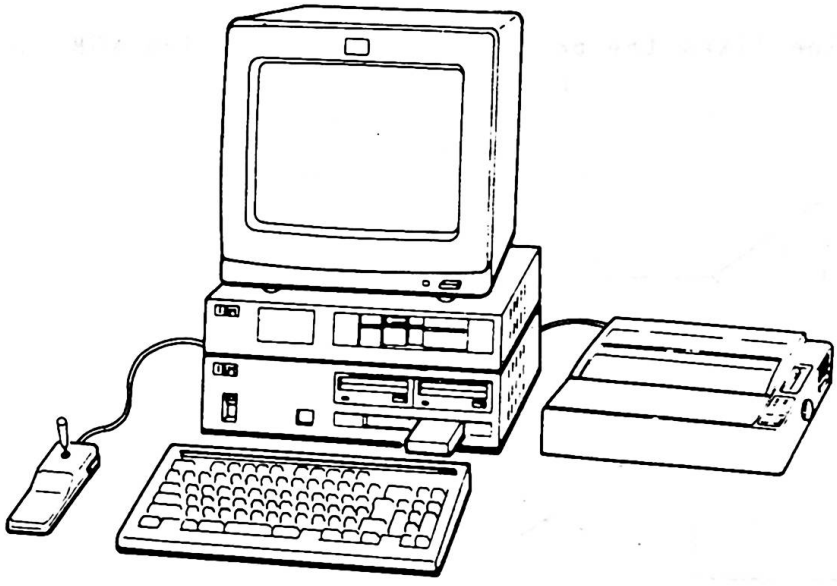
B. Logic Diagrams....B-1

C. Character Code Table/Character Font....C-1

Index....X-1

1. Introduction to the System

The IBM 5510 System basically consists of a System Unit and Keyboard. By adding optional units and/or features, various system configurations can be defined. In this Chapter, these units/features are introduced.



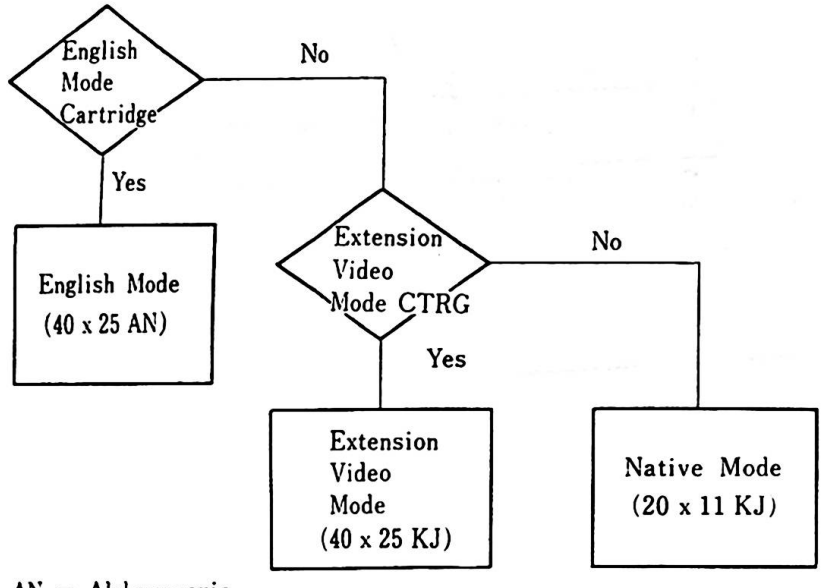
System Configuration Sample

The IBM 5510 is a compact system with a single system board which holds most of the logical circuits. It has many functions and value-added options.

One of the major characteristics of the IBM 5510 is that it is designed to operate in three different modes: English Mode, Native Mode and Extension Video Mode. Switching modes can be performed only by inserting a ROM Cartridge. When no cartridge is inserted, the system operates in Native Mode.

Mode Setting

A BIOS Routine fixes the mode depending upon which ROM Cartridge is inserted.



AN = Alphanumeric
KJ = Kanji

As shown in the above chart, mode can be selected by inserting ROM Cartridges. Users can also make their own cartridges (For instance for game programs) and can run the programs. For this purpose, there are some rules to follow in relation to the ROM Cartridge. These rules are described in Chapter 2.0 Base System - ROM Cartridge in this manual.

The IBM 5510 Video System supports a wide variety of displays ranging from a TV set normally installed in the home to a high resolution CRT display. The Video System can display up to 40 Kanji characters across and 25 lines vertically and 720 x 512 dots in two color graphic mode. In Text mode, Alphanumerics, special charcters, Katakana, Hiragana and Kanji can be displayed.

The Video System functionally consists of three Video Processors (VP1, VP2, VP3). Which of these processors is active depends on th operational mode. By using Video RAM, such functions as Animation are possible. By operating VP1 and VP2, superimposing screens is possible.

The IBM 5510 Video System holds up to eight "pages" as a standard and a maximum of twelve "pages" when additional RAM is installed. VP1 supports pages 0 through 7 for ASCII code, while VP2 and VP3 support pages 8 through 11 for JIS code. Here, "page" refers to 16KB units in memory which contain data to be displayed on the screen.

The IBM 5510 system unit has the following optional features and provides interfaces for them:

64KB RAM Card
increases the system memory size by 64KB to a total of 128KB and enables users to work with a higher resolution video mode in VP1.

128KB RAM Card
increases the system memory size by 128KB to a total of 256KB and holds Address Decode Logic. Up to three cards can be installed when the Expansion Unit is used, making the maximum memory size 512KB.

Extension Video Card
makes it possible to display up to 40 Kanji characters across and 25 lines vertically. Grid Line, 720 x 512 dot two color and 320 x 512 four color graphic displays are also possible. The maximum Video RAM size is 64KB, including the 32KB provided with the base system.

Diskette Drive Adapter Card
controls up to three diskette drives. Diskettes are formatted to 360 KB in English mode and 720 KB in Native and Extension Video mode.

3-1/2" Diskette Drive
supports a 3-1/2" 2DD Diskette.

5-1/4" Diskette Drive
supports a 5-1/4" 2DD Diskette.

RS-232C Card
is a Serial Interface Card which plugs into the IBM 5510 System Board and supports Start-Stop transmission.

RS-232C Cable
connects the RS-232C Card with a Serial I/O Unit.

12" Color Display
is a medium resolution Red/Green/Blue/Intensity direct-drive display which can display a maximum of 16 colors.

12" Monochrome Display
is a high-resolution direct-drive display which is dual-mode and can display English, Native and Extension Video modes. Operational Frequency can be changed by changing the signal transmitted from the System Unit.

14" Color Display
is a high resolution Red/Green/Blue/Intensity direct-drive display which is dual-mode and can display English, Native and Extension Video modes. Operational Frequency can be changed by changing the signal transmitted from the system unit.

Joystick
is an input device which provides the user with two dimensional positioning control. Two push buttons provide the user with additional input capability. It is center-loaded and is calibrated to 100,000 Ohms.

TV Adapter
allows an ordinary home TV set to be connected to the IBM 5510 System. It includes an RF Modulator, When the system unit is turned on, the connection is switched from normal TV broadcasting.

Keyboard Cable
is used to connect the keyboard to the system unit.
When the keyboard cable is not used, an infrared link provides cordless communication between the keyboard and the system unit.

CMT Cable
connects a cassette tape recorder unit with the system unit.

5512 Printer
is a desk-top, non-impact printer. The thermal transfer print head, consisting of 24 heating elements, makes a 24 x 24 Kanji, 12 x 24 Hankaku, 16 x 24 PICA, or 12 x 24 ELITE character font matrx.

5513 Printer
is a desk-top thermal/matrix printer which uses roll paper.

Printer Cable
connects printer units with the system unit.

Expansion Unit
is the same in size as the system unit and is placed on top of the system unit. It contains a Power Unit and allows the user to expand the the number of diskette drives to two or three. Through installation of an Expansion Board Kit, the number of I/O Channels and the memory size can be increased.

Expansion Board Kit
consists of an Expansion Board Adapter and Expansion Board. It holds five I/O Channel slots.

2. Base System

The IBM 5510 consists of a System Unit and Keyboard. The keyboard is provided with an infrared optical link and therefore can make key entries without a keyboard cable. It can also use a keyboard cable to connect to the system unit. The cable is available as an optional feature.

Various kinds of optional features are available for installation with the system unit. Their descriptions are in Chapter 4.0 (4.0 System Option). In this chapter, system functions are described based on the System Board, which is the fundamental part of the system unit.

Video displays are described in Chapter 3.0 (Video Subsystem).

Figure 2-1

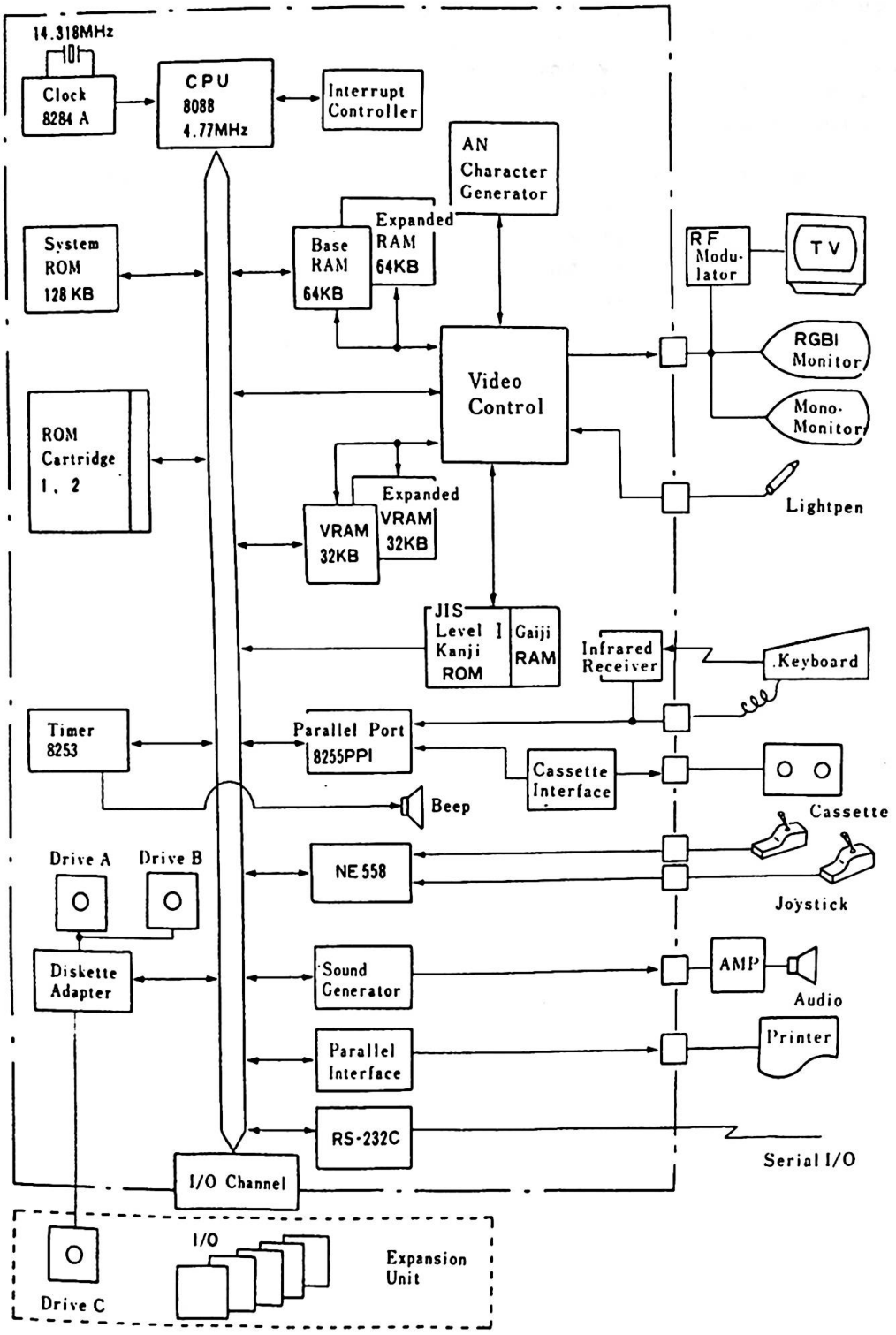


Figure 2-1 System Block Diagram

2.1. System Board

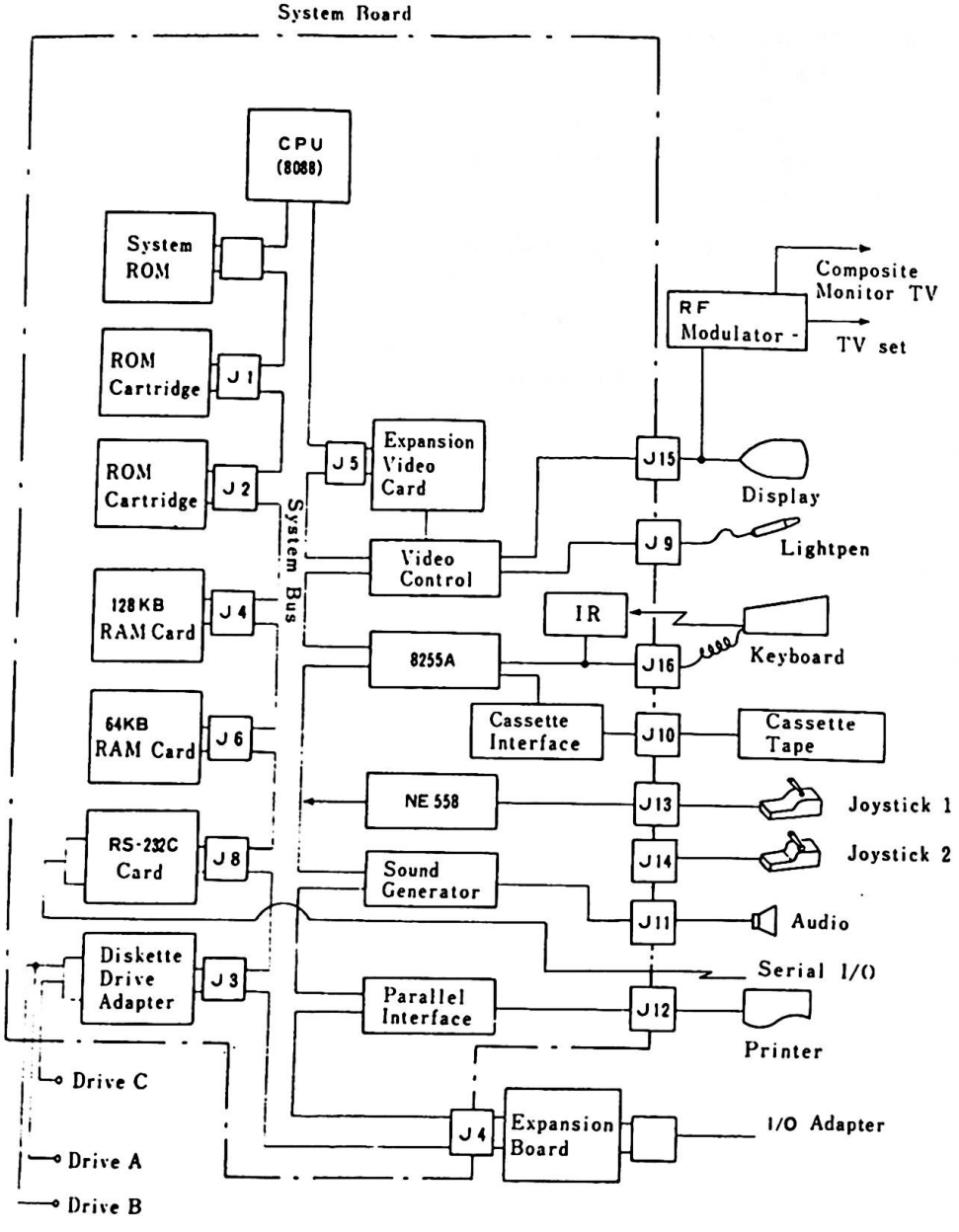
The IBM 5510 System Board is a single board which performs most of the system functions and is the nucleus of the system. The system board fits horizontally in the base of the system unit. It uses double-sided four layered printed circuit boards with an internal power/ground plane. It is designed with highly integrated circuits.

The logical circuits use 8088 family dedicated LSI, seven custom-made LSI consisting of CMOS gate arrays and various I/O LSI to make the system compact while performing many sophisticated functions. The System Board provides the following interface connectors for optional features;

- 64KB RAM Card
- 128KB RAM Card or Expansion Board Kit
- Extension Video Card
- Diskette Drive Adapter
- ROM Cartridge
- Keyboard
- RS-232C Card
- Display(3 types)
- TV Adapter
- Light Pen
- Audio Amplifier(connector only)
- Cassette Tape Cable
- Joystick
- Printer(2 types)

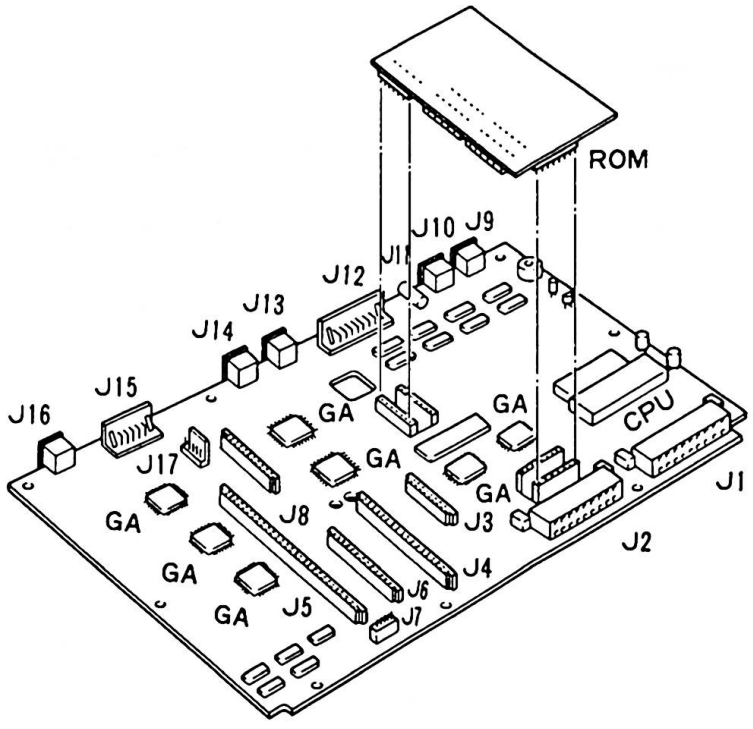
AC power(+5V, +12V, -12V) enters the system board through the power supply connector.

Figure 2-2



Remark) ---: Within System Board
JXX : Connector Number

Figure 2-2 I/O Connection Diagram

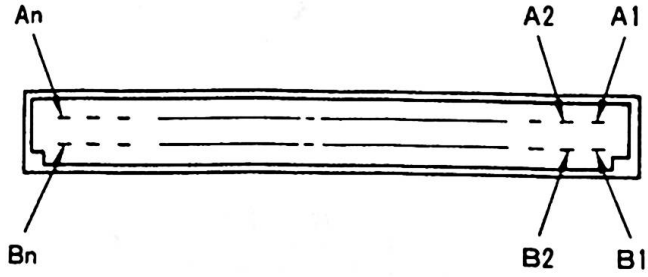


Remark) GA : Gate Array
ROM Board : System ROM and Kanji Character Generator (CG2) included

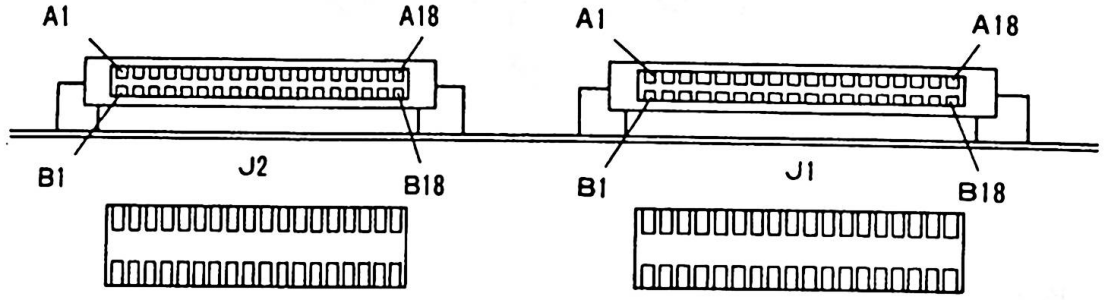
No.	Connecting Feature	No.	Connecting Feature
J 1	ROM Cartridge	J10	Cassette Tape Recorder
J 2	ROM Cartridge	J11	Audio
J 3	Diskette Drive Adapter	J12	Printer
		J13	Joystick (1)
J 4	128KB RAM Card	J14	Joystick (2)
J 5	Extension Video Card	J15	Display
J 6	64KB RAM Card	J16	Keyboard
J 7	Infrared Receiver	J17	Power Connector
J 8	RS-232C Card		
J 9	Light Pen		

Figure 2—3 Connectors on System Board

Figure 2-4



Card Connector(J3, J4, J5, J6, J8)



ROM Cartridge Connector (J1, J2)

Figure 2-4 Connector Pin Numbers

The nucleus of the system board is the Intel 8088 microprocessor. This processor (hereafter called CPU) is software-compatible with the 8086 micro-processor. The 8088 supports 16 bit operations, including multiplication and division and supports 20 bits of addressing (1 megabyte storage). It operates in the minimum mode at 4.77 MHz.

The three programmable timer/counters are provided by an Intel 8235-5 programmable interval timer and are used by the system in the following manner;

Channel 0 : Used as a general-purpose timer to provide constant time base for implementing a time-of-day clock.

Channel 1 : Used for deserializing the keyboard data and for time-of-day overflow during diskette operations.

Channel 2 : Used to support the tone generation for the audio speaker and to write data to the cassette.

There are nine prioritized hardware interrupt levels and out of them, four are bussed to the system I/O channel for use by adapters. The non-maskable interrupt (NMI) of the 8088 is attached to the keyboard-interface circuits and receives an interrupt for each scan code sent by the keyboard.

The system board has space for 128K bytes by 8 bits of ROM and the ROM is aligned at the top of the 8088's address space.

The system board makes it possible to process complex screen handling and sound generation through its sound generator and video subsystem.

2.2. Fundamental System Function

In this Chapter, fundamental system functions and the hardware to support the functions are described.

2.2.1. System Clock

The CPU operates in the minimum mode using a 4.77 MHz clock. The time of one clock cycle is 210 nsec. Normally, four clock cycles are required for a bus cycle, so that an 840 nsec ROM memory cycle time is achieved. When RAM memory is shared with video memory RAM, write and read cycles will take an average of 2 wait cycles, leading to an average of 6 clock cycles. The bus cycle time is 1.260 micro-seconds. The bus cycle time for I/O reads and writes is also 1.260 micro-seconds. The 4.77 MHz clock, whose frequency is derived from a 14.31818 MHz crystal is divided by 3 for the processor clock, and by 4 to obtain the 3.58 MHz color burst signal required for color televisions. The 1.789 MHz clock, which is divided by 8 is used as a baud rate clock of an RS-232C card.

2.2.2. Interrupt Controller

Eight hardware levels of interrupts are available for the system. The highest priority interrupt is the NMI in the 8088. The NMI is followed by eight prioritized interrupt levels (0 - 7) in the 8259A Programmable Interrupt Controller. The priority of the interrupts are as follows:

Priority	Interrupt	Function
1	NMI	Keyboard Interrupt
2	IRQ 0	Time Clock Interrupt
3	IRQ 1	I/O Channel, Keyboard (INT 9 Software Interrupt)
4	IRQ 2	I/O Channel
5	IRQ 3	Asynchronous Port Interrupt (RS-232C)
6	IRQ 4	External I/O Channel
7	IRQ 5	Vertical Retrace Interrupt(Video)
8	IRQ 6	Diskette Interrupt
9	IRQ 7	I/O Channel and Printer

8259A Programming Considerations

(1) 8259A is initially set up with the following characteristics:

- Buffered Mode
- 8088/86 Mode
- Single Mode Master(No cascading is allowed)
- Edge Triggered Mode
- I/O Address is 20 (Hex)
- can issue Hardware Interrupt types Hex 8(IRQ 0) to Hex 0F(IRQ 7)

The following is an example setup:.

```
0263      B0 13     MOV AL,13H      ; ICW1 - RESET EDGE SENSE
                                     ; CIRCUIT, SET SINGLE MODE
                                     ; 8259 CHIP AND ICW4 READ
0265      E6 20     OUT INTA00,AL
0267      B0 08     MOV AL,8        ; ICW2  - SET  INTERRUPT
                                      TYPE 8 - F
0269      E6 21     OUT INTA01,AL
026B      B0 09     MOV AL,9        ;ICW4 - SET BUFFERED MODE/MASTER
                                     AND 8088 MODE
026D      E6 21     OUT INTA01,AL
```

(2) IRQ 1,2,4 and 7 can be used by the I/O Channel. Care should be taken when IRQ1 and IRQ 7 are used, as they are shared by several I/O devices.

(3) NMI can be masked by operation of port A0.

2.2.3. Parallel Port

8255 PPI (Programmable Peripheral Interface) is used as one of the I/O Interfaces and supports control of the Keyboard, Timer, and Cassette Motor. It also checks whether there is an option card or not. Bit Assignments for each Port(A, B, and C) and corresponding functions are as follows;

Port A : For Output

Bit	Description
PA0 :	Reserved for Keystroke Storage
|	
PA7 :	

Port B : For Output

Bit	Name	Description
PB0	Timer 2 Gate	This line is routed to the gate input of timer 2 on the 8253-5. When this bit is "low", the counter operation is halted. This bit and PB1 (Speaker Data) control operation of the 8253-5 sound source.
PB1	Speaker Data	This bit ANDs "off" the output of the 8253-5 Timer 2. It can be used to disable the 8253-5 sound source, or modify its output. When this bit is high, it enables the output. A "low" value forces the output to 0.
PB2	Text/Graphics	This bit is used to steer data from the memory into Video Processor 1. 1 --- Text Modes 0 --- Graphics Modes
PB3	Cassette Motor Control	When this bit is a 1, the cassette relay is open and the cassette motor is off. When this bit is a 0 and PB4=0, the cassette motor is on.

PB4	Beeper & Cassette Control	When this bit is 1, the internal beeper and cassette motor are disabled.
PB5 PB6	Speaker Switch 0,1	These bits steer one of 4 sound sources. The selected sound source is in the display or in the external speaker. Selected sound sources are as follows;

PB6	PB5	Sound Sources
0	0	Timer 2
0	1	Cassette Audio
1	0	I/O Channel Audio
1	1	Sound Generator

PB7	Open	Reserved for future use

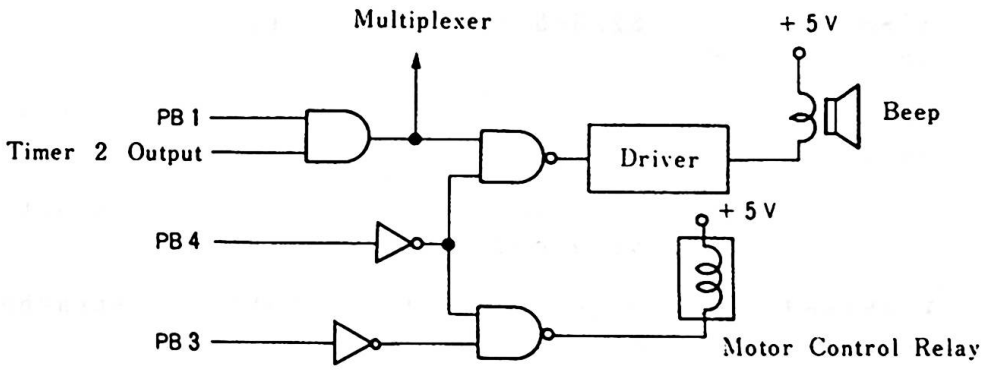


Figure 2—5 PB Bit Function

Port C : For Input

Bit	Name	Description
PC0	NMI Latch Input	This Input comes from a latch which is set to a one on the first leading edge of the Keyboard Data Stream.
PC1	RS-232C Card	When the RS-232C Card is installed, this bit is a 0.
PC2	Diskette Adapter	When the Diskette Adapter Card is installed, this bit is a 0.
PC3	64KB RAM Card	When the 64KB RAM Card is installed, this bit is Low Level.
PC4	Cassette Data	If the Cassette Motor Relay is "closed", and the Cassette Motor is "on", this pin will contain data which has been wave-shaped from the cassette. When the Cassette Motor Relay is off, The Timer 2 output will become PC4 input.
PC5	Timer 2 Input	8253-5 Timer 2 input.
PC6	Keyboard Data	This input contains keyboard data. The keyboard data comes from the cable, when it is attached, or from the IR Receiver Card if the cable is not attached.
PC7	Keyboard Cable	When the Keyboard Cable is attached, the bit is a 0.

2.2.4. Port A0 Output Description

Port output of I/O Address A0 (Hex) allows NMI interrupt and clock input selections. (Reference : 2.2.11 Keyboard Interface)

Bit	Description
Bit 7 (NMI Enable)	When this bit is a one, NMI is enabled. When this bit is a zero, NMI is disabled.
Bit 6 (IR Test ENA)	This bit enables the 8253-5 Timer 2 output into an IR diode on the IR Receiver Card. This information is then wrapped back to the the keyboard input. If the cable is connected, timer 2 should be set for 40 KHz which is the IR-modulation frequency. This feature is used only for a diagnostic test of the IR Receiver Card.
Bit 5 (Clock 1 Input Selection)	This bit selects input clocks to the 8253-5 timer 1. A zero selects a 1.1925MHz clock input (for deserializing the keyboard data). A one selects the timer 0 output to be used as the clock input to timer 1. This is used to catch timer 0 overflows during diskette drive operations, while interrupts are masked off. This is then used to update the time of day.

2.2.5. Memory

The IBM 5510 has two kinds of memory, ROM(Read Only Memory) and RAM(Random Access Memory). Memory is categorized functionally as follows;

- General-use Memory
- System ROM
- Video RAM
- Character Generator 1 (CG1)
- Character Generator 2 (CG2)
- Gaiji RAM

General-use memory

64KB of R/W memory resides on the system board. R/W Memory can be expanded to a 512KB maximum. The standard 64KB memory consists of eight 64K bit modules and has no parity bit. Sources of these memory modules include the Texas Instruments TMS4164-15 or equivalent. These are dynamic RAM with 150 ns access time. Memory size can be expanded by installing additional 64KB or 128KB cards.

Address space of 00000 - 7FFFF (Hex) is always reserved for RAM. Normally, the standard 64KB uses address space of 00000 - 0FFFF (Hex). If an additional 64 KB memory card is installed, address space of 00000 - 1FFFF (Hex) is used for the total of 128 KB space. The 64 KB system board memory is mapped to the EVEN memory address, while the 64 KB additional memory card is mapped to the ODD memory address within the 128 KB reserved space. When an additional 128KB RAM card is installed, the address space will be changed accordingly.
(Reference : 5.5 Memory Map)

System ROM

The ROM subsystem is made up of 128KB of ROM aligned at E0000 - FFFFF(Hex) and has the following functions:

- Power on Self-Diagnostic test
- Initialization (ROM, RAM, I/O Port Configuration set-up)
- Basic Interpreter
- BIOS
- Diskette Boot Strap Loader
- Kanji Dictionary
- ROM Cartridge auto-link

The access time of this ROM Cartridge is 250ns and the cycle time is 375ns.

Video RAM

The system board has 32KB Video RAM as a standard feature. The Video RAM consists of four 16K x 4 modules. These modules include the TMS 4416-15 or its equivalent and are dynamic RAM with 150ns access time.

Character Generator (CG1)

2KB ROM space is reserved for Alphanumeric and Special character fonts used in English Mode.

Character Generator (CG2)

128KB ROM space is provided to for Kanji, Alphanumeric, Special character, Hiragana and Katakana fonts to be used in Native and Extension Video Mode.

Gaiji RAM

2KB static RAM is reserved for storage of up to 62 fonts (15 x 16). They can be accessed by Video Processor 2 and 3. The CPU can also read/write data. The modules are static RAM with 200ns access time.

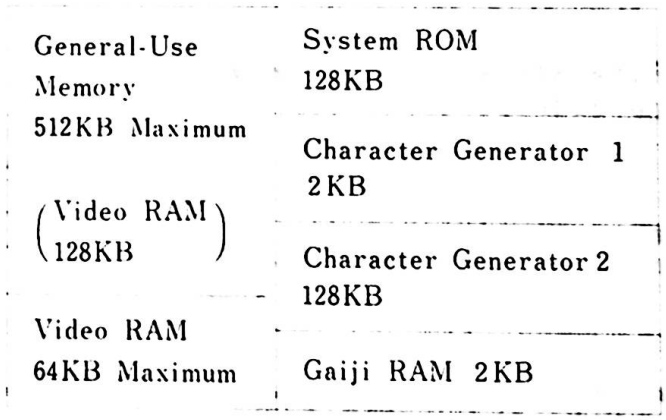


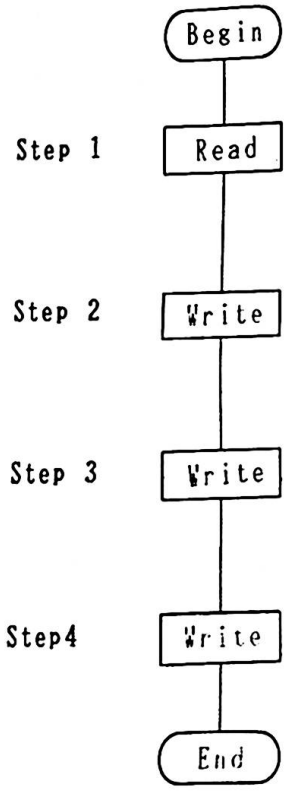
Figure 2-6 Memory Classification

2.2.6. Memory Space and I/O Address Setting

The IBM 5510 is provided with memory space and dedicated custom LSI. The memory space is separated into 11 blocks and the addresses each memory block uses are defined by the software. The minimum unit of definition is 32KB. There are 19 blocks for I/O addresses. The addresses for most I/O devices can be change by software. These controls are performed by setting the appropriate parameter in I/O address 1FF (Hex).
Parameters are set in two of the control registers which reside in each block. Memory space and I/O addresses are set by the following hardware elements:

- Status Register
- Block Definition
- Control Register 1(REG1)
- Control Register 2(REG2)

Method for setting



Status is reset by reading Status Register in I/O address 1FF(Hex)

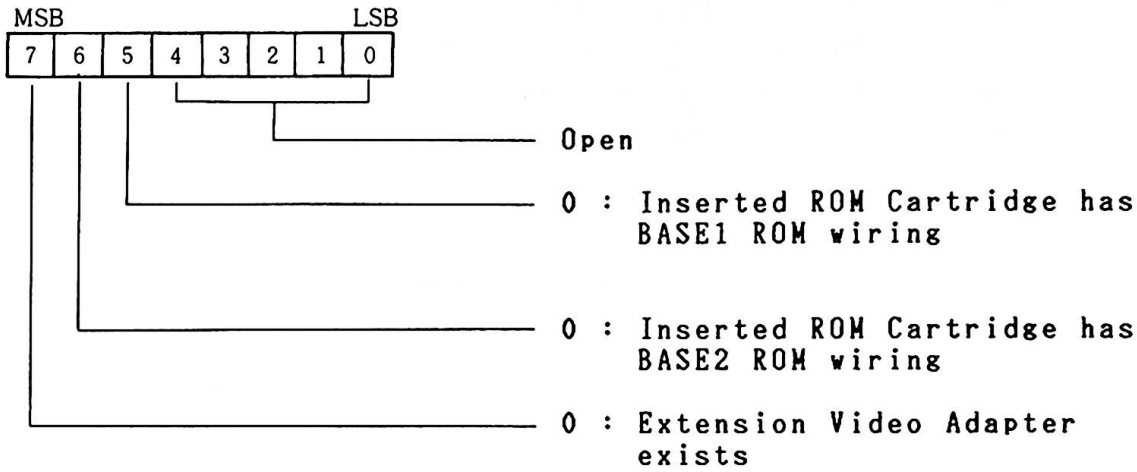
Write Block definition value in I/O address 1FF(Hex)

Write parameter to be set in REG1 in I/O address 1FF(Hex)

Write parameter to be set in REG2 in I/O address 1FF(Hex)

Status Register

When I/O address 1FF(Hex) is read, the following status is obtained.



Block Definition Value

Each memory block is defined by Block Definition Value 00 - 0A (Hex) and I/O Block by 80 - 93 (Hex).

Block Definition Value	Memory Block	Remarks	Address (Default Value)
00	IROM 7	System ROM	E0000~FFFFF
01	EROM 2	Cartridge	D0000~D7FFF
02	EROM 3	Cartridge	D8000~DFFFF
03	EROM 4	Cartridge	E0000~E7FFF
04	EROM 5	Cartridge	E8000~EFFFF
05	EROM 6	Cartridge	F0000~F7FFF
06	EROM 7	Cartridge	F8000~FFFFF
07	KJCS	Kanji ROM and Gaiji RAM Font	80000~BFFFF
08	MAIN	General-use Memory (128KB)*	Changed by Option
09	VRAM 1	Co-use of general-use Memory & Video RAM	B8000~BFFFF
0A	VRAM 2	Video RAM	B8000~BFFFF

Remark) * --- The total of 64 KB Base and 64 KB Expanded Memory.

Figure 2-7 Memory Block & Definition Value

Block Value	I/O	Remarks	Address (Default Value)
80	8259	Interrupt Controller	020~027
81	8253	Timer	040~047
82	8255	Parallel Port	060~067
83	NMI		0A0~0A7
84	SOND	Sound Generator	0C0~0C7
85	FDC	Diskette Drive Adapter	0F0~0F7
86	JOYW	Joystick Write	200~207
87	JOYR	Joystick Read	200~207
88	PRNT	Printer	378~37F
89	8250	RS-232C Card	2F8~2FF
8A	CRTC	CRT Controller	3D0~3D7
8B	GA01	Gate Array 1	3D8
8C	GA2A	Gate Array 2 (VP1)	3DA
8D	GA2B	Gate Array 2 (VP2)	3DA
8E	GA03	Gate Array 3 (VP3)	3DD
8F	LPGT	Light Pen Strobe	3DE
90	PG2	Page Register 2	3D9
91	PG1	Page Register 1	3DF
92	MODM	Open	3F8~3FF
93	ETSC	I/O Address in Expansion Unit	Undefined

Remark) I/O Blocks 80 - 92 are those I/O within the system unit When one of these are selected, bus line of the Expansion Unit is separated from the system bus.

Figure 2-8 I/O Block and Definition Value

Address Change

As the address bits of each memory block or I/O blocks are programmable, their addresses can be changed.

Block Definition Value	Block Name	ON/ OFF	I/O	Mem.	RD	WR	A19	A18	A17	A16	A15
00	IROM 7	P	0	1	1	0	1	1	1	P	P
01	EROM 2	P	P	P	P	P	1	P	P	P	P
02	EROM 3	P	P	P	P	P	1	P	P	P	P
03	EROM 4	P	0	1	1	0	1	P	P	P	P
04	EROM 5	P	0	1	1	0	1	P	P	P	P
05	EROM 6	P	0	1	1	0	1	P	P	P	P
06	EROM 7	P	0	1	1	0	1	P	P	P	P
07	KJCS	P	0	P	P	P	P	P	P	P	P
08	MAIN	P	0	1	1	1	P	P	P	P	P
09	VRAM 1	P	1	1	1	P	P	P	P	P	P
0A	VRAM 2	P	0	1	1	1	P	P	P	P	P

P : Programmable

Figure 2-9 Memory Address Change

Block Definition Value	I/O Name	ON/ OFF	A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0
80	8259	P	0	0	0	0	1	0	0	X	X	X
81	8253	P	0	0	0	1	0	0	0	X	X	X
82	8255	P	0	0	0	1	1	0	0	X	X	X
83	NMI	P	0	0	1	0	1	0	0	X	X	X
84	SOND	P	0	0	1	1	0	0	0	X	X	X
85	FDC	P	P	P	P	P	P	P	P	X	X	X
86	JOYW	P	1	0	0	0	0	0	0	X	X	X
87	JOYR	P	1	0	0	0	0	0	0	X	X	X
88	PRNT	P	1	1	0	1	1	1	1	X	X	X
89	8250	P	1	0	1	1	1	1	1	X	X	X
8A	CRTC	P	1	1	1	1	0	1	0	X	X	X
8B	Reserved	P	1	1	1	1	0	1	1	0	0	0
8C	GA 2A	P	1	1	1	1	0	1	1	0	1	0
8D	GA 2B	P	1	1	1	1	0	1	1	0	1	0
8E	GA 03	P	1	1	1	1	0	1	1	1	0	1
8F	LPGT	P	1	1	1	1	0	1	1	X	1	0
90	PG 2	P	1	1	1	1	0	1	1	0	0	1
91	PG 1	P	1	1	1	1	0	1	1	1	1	1
92	MODM	P	1	1	1	1	1	1	1	X	X	X
93	ETSC	P	X	X	X	X	X	X	X	X	X	X

P : Programmable
X : Neglected
0, 1 : Fixed

Remarks) In order to specify I/O address of Expansion Unit, the bit of Block Definition Value 93 "P" (on/off) should be set as 1.

Figure 2-10 I/O Address Change

Control Register (REG1, REG2)

Each memory block or each I/O address block has one REG1 and one REG2.

Memory Block Control Register

D7	D6	D5	D4	D3	D2	D1	D0	
ON/OFF	I/O	MEM	A 19	A 18	A 17	A 16	A 15	REG 1
	WR	RD	A 19	A 18	A 17	A 16	A 15	REG 2

REG1

ON/OFF : "1" makes REG1 available
I/O : When "1", this block is allocated to I/O space
MEM : When "1", this block is allocated to memory space
A19-A15 : These correspond to address lines A19-A15 and set the first five High Storage Bits which the memory block uses with REG2 A19-A15.

REG2

WR : When "1", writable memory
RD : When "1", readable memory
A19-A15 : REG1 specifies the bit (A19-A15) for comparison with the address bus output. When this bit is "1", comparison is not performed and when "0", comparison is done. When the comparison result is equal, these are valid addresses for this memory block. As for A14-A0, bit shown in the address bus is used as available addresses.

For instance, if REG1 is B7(Hex) and REG2 is 61(Hex), this memory block uses addresses B0000-BFFFF(Hex).

	D7	D6	D5	D4	D3	D2	D1	D0	
				A 19	A 18	A 17	A 16	A 15	A 14—A0
Address Bus →				1	0	1	1	0	X
REG 1 ——→	1	0	1	1	0	1	1	1	
REG 2 ——→	0	1	1	0	0	0	0	1	
Avail.address				1	0	1	1	X	X
				B				X X X X	

I/O Block Control Register

D7	D6	D5	D4	D3	D2	D1	D0	
ON/OFF	A 9	A 8	A 7	A 6	A 5	A 4	A 3	REG 1
	A 9	A 8	A 7	A 6	A 5	A 4	A 3	REG 2

The I/O Block Control Register and Memory Block Control Register have the same functions. I/O Address Lines compared are I/O Control REG1 and REG2 but the extent of the comparison is limited to A9 - A3.

2.2.7. Expansion Channel

The Expansion Channel is the means through which the CPU bus lin is expanded. It can be connected with various I/O adapters whic users can develop themselves.

Major Characteristics

- 64 Pin Connector is used
- "READY" signal is available for low-speed I/O features and slower access time memory
- Maximum load is one standard TTL
- 128KB RAM card, Expansion Board Adapter and other I/O feature are connected.

The following describes connector pin(J4):

Signal Name	Pin (A)	Pin (B)	Signal Name
D0	A 1	B 1	D1
D2	2	2	D3
D4	3	3	D5
D6	4	4	D7
GND	5	5	GND
A0	6	6	A1
A2	7	7	A3
A4	8	8	A5
A6	9	9	A7
GND	10	10	GND
A8	11	11	A9
A10	12	12	A11
A12	13	13	A13
A14	14	14	A15
+5 V	15	15	+5 V
A 16	16	16	A17
A 18	17	17	A19
−IOR	18	18	−IOW
−MEMR	19	19	−MEMW
ALE	20	20	HLDA
CPU CLK	21	21	GND
IO/−M	22	22	DOT CLOCK
OPEN	23	23	OPEN
READY	24	24	RESET
−EXRD	25	25	−ETSC
IRQ 1	26	26	IRQ 2
IRQ 4	27	27	IRQ 7
−HRQ	28	28	−DEN
R/−DT	29	29	REFC
AUDIO IN	30	30	GND
AMODE	31	31	OPEN
+12V	32	32	−12V

Figure 2-11 Expansion Channel Connector(J4)

Remarks) I/O refers to the direction seen from the system board.
Each signal is at standard TTL level except for AUDIO.

Signal	I/O	Description
D0 - D7	I/O	Data Lines 0 - 7: These lines provide data bus bits 0 to 7 for the processor, memory and I/O Devices. D0 is the Least Significant Bit (LSB) and D7 is the Most Significant Bit(MSB).
A0 - A19	O	Address Lines 0 - 19: These 20 address lines allow access of up to 1 megabyte of memory. A0 is the least significant bit (LSB) and A19 is the most significant bit (MSB).
-IOR	O	I/O Read Command: When this command line output is "0", it instructs an I/O device to drive its data onto the data bus. This signal may be driven by the 8088 microprocessor or by an external bus master(such as the DMA controller) after it has gained control of the bus.
-IOW	O	I/O Write Command: When this command line output is "0", it instructs an I/O device to read the data on the data bus. This signal may be driven by the CPU or by an external bus master(such as the DMA controller) after it has gained control of the bus.
-MEMR	O	Memory Read Command: This command line instructs the memory to drive its data onto the data bus.
-MEMW	O	Memory Write Command: When this command line output is "0", it instructs the memory to store the data present on the data bus.
ALE	O	ADDRESS LATCH ENABLE: This line provides the timing signal to latch the address bus.
HLDA	O	HOLD ACK : This line indicates to a bus master on the channel that -HRQ has been honored and that the 8088 has floated its bus and control lines.

Signal	I/O	Description
CPU CLK	O	System Clock: It is a "devide-by-three" of the 14.31818 MHz oscillator and has a period of 210 ns (4.77MHz). The clock has 33 % of the duty cycle.
IO/-M	O	I/O or Memory Status: This status line is used to distinguish a memory access from an I/O access. This line should be driven by a bus master after it has gained control of the bus. If this line is "high", it indicates an I/O access; if this line is "low", it allows memory access.
READY	I	This line, normally "high"("ready"), is pulled "low"(not ready) by a memory or I/O device to lengthen I/O or memory cycles. It allows slower devices to attach to the I/O channel with a minimum of difficulty. Any slow device requiring this line should drive it low immediately upon detecting a valid address and IO/-M signal. Machine cycles (I/O and memory) are extended by an integral number and CLK cycles (210 ns). Any bus master on the I/O channel should also honor this "ready" line. It is pulled "low" by the system board on memory and write cycles and outputting to the sound subsystem.
RESET	O	This line is used to reset or initialize system logic upon power-up. This line is synchronized to the trailing edge of the clock and is "active high". Its duration upon power up is 26.5 micro secs.
-EXRD	O	EXPANSION BUS READ CONTROL: This line is used to control the direction of data flow on the expansion channel. The read status makes it "0".
-ETCS	O	EXPANSION BUS Tri-state Control: It is used to control the Expansion Bus to Tri-state.

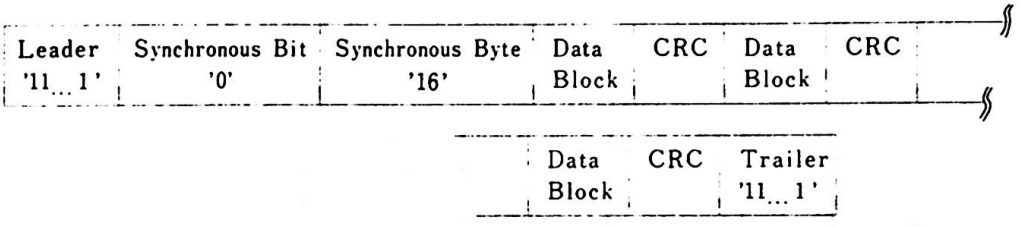


Signal	I/O	Description
IRQ1 IRQ2 IRQ4 IRQ7	I	Interrupt Request 1, 2, 4 and 7: These line are used to signal the processor that an I/O device requires attention. They are prioritized with IRQ1 as the highest priorit and IRQ7 as the lowest. An interrupt reques is generated by raising an IRQ line from "lo to "high" and holding it high until it is acknowledged by the processor.
-HRQ	I	Hold Request: This line indicates that another bus master is requesting the I/O channel. When an I/O channel issues a -HRQ signal, the 8088 processor will respond to the -HRQ by asserting an HLDA, after it lets the data bus and address bus relinguish to an external bus master (such as the DMA controller). As a -HRQ is not an asynchronous signal, it should be synchronized to the system clock.
-DEN	O	DATA ENABLE: This signal makes the data bus available.
R/-DT	O	This signal is used to control the direction of data flow through the transceiver.
AUDIO	I	I/O devices connected to the expansion chann may provide sound sources to the multiplexer of the sound subsystem which resides on the system board. The signal level is 1V P-P.
A MODE	O	This signal is used to make the memory space which 128 KB RAM card uses fixed after the memory on the board when in English mode.
REFC	O	This line is used for dynamic RAM refresh.
DOT CLOCK	O	14.318 MHz Clock

2.2.8. Cassette Interface

The cassette interface is controlled by software. Output from the 8253 Timer 2 controls data to be written to the cassette recorder. The digital signal read from the cassette is input at PC4, 8255A PPI. PB3 and PB4 output controls the relay. When the relay is inactive(Motor is disabled), the cassette output signal has a wrap feature which connects the output to the input.

Data Format



Contents	Description .
Leader	FF(Hex) of 256 bytes
Synchronous bit	1 "0" bit
Synchronous byte	"16" (Hex)
Data Block	256 byte unit Data
CRC	2 byte check Character(each Data Block)
Trailer	4 byte FF(Hex)

Figure 2-12 Cassette Data Format

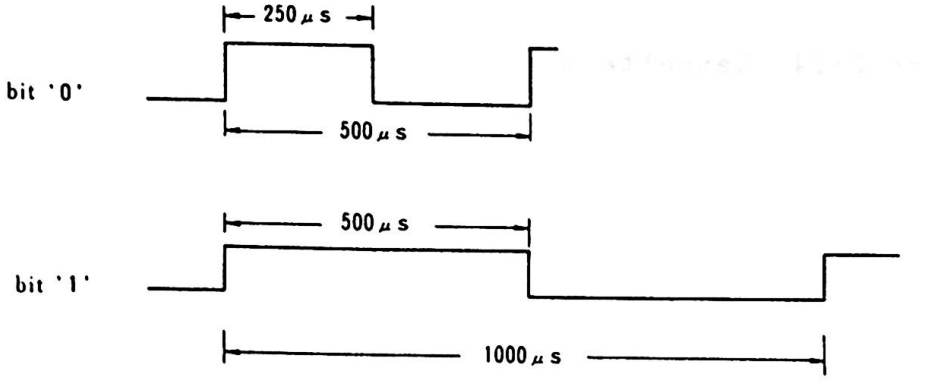


Figure 2-13 Bit Format

Cassette Tape Write

Cassette tape write is performed from MIC DATA OUT (Timer 2 output). The cycle of Timer 2 is 0.5ms for bit 0 and is 1ms for bit 1. Data are written in units of 256-byte (Block) and when the length of data is shorter than multiplies of 256-byte, the last available data are repeatedly written to complete the block.

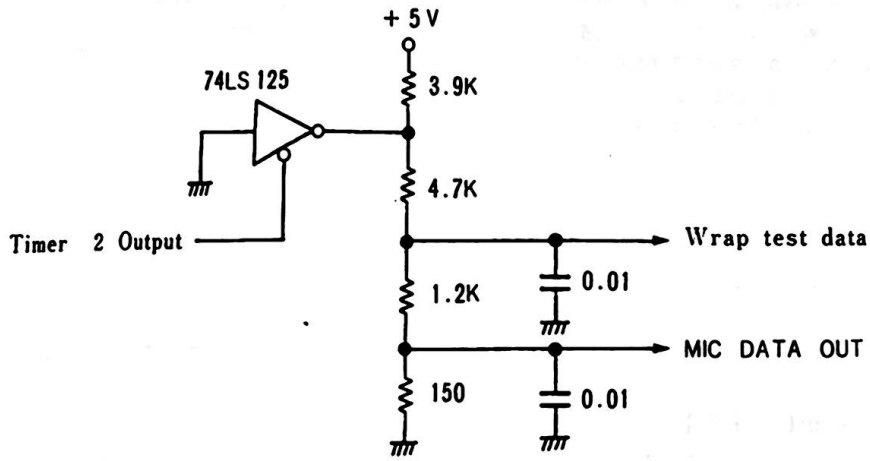


Figure 2-14 Cassette Write

Cassette Tape Read

When more than 1/4 of leader is correctly read, the synchronous bit ("0") and synchronous byte are read. At the point where synchronous bytes are correctly read, data blocks are then read. If data are not read correctly up to a point of the synchronous bytes, the leader is searched again.
When data are read, a CRC check is performed for each block.
Level 0 interrupt is not allowed while reading the cassette tape.
Data read is input from CASS AUDIO to PC4.

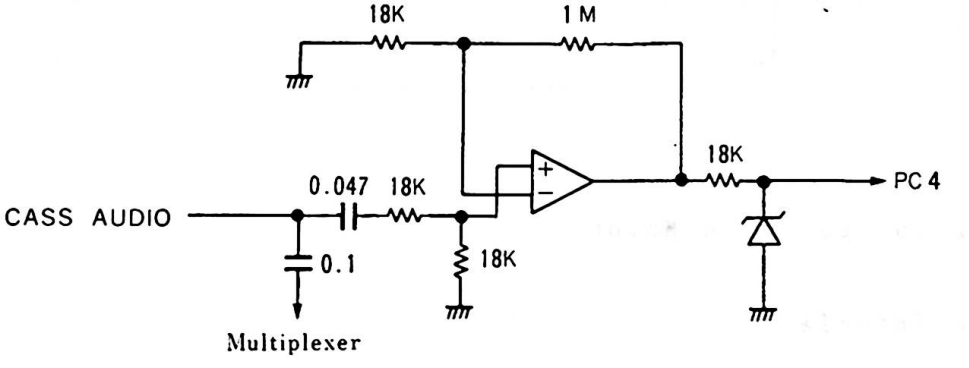


Figure 2-15 Cassette Read

Motor Control

When both PB3 and PB4 are "0", the relay is active and the motor is turned on.

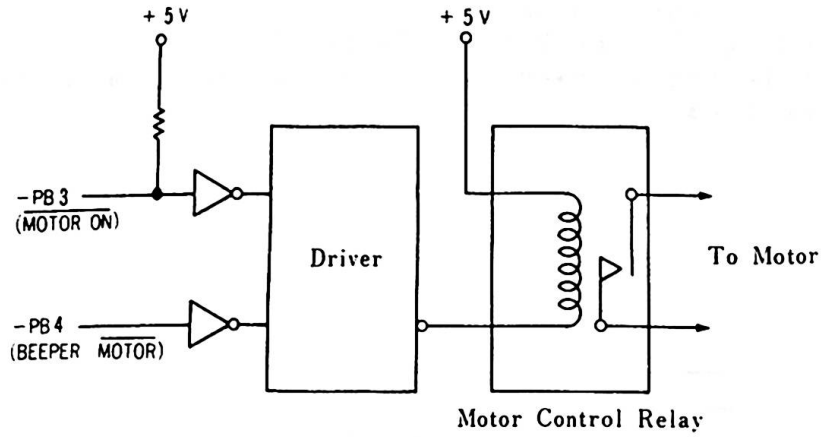
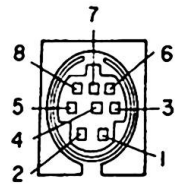


Figure 2-16 Cassette Motor Control

Hardware Interface



(System-Side)

Pin No.	Signal Name	I/O
1	GND	—
2	OPEN	—
3	OPEN	—
4	CASS AUDIO	I
5	MIC DATA OUT	O
6	MOTOR CTL	—
7	CASS MTL CTL	—
8	OPEN	—

Figure 2-17 Interface Connector(J10)

2.2.9. Sound Subsystem

The nucleus of the sound subsystem is an analog multiplexer(MPX) which allows 1 to 4 different sound sources to be selected, amplified and sent to the audio output. The MPX and amplifier are configured so the amplifier's gain is unique to and consistent with each sound source. The amplifier is configured as a single-pole low pass filter with a 3dB cut-off frequency of 4.8 KHz. This filter is used to "round" off the corners of the square-wave signals. The output of the amplifier is supplied to the display interface and audio interface connector. If an external speaker is used, an external amplifier must be used to drive it. The audio output is 1V P-P alternating current and can drive a 10K Ohm or greater input impedance.

Sound source selection depends on the configuration of port bits PB5 and PB6of 8255A PPI. Power-on selects Timer 2.

PB6	PB5	Selected Sound Source
0	0	Timer 2 output
0	1	Cassette Audio
1	0	I/O Channel Audio
1	1	Sound Generator

(1) Timer 2 Output
Beep or simple tone is output.

(2) Cassette Audio
Allows use of recorded sound on the cassette tape as the sound source.

(3) I/O Channel Audio
Allows the Expansion Channel to connect such I/O devices as a sound synthesizer and have an output in Expansion Channel Connector (J4) Pin A30 "Audio" for input to multiplexer.

(4) Sound Generator
SN76489A is used.

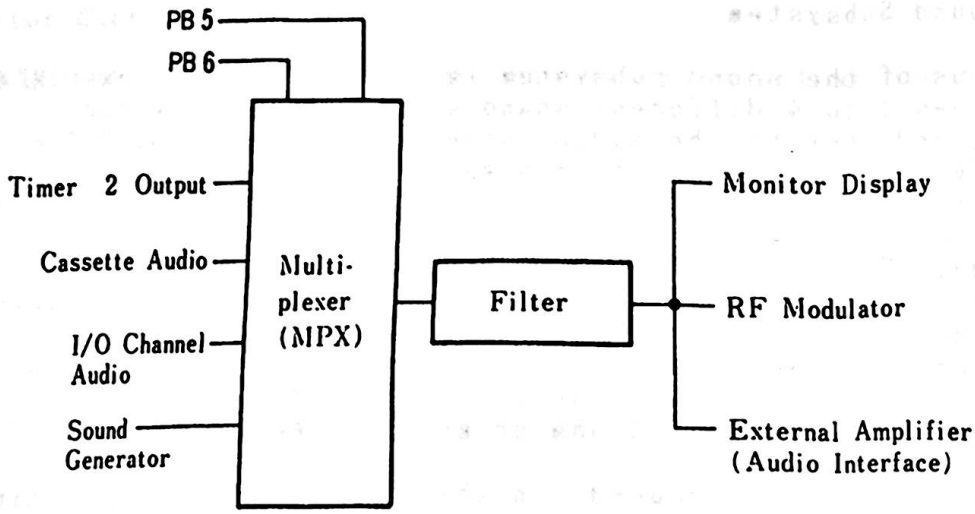


Figure 2-18 Sound Source

The RF modulator is included in the TV Adapter. It modulates the the audio signal and sends it with the screen signals to the TV set.

The IBM 5510 uses SN76489A as its sound generator. Its I/O address is C0 (Hex). This chip consists of a CPU Interface, Control Register, three tone generators, a Noise Generator,and an Audio Mixer. It is controlled by software and allows 3 simultaneous tone. A 3.579 MHz Clock Input is used.

The following is a description of each configured element:

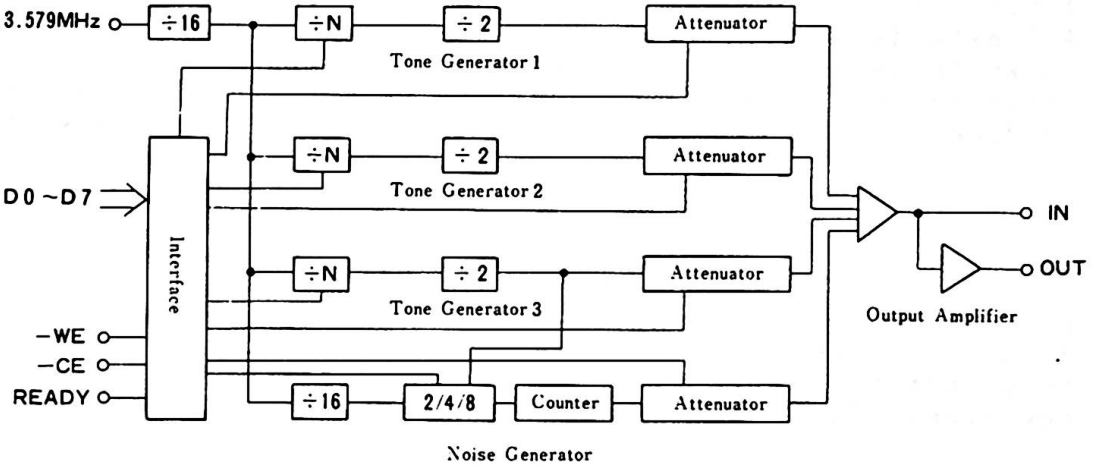


Figure 2-19 Sound Generator Block Diagram

CPU Interface

The system microprocessor interfaces with the SN76489A by means of the 8 data lines and 3 control lines (WE,CE and READY).
WE stands for Write Enable and CE for Chip Enable.
The SN76489A is controlled by 1 or 2 bytes of data issued by the CPU. Each tone generator requires 10 bits of information for the frequency. The data transfer is made twice for each byte. 4 bits of information are required to select the attenuation. A 1 byte data transfer is required. It requires 32 clock cycles to write data on the register and uses a READY signal for synchronization. The READY signal becomes "Low Level" at the CE leading edge and "High Level" at the end of the data write.

Control Register

The sound generator has 8 internal registers which are used to control the 3 tone generators and the noise source. During all data transfers to the sound generator, the first byte contains a three bit field which determines the destination control register. The register address codes are as follows;

Bit				Control Register Destination	Hex.
MSB	R2	R1	R0		
1	0	0	0	Tone 1 Frequency	8
1	0	0	1	Tone 1 Attenuator	9
1	0	1	0	Tone 2 Frequency	A
1	0	1	1	Tone 2 Attenuator	B
1	1	0	0	Tone 3 Frequency	C
1	1	0	1	Tone 3 Attenuator	D
1	1	1	0	Noise Control	E
1	1	1	1	Noise Attenuator	F

7	6	5	4	3	2	1	0
1	R2	R1	R0	F3	F2	F1	F0
MSB	1 st byte						LSB

7	6
0	X
2 nd byte	

Figure 2-20 Control Register Destination

Tone Generator

Each tone generator consists of;

- Frequency synthesis section (Programmable Counter)
- Attenuation section (Programmable Attenuator)

The frequency synthesis section requires 10 bits of information (Hex F0 - F9) from 2 bytes of data issued from the CPU.

Data Format (2 byte transfer)

MSB 1st Byte LSB

	Register			Data			
1	R2	R1	R0	F3	F2	F1	F0

MSB 2nd Byte LSB

		Data					
0	X	F9	F8	F7	F6	F5	F4

(1) Register is selected by R0 - R2

R2	R1	R0	Register Assignment
0	0	0	Tone 1 Frequency
0	1	0	Tone 2 Frequency
1	0	0	Tone 3 Frequency

(2) Frequency is set by F0 - F9 (10 bit binary number). F9 is the most significant bit and F0 is the least significant bit.

F9	F8	F7	F6	F5	F4	F3	F2	F1	F0

The frequency can be calculated by the following;

$$f = \frac{N}{32n} \quad (HZ)$$

Where N=Base clock frequency (3.579MHz)
n=10 bit binary number (F0 - F9)

The following is an example of obtaining a 440HZ frequency:

$$440 = \frac{3579000}{32n}$$

n = 254

Bit positions of F9 - F0 are 0 0 1 1 1 1 1 1 1 0.

The attenuator allows 15 stages of sound volume, ranging from 0dB to 28dB, based on the 1 byte of data issued from the CPU.

Data Format (1 byte transfer)

MSB LSB

	Register			Data			
1	R2	R1	R0	A3	A2	A1	A0

(1) Register is selected by R0 - R3.

R2	R1	R0	Register Assignment
0	0	1	Tone 1 Attenuator
0	1	1	Tone 2 Attenuator
1	0	1	Tone 3 Attenuator

(2) Sound volume is set every 2dB each by bits A0 - A3.

dB	A3	A2	A1	A0	Hex	dB	A3	A2	A1	A0	Hex
0	0	0	0	0	0	16	1	0	0	0	0
2	0	0	0	1	1	18	1	0	0	1	9
4	0	0	1	0	2	20	1	0	1	0	A
6	0	0	1	1	3	24	1	0	1	1	B
8	0	1	0	0	4	26	1	1	0	0	C
10	0	1	0	1	5	26	1	1	0	1	D
12	0	1	1	0	6	28	1	1	1	0	E
14	0	1	1	1	7	OFF	1	1	1	1	F

Noise Generator

The noise Generator consists of:
- Noise Source
- Attenuator

and can control the sound tone. (NF0:1, NF1:1 mode)
The noise source is a shift register with an exclusive -OR feedback-network. The feedback network has provisions to protect the shift register from being locked in the zero state (periodic noise).

Data Format (1 byte transfer)

MSB							LSB
1	R2	R1	R0	x	FB	NF1	NF0

(1) Register is selected by R0 - R2

R2	R1	R0	Register Assignment
1	1	0	Noise control
1	1	1	Noise Attenuator

(2) The kind of noise is selected by the bit FB

FB	Configuration
0	"Periodic" Noise
1	"White" Noise

(3) Shift rate is set by NF1 and NF0

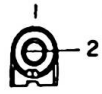
NF1	NF0	Shift rate
0	0	N/512
0	1	N/1024
1	0	N/2048
1	1	Tone 3 frequency output

(4) Attenuator control is the same as the Tone Generator.

Audio Mixer

The audio mixer sums the three tone-generator outputs and the noise generator output. The output buffer will generate up to 10mA.

Hardware Interface



Pin No.	Signal Name
1	GND
2	AUDIO

Figure 2-21 Audio Interface Connector(J11)

2.2.10. Beep Subsystem

The system beeper is a small piezoelectric speaker, which can be driven from one or two sound sources.
The two sound sources are as follows;

- 8255A PPI (PB4) Output
- 8253-5 Timer 2 Output (Timer clock)

  The input clock for this timer is 1.19MHz.

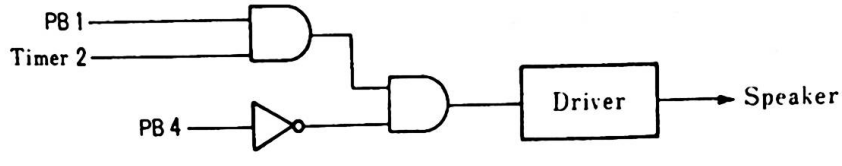


Figure 2-22 Beep Circuit

2.2.11. Keyboard Interface

The infrared link or keyboard cable provides communications between the keyboard and the system unit.

After the system unit reads serially encoded keyboard data from PC6, the software de-serializes them using the 8253 Timer 1.

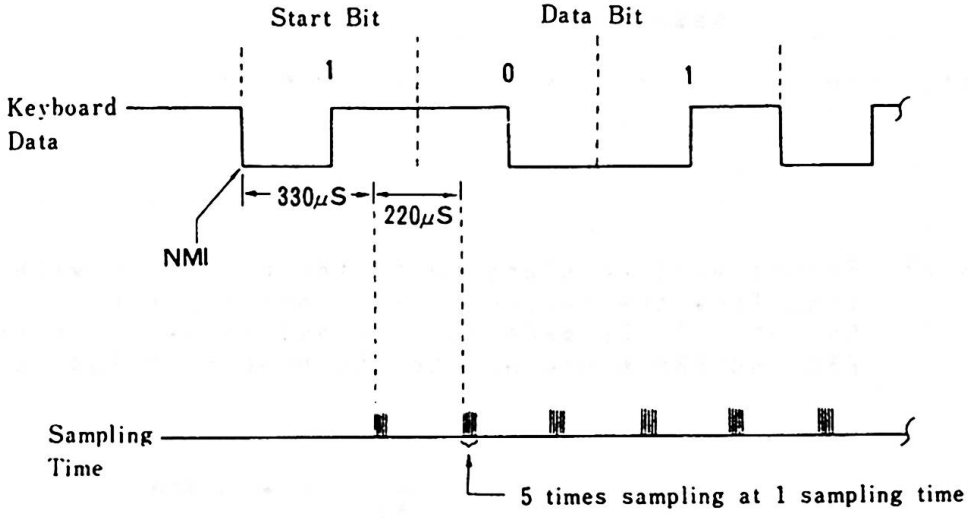


Figure 2-23 Sampling

The leading edge of the start bit will generate a non-maskable interrupt (NMI). Once the processor enters the NMI routine to handle the deserialization, the keyboard data line is sampled and the processor is waiting to sample the trailing edge of the start bit. When the trailing edge of the start bit is sampled, the processor will wait for 330 micro-sec and sample the first half of the first data bit. The processor then samples the keyboard data every half bit cell-time. The sampling interval is 220 micro-sec. The processor samples each half bit sample 5 times and will determine the logical level of the sample by majority rule. This enables the processor to discriminate against transient glitches and to filter out noise.

Detectable Error Conditions

The software checks whether the received data has an error or not. There are two kinds of errors, Phase and Parity errors.

Error	Cause
Phase Errors	The 1st half of the bit-cell sample is not equal to the inverse of the 2nd half of the bit-cell sample.
Parity Errors	The received encoded word did not maintain odd parity.

Remark) Errors will be signaled by the processor with a short tone from the beeper or external speaker.
Output will be made to external speaker, if both of PB5 and PB6 are 0 and to the beeper if PB4 is 0.

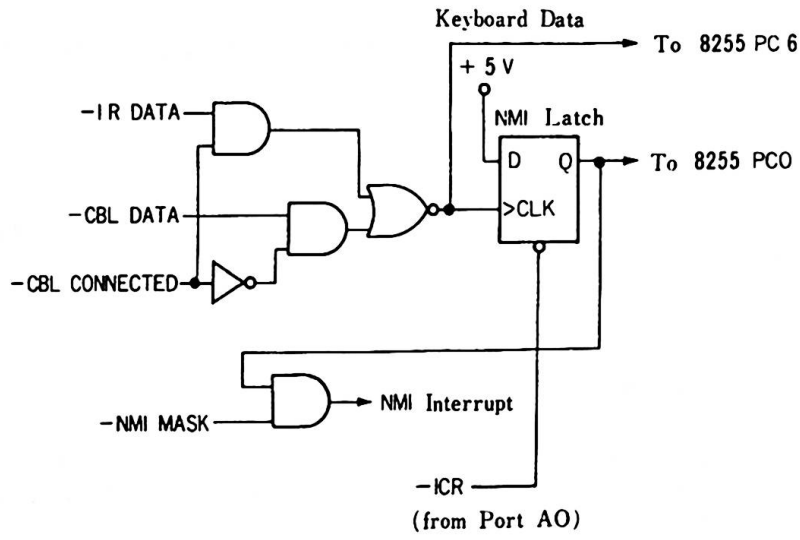


Figure 2-24 Keyboard Interface Block Diagram

Operational Parameters

The operational distance from infrared devices to the system is 5 meters (line of sight). Operational efficiency can be impaired by outside sources. Those sources are, excessively bright lights and high voltage lines. A keyboard cable is recommended for those instances.

Port A0

Port A0 is strongly related to the keyboard data handling and is described here. When a key is depressed, a serialized scan code is sent to the system unit. The latch causes an NMI on the first leading edge of the keyboard data if the enable NMI bit (port A0 bit D7) is on. The 8088 then reads the 8253 timer to determine when to interrogate the serial stream. Depending on whether the bit is on or off, 1 character of data is deserialized and the NMI Interrupt service routine resets the NMI latch to read the next NMI interrupt. The NMI latch is cleared by reading I/O port A0(Hex). As the latch can also be read on the 8255 PC0, the program can determine if a keystroke occurred while NMI was disabled by reading the status latch. For instance, during certain critical operations such as diskette I/O, the processor will mask off the NMI interrupt. As the keyboard input during this time cannot be serviced, the NMI latch should be checked later and the appropriate action should be taken. The system board provides for selection of keyboard data from either the cable or the IR receiver board. When the signal level -CBL CONNECTED on the keyboard cable connector is "0", data are read from the cable and when "1", from the IR receiver board. The 8088 processor uses one 8255 PPI bit (PC6) to do the software de-serialization of the keyboard.

Infrared Receiver

The infrared receiver is located in the system unit and has an infrared sensitive device that demodulates the signal transmitted from the keyboard and sends it to the system. The infrared sensitive device is located on the front of the card and receives its input through an opening in the front of the system unit box. There is also an infrared transmitter mounted on the receiver card for diagnostic purposes.

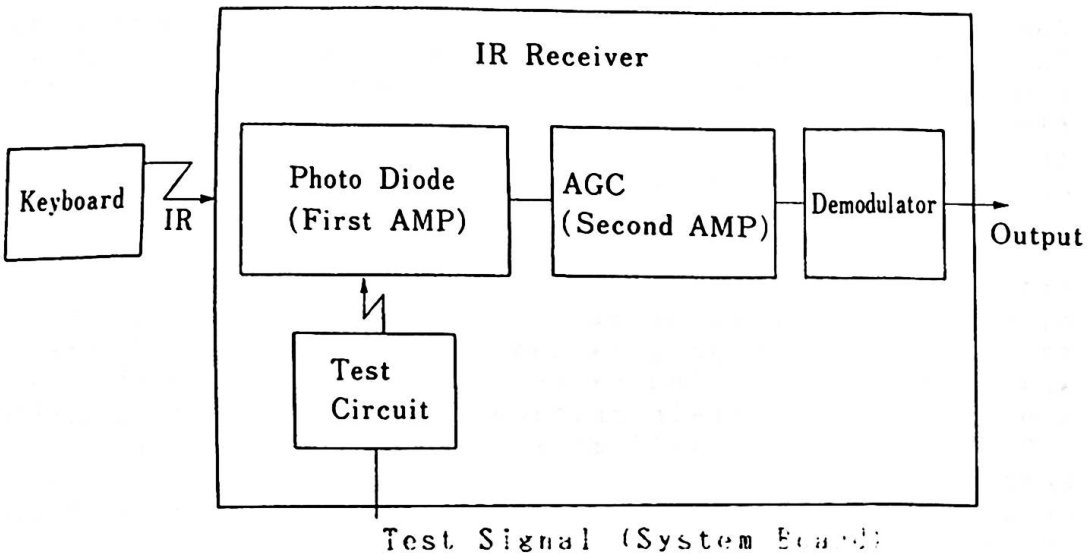
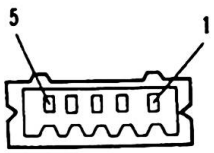


Figure 2-25 IR Block Diagram

The infrared receiver is mounted on the system board with a 5-pin connector.



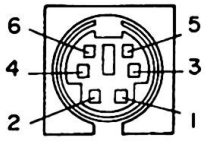
(IR-Side)

Pin No.	Signal Name	I/O
1	−IR KBD DATA	O
2	−IR TEST DATA	I
3	+5V	I
4	GND	−
5	+12 V	I

Figure 2-26 IR Connector specifications (J7)

Keyboard Cable Interface

The keyboard cable connects to the sytem unit connector J16. The -CBL CONNECTED signal notifies the system unit that the cable is connected. The system unit's IR receiver circuit is "disabled" by this signal. When the keyboard cable is used, the power is supplied from the system unit.



(System Side)

Pin No.	Signal Name	I/O
1	+5V	O
2	FG	–
3	GND	–
4	-CBL KBD DATA	I
5	-CBL CONNECTED	I
6	Open	–

Figure 2-27 Keyboard Cable Connector (J16)

2.2.12. Joystick Interface

Interface connectors are provided for two joysticks. The I/O address is common and is 201 (Hex).
When the data are read from the joystick, the following data are sent on the data bus:

Joystick 2		Joystick 1		Joystick 2		Joystick 1	
Switch #2	Switch #1	Switch #2	Switch #1	Coordinate Y	Coordinate X	Coordinate Y	Coordinate X
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Switch Input				Resistor Input		

Switch Input

Switch input is shown in data bits 7,6(or 5,4) when I/O address 201(Hex) is read. Each of the four switch inputs has a 1K ohm pull up resistor connected to +5V. When the button is depressed it is read as 0 and when the button is not depressed, it is read as 1.

Resistor Input

The joystick position is indicated by X and Y coordinates which are set by the two (for 1 joystick) variable resistors. When software writes data (dummy) in I/O address 201 (Hex), the hardware issues two kinds of pulses in accordance with the resistor value. This pulse output is shown in data bit 3,2 (or 1,0) when I/O address 201(Hex) is read. The software can determine the coordinate value based on this.

The width of the pulse is calculated by the following formula:

Time = 24.2 micro sec + 0.011 X R micro sec

where R is the resistance in ohms.

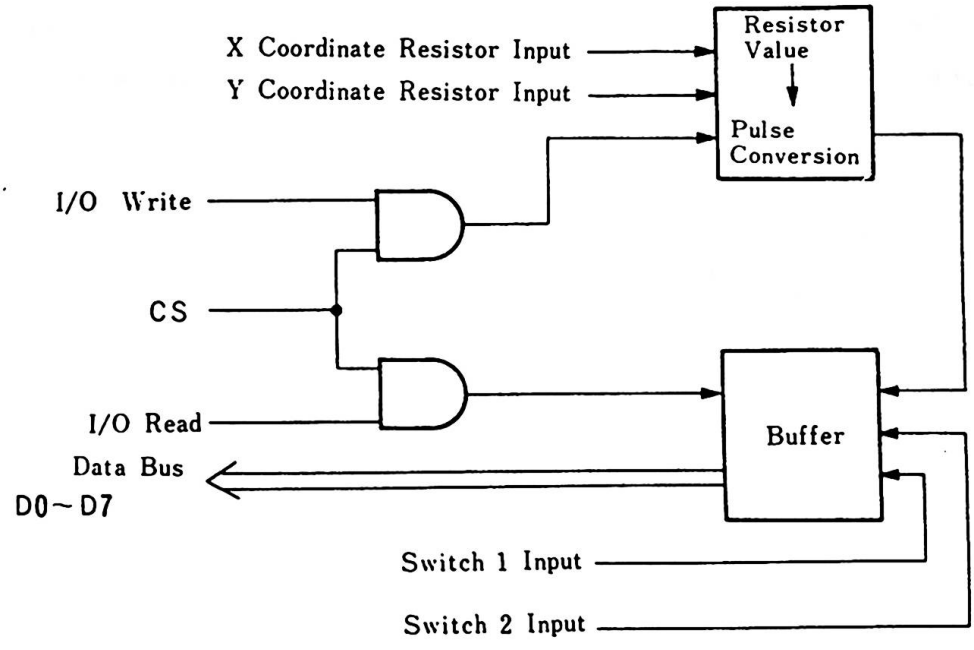
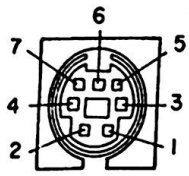


Figure 2-28 Joystick Interface Block Diagram

Hardware Interface

The joystick interface has two connectors in the rear of the system unit to connect two joysticks.



(System Side)

Pin No.	Signal Name	I/O
1	SW 1	I
2	SW 2	I
3	RX	I
4	RY	I
5	+5V	O
6	GND	–
7	Open	–

Figure 2-29 Joystick Interface Connector (J13, J14)

2.2.13. ROM Cartridge

When a ROM cartridge is inserted into either one of two identical slots in the front of the machine, ROM is added to the system unit or is replaced.

Each cartridge can hold up to 96 KB ROM. Cartridge selection is accomplished by the chip selects, each of which addresses one of the high 32KB memory blocks. Each cartridge uses up to three of the six chip selects and the selection is determined based on the intended use of the cartridge.

The ROM modules used are 250 ns access time devices. Typical modules are the Mostek MK37000, TMM23256, SY23128 or equivalent.

Cartridge ROM Storage Allocations

Address map of system ROM and ROM cartridge is as follows:

Address	System ROM		ROM Cartridge	
	Chip Select	Contents	Chip Select	Contents
D0000~D7FFF	—	—	EROM 2	Application Cartridge
D8000~DFFFF			EROM 3	
E0000~E7FFF		BASIC	EROM 4	
E8000~EFFFF	IROM 7		EROM 5	Special Cartridge
F0000~F7FFF		Kanji Dictionary	EROM 6	
F8000~FFFFF		BIOS	EROM 7	

Figure 2-30 Cartridge ROM Chip Select

- The system ROM on the board is selected by chip select IROM 7.
- System ROM address space is E0000 - FFFFF (Hex).
- Some portions of the address space for ROM cartridges will be used by the system ROM and some will not.

- Normally, application cartridges should use EROM 2 and EROM (D0000 - DFFFF).

- When EROM 4 - EROM 7 (E0000 - FFFFF) whose address space is shared by the system ROM, is used, addressing must be done carefully.

- EROM 4 and 5 are used for Language compilers which will not compete with BASIC.

- When EROM 6,7 are used for replacement of BIOS, the system characteristics can be changed. In this case, BASE 1 ROM (A04 Pin) or BASE 2 ROM (A09 Pin) should be wired to GND GND (A01 Pin) to notify the system.

- When EROM 4 - 7 chip selects which will compete with the system ROM is used, ROM space allocation of the system ROM will be as follows:

  1. F0000 - FFFFF (64KB)
  2. F8000 - FFFFF (32KB)
  3. All area deletion

- This change of ROM space allocation is made at the time of self-diagnostic test right after the system power-on or syst reset.

- When two cartridges of different modes are used in combination, "Error L" will be displayed. To continue, the carriage return key should be pressed.

There are some ground rules to follow in making application cartridges. The major rules are:

Conventions

(1) These conventions should be followed for "Initial Program Load-able" program cartridges. These cartridges should include:

Location	Contents
0, 1	55AA (Hex) : For English Mode
	AA55 (Hex) : For Native and Extension Video Mode
2	Length
3,4,5	Jump to initializing code
6	00 (Hex)
Last 2 bytes	CRC Bytes

1. Locations 0 and 1 are used to check whether there is a cartridge inserted during self-diagnostic test. 55AA (Hex) or AA55 (Hex) indicates that a cartridge is inserted.

2. Location 2 contains a length indicator representing the entire address space taken by the ROM on the cartridge. The algorithm for determining the contents of this byte is (ROM data size/512). The contents of this byte are used by the CRC check routine to determine how much ROM to check. Address space for a ROM cartridge is on a 2048 byte boundary.

3. Locations 3, 4, and 5 contain a Jump call which is used to jump during an initialization routine. For cartridge programs that are "IPL-able" (Basic or Assembler programs), this routine should set the INT 18 vector to point to their entry points. Other types of cartridges should merely return to the caller. Setting the INT 18 (Hex) vector will enable transfer of control to the cartridge program by the IPL routine.

4. Location 6 should be 00.

5. CRC byte: The last two locations of the address space used by the cartridge must be blank. CRC characters will be placed in these bytes when the cartridge is built. For the CRC algorithm, refer to the routine at label "CRC Check" in the BIOS listing.

(2) For cartridges which include DOS command words and run under DOS, the following conventions should be followed:

Location	Contents
0, 1	55AA (Hex) : For English Mode AA55 (Hex) : For Native and Extension Video Mode
2	Length
3-5	Jump to initializing code
6	The length of the command word
Z	Command name
Y	Location of jump, when command name is typed
X	The length of the next command word, or 00(Hex) when no more command words exist.
Last 2 bytes	CRC Bytes

1. Locations 0 and 1 are used to check whether there is a cartridge inserted during self-diagnostic test. 55AA (Hex) or AA55 (Hex) indicates that a cartridge is inserted.

2. Location 2 contains a length indicator representing the entire address space taken by the ROM on the cartridge. The algorithm for determining the contnts of this byte is (ROM data size/512). The contents of this byte are used by the CRC check routine to determine how much ROM to che Address space for a ROM cartridge is on a 2048 byte boundary

3. Location 3 contains a Jump call which is used to jump during an initialization routine. (There might be instances where only FAR RETURN call exists.)

4. Locations 6 - Y are command name pointers. If a cartridge has a routine called "Test" at location 0FB5 (Hex. offset from the start of the segment that the cartridge is in) that needs to be executed when "Test" is entered as a DOS command, the entries (in Hex) at locations 6 - Y would be:

```
Location 6      04 (4 byte length command name)
Location Z      54(T), 45(E), 53(S), 54(T) T E S T
Location Y      B50F (offset ; 0FB5)
```

Until the count field changes to 00, which occurs after the last name pointer (6-Y), the next name pointer will be set.

5. CRC byte: The last two locations of the address space used by the cartridge must be blank. CRC characters will be placed in these bytes when the cartridge is built. For the CRC algorithm, refer to the routine at label "CRC Check" in the BIOS listing.

(3) The cartridge chip selects should specify EROM2 or EROM3, when the cartridge is designed to run application programs which are written in the Basic language, because the Basic Interpreter addresses E0000 (Hex). When the Basic Interpreter is activated, a check is made to see if there is a cartridge application at D0000 (Hex). When there is a cartridge application and its format is correct, the application is loaded into RAM and run.

The format for interpretable BASIC code must be as follows:

Location	Contents
0, 1	55AA (Hex) : For English Mode
	AA55 (Hex) : For Native and Extension Video Mode
2	Length
3	CB (Hex)
4	AA (Hex)
5	55 (Hex)
6	00 (Hex)
7	FF (Hex) for unprotected Basic Program
	FE (Hex) for protected Basic Program
8	Starting address of Basic Source Code
n	FF (Hex). Padding to next 2048 byte boundary
Last 2 bytes	CRC Bytes

1. Locations 0 and 1 are used to check whether there is a cartridge inserted during self-diagnostic test. 55AA (Hex) or AA55 (Hex) indicates that a cartridge is inserted.

2. Location 2 contains a length indicator representing the entire address space taken by the ROM on the cartridge. The algorithm for determining the contents of this byte is (ROM data size/512). The contents of this byte are used by the CRC check routine to determine how much ROM to check. Address space for the ROM cartridge is on a 2048 byte boundary.

3. Location 3 must be CB (Hex. Far Return instruction).

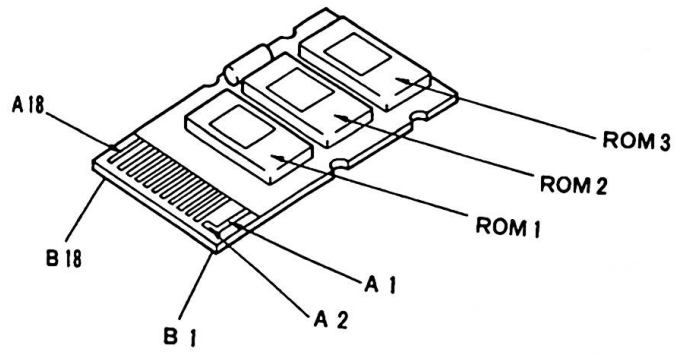
4. Locations 4 and 5 contain the word AA55 (Hex). This is used by the Basic Interpreter to check for the presence of a Basic Application cartridge.

5. Location 6 must be 00.

6. Location 7 can either FF (Hex) to indicate an unprotected Basic Program, or FE (Hex) to indicate a protected program.

7. Location 8 must be the start of the Basic Application program.

8. CRC byte : The last two locations of the address space used by the cartridge must be blank. CRC characters will be placed in these bytes when the cartridge is built.



ROM No	Address	Chip Select
ROM 3	D8000~DFFFF	EROM 3
ROM 2	E0000~E7000	EROM 4
ROM 1	E8000~EFFFF	EROM 5

Figure 2-31 ROM Cartridge

Hardware Interface

There are two slots for ROM cartridge insertion and they both have the same function. The following is a description of pin allocations and signals:

Signal Name	Pin (A)	Pin (B)	Signal Name
GND	A1	B1	−CTWR
−CARTRIDGE RESET	2	2	−EROM7
−EROM5	3	3	−EROM3
−BASE 1 ROM	4	4	A14
A13	5	5	A12
A8	6	6	A7
A9	7	7	A6
A11	8	8	A5
−BASE 2 ROM	9	9	A4
A10	10	10	A3
D7	11	11	A2
D6	12	12	A1
D5	13	13	A0
D4	14	14	D0
D3	15	15	D1
−EROM2	16	16	D2
−EROM4	17	17	−EROM6
+5V	18	18	Open

Figure 2-32 ROM Cartridge Connector (J1,J2)

Remarks) 1. I/O is referred to as "System-side" view.
2. Each signal is at a standard TTL level.

Signal	I/O	Description
A0 - A14	O	Processor Address line A0 - A14
D0 - D7	I	Processor Data lines
-EROM2 - 7	O	These chip select lines are used to select ROS modules.
-BASE 1 ROM	I	When wired to A01 pin (GND), -EROM7 is made available. Addresses F8000 - FFFFF (Hex) are used by the ROM cartridge.
-BASE 2 ROM	I	When wired to A01 pin (GND), -EROM6 is made available. Addresses F0000 - F7FFF (Hex) are used by the ROM cartridge.
-CARTRIDGE RESET	I	This input when "low" causes a reset to the system. The system will remain reset until this line is brought back to "high". This tab is usually wired with an L shaped land pattern to the GND at A02 which provides a momentary "reset" when a cartridge is inserted or removed
-CTWR	O	Memory Write Signal (open collector).

2.2.14. Printer Interface

Centronics specifications for an 8 bit parallel interface are provided.

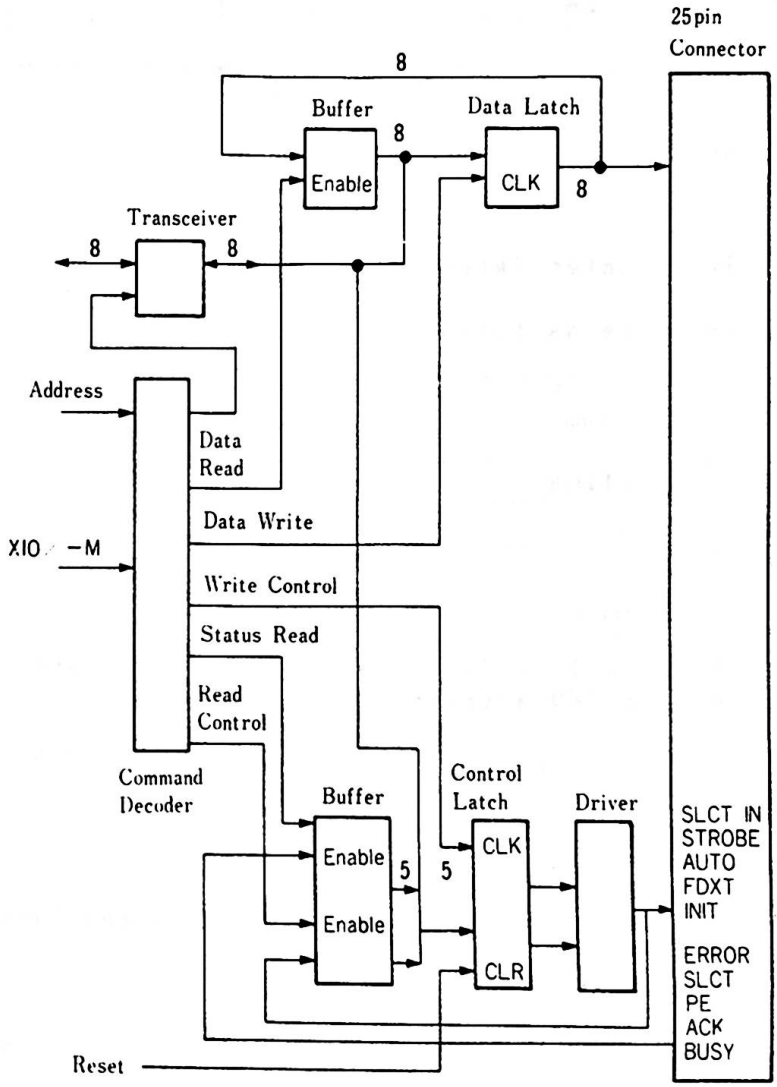


Figure 2-33 Printer Interface Block Diagram

The interrupt level is 7. To make it possible to interrupt, control latch bit 4 should be changed to 1.

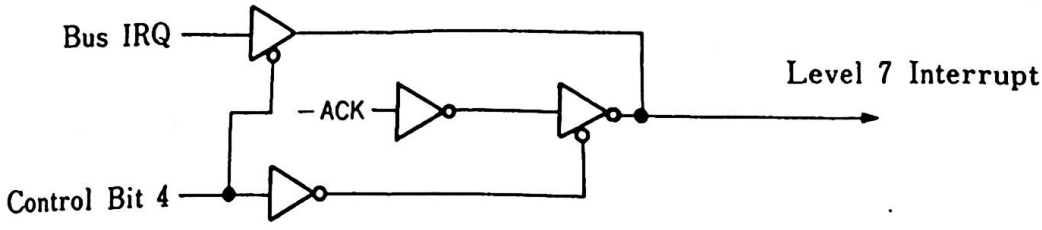


Figure 2-34 Printer Interrupt

I/O Addresses are as follows:

I/O Address (Hex)	Allocation
378.37C	Data Latch
379.37D	Status
37A.37E	Control Latch
37B.37F	Reserved

Remarks) Address line A2 is not decoded and therefore either on of two I/O addresses are to be used.

The following descriptions apply when the IBM 5513 Printer is connected with the printer interface:

Data Latch (I/O address 378 or 37C)

Printer output data are temporarily kept in the Data Latch.

Status (I/O address 379 or 37D)

Input data read at I/O address 379 or 37D contain the following printer status:

Bit 7 (BUSY): When "0", a printer cannot accept data. Following are instances where "0" is set:

1. Data Transfer
2. Printing
3. An error occurs

Bit 6 (ACK): When the printer finishes accepting data, it will return the signal -ACK to notify the system that it got ready to accept the next data.

Bit 5 (PE): When the printer paper runs out, "1" is set.

Bit 4 (SELECT): When the printer is selected, "1" is set.

Bit 3 (ERROR): When a printer error occurs, "0" is set.

Bit 2 (KJ-CD): When "1", connection to an Image Printer is assume.

Bit 1,0: Open

Control Latch (address 37A or 37E) containes the printer control signal.

Bit 7, 6, 5 : Open

Bit 4 (INTR): When "1", an interrupt is possible (Level 7).

Bit 3 (SELECT IN): When "1", it is possible to control the printer from the CPU.

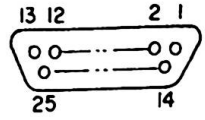
Bit 2 (INIT): When "0", the printer is initialized. (More than 50 microseconds is required.)

Bit 1 (AUTO FD): When "1", a line feed is performed after each line is printed.

Bit 0 (STROBE): 5 microsecond pulse (level "1") allows sending of data for this duration.

Hardware Interface

The connector used is a D shaped connector and the signal level is at standard TTL.



(System Side)

Pin No.	Signal Name	I/O
1	−STROBE	O
2−9	DATA 0-7	O
10	−ACK	I
11	BUSY	I
12	PE	I
13	SLCT	I
14	−AUTO FD	O
15	−ERROR	I
16	−INIT	O
17	−SLCT IN	O
18	KJ-CD	I
19−25	GND	—

Figure 2-35 Printer Interface Connector (J12)

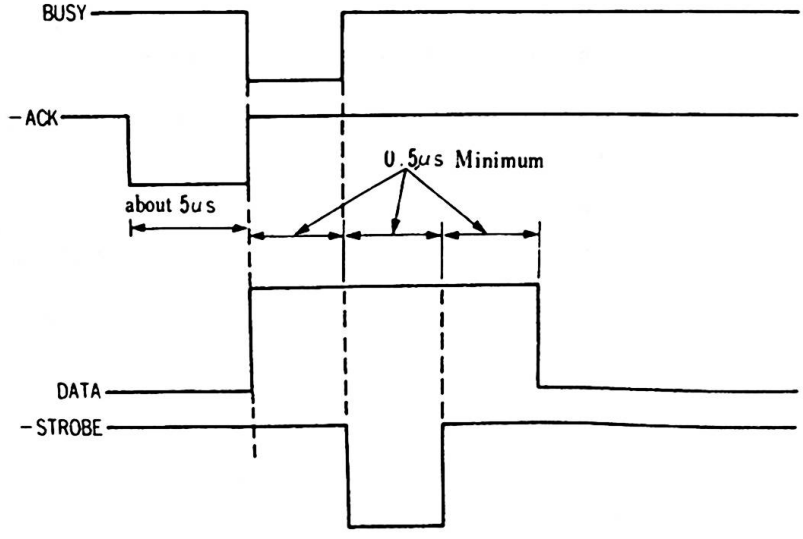


Figure 2-36 Timing Chart

2.3. Keyboard

The cordless keyboard is battery powered and interfaces to the system unit with an infrared (IR) optical link. There are two types of keyboards. One is a 83-key (Compact Keyboard) and the other is a 102-key (Full Keyboard). The keys are arranged in a standard typewriter layout with the addition of function keys and cursor keys. All keys are typamatic keys. (When a key is pressed for more than 0.6 seconds, that key code is repeatedly sent out at a speed of 11 characters per second.) The microprocessor in the keyboard is normally in a standby-power-down mode until a key is depressed. When an optional keyboard cable is used, power is supplied from the system unit and signals are transmitted via the cable.

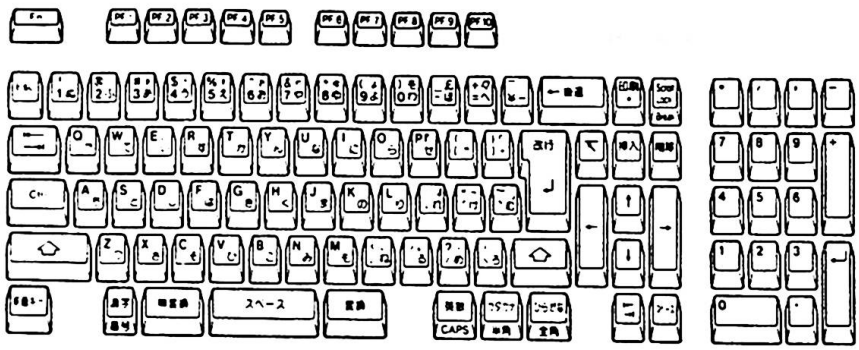


Figure 2-37 Full Keyboard

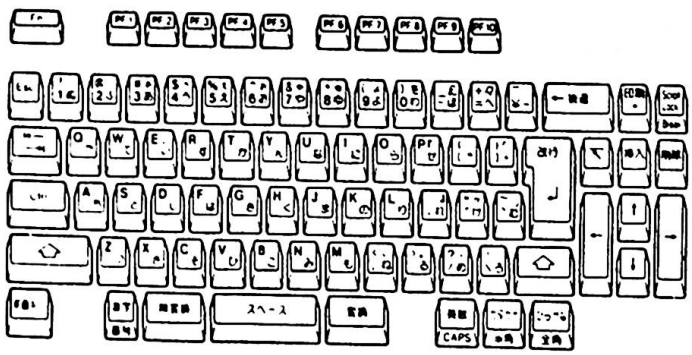


Figure 2-38 Compact Keyboard

Transmitter

Serially encoded scan codes are transmitted to the system unit with a bit cell of 440 micro-sec. (Odd parity) After each scan code is transmitted, they are added with 11 stop bits. When infrared link is used, logical 1 contains a 40 KHz(50 % duty ratio) carrier burst signals.

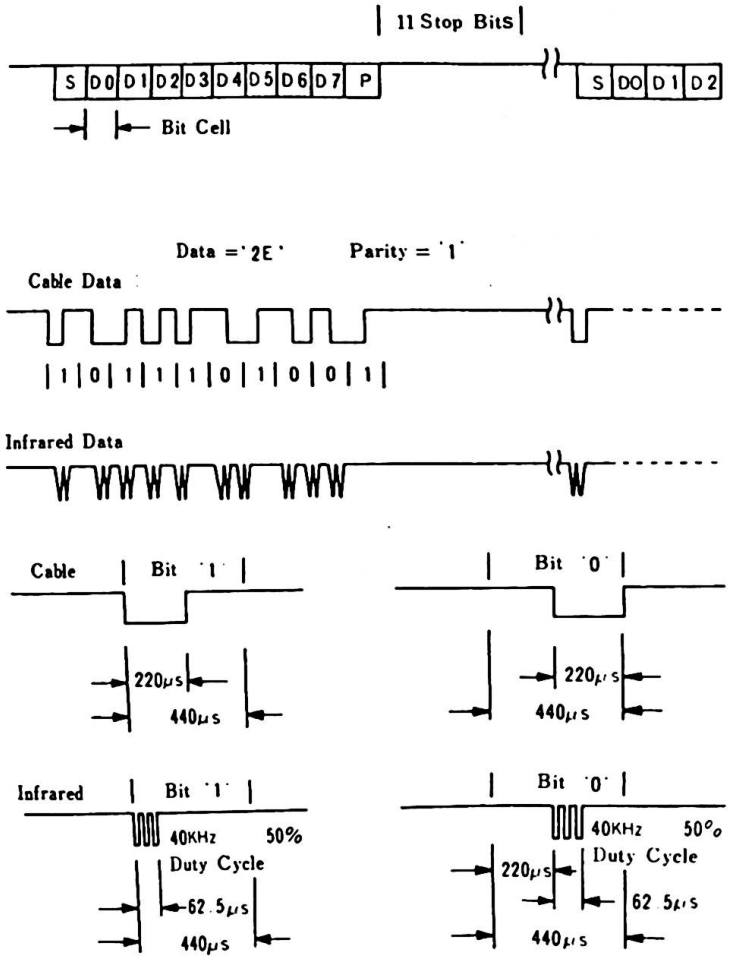


Figure 2-39 Keyboard Data Transmission Format

Micro Processor

The keyboard contains a microprocessor and its functions are as follows:

- Suppress Chattering
- Scan Code Conversion
- Typamatic
- N Key Roll-over
- Simultaneous Keystroke check
- From parallel to serialized Scan Code Conversion
- Biphase Modulation

When keys are not pressed, the keyboard is in a standby-power down mode.

Scan Code

Each key is assigned a scan code and when a key is pressed or released, serially encoded codes are sent to the system unit. The scan code when a key is released, is obtained by adding 80 (Hex) to the scan code when a key is pressed. For instance, the scan code of "ESC" when pressed is "01" and is "81" when released. (Reference : Appendix C)

Error Detect

The keyboard has an N Key Roll-over function. Every combination is possible at N = 2 and when at N = 3, more than 95 % will be correctly processed. When Keys of unavailable combination are pressed, scan code 55 (Hex) is sent to the system unit as a phantom key and notifies it that the keystrokes are in error. In this case nothing will appear on the display or incorrect words will be displayed.

2.4. Power Unit

The power unit resides in the system box and provides direct current power to a system board, a diskette drive and cooling fan. The output is +5V, +12V, -12V.

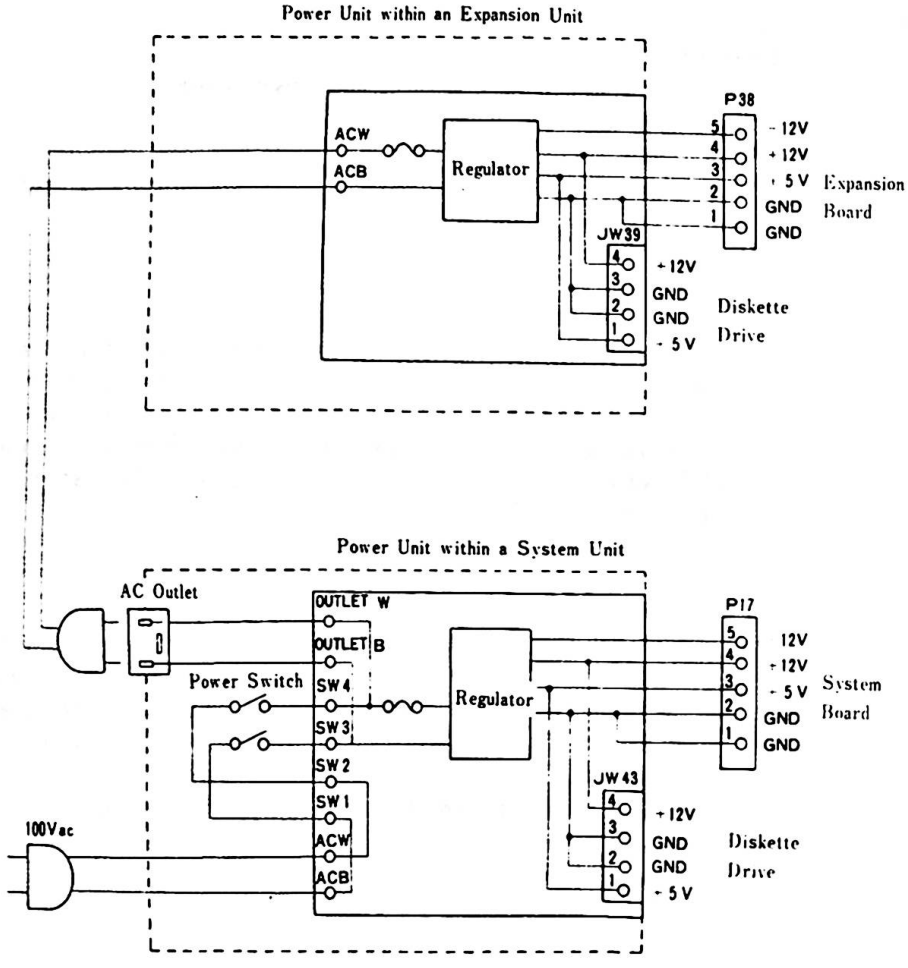


Figure 2-40 Power Unit Logic Diagram

Power Specifications

Voltage	Rating Current	Maximum Current	Ripple	Variance
+ 5V	3.6 A	5A	100mV	± 5%
+12V	0.8 A	1.15A	150mV	± 5%
−12V	0.05A	0.05A	100mV	± 5%

3. Video Subsystem

The video subsystem allows the IBM color display, a wide variety of television-frequency monitors, or ordinary home TV sets to display characters and graphics. It can operate in black-and-white as well as in color mode and provides a lightpen interface. In this chapter, an overall description of the VSS is mentioned first, followed by descriptions of the functions of the three video processors (VP1, VP2 and VP3).

3.1. Introduction

The video subsystem is implemented using a CRT controller and a video gate array. It contains three video processors.

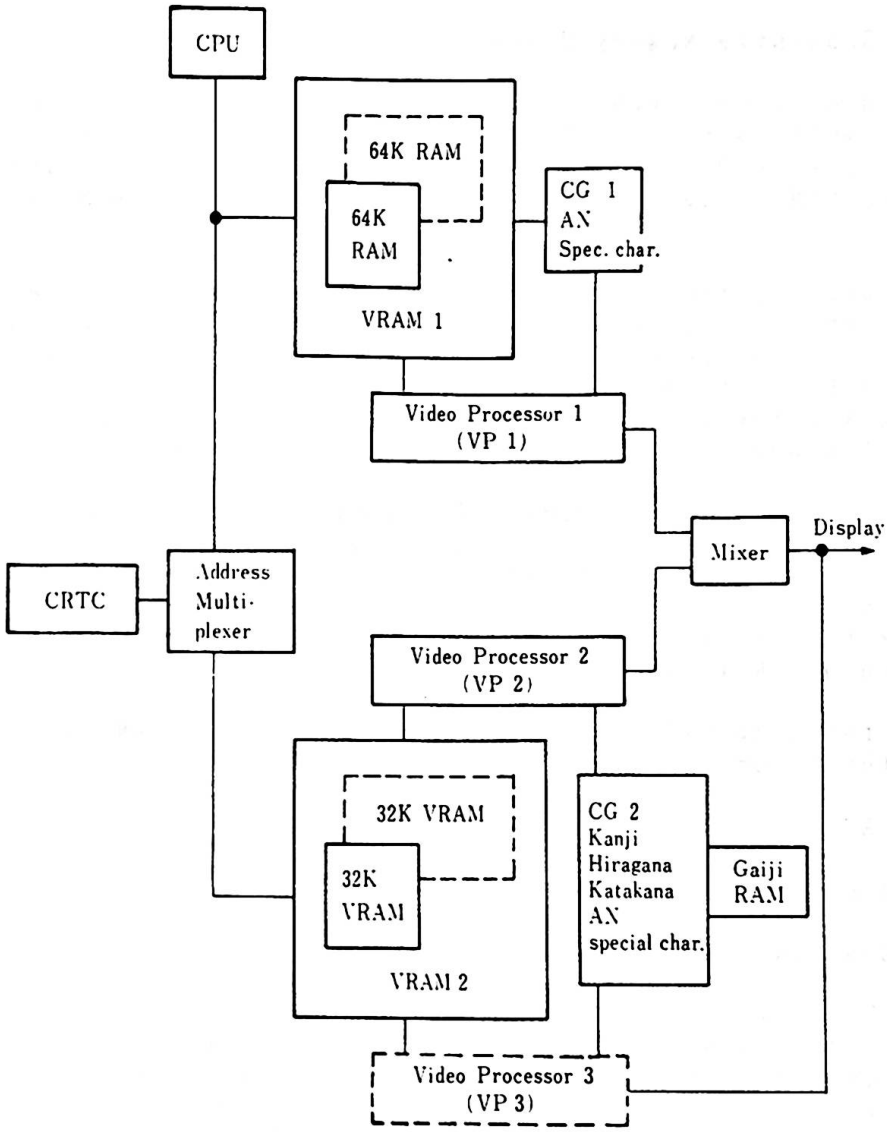
Video processor 1 (VP1) is used both in English and in Native mode. It can process alphanumerics and special characters (8 x 8 dot) which are compatible with PCjr. It can also process a maximum of 640 x 200 dot 4 color graphics.

Video processor 2 (VP2) is used in Native mode. It can process Kanji, Hiragana, full and half-size Katakana characters, alphanumerics, special characters and a maximum of 640 x 200 dot 4 color graphics.

Video processor 3 (VP3) is contained in an optional Extension Video Card and is used in Extension Video mode. It can process Kanji, Hiragana, full and half-size Katakana characters, alpha-numerics, special characters, grid lines which are compatible with the IBM Multistation 5550 and a maximum of 720 x 512 dot two color graphics.

There are four video frequencies available depending on operational mode. They are as follows:

3.5	MHz	: 160 x 200 dot
7	MHz	: 320 x 200 dot
14	MHz	: 640 x 200 dot
20	MHz	: 720 x 512 dot



Remarks) 1. CG : Character Generator
2. --- : Option
3. AN : Alphanumerics

Figure 3-1 Video Subsystem Block Diagram

3.2. Video Subsystem Memory Usage

The base video color/graphics subsystem accesses 64K bytes of system read/write memory (RAM) and 32K bytes of video RAM. When an optional RAM card (64 KB) and an Extension Video Card (32 KB video RAM included) are installed, video RAM space is expanded.

The memory used by the VSS is separated into VRAM 1 and VRAM 2. VRAM 1 consists of 64 KB memory on the system board which is provided as a standard feature and 64 KB of expanded memory. VRAM 2 is 64 KB of video RAM (32 KB of expanded memory). The reading/writing of video RAM is usually performed by first specifying the page and by using the virtual address.

If an additional 128 KB of memory is added, it can not be accessed by the VSS and cannot be used as video RAM. When 192KB of expanded memory (one 64KB RAM card and one 128KB RAM card) is added, the address space of the 128KB expansion comes first, followed by that of the 64KB on the system board and the 64KB expansion card (Reference: 5.5 Memory Map).

The memory for displaying screens by the VSS (video RAM) uses the following three concepts:

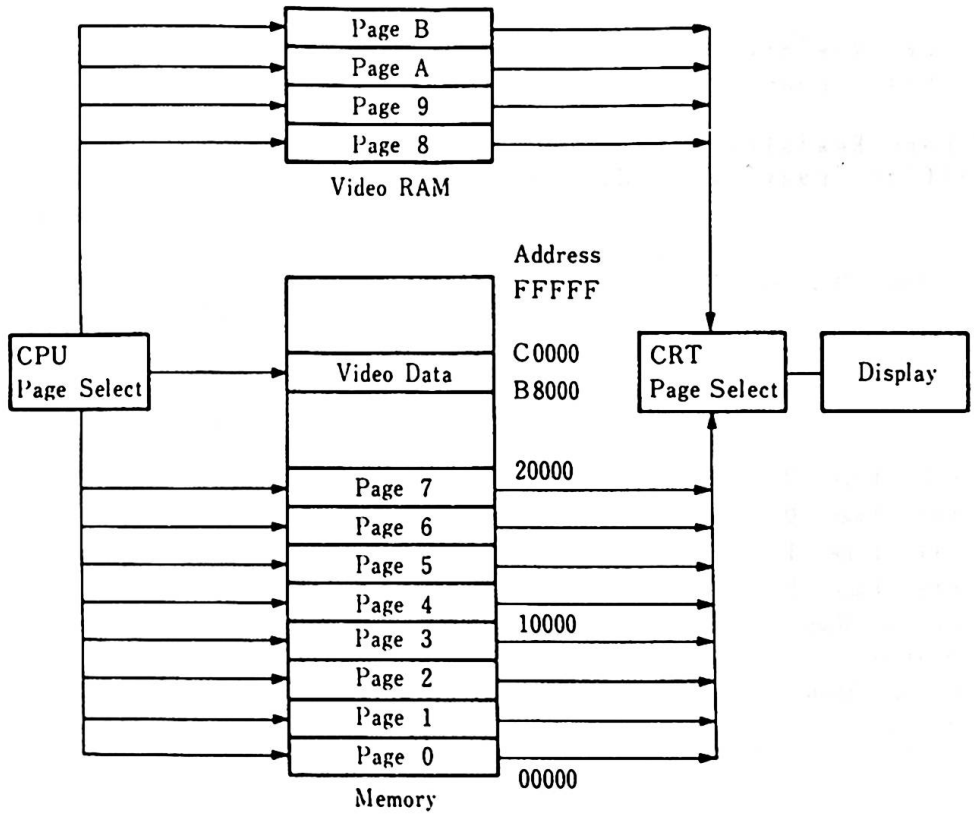
- Virtual Address
- Page
- Page Register

Virtual Addresses

B8000 - BFFFF (Hex) are virtual addresses for VP1 and VP2. A0000-AFFFF (Hex) are virtual address for VP3. The memory for these addresses does not physically exist. When a data read/write is performed from the CPU to these addresses, the read/write is performed on the pages (any of 0 - B) which are specified by the CPU page register.

Page

A page is a memory block separated into 16 KB units. It is part of a total of 192 KB memory which consists of 64 KB of base memory and 64 KB of expanded memory, making total of 128 KB (VRAM 1) plus 64 KB of Video RAM (VRAM 2). One screen is displayed with 1, 2 or 4 pages. The number of pages necessary to display a screen depends on the number of colors and the display resolution. Four pages are necessary to display a 720 x 512 dot 2 color or 360 x 512 dot 4 color screen.



Remarks) The memory addresses shown above are correct for the case where only 64KB of expanded memory is installed. (Reference : 5.5 Memory Map)

Figure 3-2 Video Memory Map

Page Register

Pages 0-7 are allocated for VRAM 1, while pages 8-B are for VRAM 2. The pages are selected by the page register. The I/O address for VRAM 1 is 3DF (Hex) and for VRAM 2 is 3D9 (Hex). Both are write-only 8 bit registers. There are two kinds of page registers, the CPU Page Register and the CRT Page Register.

- CPU Page Register
  specifies "page" for read/write from CPU to video RAM.

- CRT Page Register
  specifies "page" for display on the screen.

Page Register 1 (I/O address 3DF)			Page Register 2 (I/O address 3D9)		
Bit	Meaning	Remarks	Bit	Meaning	Remarks
0	CRT Page 0	CRT Page 0-7 specified	0	CRT Page 0	CRT Page 8-B specified
1	CRT Page 1		1	CRT Page 1	
2	CRT Page 2		2	Open	
3	CPU Page 0	CPU Page 0-7 specified	3	CPU Page 0	CPU Page 8-B specified
4	CPU Page 1		4	CPU Page 1	
5	CPU Page 2		5	Open	
6	Display Mode Selection		6	Open	
7	Display Mode Selection		7	Open	

Figure 3-3 Page Register

Bits 0 - 2
Which page in VRAM 1 (16 KB/page) to display is determined by these three bits in page register 1.
Which page in VRAM 2 (16 KB/page) to display is determined by these three bits in page register 2.

Bits 3 - 5
These three bits select the page for the data transfer in case the data reside at virtual addresses B8000-BFFFF (Hex). In Extension Video mode, bits 3 and 4 should be set to "0" and virtual addresses of A0000-AFFFF (Hex) are used.

Bit 6, 7 (Only for VP 1)
VP1 display mode is selected.

Bit 7,	Bit 6	Meaning
0	0	All text modes
0	1	Low resolution graphics mode
1	0	Open
1	1	High resolution graphics mode

Low Resolution Graphics Modes:

- 160 x 200 dots 16 colors
- 320 x 200 dots 4 colors
- 640 x 200 dots 2 colors

High Resolution Graphics Modes:

- 320 x 200 dots 16 colors
- 640 x 200 dots 4 colors

The CPU page register allows pages other than that being displayed to be selected and to be read/written. A change of the contents of the CPU register does not affect the displaying screen, but when the contents of the CRT page register are changed, the screen being displayed will change. Page changes are performed by the CRT page register by BIOS at the time of vertical line retrace of the display raster. Because of this, you can use page changes effectively to achieve an animation effect without garbling the screen.

(Reference: 2.2.6 Memory Space and I/O Address Setting)

3.3. Character Generator

The Video Subsystem has two different character generators: Character Generator 1 (CG1) and Character Generator 2 (CG2).

3.3.1. Charactor Generator 1 (CG1)

CG1 consists of 2 KB of storage, for alphanumerics and special characters and is used only by VP1. One character consists of 8 x 8 dots. CG1 has a 350 ns access time/350 ns cycle-time. An NM2364 (or equivalent) is used.

The kinds of characters contained in CG1 are as follows:

- 16 special characters for games
- 15 characters for word processor editing
- 96 ASCII graphic characters
- 48 English characters
- 48 graphic characters for business use
- 16 symbols
- 15 scientific characters

3.3.2. Character Generator 2 (CG2)

CG2, used by VP2 and VP3,uses 128 KB of ROM.
The following are its characteristics:

1. The IBM 5510 uses the JIS Level 1 Kanji character set. CG2 contains Hankaku (half-size)characters of 8 x 16 dots such as alphanumerics, special characters and Katakana, and Zenkaku (full-size) characters of 16 x 16 dots such as Hiragana and JIS Level 1 character set Kanji. CG2 contains 8 x 8 alpha/numerics, special characters and Katakana which are used for 80 x 25 text mode display by VP2.

2. CG2 consists of four 32 KB ROMs of type TM23256(or equivalent)

3. Access time and cycle-time are both 250 ns.

4. CG2 can be accessed from the VP2 and VP3 code buffer and fonts can be read through BIOS. In order to display Kanji characters in graphics mode, CG2 is read by BIOS and is written on the video RAM.

5. In order to distinguish one-byte (Hankaku) codes from two-byte (Zenkaku) code, 256KB of virtual address space are used. The codes are then compressed 128KB of Kanji ROM (CG2).

6. When a Kanji is written on video RAM, the hardware uses the Kanji code to point to a ROM Kanji address and the character at that address is displayed.

7. In memory, CG2 is allocated to 80000 - BFFFF.

3.4. Display Function of VP1 and VP2

VP1 and VP2 have the following two display modes:

- Text mode
- Graphics mode

The Graphic displaying functions of VP1 and VP2 are the same, but their text displaying functions are different:

VP1/VP2 Displaying functions

Function	VP1	VP2
Text Display	40 char. × 25 lines (AN) 80 〃 × 25 〃 (AN)*	20 char. × 11 lines (Kanji) 40 〃 × 11 〃 (〃) 40 × 25 (ANK) 80 × 25 (ANK)
Graphics Display	160 ×200 dot 16 colors 320 ×200 〃 4 〃 320 ×200 〃 16 〃 * 640 ×200 〃 2 〃 640 ×200 〃 4 〃 *	The same as VP1
Character display possible	Alphanumeric Special Characters (8 × 8 dot)	JIS Level 1 Kanji (16 × 16 dot) Hiragana (16 × 16 dot) Katakana, Alphanumeric, Special Char. (16 × 16 or 8 × 16 dot) Katakana, Alphanumeric, Special Char. (8 × 8 dot)
Super-Impose	Graphics display only	Graphics and Text display

Remarks) 1. * indicates that 64 KB expanded memory is required for VP1.
2. AN stands for Alphanumeric and ANK for Alphanumeric and Katakana.

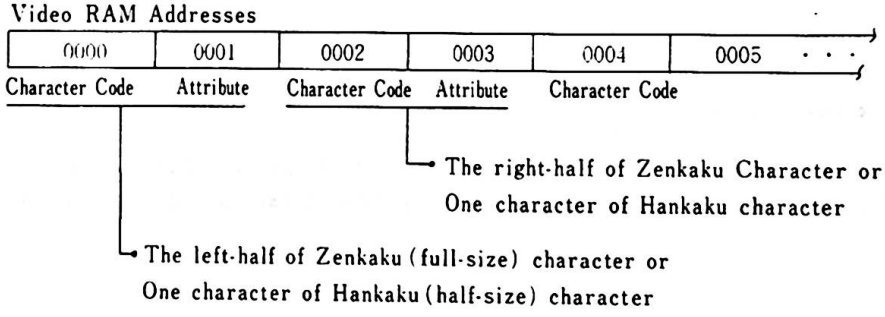
Figure 3-4 VP1 and VP2 Display Function

3.4.1. VP1 Text Display

In color-text mode, 16 colors are available either for the foreground or the background. The number of background colors, however, becomes 8 when "blink" is used. One of 16 colors is available for use in the text mode screen border.

The character generator contains 256 character fonts in its 2 KB ROM (CG1).

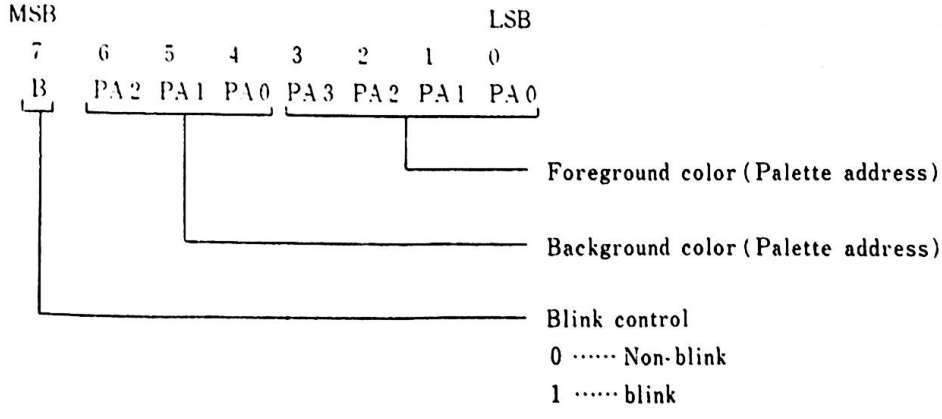
The display character codes are stored in even addresses, while their character attributes are stored in the odd addresses next to them.



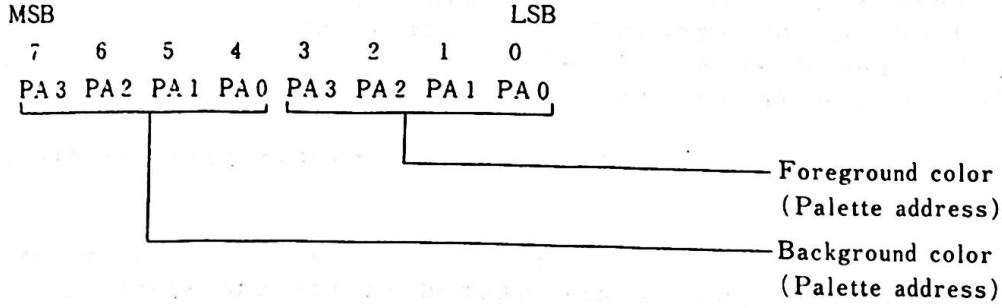
VP1 Attributes

(1) Blink is possible

In the mode control 2 register, a "1" makes blink possible. The number of background colors, however, is reduced to a maximum of 8.



(2) Blink is not possible

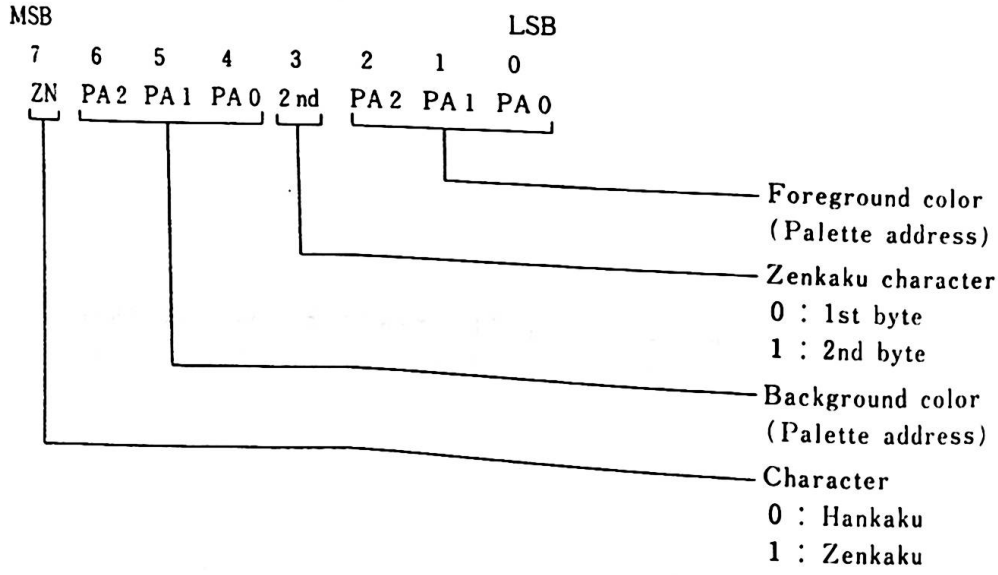


3.4.2. VP2 Text Display

VP2 Attributes

(1) Blink is possible

In the VP2 mode control 1 register, if bit 6 is "0", it is possible to display alphanumeric mode characters. (8 x 8 dots to display one character. Reference: 3.4.4 Video Gate Array.) In this display mode, an attribute has the same meaning as in VP1.



3.4.3. VP1 & VP2 Graphics Display

VP1 and VP2 support the following six graphics modes:

Graphics Mode 1

Graphics mode 1 displays on an ordinary TV set, 12" color display or 14" color display and has the following characteristics:

- 160 dot horizontal and 200 dot vertical display
- 16 colors are available for each dot
- 16 KB memory per screen are necessary, as the color is specified for each dot
- 1 byte is required to specify 2 dots
- This corresponds to Screen Mode 1 in Basic.

MSB							LSB
7	6	5	4	3	2	1	0
PA3	PA2	PA1	PA0	PA3	PA2	PA1	PA0
1st display dot				2nd display dot			

Graphics Mode 2

Graphics mode 2 displays on an ordinary TV set, 12" color display or 14" color display and has the following characteristics:

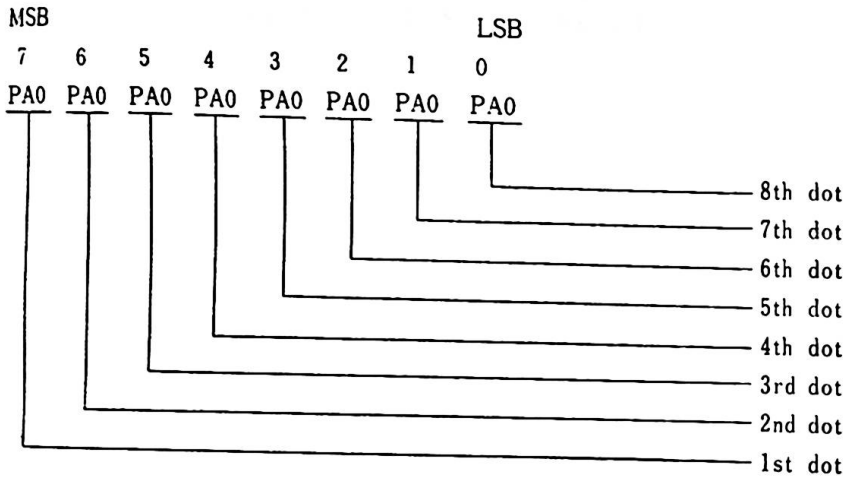
- 320 dot horizontal and 200 dot vertical display
- 4 out of 16 colors can be displayed
- 16 KB memory per screen are necessary, as the color is specified for each dot
- 1 byte is required to specify 4 dots
- This corresponds to Screen Mode 2 in Basic.

MSB							LSB
7	6	5	4	3	2	1	0
PA1	PA0	PA1	PA0	PA1	PA0	PA1	PA0
1st display dot		2nd display dot		3rd display dot		4th display dot	

Graphics Mode 3

Graphics mode 3 displays high-resolution 2 color graphics and is possible only on high-resolution displays. This mode has the following characteristics:

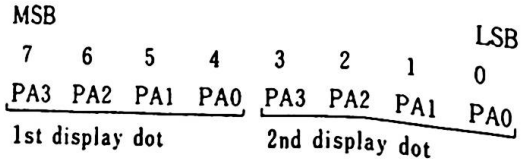
- 640 dot horizontal and 200 dot vertical display
- 2 out of 16 colors can be displayed
- 16 KB memory per screen are necessary, as the color is specified for each dot
- 1 byte is required to specify 8 dots
- This corresponds to Screen Mode 3 in Basic.



Graphics Mode 4

Graphics mode 4 displays on an ordinary TV set, 12" color display or 14" color display and has the following characteristics:

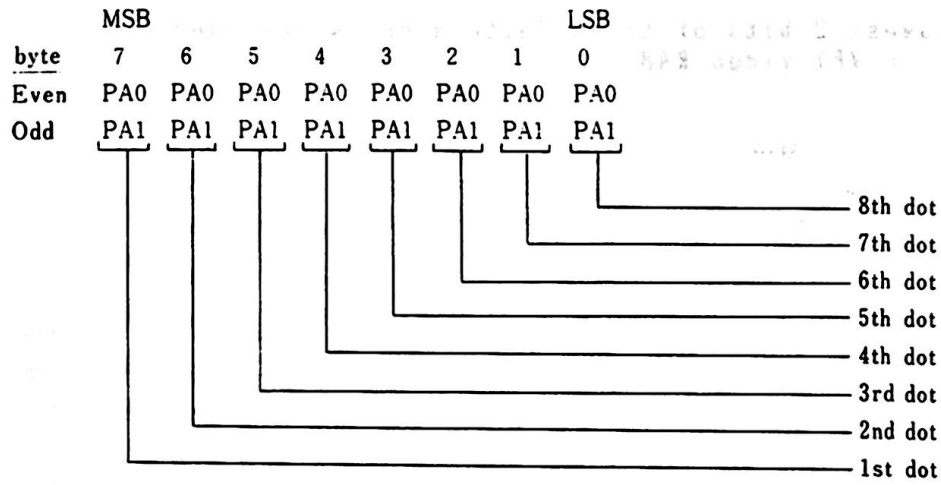
- 320 dot horizontal and 200 dot vertical display
- 16 colors are available for display
- 32 KB memory per screen are necessary, as the color is specified for each dot
- 1 byte is required to specify 2 dots
- 64 KB expanded memory is required for VP1.
- This corresponds to Screen Mode 4 in Basic.



Graphics Mode 5

Graphics mode 5 can display only on high-resolution displays and has the following characteristics:

- 640 dot horizontal and 200 dot vertical display
- 4 out of 16 colors can be displayed
- 32 KB memory per screen are necessary, as the color is specified for each dot
- 64 KB expanded memory is required for VP1
- 2 bytes are required to specify 8 dots
- This corresponds to Screen Mode 5 in Basic.

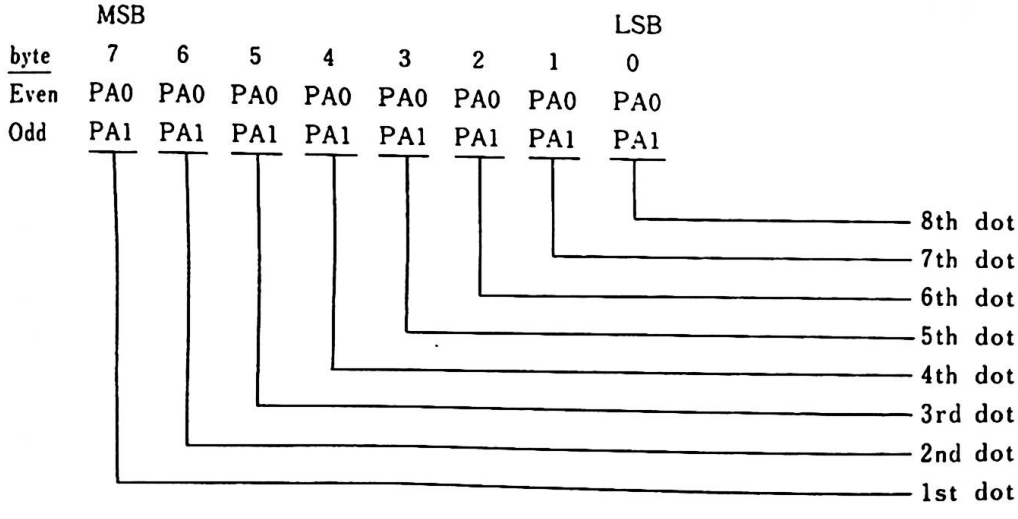


Graphics Mode 6

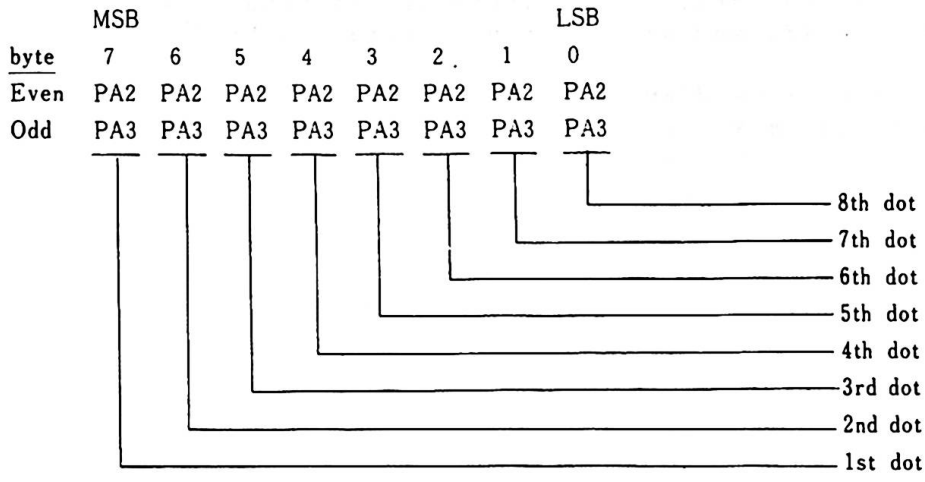
Graphics mode 6 can display only on a high-resolution display. This mode is achieved by combining two screens. One is screen mode 5 of VP1 and the other is screen mode 5 of VP2. This mode has the following characteristics:

- 640 dot horizontal and 200 dot vertical display
- 16 colors can be displayed
- 64 KB memory per screen are necessary, as the color is specified for each dot
- 64 KB expanded memory is necessary
- 4 bytes are required to specify 8 dots
- This corresponds to Screen Mode 6 in Basic.

The lowest 2 bits of the palette address are used to set the value in VP1 video RAM.



The highest 2 bits of the palette address are used to set the value in VP2 video RAM.



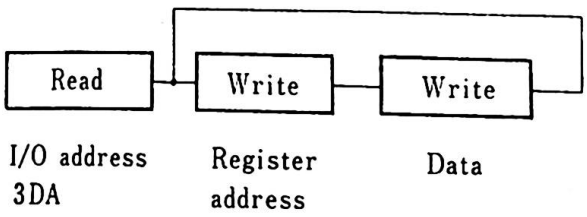
In graphics mode 6, methods of specifying the palette address differs from those of graphics modes 1 through 5.

Mode 1 – 5 address				Mode 6 address				Hex. Value
PA3	PA2	PA1	PA0	PA3	PA2	PA1	PA0	
0	0	0	0	1	0	1	0	0/A
0	0	0	1	1	0	1	1	1/B
0	0	1	0	1	0	0	0	2/8
0	0	1	1	1	0	0	1	3/9
0	1	0	0	1	1	1	0	4/E
0	1	0	1	1	1	1	1	5/F
0	1	1	0	1	1	0	0	6/C
0	1	1	1	1	1	0	1	7/D
1	0	0	0	0	0	1	0	8/2
1	0	0	1	0	0	1	1	9/3
1	0	1	0	0	0	0	0	A/0
1	0	1	1	0	0	0	1	B/1
1	1	0	0	0	1	1	0	C/6
1	1	0	1	0	1	1	1	D/7
1	1	1	0	0	1	0	0	E/4
1	1	1	1	0	1	0	1	F/5

3.4.4. Video Gate Array

The video gate array controls the video functions and the I/O address is 3DA (Hex). The gate array for VP1 and VP2 has 8 kinc of internal registers. There are internal registers for each o VP1 and VP2 and also ones used commonly by VP1 and VP2.

To control read/write to a register, a flip-flop latch is used. Each time a write is performed to an I/O register, address/data flip-flop latch will be reversed. When a read is performed, this flip-flop changes to "address status".



The following are descriptions of the registers:

Internal Register Address

The addresses for VP1 and VP2 are the same. When a register for VP1 is accessed, however, the "P" bit of memory block GA2A shoul be "1". Similarly, when a register for VP2 is accessed the "P" bit of memory block GA2B should be "1". (Reference 2.2.6 Memory Space and I/O Address Setting)

Address(Hex)	Register Name
00	Mode control 1 register (separate)
01	Palette mask register (separate)
02	Border color register (common)
03	Mode control 2 register (separate)
04	Reset register (common)
05	Transparent palette register (for VP1 only)
06	Superimpose control register (common)
10~1F	Palette register (common)

Remarks) separate --- separately used by VP1 and VP2
common ----- commonly used by VP1 and VP2

Mode Control 1 Register

This is a write-only 8 bit register and the address in the video gate array is 00 (Hex).

Bit	VP1 function	VP2 function
0	High to low bandwidth	High to low bandwidth
1	Graphics/Text	Graphics/Text
2	—	—
3	Video enable	Video enable
4	16 color graphics	16 color graphics
5	—	—
6	—	VRAM 2 Kanji/AN mode
7	—	Super 16 color

Bit 0 : The high video bandwidth display requires it to be "1". The following display modes require the high video bandwidth.
An optional 64 KB expanded RAM card is required for VP1.

- 80 char. x 25 lines text mode
- 640 x 200 dot 4 color graphics
- 320 x 200 dot 16 color graphis

Bit 1 : "1" for all graphics modes, "0" for text mode.

Bit 2 : Always "0".

Bit 3 : "1" makes the video signal available. "0" makes it not available and the screen becomes border color.

Bit 4 : "1" for 160 x 200 dot 16 color graphics and 320 x 200 dot 16 color graphics.

Bit 6 : For VP2. "1" makes it possible to display Kanji characters using VRAM 2. When it is "0" and bit 1 is "0", 80 chars. x 25 lines text display (ANK) is possible using VP2.

Bit 7 : For VP2. "1" makes it possible to display 640 x 200 dot 16 color graphics (screen mode 6). In this case, as both VRAM 1 and VRAM 2 are used, superimpose is not possible.

Palette Mask Register

This is a write-only 8 bit register and the address in the vide gate array is 01 (Hex). "0" makes it possible to mask the palette.

Bit	VP1 function	VP2 function
0	Mask bit 0	Mask bit 0
1	Mask bit 1	Mask bit 1
2	Mask bit 2	Mask bit 2
3	Mask bit 3	Mask bit 3
4	—	VRAM 1
5	—	VRAM 2
6	—	Extended video
7	—	Video bandwidth control

Bit 0 - 3 : "0" masks palette register address bit.

Bit 4 : "1" makes VP1 access VRAM 1.

Bit 5 : "1" makes VP2 access VRAM 2.

Bit 6 : "1" makes VP3 access VRAM 3.

Bit 7 : This changes the video bandwidth. "0" sets 14 MH while "1" sets 20 MHz.

Border Color Register

This is a write-only 4 bit register whose address in the video gate array is 02 (Hex). The four bits correspond to R, G, B and I and the border color is fixed by their combination.

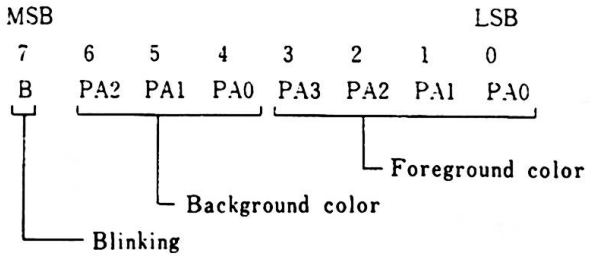
VP1, VP2	
Bit	Color
0	B (Blue)
1	G (Green)
2	R (Red)
3	I (Intensity)

Mode Control 2 Register

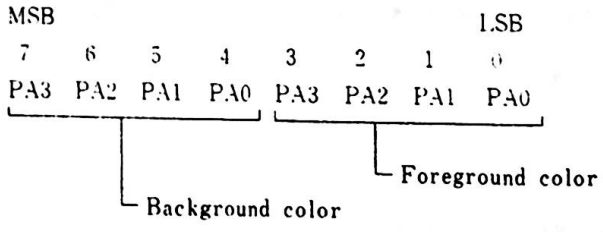
This is a write-only register whose address in the video gate array is 03 (Hex).

Bit	VP1 function	VP 2 function
0	always "0"	always "0"
1	Blinking	Blinking ("0" in Kanji)
2	always "0"	always "0"
3	2 color graphics	2 color graphics
4	—	English mode

Bit 1 : When "1", the attribute byte in text display has the following meaning:



When "0", the attribute byte in text display has the following meaning:



In graphics mode, when the control 2 register bit 1 is "1", the palette (PA3) high bit address is replaced by character the blink rate and for this, 2 colors will be displayed alternately. If the palette of the higher half and the lower half are the same, the color remains unchanged. If they are different, two colors will be displayed alternately in accordance with the blink rate. To cause this, only 8 colors are possible and bit 3 of the palette mask register becomes inactive.

Bit 3 : "1" when in 640 x 200 2 color graphics mode.

Bit 4 : "1" when in English mode. The memory space for the 128 KB expanded RAM card is placed after that of the 64 KB memory on the system board and the 64 KB expanded RAM. "0" when in Native and Extended Video mode. In this case, memory space for the 128 KB expanded RAM is always placed after memory address 20000 (Hex).

Reset Register

This is a write-only 2 bit register whose address in the gate array is 04 (Hex).

Bit	Function
0	Asynchronous Reset
1	Synchronous Reset

Bit 0 : "1" issues an asynchronous reset in the video gate array and stops all memory cycles. All output signals are tri-stated and memory is cleared. This reset should be issued only once after power-on and then the synchronous reset should be used.

Bit 1 : "1" issues a synchronous reset in the video gate array and stops all memory cycles. All output signals stop. When a synchronous reset is made after a memory refresh at the time of a display mode change, the contents of the gate array mode control register and the CRT control register will be changed. After the synchronous reset is released, a memory refresh is done.
In this way, the memory contents will not be changed even at the time of display mode change.

Transparent Palette Register

This is a register for VP1 only and specifies which palette register will become transparent. The address in the gate array is 05 (Hex).

Bit	Function
0	Register Select 0
1	Register Select 1
2	Register Select 2
3	Register Select 3

Superimpose Control Register

The address in the gate array is 06 (Hex). It controls superimposing of VRAM 1 and VRAM 2.

Bit	Function
0	Priority Control
1	Transparent Control
2	Mode Control 1
3	Mode Control 2

Bit 3, 2 = 11 --- OR
10 --- AND
01 --- XOR
00 --- has the following meanings depending on bits 0 and 1.

Bit 1, 0	Foreground	Background	Transparent (Y or N)
0 0	VRAM 1	VRAM 2	NO
0 1	VRAM 2	VRAM 1	NO
1 0	VRAM 1	VRAM 2	YES
1 1	VRAM 2	VRAM 1	Background color is displayed in the foreground

Palette Register

This is a write-only 4 bit register. The combination of the 4 bits selects a color. There are 16 registers and the addresses in the video gate array are 10 - 1F (Hex).
Each address has a corresponding color code.

Color Code (Hex)	Palette Register Address (Hex)
0	10
1	11
2	12
3	13
4	14
5	15
6	16
7	17
8	18
9	19
A	1A
B	1B
C	1C
D	1D
E	1E
F	1F

Figure 3-5 Palette Register and Color Code

Palette Bit				Color
I	R	G	B	
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Light Blue
0	1	0	0	Red
0	1	0	1	Purple
0	1	1	0	Yellow
0	1	1	1	White
1	0	0	0	Bright Black
1	0	0	1	Bright Blue
1	0	1	0	Bright Green
1	0	1	1	Bright Light Blue
1	1	0	0	Bright Red
1	1	0	1	Bright Purple
1	1	1	0	Bright Yellow
1	1	1	1	Bright White

Figure 3-6 Palette Register and the color displayed

When the palette is loaded, the video is in a "display disable" status and the color on the screen is set depending on the contents of the register which the processor specifies.
When the program finishes loading the palette, the video changes again to a "display enable" status.

3.4.5. Superimpose

VP1 and VP2 operate on the same clock at 14.31818 MHz. The two video signals can be imposed using the mixer. The video mixer includes a 4 bit x 16 word palette. The mixer sets the priority of VP1 and VP2 and allows screens to be imposed using such logic operations as AND, OR, or XOR.

		VP2								
	Page 8~B / Page 0~7	Kanji 20× 11	ANK 40× 25	160× 200 16C	320× 200 4C	640× 200 2C	ANK 80× 25	320× 200 16C	640× 200 4C	Kanji 40× 11
VP1	AN 40×25	×	×	×	×	×	×	×	×	×
	160×200 16C	○	○	○	○	○	×	×	×	×
	320×200 4C	○	○	○	○	○	×	×	×	×
	640×200 2C	○	○	○	○	○	×	×	×	×
	AN 80×25	×	×	×	×	×	×	×	×	×
	320×200 16C	×	×	×	×	×	○	○	○	○
	640×200 4C	×	×	×	×	×	○	○	○	○

Remarks) ○ ---- Imposition Possible
× ---- Impossible
AN --- Alphanumeric, Special Character
ANK -- Alphanumeric, Special Character, Katakana
C ---- Color

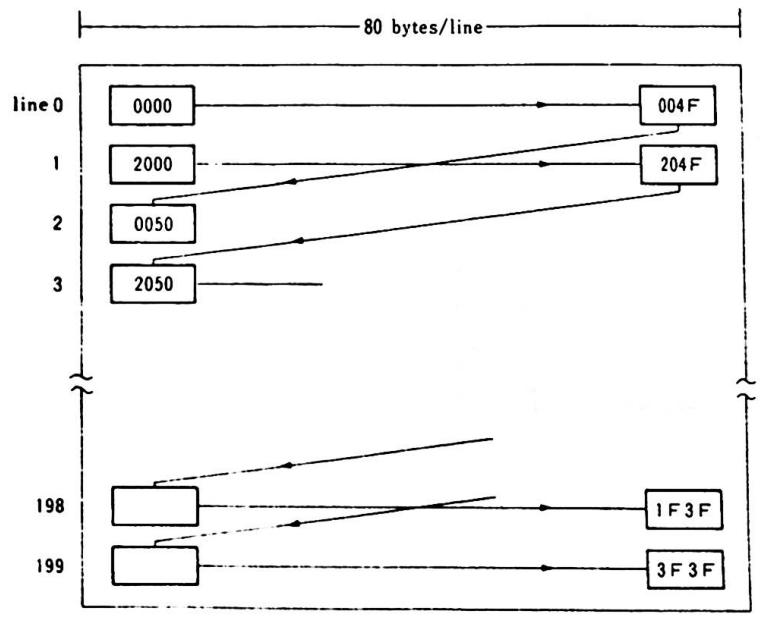
Figure 3-7 Combinations for Superimpose

3.4.6. Video RAM and Display Screen Relationships

The size of video RAM necessary to display graphics depends on what is being displayed.

The following displays require two banks of memory, each of which is 8000 bytes:

- Graphics mode 1 (160 x 200 dot 16 colors) : 2 dots/byte
- Graphics mode 2 (320 x 200 dot 4 colors) : 4 dots/byte
- Graphics mode 3 (640 x 200 dot 2 colors) : 8 dots/byte

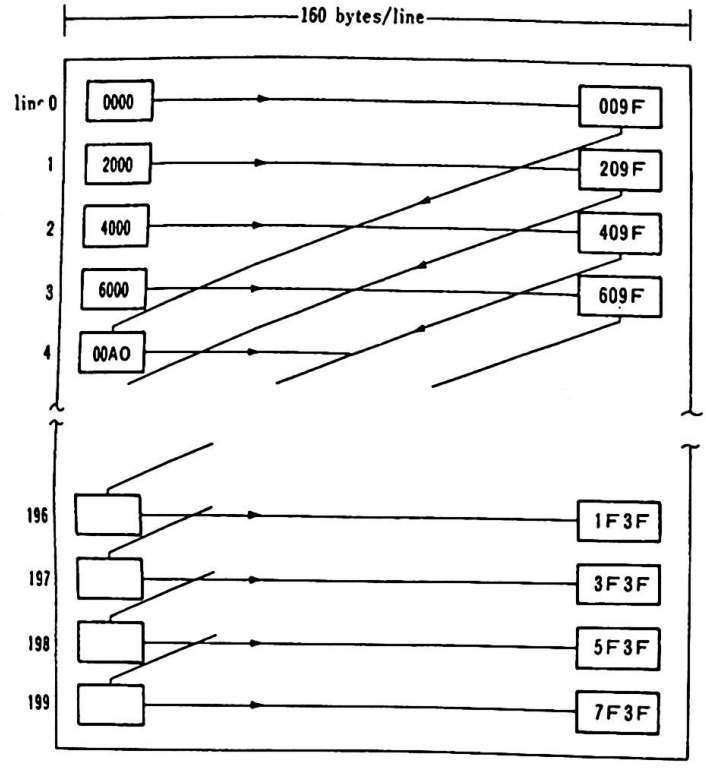


Remark) The value in the rectangle indicates the relative address (Hex) of the video RAM.

Figure 3-8 Video RAM and the screen (1 of 2)

The following displays require four banks of memory, each of which is 8000 bytes.

- Graphics mode 4 (320 x 200 dot 16 colors)
- Graphics mode 5 (640 x 200 dot 4 colors)



Remark) The value in the rectangle indicates the relative address (Hex) of the video RAM.

Figure 3-8 Video RAM and the screen (2 of 2)

3.5. VP3 Display Function

VP3 is a video processor for Extension Video mode and is included in an optional Extension Video card.

3.5.1. VP3 Text Display

VP3 supports display of the following types of text :

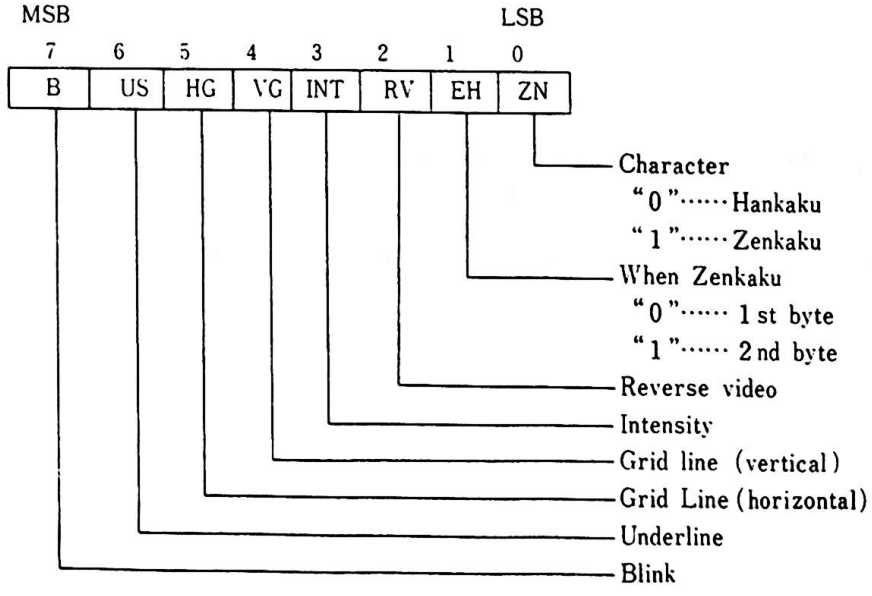
- Kanji, Hiragana
  Zenkaku(Full-size) 16 x 16 dot, 40 characters x 25 lines

- Katakana, Alphanumerics, Special Characters
  Zenkaku(Full-size) 16 x 16 dot, 40 characters x 25 lines
  Hankaku (Half-size) 8 x 16 dot, 80 characters x 25 lines

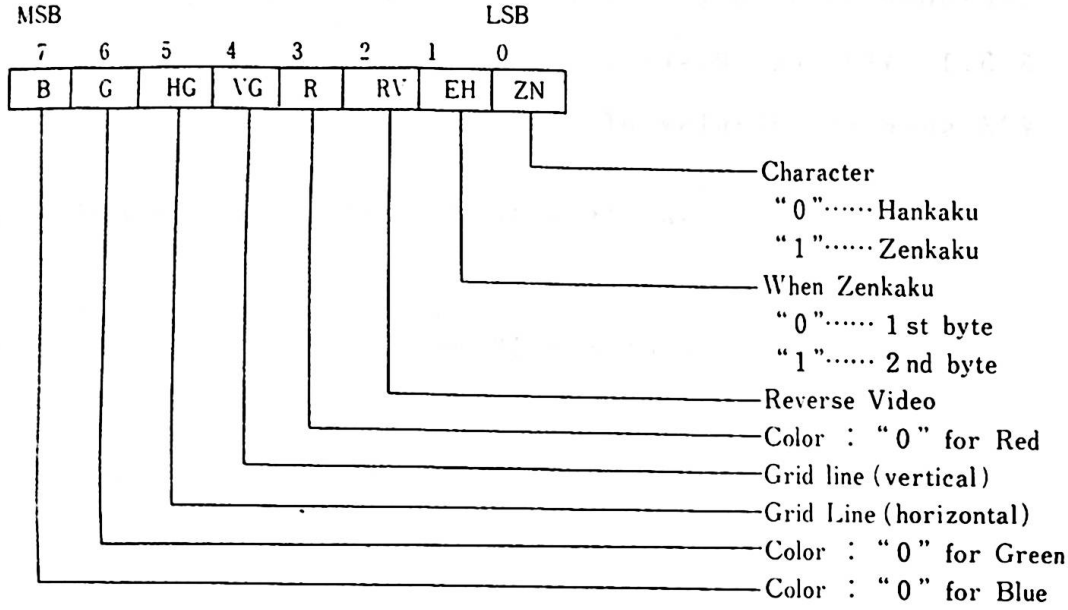
- Vertical and Horizontal Grid lines

Attributes differ between monochrome and color displays.

Monochrome display attributes:



Color Display Attributes:



Remarks) 1. The combination of bits 3, 6 and 7 makes it possible to display 8 colors.

2. Display is not possible when all of bits 2, 3, 6 and 7 are "1".

3.5.2. VP3 Graphics Display

In VP3 graphics display, the address space of A0000 - AFFFF (Hex) is allocated as the video RAM virtual addresses.

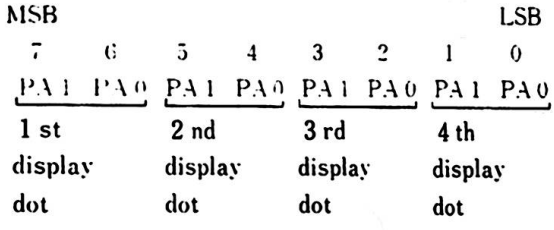
VP3 supports the following two graphics modes:

- Mode 1 : 360 x 512 dot 4 colors
- Mode 2 : 720 x 512 dot 2 colors

Mode 1

Mode 1 graphics display has the following characteristics:

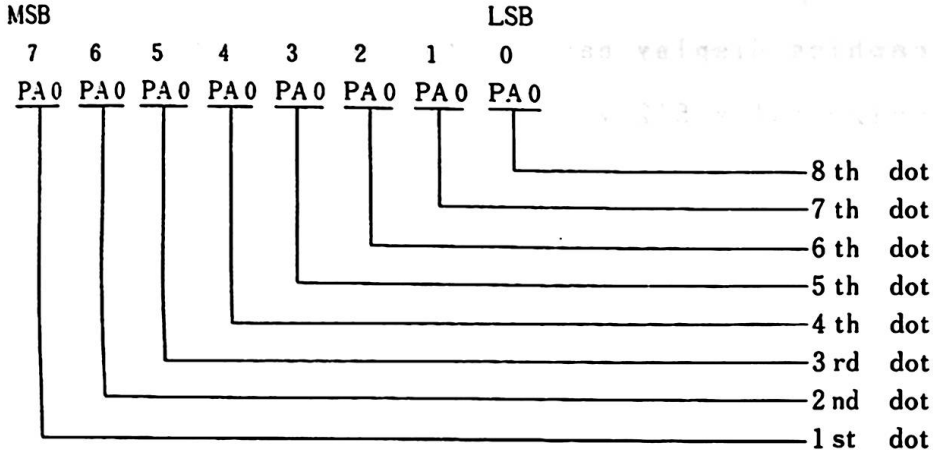
- 360 horizontal x 512 vertical dot display screen
- 4 out of 16 colors can be displayed at one time
- 46,080 bytes of memory are required, as the color is specified for each dot.
- 1 byte specifies 4 dots.



Mode 2

Mode 2 graphics display has the following characteristics:

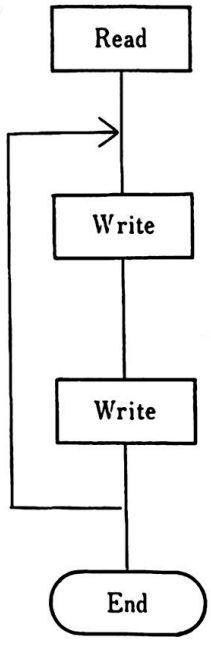
- 720 horizontal x 512 vertical dot display screen
- 2 out of 16 colors can be diplayed at one time
- 46,080 bytes of memory are required, as the color is specified for each dot.
- 1 byte specifies 8 dots.



3.5.3. Gate Array

The VP3 gate array is contained in an optional Extension Video Card and the I/O address is 3DD (Hex). The gate array has eight internal registers whose internal addresses are 00 - 07 (Hex). The internal registers perform various controls on VP3.

Write to internal register



Step 1 : Read (Dummy) at I/O address 3DD (Hex)

Step 2 : Write internal register address at I/O address 3DD (Hex).

Step 3 : Write the internal register data which was specified in the above step 2 at I/O address 3DD (Hex).

Register 0

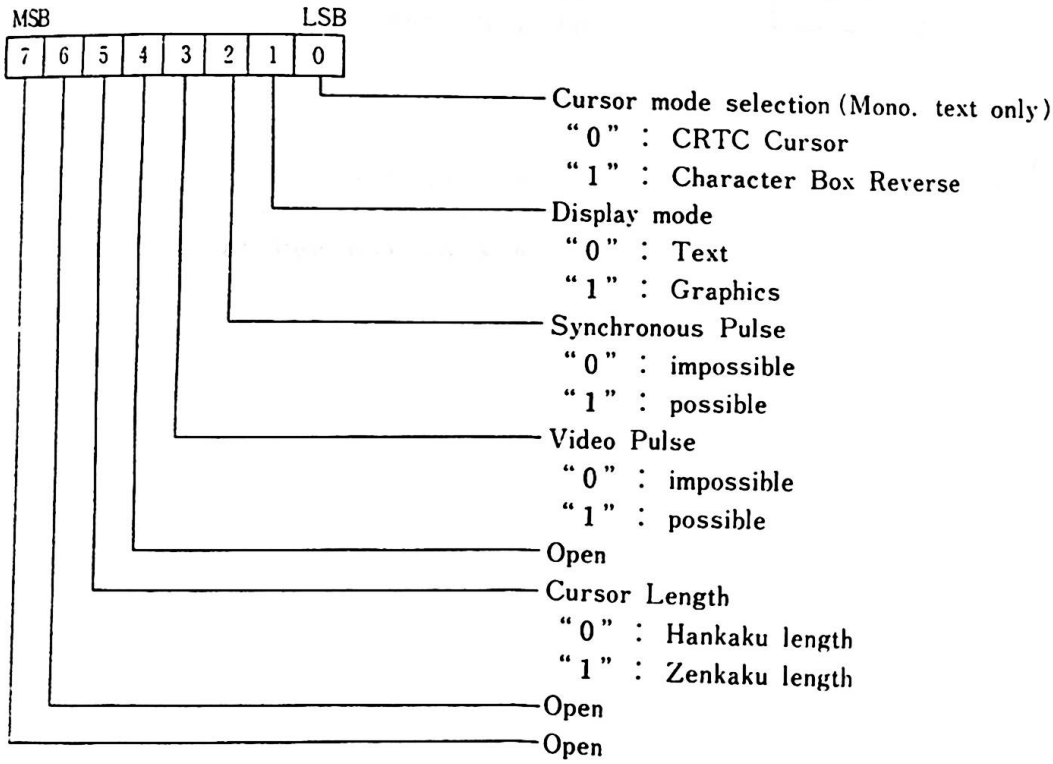
The internal address is 00 (Hex). When any data are written in this register, a write to the internal register is made possible.

Register 1

The internal address is 01 (Hex). When any data are written in this register, a write to the internal register is made impossible. Registers other than the palette register are cleared.

Register 2

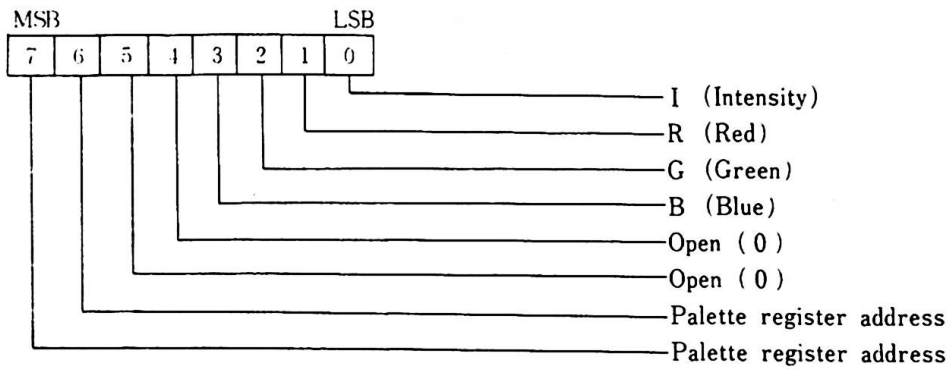
The internal address is 02 (Hex).



In color mode ("1" at bit 7 of Register 5), only "0" at bit 0 is valid.

Register 3

This register contains the information for setting the color in the palette register. The internal address is 03 (Hex).



Bit 7, 6:

00	Palette register 0
01	Palette register 1
10	Palette register 2
11	Palette register 3

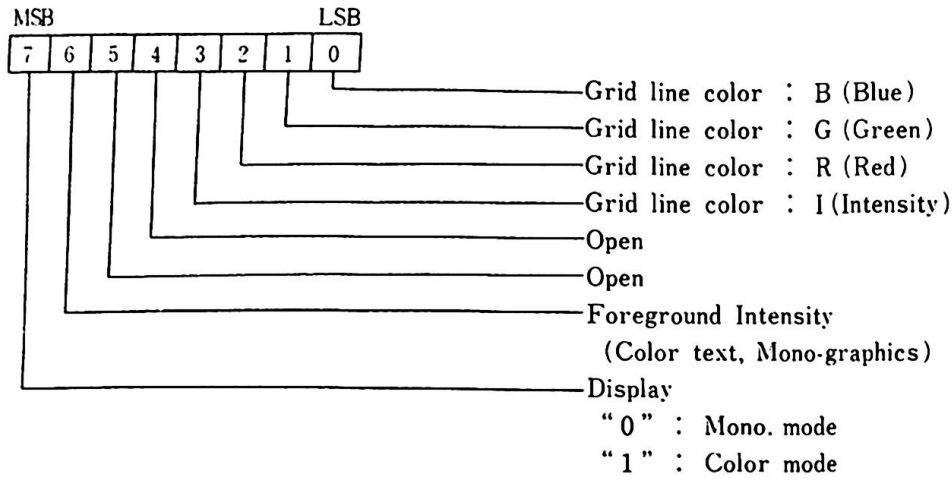
The palette registers 0 through 3 can be used in 360 x 512 dot graphics display. The palette registers 0 and 1 can be used in 720 x 512 dot graphics display.

Register 4

The internal address is 04 (Hex). A write is performed against this register to make the clock available (any data will do).

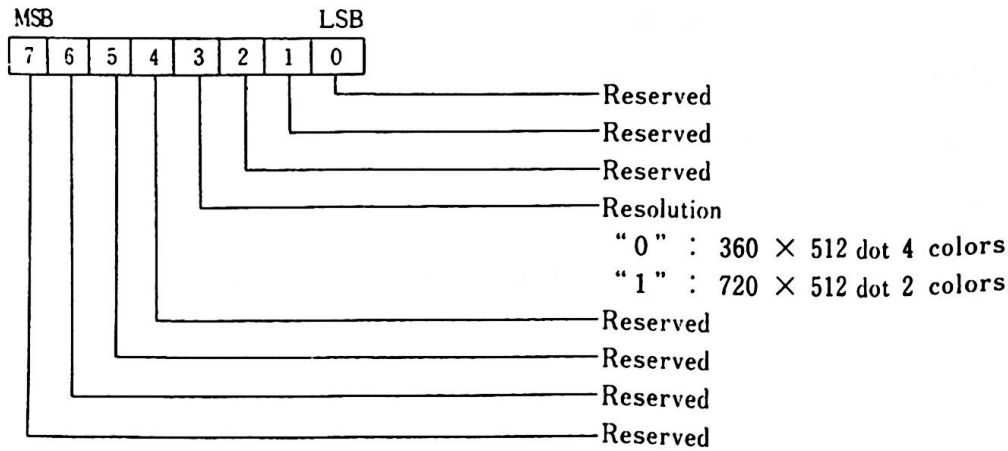
Register 5

The internal address is 05 (Hex).



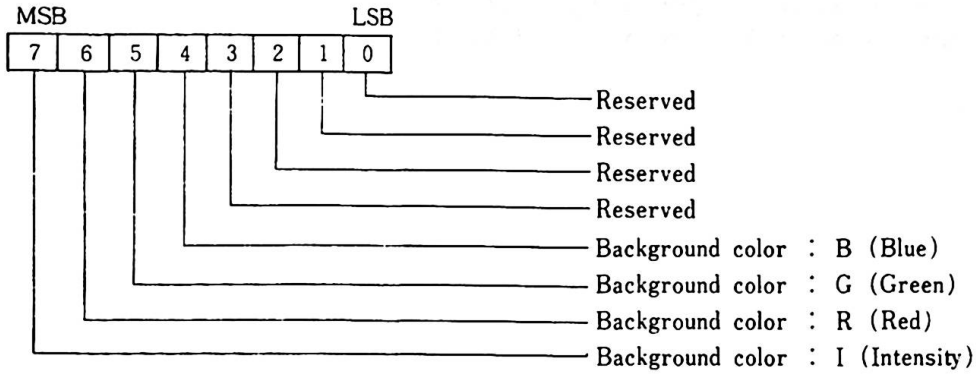
Register 6

The internal address is 06 (Hex).



Register 7

The internal address is 07 (Hex).



Display Mode Selection

The VP3 display mode is set by the combination of the bits as follows:

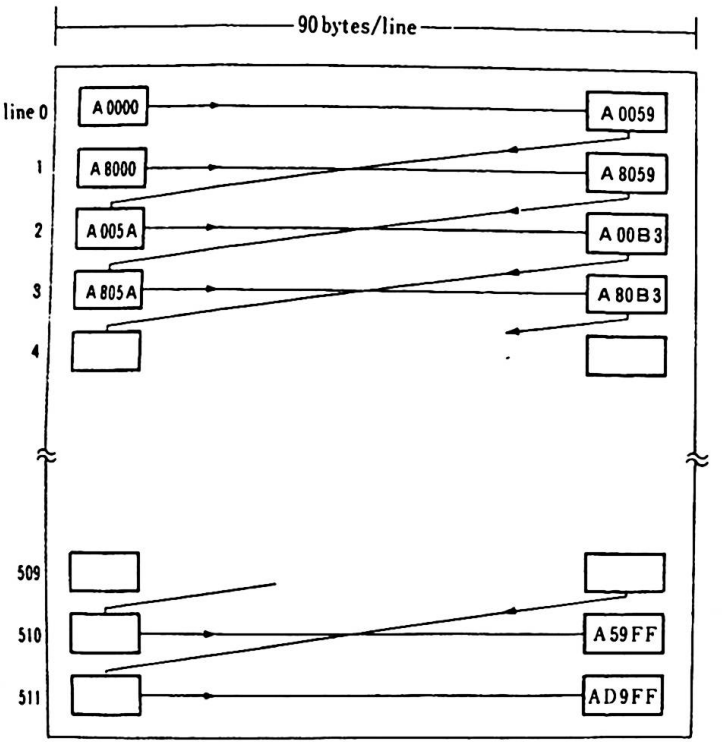
1. Register 2, bit 1 --- X
2. Register 5, bit 7 --- Y
3. Register 6, bit 3 --- Z

Display mode	X	Y	Z
Mono. Text	0	0	—
Color Text	0	1	—
Mono. Graphics	1	0	—
Color Graphics (mode 1)	1	1	0
Color Graphics (mode 2)	1	1	1

3.5.4. Video RAM and Display Screen Relationships

Two banks of memory each of 23,040 bytes are used to display the following:

- Graphics mode 1 (360 x 512 dot, 4 colors)
- Graphics mode 2 (720 x 512 dot, 2 colors)



Remark) The value in the rectangle indicates the absolute address (Hex) of the video RAM.

Figure 3-9 Video RAM and the screen (VP3)

3.6. CRT Controller

The HD46505 is used as the CRT Controller (CRTC). It has 19 accessible internal registers. One of these registers, the Index register, is actually used as a pointer to the other 18 registers. It is a write-only register and the I/O address is 3D4 (Hex). In order to write data in any of the 18 registers, the Index register is first loaded with the necessary pointer. Then, the Data register is loaded with the information to be placed in the selected register. The data register is loaded from the processor by executing an OUT instruction to I/O address 3D5 (Hex).

The following are the relationships between the modes and the registers:

Register		VP 1 & VP 2						VP 3	
		Text				Graphics		Text	Graphics
No.	Address	Kanji		AN		Low Band	High Band		
		20×11	40×11	40×25	80×25				
R0	0	38	71	38	71	38	71	65	38
R1	1	28	50	28	50	28	50	50	2D
R2	2	2F	5C	2F	5C	2E	5B	58	31
R3	3	06	DC	06	0C	06	0C	AC	A6
R4	4	0D	0D	1F	1F	7F	3F	1A	47
R5	5	0A	0A	06	06	06	06	02	00
R6	6	0B	0B	19	19	64	32	19	40
R7	7	0C	0C	1C	1C	70	38	19	41
R8	8	02	02	02	02	02	02	03	03
R9	9	11	11	07	07	01	03	13	06
R10	A	10	10	06	06	26	26	13	27
R11	B	11	11	07	07	07	07	14	07
R12	C	00	00	00	00	00	00	00	00
R13	D	00	00	00	00	00	00	00	00
R14	E	00	00	00	00	00	00	00	00
R15	F	00	00	00	00	00	00	00	00
R16	10	Lightpen data (High Address) : Read-only							
R17	11	Lightpen data (Low Address) : Read-only							

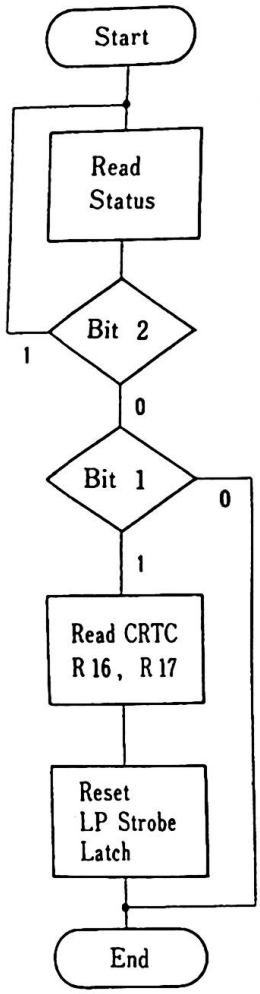
Remark) All register values are given in Hexadecimal.

Figure 3-10 Display and CRTC Register Value

3.7. Light Pen Interface

The light pen interface is designed for RGBI (Red, Green, Blue, Intensity). Due to timing differences depending on different display types, the row/column value returned from the CRT may vary. These differences must be compensated for through software.

The following show the light pen data read sequences:



Read I/O address 3DA (Hex)
Bit 1 : LP strobe latch
Bit 2 : -LP SW

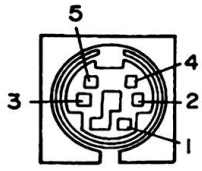
0 : Light Pen switch is on.
1 : Light Pen switch is not pressed.

0 : Light Pen is not lighted.
1 : Light Pen is lighted and address data are in R16 and R17 of CRTC.

Write data in I/O address 3DB (Hex).
(Any data will do.)

Hardware Interface

A light pen is connected to a five pin connector on the back of the system unit.



(System Side)

Pin No.	Signal	I/O
1	+12 V	O
2	−LPEN INPUT	I
3	+5 V	O
4	−LPEN SW	I
5	GND	−

Figure 3 −11 Light Pen Interface Connector (J9)

3.8. Video Subsystem I/O Addresses

In Native Mode, I/O addresses used by the Video Subsystem are initialized with the following but they can be changed by the software:

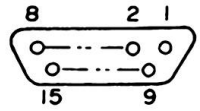
Address (Hex.)	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Register
3DA	1	1	1	1	0	1	1	0	1	0	VP1/2 Gate Array
3DB	1	1	1	1	0	1	1	0	1	1	Light Pen Latch Clear
3DC	1	1	1	1	0	1	1	1	0	0	Light Pen Latch Set
3D0,3D2 3D4,3D6	1	1	1	1	0	1	0	x	x	0	CRTC Index Register
3D1,3D3 3D5,3D7	1	1	1	1	0	1	0	x	x	1	CRTC Data Register
3DF	1	1	1	1	0	1	1	1	1	1	Page Register
3DD	1	1	1	1	0	1	1	1	0	1	VP3 Gate Array

X = N/A

Figure 3—12 Video I/O Addresses

3.9. Hardware Interface

This interface provides power to an optional TV Adapter. The signal level is standard TTL but the audio output operates with a 1V Peak-to-Peak signal biased at 0V which can drive a 10 K Ohm or greater input impedance.



(System Side)

Pin No.	Signal	I/O
1	GND	-
2	R	O
3	−HSYNC	O
4	−VSYNC	O
5	H −L SW	O
6	I	O
7	B	O
8	GND	-
9	GND	-
10	G	O
11	COLOR CLOCK	O
12	−12 V	O
13	DOT CLOCK	O
14	AUDIO	O
15	GND	-

Figure 3 −13 Display Interface (J15)

4. System Options

The IBM 5510 system provides connectors on the system board for optional feature installation. In this Chapter, optional features and their connection to the system are described. I/O devices can be connected to the Expansion Channel connectors.

4.1. 64KB RAM Card

To display 80 characters x 25 lines in text mode, or 640 x 200 dots in 4 colors or 320 x 200 dots in 16 colors in graphics mode, the system RAM size should be expanded to 128KB through installtion of this option. This card consists of one board and connects to a 50 pin connector on the system board.
(Only one 64KB card can be installed.)
When this card is inserted, the addressing method is changed. This memory option uses the ODD memory space, while the system memory is decoded as the EVEN memory. When an additional 128KB of RAM is installed, the addresses will again be changed.
(Reference : 5.5 Memory Map)
Memory refresh is performed on the system board logic.

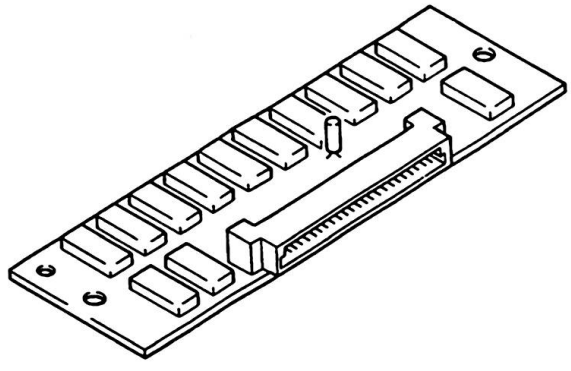


Figure 4—1 64KB RAM Card

The following are the connector specifications for the JX 64KB RAM Card.

Signal	Pin (A)	Pin (B)	Signal
D0	A1	B1	D1
D2	2	2	D3
D4	3	3	D5
D6	4	4	D7
+5V	5	5	+5V
GND	6	6	GND
A0	7	7	Open
V1A0	8	8	V1A1
V1A2	9	9	V1A3
V1A4	10	10	V1A5
V1A6	11	11	V1A7
Open	12	12	Open
–RAS1	13	13	CAS1
–WE1	14	14	SPL1
SAT1	15	15	AGPD
VIDEO MEMR	16	16	GATE
–DIS CAS0	17	17	–DIS ED
–LCG	18	18	Open
GAC0	19	19	GAC1
GAC2	20	20	GAC3
GAC4	21	21	GAC5
GAC6	22	22	GAC7
–64K CD IN	23	23	Open
Open	24	24	Open
Open	25	25	Open

Figure 4-2 64KB RAM Card Connector Specifications (J6)

I/O refers to the view from the card side.

Signal	I/O	Description
-RAS	I	Row Address Strobe. This timing pulse sets the row address for the RAM modules.
A0	I	Address line A0. When this bit is "1", the 64KB RAM card is selected.
-DISABLE EDATA	I	When the expansion RAM card is installed and the microprocessor is reading an odd byte of data, the expansion card tri-states the latch for an even byte of data on the system board using this line.
SAT1	I	This signal indicates that the expansion RAM card should latch up data from the expansion RAM into the attribute latch.
GAC0 - GAC7	O	These data lines contain VP1 data from the attribute latch.
D0 - D7	I/O	Data lines D0 - D7.
V1A0 - V1A7	I	These are multiplexed address lines and contain the row, column, and CRT addresses.
VIDEO MEMR	I	This signal when high indicates that the video RAM is being accessed.
-AGDP	I	This line when low indicates that a CPU RAM cycle is occurring.
-DIS CAS0	O	This line is used to disable the system board CAS0 when a system microprocessor write is occurring in the expansion RAM.
CAS1	I	Column Address Strobe 1
-LCG	O	This line is used to instruct the system board that attributes or graphics data should be read from the expansion RAM card.
GATE	I	This line becomes the -LCG output.

Signal	I/O	Description
-WE1	I	This line instructs the memory that the a microprocessor write cycle is occurring.
SPL1	I	This line instructs the expansion RAM card to latch the data from the expansion RAM into the microprocessor latch.
-64KB CDIN	O	When a 64KB expansion RAM card is inserted, this line is low.

4.2. 128KB RAM Card

The 128KB RAM card expands the system's memory size by 128KB increments up to 512KB maximum. The 128KB RAM card plugs into the I/O Expansion Connector on the system board or Expansion unit. This memory is 150ns dynamic RAM. Addres decode and refresh logic are included on this card.
One card can be installed in the base system. When the Expansion unit is installed, a total of three cards can be installed. When installing the 128KB RAM cards, an address decoder jumper should be connected.

128 KB RAM Card	Jumper	Address Range	Address (English Mode)
1	1	00000~1FFFF	20000~3FFFF
2	2	20000~3FFFF	40000~5FFFF
3	3	40000~5FFFF	60000~7FFFF

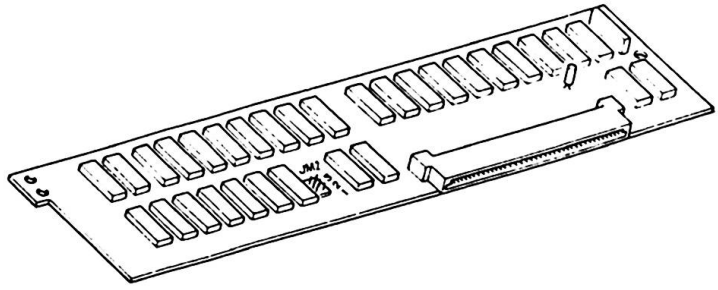


Figure 4-3 128KB RAM Card and Jumper

When the 128KB RAM card is installed with the jumper, base memory addresses are placed after the 128KB RAM addresses at the initial self-diagnostic power-on test or system reset. The base memory addresses, however, will not change in English mode.

(Reference: 2.2.7 Expansion Channel, 5.5 Memory Map)

4.3. Extension Video Card

The Extension Video Card consists of a gate array for Video Processor 3 (VP3), a 20MHz oscillator and a 32KB Expansion Video RAM. Enhanced video function (Extension Video mode) becomes possible when the Extension Video Card, Extension Video Mode Cartridge and the high-resolution display (12" monochrome or 14" color) are installed.

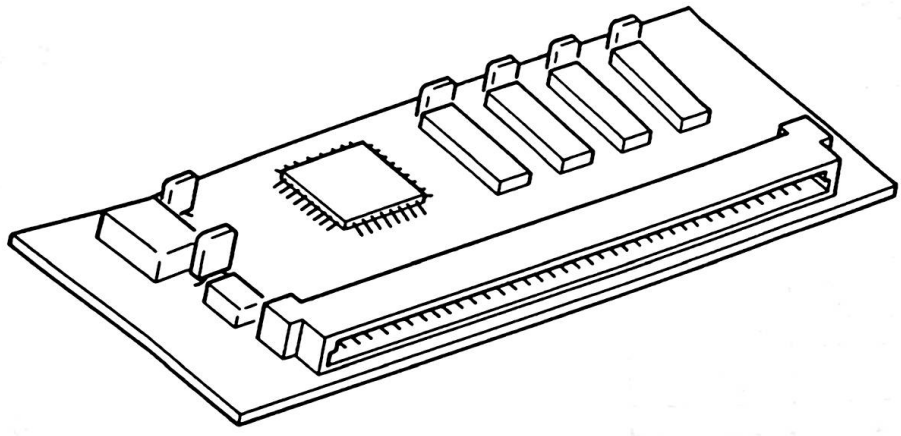


Figure 4—4 Extension Video Card

The following are the connector pin assignments for the Extension Video Card:

Signal	Pin (A)	Pin (B)	Signal
V2A0	A1	B1	V2A1
V2A2	2	2	V2A3
V2A4	3	3	V2A5
V2A6	4	4	V2A7
-RAS2	5	5	-CAS2
-W10E	6	6	-W10O
-GP10	7	7	-32K CD IN
VD0	8	8	VD1
VD2	9	9	VD3
VD4	10	10	VD5
VD6	11	11	VD7
VD8	12	12	VD9
VD10	13	13	VD11
VD12	14	14	VD13
VD14	15	15	VD15
+5V	16	16	+5V
Open	17	17	Open
GND	18	18	GND
-SX03	19	19	NCCLK3
DSPT0	20	20	CUSR
VSYC	21	21	HSYC
RA4	22	22	RA1
RA3	23	23	RA0
RA2	24	24	RPT3
20M	25	25	RQR3
DSPT3	26	26	RST3
GRM3	27	27	DST3
Open	28	28	OSC3
Open	29	29	Open
Open	30	30	Open
GD0	31	31	GD1
GD2	32	32	GD3
GD4	33	33	GD5
GD6	34	34	GD7
GATEE	35	35	Open
Open	36	36	-RESET
-IOW	37	37	-IOR
D0	38	38	D1
D2	39	39	D3
D4	40	40	D5
D6	41	41	D7
DUSW	42	42	Open
-B	43	43	-I
-R	44	44	-G
EHS	45	45	EVS

Figure 4-5 Connector for Extension Video Card (J5)

I/O refers to the view from the card side.

Signal Name	I/O	Description
V2A0 - V2A7	I	These lines are the multiplexed address of the expansion video RAM.
VD0 - VD15	I	Data lines D0 - D15
-RAS 2	I	Row Address Strobe. This line instructs the Extension Video Card to latch up the address on the first MPX'd address.
-CAS 2	I	Column Address Strobe. This line instructs the expansion VRAM to latch up the address on the second MPX'd address.
-W10E	I	This line instructs the expansion VRAM to write the even addresses of the expansion VRAM.
-W10O	I	This line instructs the expansion VRAM to write the odd addresses of the expansion VRAM.
-GP10	I	This line instructs the expansion VRAM to gate out the expansion VRAM on a read option.
-32K CD IN	O	When an extension video card is inserted, this signal is low.
VD0 - VD15	I/O	Video RAM I/O Data Bus line.
-SX03	I	When this signal is low, the VP3 gate array is selected.
NCCLK3	O	Inverted signal for"GATEE" (A35 pin)
DSPTO	I	Display Timing signal from CRTC. When high, the contents of VRAM read are displayed
CUSR	I	Video signal to display the cursor.
VSYNC	I	Vertical synchronous signal from CRTC
HSYNC	I	Horizontal synchronous signal from CRTC

Signal Name	I/O	Description
RA0 - RA4	I	These lines are raster addresses from CRTC and are available in text mode. RA0 - RA4 are decoded and then the raster to be displayed in a character box is determined. Interlace mode is used.
PRT 3	O	This line is a timing signal to determine which of the CRTC, CPU or Refresh accesses VRAM2.
20M	O	20MHz clock output
RQR 3	O	This line is used to reset requests for use to VRAM1 and 2.
DSPT 3	O	This line is used to correct the access timing from CRTC to VRAM. This signal is derived by delaying "DSPT0" (A20 Pin) by one character.
RST 3	O	RAS Time
GRM 3	O	When VP3 is in graphics mode, this signal is high.
DST 3	O	This line is used as a timing pulse to keep the timing of memory timing data with the character generator timing.
CSC 3	O	This line is used as a chip select signal of a character generator.
GD0 - GD7	I	These lines are image data signals for screen display.
-RESET	I	System Reset Signal
-IOW	I	I/O Write signal
-IOR	I	I/O Read signal
D0 - D7	I/O	Data bus D0 - D7

Signal Name	I/O	Description
DUSW	I	When this signal is high, VP3 operation is allowed. When low, the signal lines EHS, EVS, -R, -G, -B, -I float.
-B	O	Blue signal to CRT. When consecutive vertical lines are displayed, the frequency becomes maximum. (10MHz: Applied the same as the following -I, -R, -G)
-I	O	Intensity signal to CRT.
-R	O	Red signal to CRT.
-G	O	Green signal to CRT.
EHS	O	When the DUSW signal is high and bit 2 of VP3 gate array register 2 is "1" (synchronous), the HSYNC signal is output.
EVS	O	When the DUSW signal is high and bit 2 of VP3 gate array register 2 is "1" (synchronous), the VSYNC signal is output.

4.4. Diskette Drive Adapter

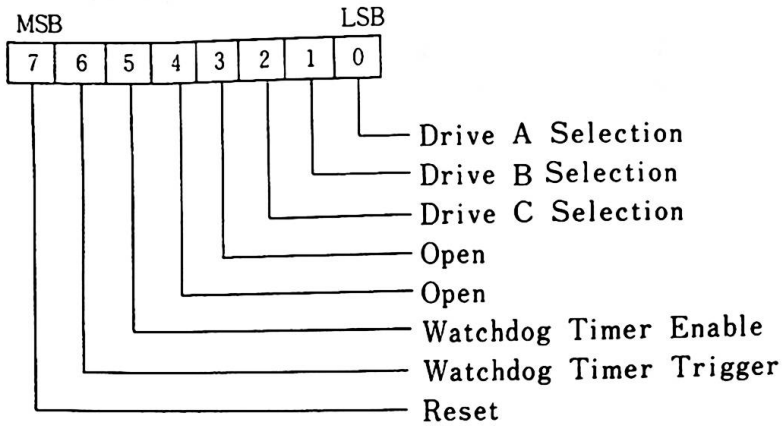
The diskette drive adapter consists of a single sub-board (card) and is connected to the system unit connector (30 pin) by a signal cable. It is attached to the three diskette drives through two kinds of signal cables. Power is supplied to each diskette drive via the diskette drive adapter connector.

The adapter uses the JPD765 or equivalent and is designed for Modified Frequency Modulation (MFM).

The following are descriptions of its components:

DOR (Digital Output Register)

The digital output register (DOR) is a write-only 8 bit register used to control the three diskette drive motors and selection of the diskette drives. The I/O address is F2 (Hex).



Bit 0 (Drive A Selection)
"1" activates drive A signal path and turns on the motor.
"0" deactivates it and turns off the motor.

Bit 1 (Drive B Selection)
"1" activates drive B signal path and turns on the motor.
"0" deactivates it and turns off the motor.

Bit 2 "1" activates drive C signal path and turns on the motor.
"0" deactivates it and turns off the motor.

Bit 5 (WATCHDOG TIMER CONTROL)
"1" activates functions of WATCHDOG TIMER CYCLE and allows interrupts.
"0" deactivates functions of WATCHDOG TIMER CYCLE and rejects interrupts.

Bit 6 (WATCHDOG TIMER TRIGGER)
By alternating 1 and 0 being written, the timer cycle gets started.

Bit 7 (reset)
"1" resets diskette controller.
"0" releases the reset status of the diskette controller.

Watchdog Timer

The Watchdog Timer (WDT) is a one-to-three second timer connected for output to IRQ 6 (Interrupt Level 6) of the 8259A PIC. The Watchdog Timer is used for the system unit to watch the stopped status of the adapter.

Diskette Controller

There are two kinds of registers:

Register	I/O Address
Status Register	F 4
Data Register	F 5

Status Register

The 8-bit status register which can be read contains the status information of the Diskette Controller.

Data Register

Data transmission between the diskette and the CPU are performed through this 8-bit data register. Commands and parameters which are transmitted between the CPU and the diskette controller are temporarily stored in this register.

Programming Considerations

1. The diskette controller is initialized with the following parameters after system power up. Parameters are stored in the addresses which the interrupt vector 1E (Hex) indicates.

Contents	Parameter (Hex)	Remarks
Sector Size	02	512 Bytes/Sector
Sector Count	09	Sectors/Track (9 sector)
Head Unload	1	Always '1' (32ms)
Head Step Rate	E	4ms/step
Head Load Time	01	Minimum Head Load Time (4ms)
Format Gap	50	
Write Gap	2A	
Non-DMA Mode	01	
Fill byte for Format	19	

2. BIOS uses the following commands for the diskette drive adapter.

   - SPECIFY
   - SEEK
   - RECALIBRATE
   - SENSE INTERRUPT STATUS
   - SENSE DRIVE STATUS
   - READ DATA
   - WRITE DATA
   - FORMAT TRACK

3. Head Load

   When a diskette is correctly inserted in a diskette drive and the drive is selected, the head is automatically loaded. Programming for head load is, therefore, not necessary. Read/Write commands cannot be accepted until after 0.5 seconds of head load.

4. Interrupt

   The system should not allow other interrupts than interrupt level 6 (IRQ6 : used by diskette drive adapter). This is due to the strict time limitations during diskette read/write.

Figure 4-6

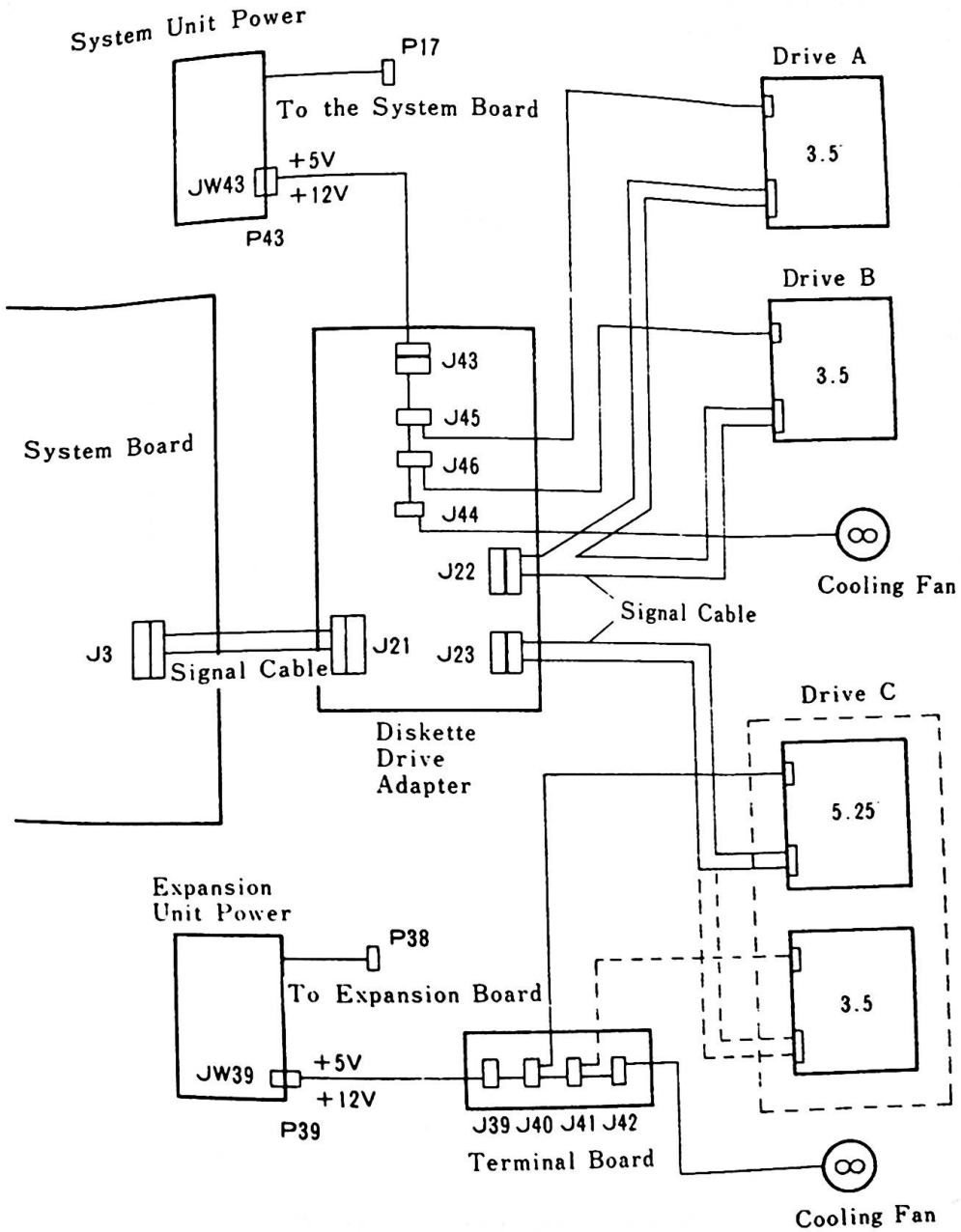
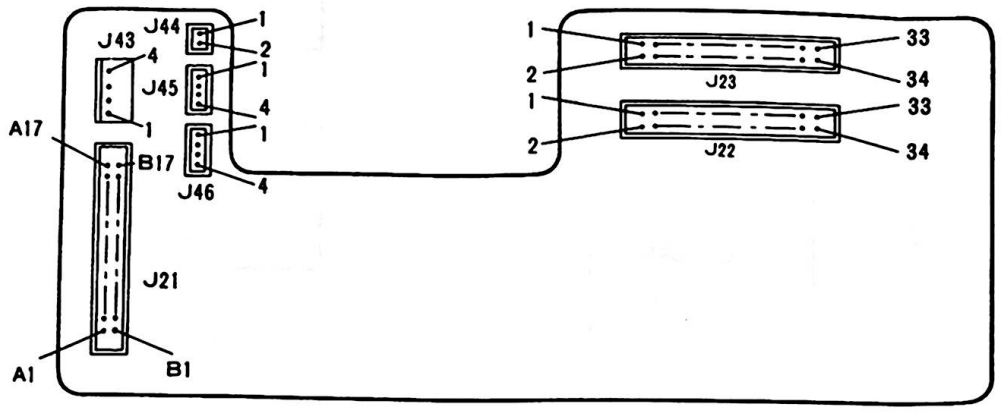


Figure 4—6 Diskette Drive Connections

Interface Connector

The adapter is provided with the signal cable connector and power-supply connectors.



J21 : Signal Cable Connector
J22 : Signal Cable Connector (Drive A, B)
J23 : Signal Cable Connector (Drive C)
J43 : Power-supply Connector (From the Power Unit)
J44 : Power-supply Connector (To the Cooling Fan)
J45 : Power-supply Connector (To the Drive A)
J46 : Power-supply Connector (To the Drive B)

Power for the diskette drive C is supplied from the power unit in the Expansion Unit.

Figure 4-7 The connector on the diskette drive adapter

System Interface

The adapter is connected using the dedicated cable to the connector (J3) on the system board. Following are the pin assignments and their descriptions:

Signal	Pin (A)	Pin (B)	Signal
D0	A1	B1	D1
D2	2	2	D3
D4	3	3	D5
D6	4	4	D7
A0	5	5	A1
A2	6	6	A9
Open	7	7	Open
-IOR	8	8	-IOW
-RESET	9	9	GND
DSKT INTR	10	10	GND
-FDC CS	11	11	GND
GND	12	12	GND
+5V	13	13	GND
+5V	14	14	GND
+5V	15	15	GND
-FDC CD IN	16	16	GND

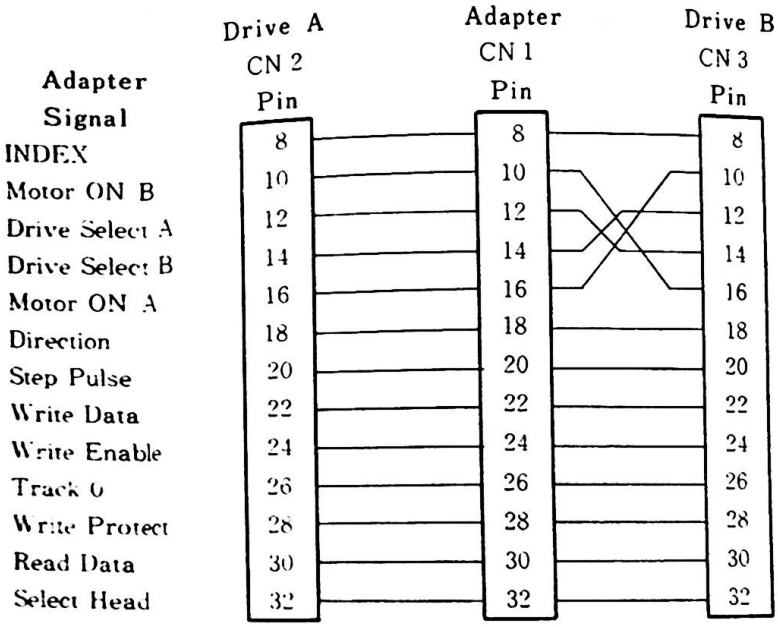
Figure 4-8 Connector for Diskette Drive Adapter (J3)

I/O refers to the view from the card side.

Signal Name	I/O	Description
D7 - D0	I/O	These eight lines form a bus through which all commands, status and data are transferred.
A0 - A3	I	These lines are address lines used in register selection.
-IOW	I	Data are written in the register by the trailing edge of this signal.
-IOR	I	This line is used to read the contents of the register.
-RESET	I	This line is used to reset the controller and to clear DOR.
DSKT INTR	O	This line becomes "high" by time-out of WATCHDOG TIMER.
-FDC CS	I	This line remains "low" while the CPU performs read/write to a diskette drive adapter
A9	I	Address line A9.

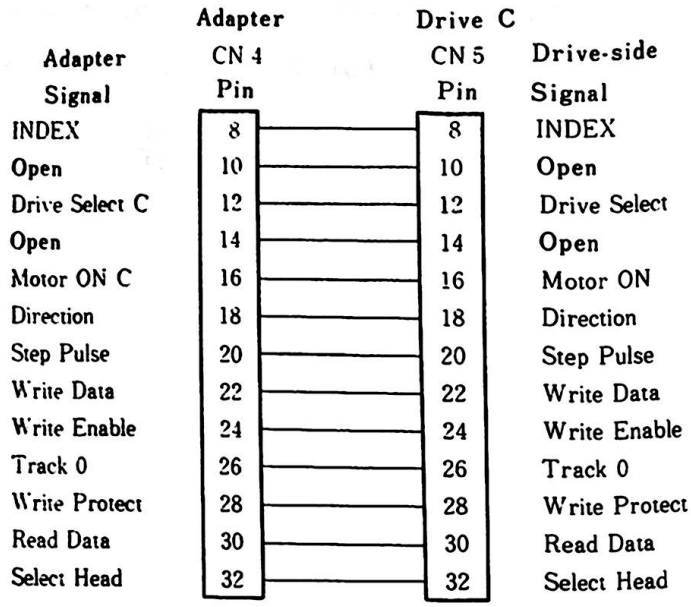
Drive Interface

Two connectors are provided on the card for connection to the diskette drive. One is for the two diskette drives in the system unit (Fig. 4 - 9, 1 of 2) and the other is for the diskette drive in an Expansion unit (Fig. 4 - 9, 2 of 2). Signals are at standard TTL levels.



Remarks) 1. Odd number pins from 5 through 33 are GND, and pins 1 through 4 are open.
2. CN2 is connected to drive A, while CN3 is connected to drive B.

Figure 4-9 Diskette Drive Signal Cable (1 of 2)



Remarks) 1. Odd number pins from 5 through 33 are GND, and pins 1 through 4 are open.
2. CN5 is connected to drive C.

Figure 4-9 Diskette Drive Signal Cable (2 of 2)

I/O refers to the view from the adapter side.

Signal	I/O	Description
-DRIVE SELECT	O	This line is used to select the diskette drive.
-MOTOR ON	O	This line is used to turn on the drive motor.
-STEP PULSE	O	This line is used to move the head by 1 cylinder.
-DIRECTION	O	This line is used to set the moving direction of the head. When this line is low, the head moves inward.
-WRITE DATA	O	When WRITE ENABLE (pin 24) is low, the change of the signal is written on the diskette.
-SELECT HEAD	O	When "high", diskette side 1 is selected : When "low", diskette side 2 is selected.
-INDEX	I	Each time the diskette goes round, one pulse (index pulse) is issued.
-WRITE PROTECT	O	This line is used to indicate that the diskette is write-protected.
-TRACK 0	I	This line is used to indicate that the head is located at track 0.
-READ DATA	I	The selected drive must supply a pulse on this line each time when magnetic flux change is encountered on the diskette.
-WRITE ENABLE	O	When "low", a data write to the diskette is possible.

Power Supply Connector

The power supply connector provides each drive with direct current from the power unit. It also provides the cooling fan with +12V.

Pin No.	Signal
1	+ 5 V
2	GND
3	GND
4	+ 12 V

J43, J45, J46

Pin No.	Signal
1	GND
2	+ 12 V

J44

Figure 4—10 Power Supply Connector on the Diskette Adapter

4.5. Diskette Drive

By installing an optional diskette drive, the IBM 5510 can be expanded to include a maximum of three 3.5" diskette drives or two 3.5" and one 5.25" diskette drives. Both 3.5" and 5.25" diskettes have the same format and have the following characteristics:

- Double-sided double-density
- 80 tracks per side
- 9 sectors per track
- 512 bytes per sector
- Seek time 4ms per track
- Write-protect sensor
- Cooling fan
- Track 00 sensor
- Index sensor
- +5VDC, +12VDC used

4.6. TV Adapter

The TV adapter includes the RF Modulator and allows an ordinary TV set to be used as a display by getting the RGBI direct video signal through the encoder and modulating it to the frequencies of channel 1 or channel 2.
The adapter has two switches, one for changing the channel (channel 1 or 2) and the other for color mode (B/W or color). The RF modulator output is automatically directed to the TV set at the time of system power-on. When the system unit is turned off, the normal TV broadcasting signal is received. Encoder output becomes the external output as a composite video signal.

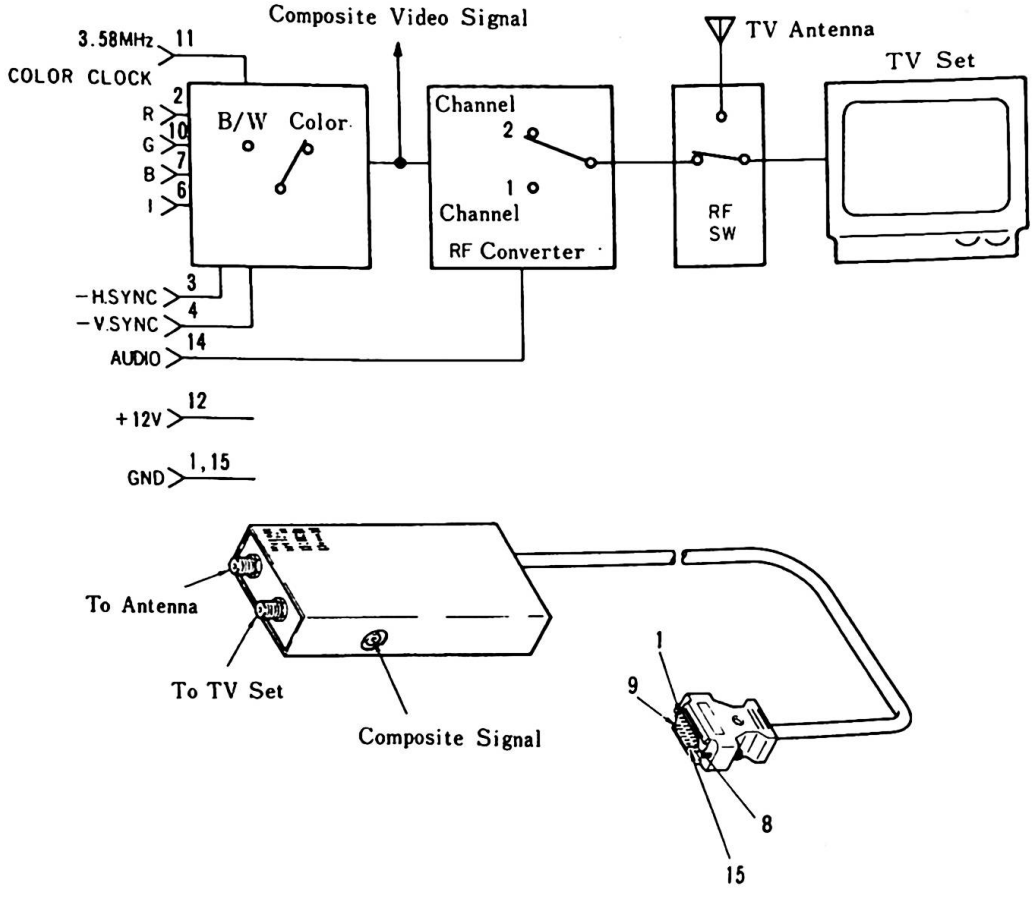


Figure 4—11 TV Adapter

4.7. Keyboard Cable

The IBM 5510 Cordless Keyboard can be attached to the JX using the optional Keyboard Cable. When the keyboard cable is connected, the signal level of -CBL CONNECTED becomes "low" (0 Volt) and the system unit's infrared (IR) receiver circuit is disabled, allowing keyboard input via the keyboard cable.

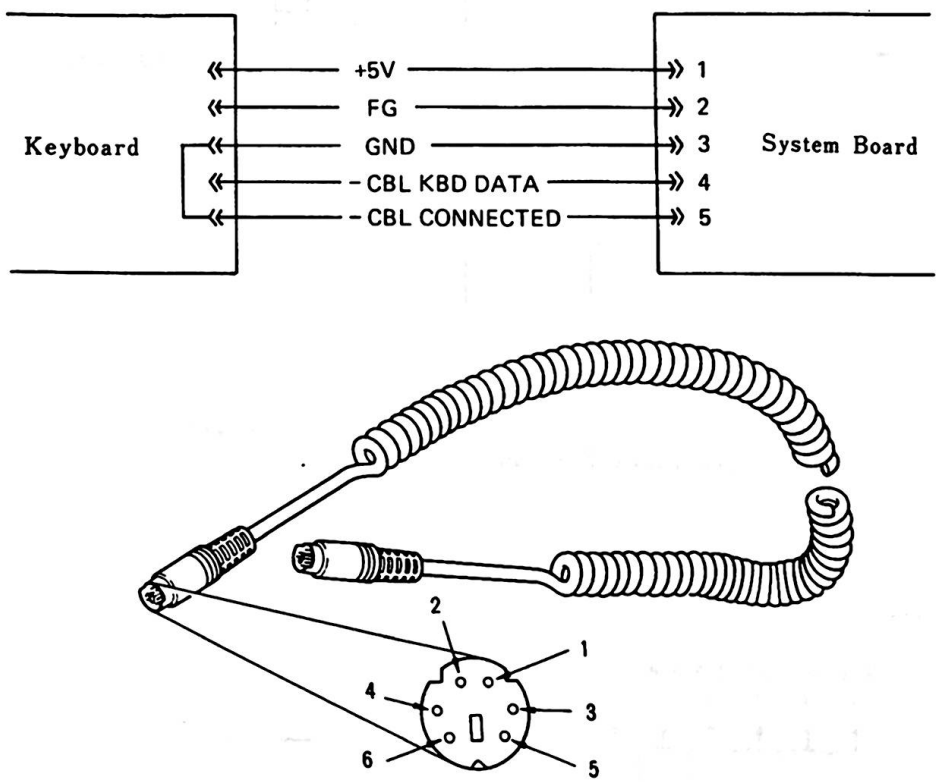


Figure 4—12 Keyboard Cable

. RS-232C Card

RS-232C card allows asynchronous communication under the ɔgraɯ control. It contains an INS8250A LSI chip.

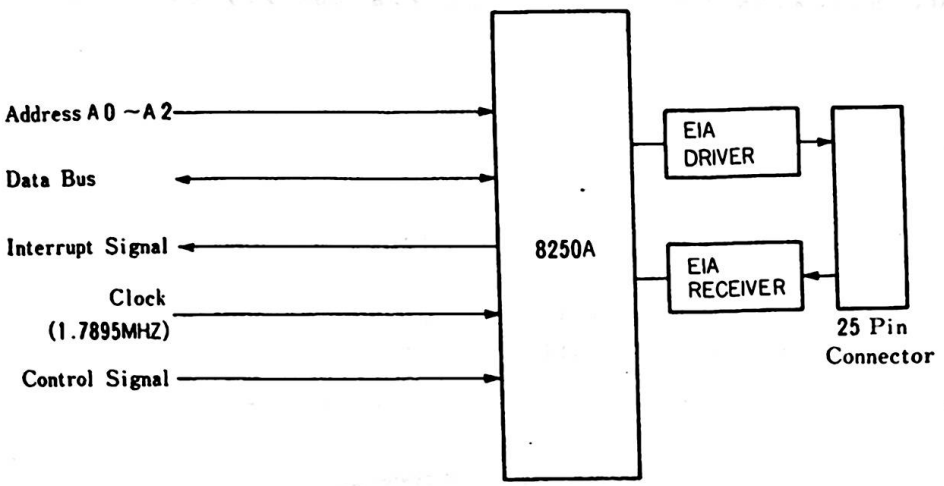


Figure 4—13 RS-232C Card Block Diagram

The following is a standard send/receive data format of asynchronous communications:

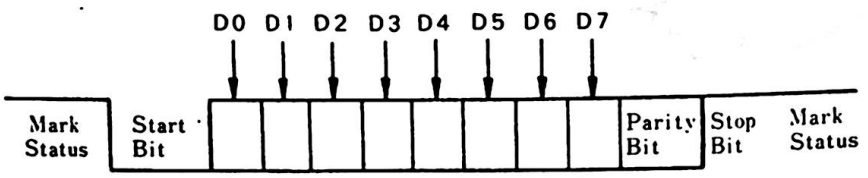


Figure 4—14 Asynchronous Communication Data Format

Following are the programming considerations and the necessary data for use with the IBM JX.

1. When a diskette read/write is performed, asynchronous communications can not be made. (Interrupts are prohibited.)

2. The speed of asynchronous communication is up to 4800 bps. A speed below 1200 bps is recommended when keyboard data are received.

3. Consecutive I/O operations to the 8250A should not be made in view of the necessary I/O time for the 8250A. An interval of more than 15 clocks is necessary.

4. The hardware interrupt level is "3".

5. 8250A pin 34 (OUT1) and pin 31 (OUT2) are not used.

6. I/O addresses are as follows:

I/O Address (Hex)	Register	DLAB Status
2F8	TX Buffer	DLAB=0 (Write)
2F8	RX Buffer	DLAB=0 (Read)
2F8	Divisor Latch(LSB)	DLAB=1
2F9	Divisor Latch(MSB)	DLAB=1
2F9	Interrupt Enable Register	DLAB=0
2FA	Interrupt Identification Registers	
2FB	Line Control Register	
2FC	Modem Control Register	
2FD	Line Satus Register	
2FE	Modem Status Register	
2FF	Scratch Register	

Figure 4—15 8250 I/O Addresses

7. The following Assembler language program initializes the 8250A. Operating conditions are:

   - 1200 bps
   - 8 bits
   - 1 stop bit
   - Odd parity

Sample program:

```
PROC    NEAR
MOV     AL,80H              ; SET DLAB=1
MOV     DX,2FBH             ; To Line Control Register
OUT     DX,AL
JMP     $+2                 ; I/O Delay
MOV     DX,2F8H             ; LSB of Divisor Latch
MOV     AL,5DH              ; LSB Value
OUT     DX,AL
JMP     $+2                 ; I/O Delay
MOV     DX,2F9H             ; MSB of Divisor Latch
MOV     AL,00H
OUT     DX,AL
JMP     $+2                 ; I/O Delay
MOV     DX,2FBH             ; Line Control Register
MOV     AL,0BH              ; 8 Bits/Word, 1 Stop Bit,
                            ; Odd Parity, DLAB = 0
OUT     DX,AL
JMP     $+2                 ; I/O Delay
MOV     DX,2F8H
IN      AL,DX
ENDP
```

8. RS-232C card connector specifications are as follows:

Signal	Pin (A)	Pin (B)	Signal
D0	A1	B1	D1
D2	2	2	D3
D4	3	3	D5
D6	4	4	D7
A0	5	5	A1
A2	6	6	A9
–SERIAL CD IN	7	7	Open
BAUD CLK	8	8	RESET
SERIAL INTR	9	9	–8250 CS
–IOR	10	10	–IOW
Open	11	11	Open
+12V	12	12	–12V
Open	13	13	Open
+5V	14	14	+5V
Open	15	15	Open
GND	16	16	GND

Figure 4—16 RS-232C Card Connector (J8)

Signal Name	I/O	Description
D0 - D7	I/O	Data Lines D0 - D7.
A0,1,2,9	I	Address Lines A0, A1, A2, and A9.
-SERIAL CD IN	O	When an RS-232C card is inserted, this signal becomes "low".
BAUD CLOCK	I	1.7895 MHz Clock input.
RESET	I	Signal for RS-232C card reset.
SERIAL INTR	O	Interrupt request signal.
-8250 CS	I	8250 LSI Chip Select Signal.
-IOR	I	I/O Read
-IOW	I	I/O Write

9. This card provides an EIA RS-232C electrically compatible interface for connection to external devices.

Pin No.	Signal Name	Signal Level
2	TRANSMIT DATA	Valid signal levels for all pins except 1 and 7 are: + : + 3 V ~ + 15 V − : − 3 V ~ − 15 V
3	RECEIVE DATA	
4	REQUEST TO SEND	
5	CLEAR TO SEND	
6	DATA SET READY	
7	GND	
8	CARRIER DETECT	
20	DATA TERMINAL READY	
1	FG	

Figure 4—17 RS-232C External Interface

10. Baud Rate Generator

The baud rate generator which resides within the 8250A generates the clock signal that is the basis for data transfer speed (baud rate) by dividing the 1.7895MHz clock input by the programmable divisor.

The output clock frequency of the baud rate generator should be set with the product of 16 multiplied by the baud rate.
The divisor of the baud rate generator is calculated as follows:

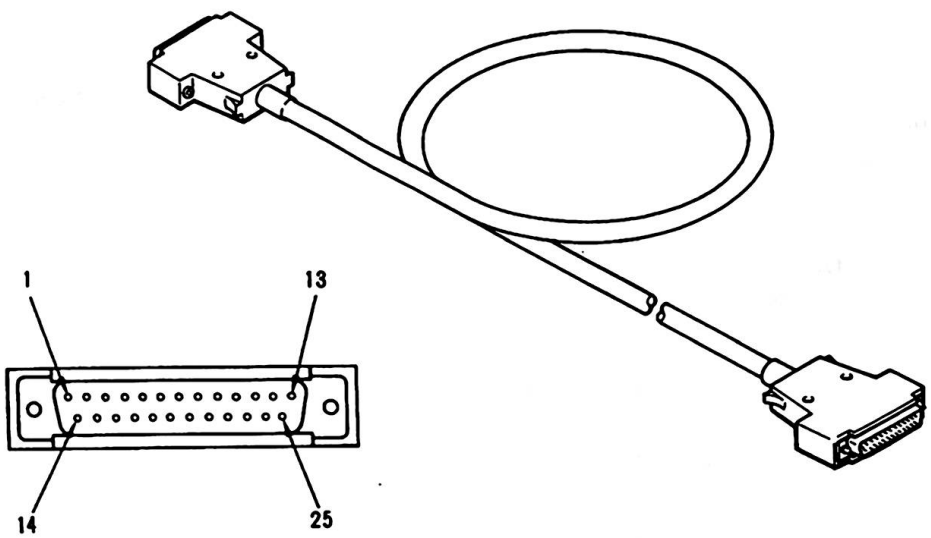
$$\text{Divisor} = (1.7895 \times 10^6) \div (\text{Baud Rate} \times 16)$$

Sample:

Required Baud Rate	Divisor Value (Hex)	Variance
50	8BD	0.006
75	5D3	0.017
110	1A1	0.023
134.5	167	0.054
150	12C	0.050
300	175	0.050
600	0BA	0.218
1200	05D	0.218
1800	03E	0.218
2000	03B	0.140
2400	02F	0.855
3600	01F	0.218
4800	017	1.291

4.9. RS-232C Cable

This IBM 5510 optional feature is provided with a 25-pin, "D" shaped connector. The cable is used to connect the RS-232C card with serial and asynchronous communication devices.



(Reference : 4.8 RS-232C Card)

Figure 4—18 RS-232C Cable

4.10. Display

There are three types of displays:

- 12" Color Display
- 12" Monochrome Display
- 14" Color Display

The 12" Monochrome or 14" Color display is required for operation in Extension Video Mode. These are dual-scan displays and they work in all operational modes. The 12" color display can be used in Native and English modes.

Their characteristics are as follows:

Contents	12″ Color	14″ Color	12″ Mono.
Video Frequency	14.318 MHz	20.000 MHz 14.318 MHz	20.000 MHz 14.318 MHz
Vertical Scan	59.92 Hz	76.68 Hz 59.92 Hz	76.68 Hz 59.92 Hz
Horizontal Scan	15.700 KHz	21.930 KHz 15.700 KHz	21.930 KHz 15.700 KHz
No. of Colors	16 Color	16 Color	Gray Scale 16
Dots (Graphics)	640 × 200	720 × 512 * 640 × 200	720 × 512 * 640 × 200
Dots (Text)	640 × 200	720 × 525 * 640 × 200	720 × 525 * 640 × 200
Characters (Hankaku)	80 × 11	80 × 25 80 × 11	80 × 25 80 × 11
Character Box (Hankaku)	8 × 18	9 × 21 8 × 18	9 × 21 8 × 18

Remark) * is for Extension Video Mode.

4.11. CMT Cable

The CMT cable connects the system unit with a cassette recorder. Connector pin assignments are as follows:

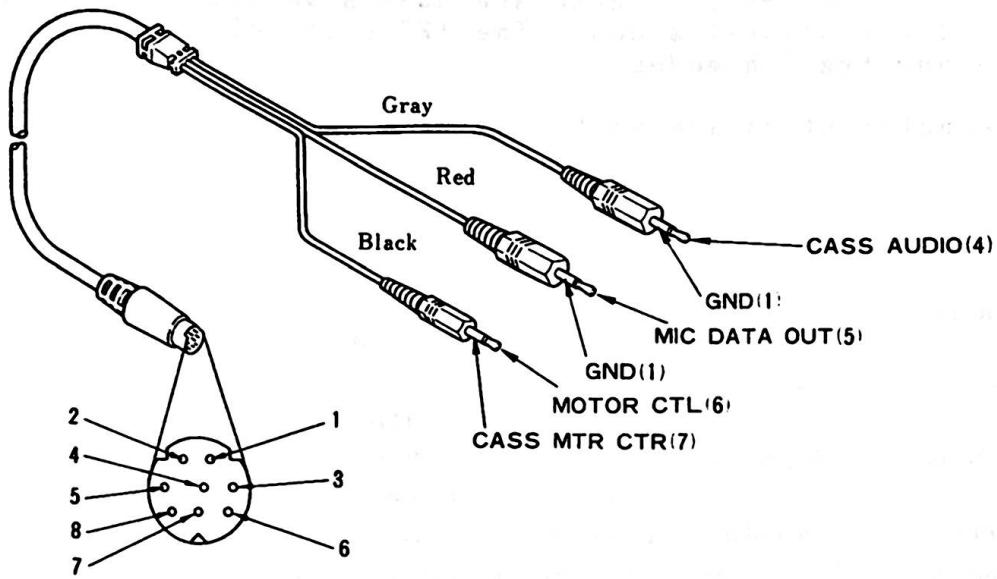


Figure 4—19 Cassette Cable Pin Assignments

4.12. Joystick

The joystick is an input device with the location control function indicated in X/Y coordinates. It consists of two switches and two potentiometers. By moving the operational stick vertically (Y coordinate) or horizontally (X coordinate), each potentiometer varies within a range from 0 to 100K Ohms. A maximum of two joysticks can be installed and they are connected to connector J13 (joystick 1) and J14 (joystick 2).

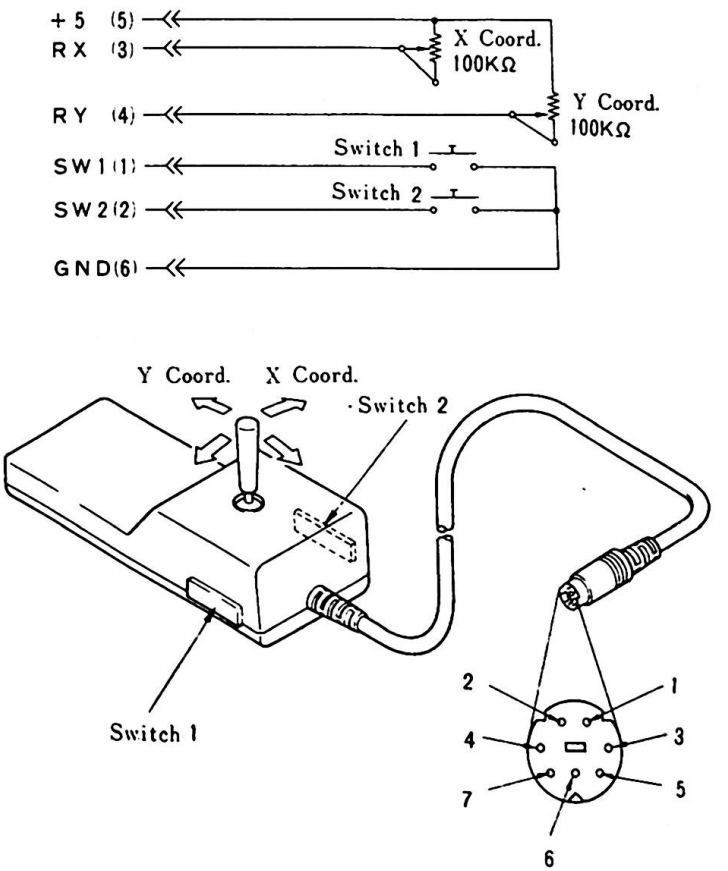
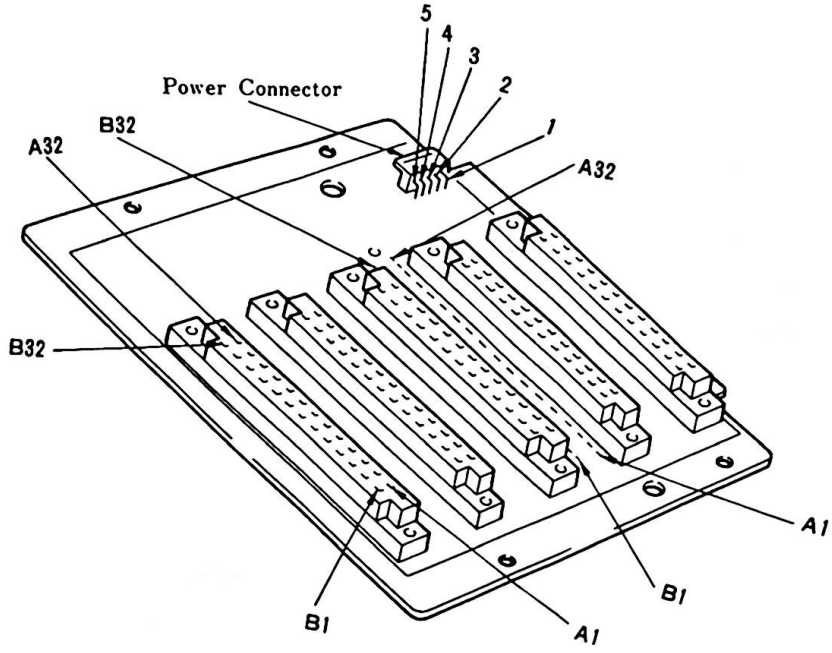


Figure 4—20 Joystick

4.13. Expansion Board

The Expansion Board is connected to the I/O channel via the expansion adapter. There are five 64-pin connectors on the board. The pin assignments are the same as those for the I/O channel. (Reference: 2.2.7 Expansion Channel)



Power Connector

Pin No.	Signal
1	GND
2	GND
3	+5V
4	+12V
5	-12V

Figure 4—21 Expansion Board and Pin Assignments

4.14. Expansion Unit

The expansion unit contains a power unit whose power capacity is the same as that of the system unit. An expansion board and diskette drive C (either 3.5" or 5.25") can be optionally installed. Alternating current is supplied from the connector on the system unit.

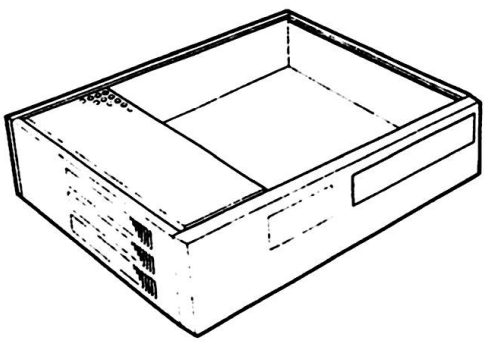


Figure 4—22 Expansion Unit

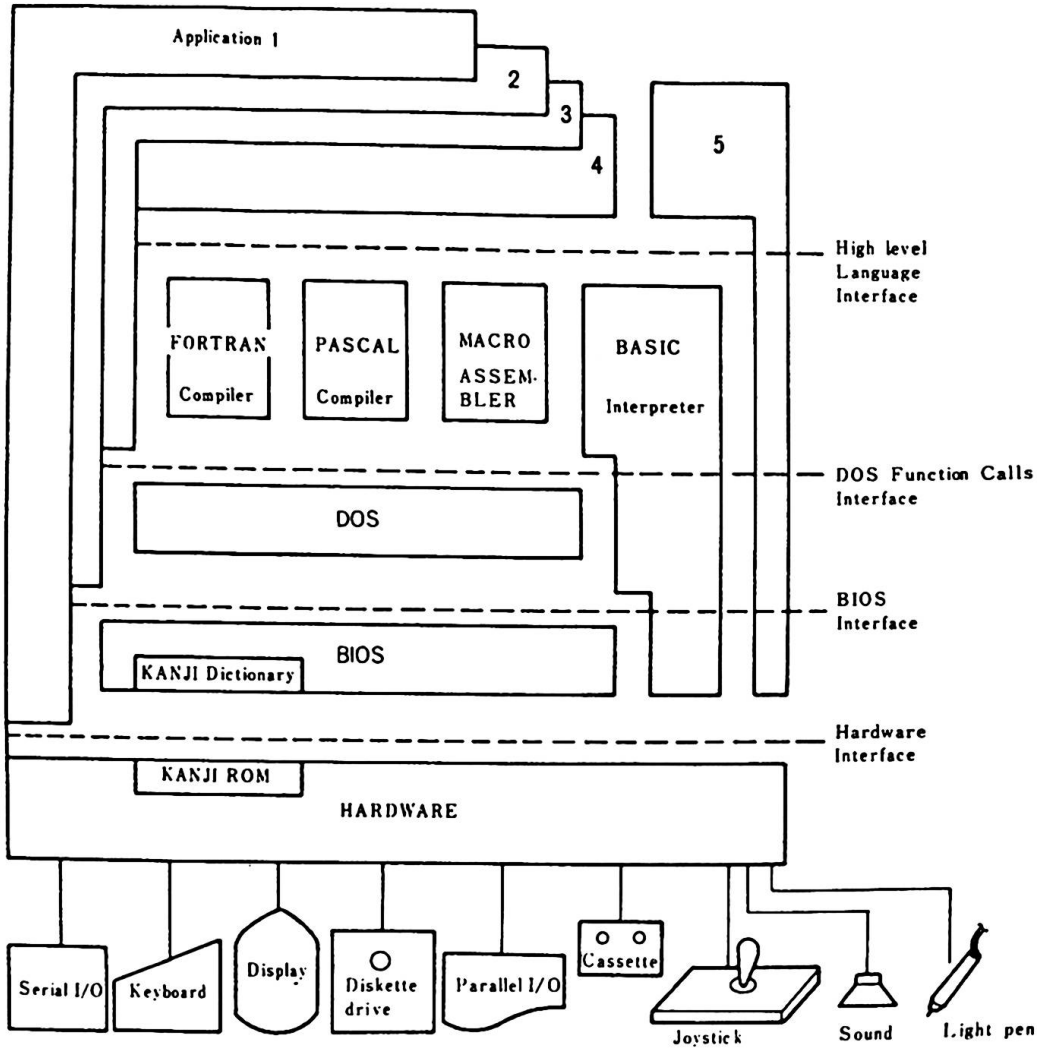


5. Software

This chapter contains information about system software, especially information about putting BIOS to practical use.

5.1. Software Structure

The IBM 5510 is utilized by the FORTRAN, PASCAL, ASSEMBLER, and BASIC programming languages in Native or Extension Video mode. BIOS is the interface between software and hardware.



Application 1: by Assembly Language or High level Language using hardware interface
Application 2: by Assembly Language using BIOS interface
Application 3: by Assembly Language using DOS interface
Application 4: by High level Language

Figure 5-1 Software structure

5.2. System Software

The Basic Input/Output System (BIOS) of JX Native mode resides in ROM on the system board and provides device level control for the major I/O devices in the system. In Extension Video mode, the initialization routines, video I/O and keyboard I/O routines are replaced by code which resides in the optional Extension Video mode cartridge. The other BIOS routines are shared with Native mode. In English mode, the initialization routines are replaced by code which resides in the optional English mode cartridge. The value at address FFFFE indicates whether English mode or one of the other two modes is currently active.

Mode	Contents of Address FFFFE (Hex)
English mode	FD (Hex)
Native mode / Extension Video mode	ED (Hex)

Distinction between Native or Extension Video mode is made possible by reading the AL register after issuing INT 11.

Bit 5,4 in AL
1 0 : Extension video mode
0 1 : Native mode

Figure 5-2 shows a map of system software routines in ROM.

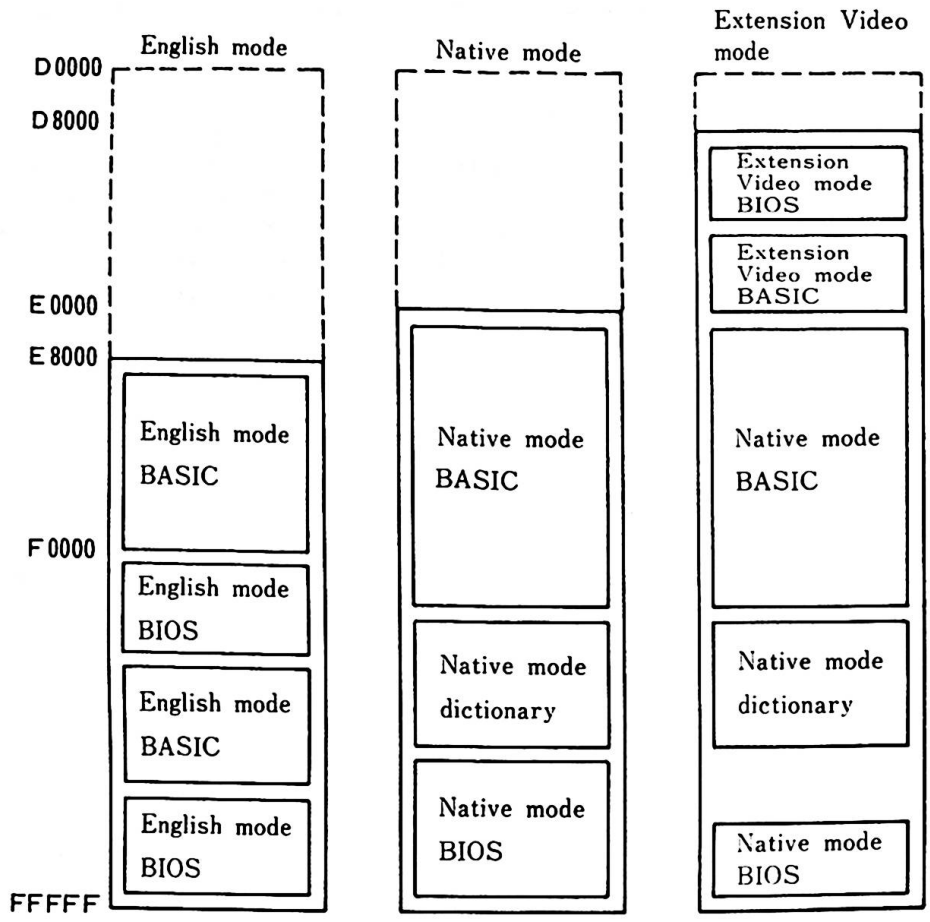


Figure 5-2 System Software Map

5.3. BIOS Usage

Access to BIOS is made through the software interrupts.
All data and parameters passed to and from the BIOS routines go through the registers of the MPU(8088).

If a BIOS routine supports several possible functions, the AH register indicates the desired function.

For example, the following code can be used to set or to read the time-of-day.

To set time-of-day:

```
MOV   AH, 1              ;  function is to set time-of-day
MOV   CX, [HIGH COUNT]
MOV   DX, [LOW COUNT]
INT   1AH                ;  software intrrupt
```

To read time-of-day: (CX and DX get values of TOD)

```
MOV   AH, 0              ;  function is to read time-of-day
INT   1AH                ;  software interrupt
```

Generally, the BIOS routines save all registers except for AX and the flags.

BIOS Programming Guidelines

1. To invoke the BIOS code use Software interrupts.
Do not 'hard code' BIOS addresses into applications. The internal workings and absolute addresses in BIOS are subject to change without notice.

2. When any error is detected in diskette operation, the diskette drive adapter must be reset before retrying the operation. A specified number of retries should be required on diskette "read" to insure that the problem is not due to motor start-up.

3. When altering I/O port bit values, change only those bits which are necessary to the current task. Upon completion, restore the original environment. Failure to adhere to this practice may cause incompatibility between present and future systems.

The following are the BIOS interrupt vectors explained in this chapter.

Vector Address	Type of Interrupts	Function
40-43	10	Video I/O
44-47	11	System configuration
48-4B	12	Memory size definition
4C-4F	13	Diskette I/O
50-53	14	ASYNC port I/O
54-57	15	Cassette I/O
58-5B	16	Keyboard I/O
5C-5F	17	Printer I/O
60-63	18	Resident(ROM) BASIC
64-67	19	System reset
68-6B	1A	Time of Day
6C-6F	1B	Keyboard Break address
70-73	1C	Timer
74-77	1D	Video parameter
78-7B	1E	Diskette Parmeter
	49	Conversion table
	7A	Dictionary pointer

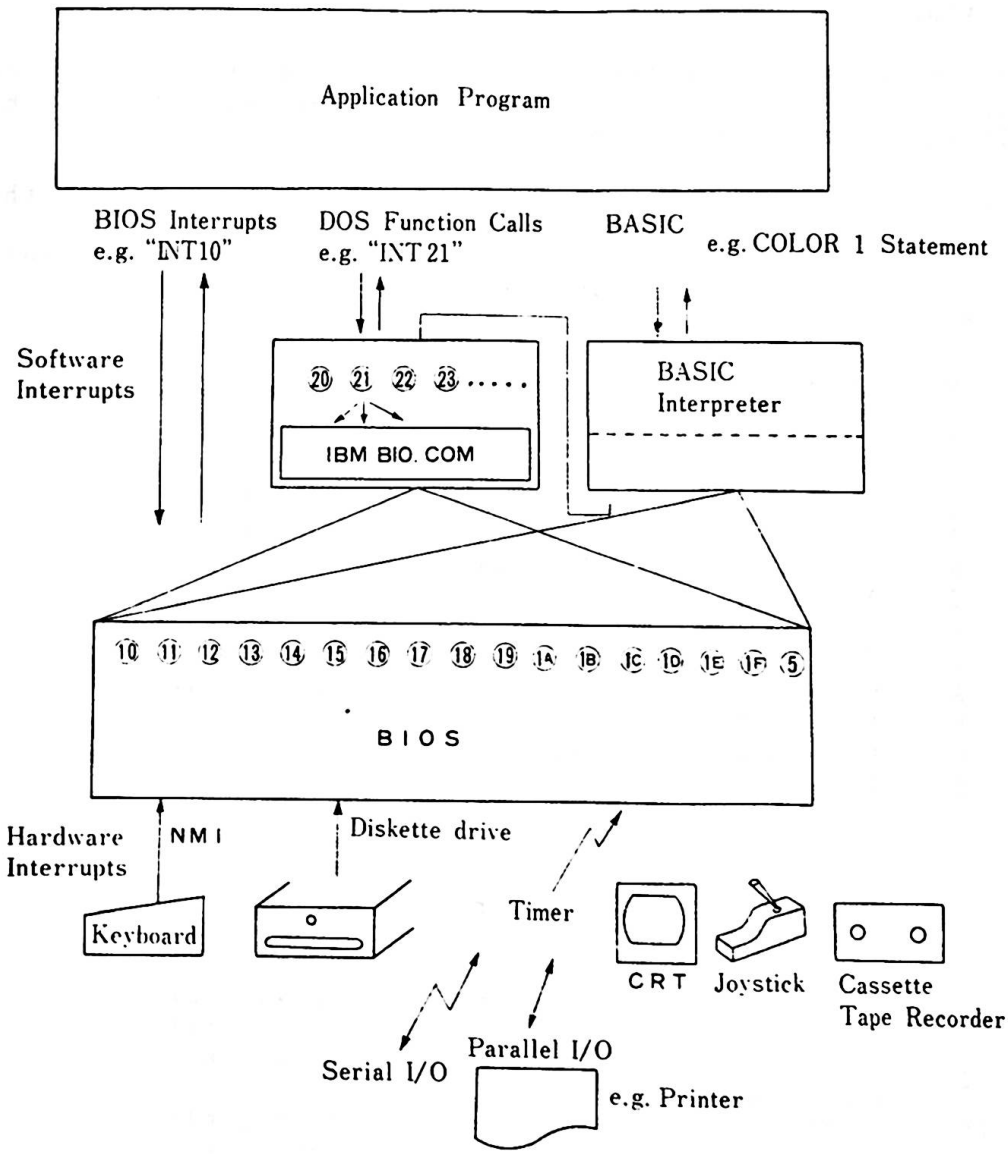


Figure 5-3 Software Interrupt Architecture

5.3.1. Native Mode BIOS Interrupts

TYPE 10 Video I/O

This vector points to the the code to be executed when video I/O operation is needed. The function performed depends on the value placed in the AH register.

(AH)= 0 : SET MODE — The AL register is used to set the desired video display mode. When AL bit 7 is "1", VRAM contents are not cleared.

(AL)	MODE		
0	40X 25	16 color	ANK
1	40X 25	16 color	ANK
2	80X 25	16 color	ANK
3	80X 25	16 color	ANK
4	320X200	4 color	(40X25 ANK)
5	320X200	4 color	(40X25 ANK)
6	640X200	2 color	(80X25 ANK)
7	Not used		
8	160X200	16 color	(20X25 ANK)
9	320X200	16 color	(40X25 ANK)
A	640X200	4 color	(80X25 ANK)
B	Not used		
C	Not used		
D	Not used		
E	Not used		
F	Not used		
10	20X 11	8 color	KJ
11	20X 11	8 color	KJ
12	40X 11	8 color	KJ
13	40X 11	8 color	KJ
14	320X200	4 color	(20X11 KJ)
15	320X200	4 color	(20X11 KJ)
16	640X200	2 color	(40X11 KJ)
17	Not used		
18	160X200	16 color	(10X11 KJ)
19	320X200	16 color	(20X11 KJ)
1A	640X200	4 color	(40X11 KJ)
1B	640X200	16 color	(40X11 KJ)

REMARKS)

ANK : Alphanumeric, Special character, Katakana
KJ : Kanji

(AH)= 1 Set cursor type. The cursor type is specified by the following bits in the CX register:

```
Bit  14,13 = 00     Non-Blink
           = 01     Non-Display
           = 10     Blink at 4 times per second
           = 11     Blink at 2 times per second
```

Blinking is not supported in graphics mode.

```
Bit  12-8   Start line of the cursor in a
            character box
Bit  4-0    End line of the cursor in a
            character box
```

(AH)= 2 Set Cursor position. The cursor position (row,col) is specified by values in (DH,DL). A sub page (16 KB page is further separated into 1 KB or 2 KB units) is specified in (BH). In graphics mode,a sub page is set to 00 in(BH). (0,0) indicates the home (upper left) position.

(AH)= 3 Read cursor position. When sub-page number is specified in (BH), the current cursor position (DH=row,DL=col) and the cursor type (CH,CL) are read in.

(AH)= 4 Read Light pen position.

```
(AL)=0 : Light pen switch is not pressed.
(AL)=1 : Valid light pen value is in registers.
(DH,DL) : light-pen position (row,col)
(CH)    : raster value  (0-199)
(BH)    : column value  (0-319,0-639)
```

(AH)= 5 Select active page.
(AL)=00-0F : sub-page (16 KB page is further separated into 1 KB or 2 KB units) is specified.
(AL)=80 : Read CRT/CPU page-registers
(AL)=81 : Write the contents of (BL) to CPU page-register. CPU page mode is specified in (CL).
(AL)=82 : Write the contents of (BH) to CRT page-register
(AL)=83 : Write the contents of (BL) and (BH) to CPU and CRT page-registers. CPU page mode is specified in (CL).

(AH)= 6 Scroll display upward. (CH,CL) specifies the upper left corner of the portion to be scrolled (CH=row,CL=column). (DH,DL) is the lower right corner(DH=row,DL=column). AL is the number of lines to be scrolled. BH holds the attributes for the space left.
AL=0 clears the area defined by CX and DX.

(AH)= 7 Scrolls display downward. Same as (AH)= 6.

(AH)= 8 Read a character (into AL) and its attributes (into AH) at the current cursor position.
A sub page is specified in (BH).
Full size characters return the following attributes:

1st byte 1XXX 0XXX (left half of a character)
2nd byte 1XXX 1XXX (right half of a character)

The attributes returned are the same as those written when specifying (AH)= 9.

All the attributes are supported in graphics mode.

In color graphics mode, the color attribute bits of AH have no affect on the color of the characters.

(AH)= 9 Writes one or more copies of the character in AL and its attributes in BL starting at the current cursor position. CX contains a count of the number of characters to be written.
The attributes of full size characters change after the 2nd byte is written.

The meanings of the attribute bits are explained in Chapter 3, " 3.4 Display function of VP1 and VP2 "

(AH)= A Write characters only. Same as (AH)=9 , except that attributes are not written.

(AH)= B Set a color in the palette.
Specify in BL the color number to be assigned to the specified palette number.
(BH)=0 : Set color number for background
(BH)=1 : Select the palette number to be used

color number	2-color		4-color		8-color	16-color
	BL=0	BL=1	BL=0	BL=1		
1	white	black	green	light blue	blue	blue
2			red	purple	green	green
3			yellow	white	light blue	light blue
4					red	red
5					purple	purple
6					yellow	yellow
7					white	white
8						gray
9						bright blue
10						bright green
11						bright light blue
12						bright red
13						bright purple
14						bright yellow
15						bright white

Remarks) Color number 0 specifies border color in 40X11 or 80X11 character mode.
In graphics mode, color number 0 specifies border and background color.

(AH)= C

Write a dot. (DX,CX) specifies the bit position (row,column) where the dot is to be written. (0,0) represents the home position on the display. Both the row and the column numbers are in units of dots, not characters. In color mode, AL specifies the palette number for the color dot to be written.
If the eighth bit (bit 7) in AL is specified as 0, the value of AL will be written directly. If the value in bit 7 is set to 1 , however, the current value of the dot will be XOR'ed (exclusive OR) with bit 0 and the result will be written.
This function works only in graphics mode.

(AH)= D

Read a dot. (DX,CX) specifies the (row,col) of the dot position to be read (both the row and column number are in units of dots, not characters). The dot value (on/off) of a dot is read into AL. The function only works in graphics mode.

(AH)= E

ASCII teletype routine for output.
Writes a character in the cursor position and advances the cursor. If the cursor is already in the rightmost position in line 10, the screen is scrolled upward.

The character to be written is specified in AL.

In graphics mode, color is specified in (BL).
THe status/mode symbols line is located outside the screen scroll area.

If the cursor is at the last position in a row and a full size character is to be written, a space is entered at the last position, and the character is written at the first position in the next line.

This function also works in graphics mode.

(AH)= F

Get the current display status. AL receives the current mode, AH receives the number of character columns displayed and BH receives the number of the sub-page.

(AH)=10 Set Palette Register. Set the parameter in (AL).

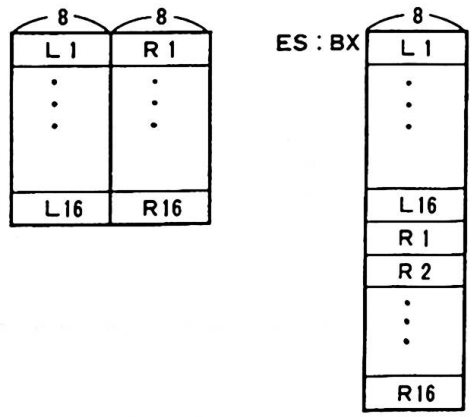
(AL)=0 : Specifies the number (00H-0FH) of the palette register in (BL), and the color number in (BH).

(AL)=1 : Sets contents of (BH) in the boreder color register.

(AL)=2 : Sets the palette register and the border color register. ES:DX points to a 17 byte list. Bytes 0-15 are written to palette registers 0-15 . Byte 16 is written to the boreder color register.

(AH)=11-12 Reserved

(AH)=13 Request a character font. Return the character font for the specified character in the user-designated memory location. CX must hold the internal code of the requested character. CH must be 0 for half size characters. AL must be 0. The font will be placed in the memory location designated by (ES:BX).



(AH)=14 Superimpose

(AL)= 0 : Mode is specified in (BH). VRAM is not cleared if bit 7 of (BH) is 1.

```
(BH)=0-3 not used
(BH)=4   320X200 4 colors : 40X25 ANK
(BH)=5   320X200 4 colors : 40X25 ANK
(BH)=6   640X200 2 colors : 80X25 ANK
(BH)=7   not used
(BH)=8   160X200 16 colors: 20X25 ANK
(BH)=9   320X200 16 colors: 40X25 ANK
(BH)=A   640X200 4 colors
(BH)=B-13 not used
(BH)=14  320X200 4 colors : 20X11 KANJI
(BH)=15  320X200 4 colors : 20X11 KANJI
(BH)=16  640X200 2 colors : 40X11 KANJI
(BH)=17  not used
(BH)=18  160X200 16 colors: 10X11 KANJI
(BH)=19  320X200 16 colors: 20X11 KANJI
(BH)=1A  640X200 4 colors : 40X11 KANJI
```

(AL)= 1 : (BH)=1 Superimpose is set to enable
(BH)=0 Superimpose is set to disable

(AL)= 2 : (BH)=0 VRAM 1 is set for a foreground page.

(AL)= 3 : Set a transparent color to the palette register specified in (BH).

(AL)= 4 : Set superimpose mode through (BH).

```
(BH)=0   priority
(BH)=1   XOR
(BH)=2   AND
(BH)=3   OR
```

TYPE 11 Device Configuration Status

The device configuration status is returned in AX.
The information specified by the bits in AX is listed below:

Bit 15,14	Number of printers (usually 1 is returned)
Bit 13	Serial printer is connected when 1
Bit 12	Game I/O is connected when 1
Bit 11-9	Number of ASYNC communication ports connected
Bit 8	Hard-disk unit is connected
Bit 7,6	Number of diskette drives (besides A)
Bit 5,4	Video mode initialized (always 01)*
Bit 3,2	RAM size on board (always 11)
Bit 1	Reserved
Bit 0	IPLed from a diskette drive

* Native or Extension Video mode is identified by a 1 in bit 4. In Native mode it is always 1.

TYPE 12 Get Memory Size

Memory size is set in AX in 1K-byte units.

TYPE 13 Diskette I/O

Executes different functions based on the value in AH.

(AH)= 0 Reset the diskette system. Bits 5-0 in DL must be set to the drive number.
Set bit 7 to 0 when diskette drive units are connected, and set bit 6 to 0 (40 tracks) or 1 (80 tracks). Set bit 7 to 1 when a hard disk unit is connected.

(AH)= 1 Read the system status. The DL setting is the same as for (AH)=0. The status from the last operation, returned in AL, is described below.

Value(Hex)	Operation status
80	Time out
40	Bad seek operation
30	No hard disk (only when bit 7 of DL is 1)
20	Device out of order
10	CRC (Cyclic Redundancy Check) error found in reading a diskette
04	Desired sectors not found
03	Attempt to write to a write-protected disk
02	Address mark not found
01	Incorrect command

(AH)= 2-7 Set registers as follows:

(AL) Number of sectors (value unchecked, not used by FORMAT)
(CH) track number (0-79, value unchecked)
(CL) sector number (value unchecked, not used by FORMAT)
(DH) head number (0-1, value unchecked)
(DL) drive number (0-3, value checked)
bit 7 =0 identifies a diskette drive,
bit 6 =1 identifies double track access.

(ES:BX) buffer location (not required for checking)

(AH)=2 Read sectors

(AH)=3 Write sectors

(AH)=4 Check sectors

(AH)= 5 Format sectors. The buffer pointer (ES:BX) must point to the set of address marks of a track. Each address mark consists of 4 bytes (C,H,R,N), where C stands for track, H for head number, R for sector number, and N for number of bytes (00=128, 01=256, 02=512, 03=1024) within a sector. Each sector in a track must have a corresponding address mark, which helps locate the desired sector in a read/write operation.

(AH)= 6-7 Reserved

If the transfer of data succeeds, the carry flag (CF)=0 will be returned. If not, (CF)=1 is returned, and the status is returned in (AH) as when (AH)=1.
For read, write, and check operations, only AX and the carry flag value will be altered. The number of sectors actually read will be returned in AL, but this value is meaningless when a time-out occurs or the system is reset with (AH)= 0. The contents of AH cannot be guaranteed in the latter case.

When an error message is received, retry after resetting the diskette adapter. More than 10 retries are required.

TYPE 14 ASYNC Communication Port Input/Output

This routine provides byte stream I/O to the communication ports as designated by the following parameters. Port number (0-1) is specified in DX.

(AH)= 0 Initializes the communications port as specified in AL :

	baud rate	parity	stop bits	character length
BIT :	7 6 5	4 3	2	1 0
	0 0 0 -110	X 0 -none	0 -1 bit	1 0 -7 bits
	0 0 1 -150	0 1 -odd	1 -2 bits	1 1 -8 bits
	0 1 0 -300	1 1 -even		
	0 1 1 -600			
	1 0 0 -1200			
	1 0 1 -2400			
	1 1 0 -4800			
	1 1 1 -9800			

The DTR signal will be ON upon completion of initialization. When the routine exits, the AL value will be set in a call for communications status (AH=3).

(AH)= 1 Send the character in AL over the communications line. The contents of AL are preserved.

If the character can not be transmitted, bit 7 of AH is set to 1. Otherwise, the current line status will be returned by the remaining bits as when (AH)=3.

(AH)= 2 Receive a character from the communications line into AL, before returning to the caller. On exit AH has the current line status, as set by the status routine (AH=3), except that the only bits left on are the error bits (7,4,3,2,1).

(AH)= 3 Returns port status in AX.

AH will contain the communications line status as shown below:

bit 7	time out
bit 6	transmitter shift register is empty
bit 5	transmitter holding register is empty
bit 4	break detect
bit 3	framing error
bit 2	parity error
bit 1	overrun error
bit 0	data ready

AL will contain the modem status as follows:

bit 7	data carrier detect
bit 6	ring indicator
bit 5	data set ready
bit 4	clear to send
bit 3	data carrier detect status changed
bit 2	ring indicator end
bit 1	data set ready changed
bit 0	clear to send changed

TYPE 15 Cassette Input/Output

The function is specified in AH as follows:

(AH)= 0 Turn Cassette motor on

(AH)= 1 Turn Cassette motor off

(AH)= 2 Read data from the cassette tape unit.

(ES,BX) contains the pointer to the data buffer.
(CX) contains the number of data to be read.

The registers and the values returned are as follows:

```
ES:BX  :  buffer address of the last byte read
          plus 1
DX     :  the number of actual bytes read
AH     :  =1  when CRC error is detected
          =2  when no signal is detected
          =4  when no leader is detected
CY     :  carry flag =0  no error is detected
          carry flag =1  some error is detected
```

(AH)= 3 Write data to the cassette tape unit.

(ES:BX) contains the pointer to the data buffer.
(CX) contains the number of data to be written.

The registers and the values returned are as follows:

```
ES:BX  : buffer address of the last byte written
          plus 1
AH     : Any other than the above values causes
         (CY)=1 and (AH)= 80 to be returned.
```

TYPE 16 Keyboard Input/Output

Executes one of the following functions as indicated by AH:

(AH)= 0 Reads next character. Reads a character from the keyboard and puts the following codes into AH and AL.

Data type	AH	AL
1-byte character	scan code	ASCII code
2-byte char. 1st-byte	scan code	1st-byte
2-byte char. 2nd-byte	scan code	2nd-byte
function key, etc.	pseudo scan code	00
input JIS 8 bit code with ALT key pressed	00	pseudo scan code
Kanji 1st-byte	FF	1st-byte
2nd-byte by Kana-Kan conversion	FF	2nd-byte

If more data remain in the buffer, the initial data are returned. Otherwise, the routine stays active for the new data input.

(AH)= 1 Indicate if a character is available to be read. ZF (zero flag) will be set as follows, to indicate whether data have been transmitted into the buffer or not:

(ZF)=1 no character is in the buffer for reading
(ZF)=0 character is in the buffer for reading

When (ZF)=0, the next character will be sent to AX. The character remains unchanged in the buffer until a call is made with (AH)=0 to read the next character.

(AH)= 2 Reads shift status. Current shift status is sent to AL and AH as follows:

AL register

bit 7	= 1	Insert mode
bit 6	= 1	CAPS Lock pressed
bit 5		Unused
bit 4	= 1	Scroll Lock pressed
bit 3	= 1	ALT key pressed
bit 2	= 1	Control key pressed
bit 1-0	= 01	Right-shift key pressed
bit 1-0	= 10	Left-shift key pressed

AH register

bit 7-3		Unused
bit 2-1	=00	Alphanumeric shift
	=01	Katakana shift
	=10	Hiragana shift
bit 0	=1	Full size mode
	=0	Half size mode

(AH)= 3 Sets typamatic rate

AL=0	Return to default values
AL=1	Increase initial delay
AL=2	Slow typamatic rate by one half
AL=3	Combine AL=1 and AL=2
AL=4	Disable typamatic

(AH)= 4

AL=0	Turn keyboard click off
AL=1	Turn keyboard click on

(AH)= 5 Alters keyboard status or mode. Sets desired status or mode in AL.

Status/Mode symbols displayed are altered.

AL register

bit 7-6	=00	Shift out Kanji-mode
	=01	Shift in Kanji-mode
	=10	Shift Kanji-mode in or out
	=11	Not switched
bit 5-4	=00	CAPS Lock off
	=01	CAPS Lock on
	=10	Switch CAPS Lock on or off
	=11	Not swithed
bit 3-2	=00	Alphanumeric shift
	=01	Katakana shift
	=10	Hiragana shift
	=11	Not changed
bit 1-0	=00	Half size mode
	=01	Full size mode
	=10	Switch Half/Full-size mode
	=11	Not switched

(AH)= 6 Reserved

(AH)= 7 Status/symbol line and Kana-Kanji conversion possible or impossible.

AL
Bit 0= 0 Kana-Kanji conversion is possible (default).
= 1 Kana-Kanji conversion is impossible
Bit 1= 0 Access to the indicator line is possible. (default)
= 1 Access to the indicator line is impossible.
When the screen mode is reset ,these parameters will revert to the default values.

(AH)= 85 Same as (AH)=5, except status/mode symbols displayed are not altered.

TYPE 17 Printer I/O

The printer BIOS supports the IBM 5512 Thermal Transfer Printer (hereafter called PT-2) and the IBM 5513 Thermal Paper Printer (hereafter called PT-1). When the printer is called by (AH)=0, the entire contents of AL are printed to the PT-2 printer.
The situation may not be the same as PT-1.
Descriptions of PT-1 output when called by (AH)=0 are provided followed by a description of the BIOS common to both printers.

When PT-1 is connected and BIOS is called by AH=0, the contents of AL determines whether they are character code, control code or data. As for the multiple number of bytes of control code, the control code sequence should strictly be followed at the time of BIOS process.

Character codes for sending one byte (ANK) or two byte characters should be IBM internal codes.
The character font output to the printer is an image of what was obtained by a request for a Video BIOS character font.
The characters use Hankaku 7 X 16 dot and Zenkaku for 15 X 16 font patterns.
The printer output is performed in units of one line. The data are stored in the BIOS character buffer until the printing commands such as LF, FF are received.
At the time when BIOS receives printing commands, character patterns for one line are output to the printer as image data. For this reason, characters and image data can not reside within the same line. (In this case, a line is automatically fed.)
The PT-1 uses an 8 dot print head. 16 dot vertical printing requires the head to move twice.
Printed characters can be large or small characters. A change from one size to the other must be made at the beginning of a line.
Small characters are the default.

Control codes used are those for the IBM 5553/5557 printers and some of these differ from those of the PT-1. BIOS converts them to PT-1 codes. Due to the PT-1 hardware limitations, those codes which BIOS can not convert are replaced by blanks.

Out of the IBM 5553/5557 control codes, the following are converted by BIOS:

1) CAN : Cancellation
After the image buffer or code buffer within BIOS are cleared, the CAN code is output to the printer.

2) CR : Carriage Return
Because the PT-1 automatically issues an LF, the CR code is neglected within BIOS.

3) LF : Line Feed
An LF is taken as a print command and after data in the buffer are output to the printer, the LF is performed.
The PT-1 automatically issues an LF.

4) FF : Page Change
FF is taken as a print command and after data in the buffer are output to the printer, the page change is performed.

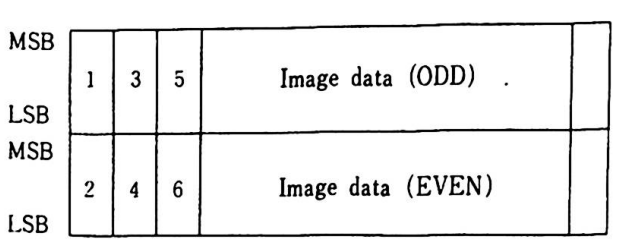
5) SP : Space
One Hankaku character space corresponds to one space output.

6) ESC % 1 : Single length image data transfer
(graphics image handling code)
This is used when graphics image data are sent.
Conversion is made as follows: (2 byte code data transfer mode should be followed.)

ESC % 1 N1 N2 D1 D2 D3 D4 .....D(2 X N1N2)

is converted to

ESC L N1 N2 D1 D3 D5 ....D(2 X N1N2)-1
+LF
+CR
+ESC L N1 N2 D2 D4 D6 ....D(2 X N1N2)



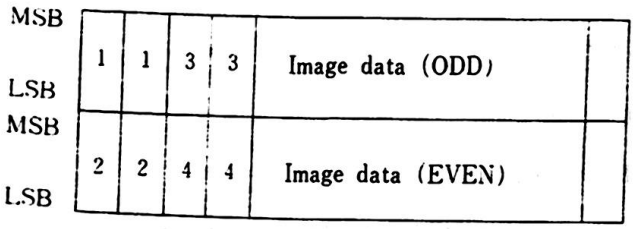
7) ESC % 2 : Double length image transfer (graphics image handling code)

This is used in graphics image data transfer as follows: (2 byte data transfer mode should be followed.)

ESC % 2 N1 N2 D1 D2 D3 D4 ....D(2 X N1N2)

is converted to

ESC L N1' N2' D1 D1 D3 D3..D(2 X N1N2)-1 D(2 X N1N2)-1
+LF
+CR
+ESC L N1' N2' D2 D2 D4 D4..D(2 X N1N2) D(2 X N1N2)



8) ESC % 3 : Horizontal skip (graphics image handling code)
This is a command to skip the dots specified and is converted as follows:

ESC % 3 N1 N2

is converted to

ESC 1 N1 N2 00 00 00 00 ..... 00 00

N1N2

9) ESC % 5 : Vertical skip (graphics image handling code)

This is a command to feed the paper vertically for the number of dots specified. As the PT-1 is unable to process dots, the conversion is made as follows:

ESC % 5 N1 N2

is converted to

ESC 0 (N1N2 / 13)

The actual paper feed is made in units of 1/9 inch (2.82 mm), N1N2 / 13 times (rounded).
(2 inch maximum (50.8mm))
Where N1N2 / 13 is less than 2, the value is rounded up to 2.

10) ESC % 6 : Set CR point (graphics image handling code)

This is a command to move the print-begin-position the dots specified and is converted as follows:

ESC % 6 N1 N2

is converted to

CR (carriage return)
+ESC L N1N2 00 00 00 00 00 00 00 00 00 00

N1N2

11) ESC % 9 : Set Line Space

The number of line spaces when LF is received is set by the value of N1N2 as follows:

0 ≦ N1N2 <	13 ·····1/9 inch	LF 2 Times
13 ≦ N1N2 <	26 ·····1/9 inch	LF 3 Times
26 ≦ N1N2 <	39 ·····1/9 inch	LF 4 Times
39 ≦ N1N2 <	52 ·····1/9 inch	LF 5 Times
52 ≦ N1N2 <	65 ·····1/9 inch	LF 6 Times
65 ≦ N1N2 <	78 ·····1/9 inch	LF 7 Times
78 ≦ N1N2 <	91 ·····1/9 inch	LF 8 Times
91 ≦ N1N2 <	104 ·····1/9 inch	LF 9 Times
104 ≦ N1N2 <	·····1/9 inch	LF 10 Times

12) ESC F : Set page length

This is a command to set the length of a page with 6 LPI for small characters and 3 LPI for large characters. Conversion is made as follows:

ESC F N1 N2

is converted to

ESC C N1 N2 (N1 N2 < 126 (SMALL),63(LARGE))

When N1N2 is greater than 126 or 63, the value is forced to 126 or 63.

13) ESC £ : Set ANK Enlargement

When this command is received, a double size character will be printed.

14) ESC | : Release ANK Enlargement

When this command is received, ESC £ is released and the normal size characters will be printed thereafter.

15) FS : Fixed length image transfer (graphics image handling code)

When this command is received, the same image transfer commands (ESC % 1 , ESC % 2) as those most recently issued and the image data transfer using data numbers will be initiated.

16) Other control codes

The following control codes do not have meaning to the PT-1 and are neglected by BIOS. In this case, commands which consist of single or multiple bytes neglect all the bytes which the command takes as valid. The printer waits for the next control code to be presented.

Control Code overlooked		No. of bytes for Command generation
CR	CARRIAGE RETURN	(1)
BS	BACK SPACE	(1)
DC 1	SELECT	(1)
DC 3	DESELECT	(1)
ESC %4 N 1 N 2	HORIZONTAL REVERSE SKIP	(5)
ESC %8 N 1 N 2	VERTICAL REVERSE INDEX	(5)
ESC %B	BIDIRECTIONAL PRINT	(3)
ESC %U	NORMAL PRINT	(3)
ESC S	SHEET FEED	(2)
ESC V	SHEET EJECT	(2)
ESC O	HIGH SPEED PRINT START	(2)
ESC P	HIGH SPEED PRINT RELEASE	(2)
ESC (	3BYTE DATA TRANSFER	(2)
ESC)	2BYTE DATA TRANSFER	(2)

REMARKS) 1. As ESC % 1, ESC % 2, ESC % 3, ESC % 5, ESC % 6,FS are operated within BIOS as the graphics image handling codes, another LF is added when thay are used together with character data.

2. When ESC % 1, ESC % 2, ESC % 3, ESC % 6 as used, the total of N1N2 within a line should not exceed 1120 dots, larger values are ignored.

3. When multiple ESC commands are received, the first ESC is valid and other ESC are ignored until a code other than ESC is issued.

The value in AH indicates to BIOS which of the following functions to execute.
PT-1 and PT-2 have the same function unless otherwise stated.

When returning to the calling program, DX must be 0. AH will contain status values, while other registers remain unchanged.

(AH)= 0

Prints the character specified in AL. A hex 7F (DEL) prints a special character. Also sends control codes to the printer through AL. In processing output data, there are differences between PT-1 and PT-2, as stated before.

(AH)= 1

Initializes the printer. Initializes the hardware, resets the software status, and then sets the initial control values, as shown below.
PT-1 values are in parentheses.

- Alphanumeric	10 (12)	characters per inch
- Kanji	5 (6)	characters per inch
- Line feed	6 (4.5)	lines per inch
- Page length	66 (49.5)	lines per page
- Speed	normal	
- Print character	SMALL character	

(AH)= 2

Reads status-1. Reads printer status into AH as follows:

bit 7	=0	In use.
bit 6		Reserved
bit 5	=1	Out of paper (EOF) or paper jam in the automatic sheet feed.
bit 4	=1	Printer ready.
bit 3	=1	An error such as EOF, printer disconnected, CANCEL key pressed or time-out has occurred.
bit 2,1	=1	Reserved
bit 0	=1	Time out. The printer is taking an abnormally long period of time to print characters.

(AH)= 3 Reads status-2. Reads the printer status into AH as follows:

```
bit 7          Always "1"
bit 6,5        Unused
bit 4,3 =11      7.5 LPI
          10     6   LPI
          01     5   LPI
          00     4   LPI
bit 2,1 =11      7.5/15  CPI
        =10      6.7/13.3 CPI
        =01      6/12 CPI
        =00      5/10 CPI
bit 0   =1       PT-1 is connected
        =0       PT-2 is connected
```

(AH)= 4 Prints contents of AL register directly.

(AH)= 5 Prints double wide. Same as in (AH)=0 except that the horizontal size of the character is doubled.

(AH)= 6-A Unused

(AH)= B Prints a line (including associated attributes) in Extension Video mode. (PT-2 only)
Prints the character string in the character buffer indicated by ES:DI. The nth character in the character buffer will be printed with the nth attribute in the attribute buffer. The length of the buffer is specified in CX.

The attributes for this function are one or two byte values indicating how characters are printed. The meaning of each bit in an attribute byte are as follows:

bit		Meaning
7		Reserved. Must be set to 0.
6		Underline
5		Reserved.
4		Reserved. Must be set to 0.
3,2		Vertical grid line
	=00	None
	=01	Single solid line
	=10	Heavy solid line
	=11	Single dotted line
1,0		Horizontal grid line
		The values are the same as for a vertical grid line.

A one-byte attribute is needed for a half-size character; a 2-byte attribute is needed for a full-size character. Horizontal lines are printed above the characters involved; vertical lines are printed to the left of the characters involved.

After the number of bytes specified in the CX register is printed, BIOS will automatically print a carriage return character and a line feed character unless the character buffer ends with a carriage return character or a line-feed character, or both. Thus the character buffer must contain all the characters to be printed in one line. The control characters that can be included in the character buffer are limited to the carriage return, line-feed, ESC [and ESC] that are used at the end of the character buffer.

(AH)= C

Sets printer control values.
The control values are specified in AL as follows.

(AL) Specified control value

0 Reset printer default values. See (AH)=1.

1 Change character pitch

(BH)=90	5	Full size characters/inch
(BH)=78	6	Full size characters/inch
(BH)=6C	6.7	Full size characters/inch
(BH)=60	7.5	Full size characters/inch

The value for half size characters is double that of full size characters.

2 Change line feed pitch

(BH)=1E	4	(3)	lines per inch
(BH)=18	5	(3)	lines per inch
(BH)=14	6	(3)	lines per inch
(BH)=10	7.5	(4.5)	lines per inch

3 Change page length.
(BX) number of lines per page
(based on 6 lines/inch)

4 Set double/normal speed mode.
(BH)=0 set double speed mode
(BH)=1 set normal speed mode

5 Set unidirectional/bidirectional printing mode
(PT-2 only)
(BH)=0 set unidirectional printing mode
(BH)=1 set bidirectional printing mode

6 Change printing character (PT-1 only)
(BH)= 0 SMALL character
(BH)= 1 LARGE character

TYPE 18 ROM BASIC
This interrupt executes a BASIC program.

TYPE 19 System reset

DOS is restarted by the BOOT program, or by issuing INT 18.

TYPE 1A Timer Support. Reads or sets the time as specified in AH.

(AH)= 0 Reads the current value of the time-of-day counter and returns the following:

CX most significant word of count
DX least significant word of count
AL=0 count has not passed 24 hours since the last time it was read
AL<>0 24 hours have passed

(AH)= 1 Set time-of-day counter to the value specified in the CX and DX registers.

CX most significant word of count
DX least significant word of count
note: Counting rate is 1193180/65536 per sec,i.e., there are 18.2 counts per sec.

5.3.2. Extension Video Mode BIOS Interrupts

INT10, INT11, and INT16 have different meanings in Extension Video mode from in Native mode. An explanation of these three interrupts in Extension Video mode are as follows.
Refer to chapter 3 " 3.5 VP3 Display Function ", for more information on Extension Video mode.

TYPE 10 Display Input/Output

Execute the following functions as specified in AH.

(AH)= 0 Set mode. Sets display mode as specified in AL.

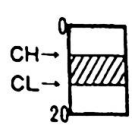
(AL)	Mode
0-7	Reserved
8	80X25 mono character mode (initial mode)
9	720X512 mono graphics mode 80X25 characters displayed
10	Reserved
11	360X512 4 color graphics - 40X25 characters displayed - double size character width - palette 11 used for character color - palette 00 used for background color - normal size characters printed except for screen print. - All palettes, except palette 0 are printed. Refer to(AH)=11, "set palette color".
12-13	Reserved
14	80X25 color characters

(AH)= 1 Set cursor-type. The cursor type is specified by bits in the CX register.

bit 14,13	=00	non-blink
	=01	non-display
	=10	blink at 4 times per second
	=11	blink at 2 times per second

Blinking is not supported in graphics mode.

bit 12-8 start line of the cursor in a character box
bit 4-0 end line of cursor in a character box



(AH)= 2 Set cursor position. The cursor position (row,col) is specified by values in (DH,DL). (0,0) indicates the home (upper left) position.

(AH)= 3 Read cursor position. (DH,DL) contains the current cursor position (DH=row,DL=col). (CH,CL) contain the cursor type.

(AH)= 4 Reserved

(AH)= 5 Reserved

(AH)= 6 Scrolls display upward. (CH,CL) specifies the upper left corner of the position to be scrolled (CH=row,CL=column), (DH,DL) the lower right corner (DH=row,DL=column), AL the number of lines to be scrolled, and BH the attributes for the space left. AL=0 clears the area defined by CX and DX.

(AH)= 7 Scrolls display downward. Same as (AH)=6.

(AH)= 8 Reads the character at the current cursor position into AL and its attributes into AH.

Full-size characters return the following attributes.

1st-byte	XXXX XX01
2nd-byte	XXXX XX11

The attributes returned are the same as those written by (AH)=9.

All attributes are supported in graphics mode.

In color graphics mode the color attribute bits of AH have no affect on the color of the characters.

(AH)= 9 Writes cursor posision attributes and characters. Characters to write in AL and their attributes, with the number of characters indicated units of half-size characters, should be specified in CX. For full-size characters, the attributes change after the 2nd byte is written.

Bit 1 or 0 of the attribute is set depending on the contents of the AL register.

Half-size character	XXXX XXX0
Full-size character	
1st byte	XXXX XX01
2nd byte	XXXX XX11

All the attributes except high-intensity and blink are supported in graphics mode.

In color graphics mode, the color attribute bits of BL have no affect on the color of the characters.

(AH)= A Write character only. Same as (AH)=9, except that attributes are not written (nothing specified in BL).

(AH)= B Set a color in the palette. Specify in BH the palette number to be set (0-3). Specify in BL the color number (0-15) to be assigned to the specified palette number.

(AH)= C Write a dot. (DX,CX) specifies the bit position (row,column) where the dot is to be written, (0,0) represents the home position on the display. Both the row and the column number are in units of dots, not characters.
In color mode, AL specifies the palette number (0-3) for the color dot to be written.
In monochrome graphics mode, bit 0 or 1 is specified in AL. If the eighth bit (bit 7) in AL is specified as 0, the value in bit 0 (or bit 0,1) of AL will be written directly.
If the value in bit 7 is set to 1, however, the XOR (exclusive OR) value of the current value and the value of bit 0 in AL will be written.

This function works only in graphics mode.

(AH)= D

Read a dot. (DX,CX) specify the (row,column) of the dot position to be read (both the row and column number are in units of dots, not characters). The dot value (on /off) is read into bit 0 (or 0,1) of AL.

This function only works in graphics mode.

(AH)= E

ASCII teletype routine for output.
Writes a character at the cursor position and advances the cursor. If the cursor is already in the rightmost position in line 24, the screen is scrolled upward.
The character to be written is specified in AL.

If the cursor is at the last position in a row and a full-size character is to be written, a space is placed at this position and at the 1st position in the next row the full-size character is written.

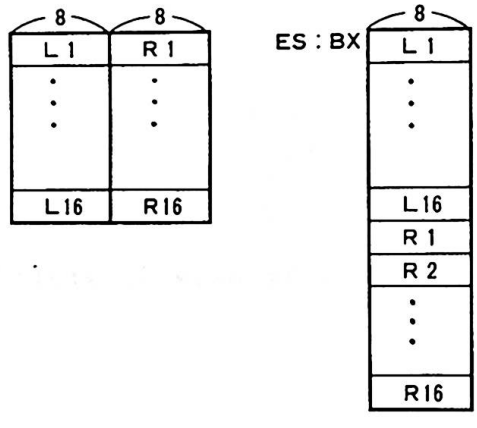
This function also works in graphics mode.

(AH)= F

Get the current display status. AL receives the current mode and AH receives the number of character columns displayed.

(AH)=10

Request a font pattern. Returns the font pattern for the specified character in the user-designated memory location. CX must hold the internal code of the requested character. CH must must be 0 for half-size characters. AL must be 0. Fonts will be placed in the memory location designated by (EX:BX), as follows:



(AH)=11 Sets the display attributes. The display attributes are specified in BH as follows:

(BH)	Display attributes
bit 6 =1	display in high-intensity
bit 3-0	grid line color (0-15). The colors are the same as those described in "Setting the palette ".

TYPE 11 Device Configuration Status

The device configuration status is returned in AX.
The information, specified by the bits in AX, is listed below:

bit 15,14		Number of printers (usually 1 is returned)
bit 13 =1		Reserved
bit 12		Reserved
bit 11-9		Number of ASYNC communication ports connected
bit 8 =1		Reserved
bit 7,6		Number of diskette drives
bit 5,4		Video mode initialized (always =10)*
bit 3,2		Type of display
	=00	12 inch monochrome display(always =00)
bit 1		Reserved
bit 0		Reserved (=1 diskette drive is connected)

* Extension mode is identified by checking bits 5 and 4 for values of 1 and 0, respectively.

RAM addresses 700-701(Hex) are reserved for system use in Extension Video mode.

TYPE 16 Keyboard Input/Output

One of the following functions is executed depending on the value in AH:

(AH)= 0 Reads next character. Reads a character from the keyboard and puts the following codes into AH and AL.

Data type	AH	AL
1 byte code character	scan code	ASCII code
2 byte code 1st byte 2 byte code 2nd byte	scan code scan code	1st byte 2nd byte
Function key, etc. JIS code Using ALT key	pseudo scan code	00
JIS-8 bit code input	00	pseudo scan code
Kanji 1st byte by kana-kan 2nd byte conversion	FF FF	1st byte 2nd byte

If more data remain in the buffer, the initial data are returned. Otherwise, the routine stays active for the next data input.

(AH)= 1 Indicates if a JIS-8 bit code character is available to be read. ZF (zero flag) will be set as follows, to indicate whether data have been transmitted into the buffer or not:

(ZF)=1 No character is in the buffer for reading
(ZF)=0 Character is in the buffer for reading

When (ZF)=0, the next character will be sent to AX. The character remains unchanged in the buffer until a call is made with (AH)=0 to read the next character.

(AH)= 2 Reads shift status. Current shift status is sent to AL and AH as follows:

AL register

```
bit 7   = 1     insert mode
bit 6   = 1     CAPS Lock pressed
bit 5           unused
bit 4   = 1     Scroll Lock pressed
bit 3   = 1     ALT key pressed
bit 2   = 1     Control key pressed
bit 1,0= 01     right shift key pressed
       = 10     left shift key pressed
```

AH register

bit 7-3		Unused
bit 2,1	=00	Alphanumeric shift
	=01	Katakana shift
	=10	Hiragana shift
bit 0	=1	Full-size mode
	=0	Half-size mode

(AH)= 3 Clicker on : Causes the speaker to generate sound with frequencies of 31-32767 Hz as specified in CX.

(AH)= 4 Clicker off : Turns keyboard click off.

(AH)= 5 Alters keyboard status or mode. Sets desired status or mode in AL.

AL register

bit 7-6	=00	Shift out Kanji-mode
	=01	Shift in Kanji-mode
	=10	Shift Kanji-mode in or out
	=11	Do not switch
bit 5-4	=00	CAPS Lock off
	=01	CAPS Lock on
	=10	Switch CAPS Lock on or off
	=11	Do not switch
bit 3-2	=00	Alphanumeric shift
	=01	Katakana shift
	=10	Hiragana shift
	=11	Do not change shift status
bit 1-0	=00	Half-size mode
	=01	Full-size mode
	=10	Switch Half/Full size mode
	=11	Do not switch

Status/Mode symbols displayed are altered.

(AH)= 6 Reserved

(AH)= 7 Kana-to-Kanji conversion and status/mode symbol line possible or impossible.

AL
Bit 0 =0 Kana-to-Kanji conversion is enabled (default)
=1 Kana-to-Kanji conversion is disabled
Bit 1 =0 Indicator line is enabled (default)
=1 Indicator line is disabled

When display mode is reset, these parameter are reset to default values.

(AH)=85 Same as (AH)=5, except that status/mode symbols displayed are not altered.

5.3.3. Interrupt Routines For Special Use

The following are descriptions of the BIOS routines for special use:

INT 5 : Screen print
When screen prints, either in Native mode or Extension Video mode are required the screen mode (character or graphics mode) is automatically set for the printer, and the printing is done in the correct mode. The printing direction for character mode is the same as that displayed. For the graphics mode, the printing line is rotated 90 degrees.

INT 1B : Keyboard Break Address
This vector points to the code to be executed when Break is pressed on the keyboard. The vector is invoked while responding to the keyboard interrupt, and control should be returned through an IRET instruction. The POWER-ON routines initialize this vector to an IRET instruction, so that nothing occurs when Break is pressed unless the application program sets a different value.

Control may be retained by this routine, with the following problem. The 'Break' may have occurred during interrupt processing, so that one or more 'End of Interrupt' commands must be issued in case an operation was underway at the time.

INT 1C : Timer
This vector points to the code to be executed on every system-clock tick. This vector is invoked while responding to the 'timer' interrupt, and control should be returned through an IRET instruction. The POWER-ON routines initialize this vector to point to an IRET instruction, so that nothing occurs unless the application modifies the pointer. It is the responsibility of the application to save and restore all registers that are modified.

INT 1D : Video Parameter
This vector points to a data region containing the parameters required for the initialization of the CRT Controller. Note that there are six separate tables, and all six must be reproduced if all modes of operation (ANK, Graphics, Kanji) are supported. The POWER-ON routines initialize this vector to point to the parameters contained in the ROM video-routines. It is recommended that if a programmer wishes to use a different parameter table, that the table contained in ROM be copied to RAM and just modify the values needed for the application.

INT 1E : Diskette Parameter
This vector points to a data region containing the parameters required for the diskette drive. The POWER-ON routines initialize the vector to point to the parameters contained in the ROM DISKETTE-routine. It is recommended that if a programmer wishes to use a different parameter table, that the table contained in ROM be copied to RAM and just modify the values needed for the application. The motor start-up-time parameter (parameter 10) is overridden by BIOS to force a 500-ms delay (value 04) if the parameter value is less than 04.

INT 1F : RESERVED

INT 49 : Conversion Table
This interrupt contains the address of a table used to translate non-keyboard scan-codes (scan codes from 56(Hex) to 69(Hex).) If Interrupt hex 48 detects a scan code between 56 - 69 (Hex) it translates it using the table pointed to by Interrupt Hex 49. The address that Interrupt Hex 49 points to can be changed by users to point to their own table if different translations are required.

INT 7A : Pointer To Dictionary
This routine includes pointers to the dictionary for Kana-to-Kanji conversion.

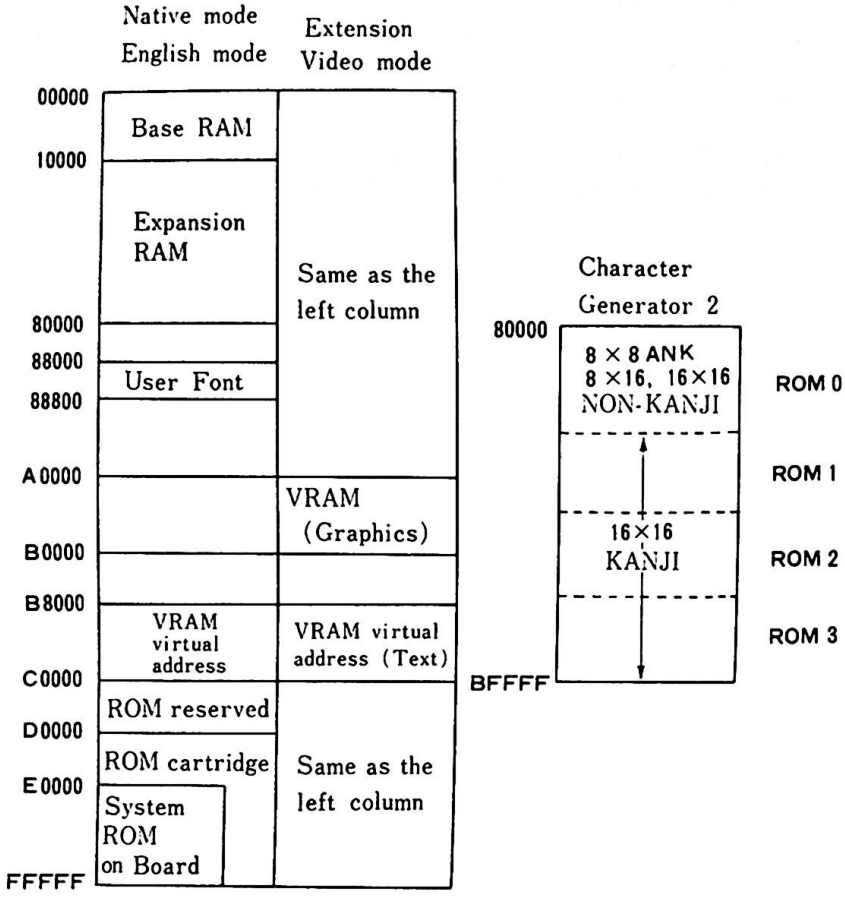
5.4. Keyboard Scan Codes

Scan codes are used to transfer keyboard data to the system unit. A different scan code is generated by pushing or releasing a key. Functions not represented by a single keystroke can be achieved by pressing two or more keys simultaneously. The BIOS keyboard routine converts them and returns one single character code. BIOS sends scan codes and converted character codes to the CPU.

The BIOS routines for processing keyboard data are INT2, INT48, INT49, INT16, INT9, INT78, INT79, and INT7A.

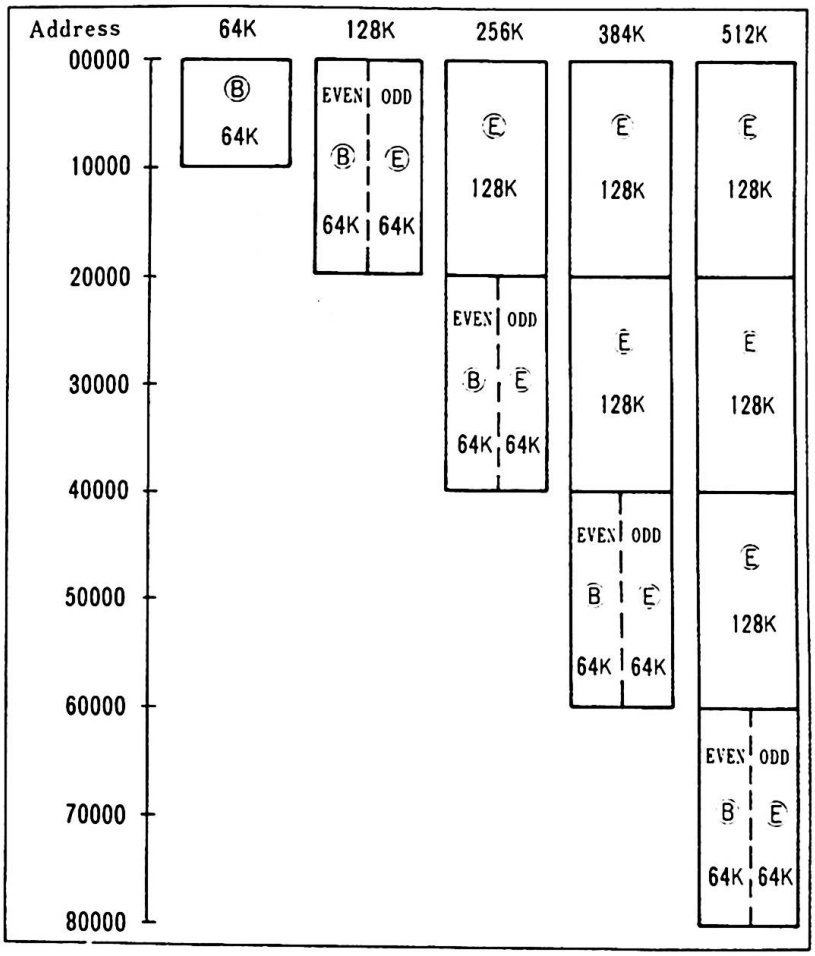
5.5. Memory Map

RAM for general use can be expanded to a 512KB maximum.
The memory space allocation is referred to in Chapter 2, " 2.2.6 Memory Space and I/O address Setting."



Remarks) The map of addresses 00000-80000 (Hex) changes depending on whether or not a 128KB RAM card is installed. Refer to Figure 5-5 and 5-6.

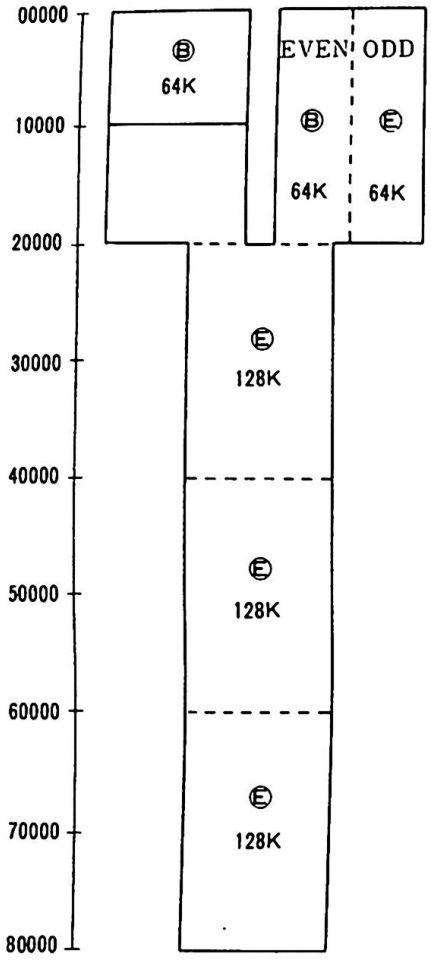
Figure 5-4 Memory Map



Meaning of abbreviations :
Ⓑ 64K : 64KB Base Memory
Ⓔ 64K : 64KB RAM Card
Ⓔ 128K : 128KB RAM Card
EVEN : Even address used
ODD : Odd address used

Figure 5—5 Memory Space (Native and Extension Video mode)

In English mode the base memory and 64K bytes of expansion memory are always located at addresses 00000-FFFFF(Hex).



Meaning of abbreviations:
Ⓑ 64K : 64KB Base Memory
Ⓔ 64K : 64KB RAM Card
Ⓔ 128K : 128KB RAM Card
EVEN : Even address used
ODD : Odd address used

Figure 5—6 Memory Space (English mode)

5.6. I/O Map

I/O addresses are initialized as follows. Addresses can be changed for some I/O. The relocation of I/O addresses is referred to in Chapter 2, " 2.2.6 Memory Space and I/O Address Setting".

Address	I/O Name
1X	Reserved
20	8259 PIC
21	8259 PIC
40	8253 Timer 1
41	8253 Timer 2
42	8253 Timer 3
43	8253 Mode
60	8255 Port A
61	8255 Port B
62	8255 Port C
63	8255 Control
A0	NMI Control
C0	76489A Sound generator
F2	Diskette controller
F4	Diskette status register
F5	Diskette data register
1FF	Gate Array-08
201	Joystick
278	Reserved
279	Reserved
27A	Reserved
2F8-2FF	8250 register addresses
32X	Reserved
378, 37C	Parallel interface
379, 37D	Parallel interface (status)
37A, 37E	Parallel interface (command)

Figure 5—7 I/O Addresses (1 of 2)

Address	Type of I/O
3D0, 2, 4, 6	CRTC address registers
3D1, 3, 5, 7	CRTC data registers
3D8	Reserved
3D9	Page register 2
3DA	Video Gate Array VP1, VP2
3DB	Clear light pen latch
3DC	Set light pen latch
3DD	Video gate array VP3
3DE	Light pen gate
3DF	Page register 1
3F8-3FF	Reserved

Figure 5—7 I/O Addresses (2 of 2)

6. Compatibility

This chapter describes points to keep in mind to maintain compatibility among the IBM 5510, IBM 5550 and PCjr systems. The differences among these systems are also described.

To have compatibility between the IBM 5510 and PCjr, it is necessary to operate the IBM 5510 in English mode. In order to be compatible with the IBM Multistation 5550, the 5510 must be operated in Extension Video mode. ROM cartridges are available for changing the mode of operation.

It is recommended that an application program use only the BIOS and DOS interrupt interfaces in order to achieve compatibility with the PCjr and IBM 5550, since absolute addresses vary among the three machines.

There are several factors to keep in mind to maintain compatibility. They are:

1. Unequal Configurations
2. Hardware Differences
3. Diskette Compatibility

Discussions of these topics follow.

6.1. Unequal Configurations

From the configuration/hardware point of view, there exist some functions which the IBM 5550 has, while the IBM 5510 does not have. Application programs which call for those functions might not work on the IBM 5510. The following are functions unique to the IBM 5550:

1. Hardware
   - 5550 ROM Function
   - DMA
   - 1024 x 768 dot display function
   - IBM 5550 unique key-tops
   - IBM 5550 unique printer function

2. BIOS
   - 1024 x 768 dot graphics mode
   - Hard Disk

3. DOS
   - User fonts of 63 or more characters
   - Hard disk support commands (SWITCH, BACKUP, RESTORE)

6.2. Hardware Differences

The IBM 5510, PCjr, and the IBM 5550 differ in hardware design and there might be instances where application program compatibility is not maintained because of the differences in the hardware. Figure 6-1 shows the hardware features of the IBM 5550 and PCjr compared with those of the IBM 5510.
Due to different shapes in connectors, some hardware might need modification before being used, but when the hardware is functionally compatible, it is classified as compatible.

Hardware/Function	Comparison	
	5550	PC-jr
User Memory	○ *	○ *
Compact Keyboard	× *	○ *
Full Keyboard	× *	N/A
Diskette Drive	○ *	× *
Printer Interface	× *	○
RS-232C Interface	○ *	○
Joystick Interface	N/A	○
Cassette Interface	N/A	○
Color Graphics	○ *	○
Light Pen	N/A	○
8253 Timer	○ *	○
Interrupt Controller	○ *	○
Beep Subsystem	○ *	○
Sound Generator	N/A	○ *
ROM Cartridge Interface	N/A	○

Remarks) ○ --- Compatible
X --- Incompatible
* --- With Conditions (Mentioned hereafter)
N/A - Not Applicable

Figure 6-1 Hardware Configuration Comparison

Hardware Differences from PCjr

1. User RAM
   PCjr always shares the user RAM with the video RAM, while the IBM 5510 has dedicated video RAM and only utilizes user RAM as video RAM in special cases. When a program uses video pages 0 through 7 on the 5510 system, user memory of 16 KB per page is required. Pages 8 through B (Hex) are areas for the dedicated video RAM and when these are accessed, user RAM is not used.

2. Diskette Drive
   - IBM 5510 : 80 tracks/side or
     40 tracks/side, 3.5" or 5.25"

   - PCjr : 40 tracks/side, 5.25"

   The size difference between the 3.5" and 5.25" diskettes makes physical compatibility impossible, but the diskette formats are compatible.

3. Keyboard
   The IBM 5510 keyboard has added keys for Kanji handling but other keys are compatible with the PCjr keyboard.

4. Sound Generator
   The IBM 5510 and the PCjr use compatible chips.

5. RS-232C Card
   This is provided as a standard feature on the PCjr, while on the IBM 5510, it is an optional feature. English mode applications using either COM1 or COM2 mode should have this optional feature to run on the IBM 5510.

Hardware Differences from IBM 5550

1. User RAM
   The IBM 5510 has the concept of multiple pages and can have up to 12 pages, while the IBM 5550 does not (one page is assumed). The memory maps for the two systems are different. However, if an application program accesses the hardware through BIOS calls, the memory map differences should be unimportant. Memory size should also be taken into account when considering compatibility because the minimum memory size of the IBM 5550 is larger than that of the 5510.

2. Diskette Operation
   The IBM 5510 does not have DMA capability, while the IBM 5550 has. The IBM 5510 uses a level 6 hardware interrupt. When diskette I/O takes place, the entire system is masked and other I/O devices are inactivated (operator keystrokes and RS-232C etc.).

3. Keyboard
   The IBM 5510 keyboard scan codes differ from those of the IBM IBM 5550. Compatibility is maintained by using interrupt type 16 (keystroke read). The keyboard operation, however, differs due to differences in the number and the functions of the keytops.

4. Video Display Function
   There is quite a high degree of compatibility between the IBM 5550 and Extension Video mode, but the font sizes of the two systems differ. The IBM 5550 has a wrap function for displaying screens, while the IBM 5510 does not.

5. Kanji Font
   The IBM 5510 requires 32 bytes/character, while the IBM 5550 requires 72 bytes/character.

6. Dictionary
   The IBM 5510 dictionary resides in ROM, while the IBM 5550 dictionary resides on a diskette and is loaded into RAM on demand.

7. RS-232C Interface
   The IBM 5550 supports up to 9600 BPS communications through use of DMA, while the IBM 5510 supports up to 4800 BPS communications through use of a level 3 interrupt. When diskette I/O takes place, interrupts other than level 6 are rejected and therefore, RS-232C interface I/O will not occur.

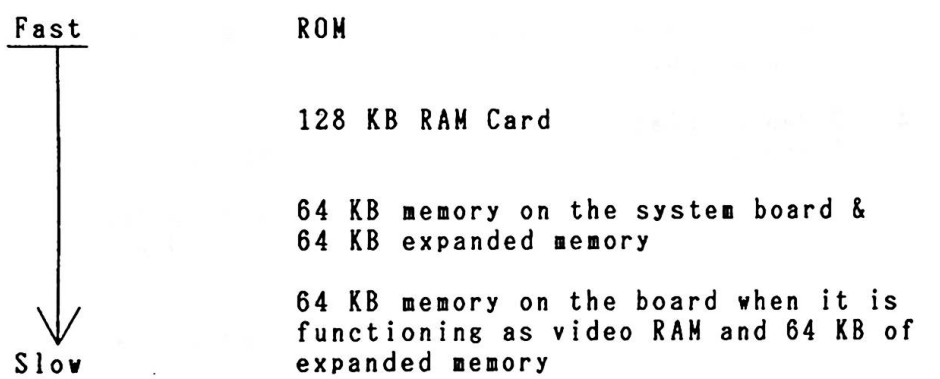
8. Timer (8253) Input
   The IBM 5510 clock is 1.19 MHz, while that of the IBM 5550 is 2.0 MHz.

9. Interrupt Controller
   Both systems use a 8259A PIC, but as the interrupt level assignments for the two systems differ, there might be instances where compatibility is not maintained.

10. Beep
    As there is a difference in the frequency of the Timer Input clock between the two systems, their beep sounds are slightly different.

11. Timing
    When an IBM 5550 application program is dependent on the processing speed or timing, it might not run on the IBM 5510. In this case, the program should be modified in accordance with the IBM 5510 specifications. In developing application programs, the following points should be taken into consideration:

    - The processing speed differs with the size and type of memory in which an application program is running.



    - The highest degree of application compatibility can be achieved by using a common high level language and accessing the system only through BIOS and DOS interrupts.

6.3. Diskette Compatibility

When a 5.25" diskette drive is installed in an Expansion Unit, there will be instances where the PCjr or the IBM 5550 is completely compatible with the IBM 5510. When only 3.5" diskette drives are provided, the physical shape and size of the diskettes makes it impossible to have compatibility. 3.5" and 5.25" diskettes are, however, the same in format and logically they are compatible. The following chart shows compatibility when a 5.25" diskette drive is installed in an IBM 5510 system:

			SYSTEM B			
			PC-jr	JX Operation Mode		IBM5550
				English	Native	Ext.Video	
SYSTEM A	PC-jr		⌟	⌟	⌟	×
	JX	English	⌟	⌟	*⌟	*⌟	*⌟
		Native	⌟	⌟*	⌟	⌟	⌟
		Ext.Video	⌟	⌟*	⌟	⌟	⌟
	IBM5550		×	⌟*	⌟	⌟	⌟

Remarks) A <----> B : Perfectly compatible (both systems can read/write)

--------> : Reading of a diskette which has been written by the other systems is possible. (The arrow points to the system which reads.)

* : Diskettes formatted with 40 tracks can be read. Ones with 80 tracks cannot.

X : No Compatibility

Figure 6-2 Diskette Compatibility

Appendix A. BIOS

Appendix A.

Absolute Address	PUBLIC Name	Module	Relative Address	Ref. Page
---------	(EQUATES & DATA AREA)	-	----	A- 2
F800:0000	--------	1	0000	3
:0030	--------	2	0000	15
:0076	VIDEO_IO	2	0046	19
:01CD	VIDEO_PARMS	2	019D	21
:2030	EQUIPMENT	3	0000	93
:203C	MEMORY_SIZE_DETERMINE	3	000C	93
:2048	DISKETTE_IO	3	0018	94
:248D	SEEK	3	045D	103
:2539	DISK_BASE	3	0509	104
:2544	DISK_INT	3	0514	104
:25B5	RS232_IO	3	0585	105
:2696	CASSETTE_IO	3	0666	107
:27AC	READ_HALF_BIT	3	077C	110
:2900	KBDNMI	4	0000	113
:2A27	EXTAB	4	0127	115
:2A50	KEY62_INT	4	0150	115
:2C9E	KEYBOARD_IO	4	039E	119
:2ED5	KB_INT	4	05D5	122
:385D	BUFFER_QUEING	4	0F5D	133
:3A00	----------	5	0000	135
:3AA0	PRINT_SCREEN	5	00A0	136
:3DC0	----------	6	0000	141
:3E20	PRINTER_IO	6	0060	141
:4E00	BOOT_STRAP	7	0000	163
:4F00	KKKFDM	8	0000	169
:6700	DDS	9	0000	214
:6708	TIME_OF_DAY	9	0008	214
:6752	READ_TIME	9	0052	214
:677D	KB_NOISE	9	007D	215
:679E	BAS_ENT	9	009E	215
:67B3	TIMER_INT	9	00B3	215
:6A00	POST	10	0000	218
:6A6B	RESET	10	006B	220
:6F99	D11	10	0599	229
:6FC0	DUMMY_RETURN	10	05C0	229
:79A1	PRT_HEX	10	0FA1	246
:79C0	BEEP	10	0FC0	247
:7FC0	VECTOR_TABLE	10	15C0	255
:7FFF	POST_END	10	15FF	256

Remark) Refer to the table shown above, when you find the character "E" for the operand in the BIOS listing.

```
                        ;--------------------------------------------------------------
                        ; <CAVEAT EMPTOR>:                                            :
                        ;                                                            :
                        ;       THE  BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH  :
                        ;       SOFTWARE INTERRUPTS ONLY.  ANY ADDRESSES PRESENT IN  :
                        ;       THE LISTINGS  ARE INCLUDED  ONLY FOR  COMPLETENESS,  :
                        ;       NOT FOR  REFERENCE.   APPLICATIONS WHICH  REFERENCE  :
                        ;       ABSOLUTE  ADDRESSES  WITHIN THIS  CODE  VIOLATE THE  :
                        ;       STRUCTURE AND DESIGN OF BIOS.                        :
                        ;--------------------------------------------------------------
                        ;--------------------------------------------------------------
                        ;       SYSTEM ID                                            :
                        ;--------------------------------------------------------------
= 00FD                  PJSYSID         EQU     0FDH            ; PCjr SYSTEM ID
= 00ED                  IXSYSID         EQU     0EDH            ; JX SYSTEM ID
                        ;--------------------------------------------------------------
                        ;       EQUATES                                              :
                        ;--------------------------------------------------------------
= 0010                  MFG_PORT        EQU     10H             ; MFG TESTER PORT
= 0060                  PORT_A          EQU     60H             ; 8255 PORT A ADDR
= 0038                  CPUREG          EQU     38H             ; MASK FOR CPU REG BITS
= 0007                  CRTREG          EQU     7               ; MASK FOR CRT REG BITS
= 0061                  PORT_B          EQU     61H             ; 8255 PORT B ADDR
= 0062                  PORT_C          EQU     62H             ; 8255 PORT C ADDR
= 0063                  CMD_PORT        EQU     63H
= 0089                  MODE_8255       EQU     10001001B
= 0060                  KBPORT          EQU     60H             ; KEYBOARD PORT
= 0020                  INTA00          EQU     20H             ; 8259 PORT
= 0021                  INTA01          EQU     21H             ; 8259 PORT
= 0020                  EOI             EQU     20H
= 0040                  TIMER           EQU     40H
= 0043                  TIM_CTL         EQU     43H             ; 8253 TIMER CONTROL PORT ADDR
= 0040                  TIMER0          EQU     40H             ; 8253 TIMER/CNTER 0 PORT ADDR
= 00A0                  NMI_PORT        EQU     0A0H            ; NMI CONTROL PORT
= 00C0                  SND_CTL         EQU     0C0H            ; SOUND GENERATOR CONTROL PORT
= 01FF                  S8STATUS        EQU     1FFH            ; 58 STATUS REGISTER
= 0080                  VRAM2IN         EQU     80H             ; 32K VRAM CARD IN
= 0040                  EROM6IN         EQU     40H             ; EXT. ROM 6 (F0000-) IN
= 0020                  EROM7IN         EQU     20H             ; EXT. ROM 7 (F8000-) IN
= 0201                  JOY_PORT        EQU     201H            ; JOYSTICK CONTROL PORT
= 03F8                  RS232_1_PORT    EQU     3F8H            ; RS232C-1 PORT
= 02F8                  RS232_2_PORT    EQU     2F8H            ; RS232C-2 PORT(ON BASE BOARD)
= 0378                  PARAL_PORT      EQU     378H            ; PARALLEL PORT
= 03D4                  CRT_CTL         EQU     3D4H            ; CRT CONTROLLER CTRL PORT
= 03DA                  VGA_CTL         EQU     3DAH            ; VIDEO GATE ARRAY CTRL PORT
= 03DD                  VGA_CTL_E       EQU     3DDH            ; VIDEO GATE ARRAY CTRL PORT
                                                                ; FOR EXTENSION MODE
= 03DF                  PAGREG          EQU     3DFH            ; CRT/CPU PAGE REGISTER
= 03D9                  PAGREG2         EQU     3D9H            ; CRT/CPU PAGE REGISTER 2
                        ;--------------------------------------------------------------
                        ;       INITIALIZE EQUATES                                   :
                        ;--------------------------------------------------------------
= 2000                  MINI            EQU     2000H           ; FUTURE USE
= 0703                  KKOFF           EQU     7*256+KANAKAN_OFF+INDICATOR_OFF ;
= 0000                  INIT_CODE       EQU     0000H           ; BOOT INITIA;L JUMP CODE
= 0003                  BOOT_LOCN1      EQU     0003H           ; IB AREA
= 0005                  BOOT_LOCN2      EQU     0005H           ; M  AREA
= FF20                  KKNINIT         EQU     0FF20H          ; KANA KANJI CONVERSION INITIALIZE
= 3A00                  DICT_ADDR       EQU     3A00H           ; DICTIONARY ADDRESS
                        ;--------------------------------------------------------------
                        ;       DISKETTE EQUATES                                     :
                        ;--------------------------------------------------------------
= 00F2                  NEC_CTL         EQU     0F2H            ; CONTROL PORT FOR THE DISKETTE
= 0080                  FDC_RESET       EQU     80H             ; RESETS THE NEC (FLOPPY DISK
                                                                ; CONTROLLER).  0 RESETS,
                                                                ; 1 RELEASES THE RESET
= 0020                  WD_ENABLE       EQU     20H             ; ENABLES WATCH DOG TIMER IN NEC
= 0040                  WD_STROBE       EQU     40H             ; STROBES WATCHDOG TIMER
= 0001                  DRIVE_ENABLE    EQU     01H             ; SELECTS AND ENABLES DRIVE
= 00F4                  NEC_STAT        EQU     0F4H            ; STATUS REGISTER FOR THE NEC
= 0020                  BUSY_BIT        EQU     20H             ; BIT = 0 AT END OF EXECUTION PHASE
= 0040                  DIO             EQU     40H             ; INDICATES DIRECTION OF TRANSFER
= 0080                  RQM             EQU     80H             ; REQUEST FOR MASTER
= 00F5                  NEC_DATA        EQU     0F5H            ; DATA PORT FOR THE NEC
= 000F                  DRV_SUPPORT     EQU     0FH             ; 4 BIT SUPPORT UP TO FOUR DRIVE
= 0003                  DRV_RANGE       EQU     03H             ; RANGE CHECK 0 TO 3
= 003F                  OFF_DBL_TRK     EQU     3FH             ; J PARAMETER : DOUBLE TRACK SUPPORT
                        ;--------------------------------------------------------------
                        ;       8088 INTERRUPT LOCATIONS                             :
                        ;--------------------------------------------------------------
= 0010                  INT_10          EQU     10H             ; VIDEO_IO
= 0011                  INT_11          EQU     11H             ; EQUIPMENT
= 0012                  INT_12          EQU     12H             ; MEMORY_SIZE_DETERMINE
= 0013                  INT_13          EQU     13H             ; DISKETTE_IO
= 0014                  INT_14          EQU     14H             ; RS232C_IO
= 0015                  INT_15          EQU     15H             ; CASSETTE_IO
= 0016                  INT_16          EQU     16H             ; KEYBOARD_IO
= 0017                  INT_17          EQU     17H             ; PRINTER_IO
= 0018                  INT_18          EQU     18H             ; RESIDENT BASIC
= 0019                  INT_19          EQU     19H             ; BOOT_STRAP
= 001A                  INT_1A          EQU     1AH             ; TIME_OF_DAY
= 001B                  INT_1B          EQU     1BH             ; DUMMY_RETURN
= 001C                  INT_1C          EQU     1CH             ; DUMMY_RETURN
= 001D                  INT_1D          EQU     1DH             ; VIDEO_PARMS
= 001E                  INT_1E          EQU     1EH             ; DISK_BASE
= 001F                  INT_1F          EQU     1FH             ; CRT_CHARH
= 0080                  INT_80          EQU     80H             ; DIAGNOSTICS

0000                    ABS0    SEGMENT AT 0
0008                            ORG     2*4
0008                    NMI_PTR         LABEL   WORD
000C                            ORG     3*4
000C                    INT3_PTR        LABEL   WORD
```

```
                                ORG     5*4
0014                    INT5_PTR        LABEL   WORD
0014                            ORG     8*4
0020                    INT_PTR         LABEL   DWORD
0020                            ORG     10H*4
0040                    VIDEO_INT       LABEL   WORD
0040                            ORG     1CH*4
0070                    INT1C_PTR       LABEL   WORD
0070                            ORG     1DH*4
0074                    PARM_PTR        LABEL   DWORD           ; POINTER TO VIDEO PARMS
0074                            ORG     18H*4
0060                    BASIC_PTR       LABEL   WORD            ; ENTRY POINT FOR CASSETTE BASIC
0060                            ORG     1EH*4                   ; INTERRUPT 1EH
0078                    DISK_POINTER    LABEL   DWORD
0078                            ORG     1FH*4                   ; LOCATION OF POINTER
007C                    EXT_PTR LABEL   DWORD                   ; POINTER TO EXTENSION
007C                            ORG     44H*4
0110                    CSET_PTR        LABEL   DWORD           ; POINTER TO DOT PATTERNS
0110                            ORG     48H*4
0120                    KEY62_PTR       LABEL   WORD            ; POINTER TO 62 KEY KEYBOARD CODE
0120                            ORG     49H*4
0124                    EXST            LABEL   WORD            ; POINTER TO EXT. SCAN TABLE
0124                            ORG     400H
0400                    DATA_AREA       LABEL   BYTE            ; ABSOLUTE LOCATION OF DATA SEGMENT
0400                    DATA_WORD       LABEL   WORD
0400                            ORG     7C00H
7C00                    BOOT_LOCN       LABEL   FAR
7C00                    ABS0    ENDS
                        ;--------------------------------------------------------------------
                        ;       STACK -- USED DURING INITIALIZATION ONLY                    :
                        ;--------------------------------------------------------------------

0000                    STACK   SEGMENT AT 30H
0000    80 [                    DW      128 DUP(?)
            ????
               ]

0100                    TOS     LABEL   WORD
0100                    STACK   ENDS
                        ;--------------------------------------------------------------------
                        ;       ROM BIOS DATA AREAS                                         :
                        ;--------------------------------------------------------------------

0000                    DATA    SEGMENT AT 40H
0000    04 [            RS232_BASE      DW      4 DUP(?)        ; ADDRESSES OF RS232 ADAPTERS
            ????
               ]

0008    04 [            PRINTER_BASE    DW      4 DUP(?)        ; ADDRESSES OF PRINTERS
            ????
               ]

0010  ????              EQUIP_FLAG      DW      ?               ; INSTALLED HARDWARE
0012  ??                KBD_ERR         DB      ?               ; COUNT OF KEYBOARD TRANSMIT ERRORS
0013  ????              MEMORY_SIZE     DW      ?               ; USABLE MEMORY SIZE IN K BYTES
0015  ????              TRUE_MEM        DW      ?               ; REAL MEMORY SIZE IN K BYTES
                        ;--------------------------------------------------------------------
                        ;       KEYBOARD DATA AREAS                                         :
                        ;--------------------------------------------------------------------

0017  ??                KB_FLAG         DB      ?
                                                ;
                        ;       SHIFT FLAG EQUATES WITHIN KB_FLAG
= 0040                  CAPS_STATE      EQU     40H             ; CAPS LOCK STATE HAS BEEN TOGGLED
= 0020                  NUM_STATE       EQU     20H             ; NUM LOCK STATE HAS BEEN TOGGLED
= 0008                  ALT_SHIFT       EQU     08H             ; ALTERNATE SHIFT KEY DEPRESSED
= 0004                  CTL_SHIFT       EQU     04H             ; CONTROL SHIFT KEY DEPRESSED
= 0002                  LEFT_SHIFT      EQU     02H             ; LEFT SHIFT KEY DEPRESSED
= 0001                  RIGHT_SHIFT     EQU     01H             ; RIGHT SHIFT KEY DEPRESSED
0018  ??                KB_FLAG_1       DB      ?               ; SECOND BYTE OF KEYBOARD STATUS
= 0080                  INS_SHIFT       EQU     80H             ; INSERT KEY IS DEPRESSED
= 0040                  CAPS_SHIFT      EQU     40H             ; CAPS LOCK KEY IS DEPRESSED
= 0020                  NUM_SHIFT       EQU     20H             ; NUM LOCK KEY IS DEPRESSED
= 0010                  SCROLL_SHIFT    EQU     10H             ; SCROLL LOCK KEY IS DEPRESSED
= 0008                  HOLD_STATE      EQU     08H             ; SUSPEND KEY HAS BEEN TOGGLED
= 0004                  CLICK_ON        EQU     04H             ; INDICATES THAT AUDIO FEEDBACK IS
                                                                ; ENABLED
= 0002                  CLICK_SEQUENCE  EQU     02H             ; OCURRNCE OF ALT-CTRL-CAPSLOCK HAS
                                                                ; OCCURED
0019  ??                ALT_INPUT       DB      ?               ; STORAGE FOR ALTERNATE KEYPAD
                                                                ; ENTRY
001A  ????              BUFFER_HEAD     DW      ?               ; POINTER TO HEAD OF KEYBOARD BUFF
001C  ????              BUFFER_TAIL     DW      ?               ; POINTER TO TAIL OF KEYBOARD BUFF
001E    10 [            KB_BUFFER       DW      16 DUP(?)       ; ROOM FOR 15 ENTRIES
            ????
               ]
                        ;       HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
= 0045                  NUM_KEY         EQU     69              ; SCAN CODE FOR NUMBER LOCK
= 0046                  SCROLL_KEY      EQU     70              ; SCROLL LOCK KEY
= 0038                  ALT_KEY         EQU     56              ; ALTERNATE SHIFT KEY SCAN CODE
= 001D                  CTL_KEY         EQU     29              ; SCAN CODE FOR CONTROL KEY
= 003A                  CAPS_KEY        EQU     58              ; SCAN CODE FOR SHIFT LOCK
= 002A                  LEFT_KEY        EQU     42              ; SCAN CODE FOR LEFT SHIFT
= 0036                  RIGHT_KEY       EQU     54              ; SCAN CODE FOR RIGHT SHIFT
= 0052                  INS_KEY         EQU     82              ; SCAN CODE FOR INSERT KEY
= 0053                  DEL_KEY         EQU     83              ; SCAN CODE FOR DELETE KEY
                        ;--------------------------------------------------------------------
                        ;       DISKETTE DATA AREAS                                         :
                        ;--------------------------------------------------------------------

003E  ??                SEEK_STATUS     DB      ?               ; DRIVE RECALIBRATION STATUS
                                                                ; BIT 0 = DRIVE NEEDS RECAL BEFORE
003F  ??                MOTOR_STATUS    DB      ?               ; NEXT SEEK IF BIT IS = 0
                                                                ; MOTOR STATUS
                                                                ; BIT 0 = DRIVE 0 IS CURRENTLY
0040  ??                MOTOR_COUNT     DB      ?               ; RUNNING
                                                                ; TIME OUT COUNTER FOR DRIVE
```

```
= 0025                          MOTOR_WAIT      EQU     37              ; TURN OFF
                                                                        ; 2 SECS OF COUNTS FOR MOTOR
0041  ??                       DISKETTE_STATUS DB      ?               ; TURN OFF
                                                                        ; RETURN CODE STATUS BYTE
= 0080                          TIME_OUT        EQU     80H             ; ATTACHMENT FAILED TO RESPOND
= 0040                          BAD_SEEK        EQU     40H             ; SEEK OPERATION FAILED
= 0020                          BAD_NEC         EQU     20H             ; NEC CONTROLLER HAS FAILED
= 0010                          BAD_CRC         EQU     10H             ; BAD CRC ON DISKETTE READ
= 0009                          DMA_BOUNDARY    EQU     09H             ; ATTEMPT TO DMA ACROSS 64K
                                                                        ; BOUNDARY
= 0008                          BAD_DMA         EQU     08H             ; DMA OVERRUN ON OPERATION
= 0004                          RECORD_NOT_FND  EQU     04H             ; REQUESTED SECTOR NOT FOUND
= 0003                          WRITE_PROTECT   EQU     03H             ; WRITE ATTEMPTED ON WRITE
                                                                        ; PROTECTED DISK
= 0002                          BAD_ADDR_MARK   EQU     02H             ; ADDRESS MARK NOT FOUND
= 0001                          BAD_CMD         EQU     01H             ; BAD COMMAND GIVEN TO DISKETTE I/O
0042      07 [                  NEC_STATUS      DB      7 DUP(?)        ; STATUS BYTES FROM NEC
                ??
                  ]

= 0020                          SEEK_END        EQU     20H
= 012C                          THRESHOLD       EQU     300             ; NUMBER OF TIMER-0 TICKS TILL
                                                                        ; ENABLE
= 00AF                          PARM0           EQU     0AFH            ; PARAMETER 0 IN THE DISK_PARM
                                                                        ; TABLE
= 0003                          PARM1           EQU     3               ; PARAMETER 1
= 0019                          PARM9           EQU     25              ; PARAMETER 9
= 0004                          PARM10          EQU     4               ; PARAMETER 10
                                ;---------------------------------------------------------------
                                ;       VIDEO DISPLAY DATA AREA                                 :
                                ;---------------------------------------------------------------
0049  ??                        CRT_MODE        DB      ?               ; CURRENT CRT MODE
004A  ????                      CRT_COLS        DW      ?               ; NUMBER OF COLUMNS ON SCREEN
004C  ????                      CRT_LEN         DW      ?               ; LENGTH OF REGEN IN BYTES
004E  ????                      CRT_START       DW      ?               ; STARTING ADDRESS IN REGEN BUFFER
0050      08 [                                  DW      8 DUP(?)        ; CURSOR_POSN IS DEFINDED LATER
                ????
                  ]

0060  ????                      CURSOR_MODE     DW      ?               ; CURRENT CURSOR MODE SETTING
0062  ??                        ACTIVE_PAGE     DB      ?               ; CURRENT PAGE BEING DISPLAYED
0063  ????                      ADDR_6845       DW      ?               ; BASE ADDRESS FOR ACTIVE DISPLAY
                                                                        ; CARD
0065  ??                        CRT_MODE_SET    DB      ?               ; CURRENT SETTING OF THE
                                                                        ; CRT MODE REGISTER
0066  ??                        CRT_PALLETTE    DB      ?               ; CURRENT PALETTE MASK SETTING
                                ;---------------------------------------------------------------
                                ;       CASSETTE DATA AREA                                      :
                                ;---------------------------------------------------------------
0067  ????                      EDGE_CNT        DW      ?               ; TIME COUNT AT DATA EDGE
0069  ????                      CRC_REG         DW      ?               ; CRC REGISTER
006B  ??                        LAST_VAL        DB      ?               ; LAST INPUT VALUE
                                ;---------------------------------------------------------------
                                ;       TIMER DATA AREA                                         :
                                ;---------------------------------------------------------------
006C  ????                      TIMER_LOW       DW      ?               ; LOW WORD OF TIMER COUNT
006E  ????                      TIMER_HIGH      DW      ?               ; HIGH WORD OF TIMER COUNT
0070  ??                        TIMER_OFL       DB      ?               ; TIMER HAS ROLLED OVER SINCE LAST
                                                                        ; READ
                                ;---------------------------------------------------------------
                                ;       SYSTEM DATA AREA                                        :
                                ;---------------------------------------------------------------
0071  ??                        BIOS_BREAK      DB      ?               ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072  ????                      RESET_FLAG      DW      ?               ; WORD=1234H IF KEYBOARD RESET
                                                                        ; UNDERWAY
                                ;---------------------------------------------------------------
                                ;       EXTRA DISKETTE DATA AREAS                               :
                                ;---------------------------------------------------------------
0074  ??                        TRACK0          DB      ?
0075  ??                        TRACK1          DB      ?
0076  ??                        TRACK2          DB      ?
0077  ??                                        DB      ?
                                ;---------------------------------------------------------------
                                ;       PRINTER AND RS232 TIME-OUT VARIABLES                    :
                                ;---------------------------------------------------------------
0078      04 [                  PRINT_TIM_OUT   DB      4 DUP(?)
                ??
                  ]

007C      04 [                  RS232_TIM_OUT   DB      4 DUP(?)
                ??
                  ]


                                ;---------------------------------------------------------------
                                ;       ADDITIONAL KEYBOARD DATA AREA                           :
                                ;---------------------------------------------------------------
0080  ????                      BUFFER_START    DW      ?
0082  ????                      BUFFER_END      DW      ?
0084  ??                        INTR_FLAG       DB      ?               ; FLAG TO INDICATE AN INTERRUPT
                                                                        ; HAPPENED
                                ;---------------------------------------------------------------
                                ;       62 KEY KEYBOARD DATA AREA                               :
                                ;---------------------------------------------------------------
0085  ??                        CUR_CHAR        DB      ?               ; CURRENT CHARACTER FOR TYPAMATIC
0086  ??                        VAR_DELAY       DB      ?               ; DETERMINES WHEN INITIAL DELAY IS
                                                                        ; OVER
= 000F                          DELAY_RATE      EQU     0FH             ; INCREASES INITIAL DELAY
0087  ??                        CUR_FUNC        DB      ?               ; CURRENT FUNCTION
0088  ??                        KB_FLAG_2       DB      ?               ; 3RD BYTE OF KEYBOARD FLAGS
= 0004                          RANGE           EQU     4               ; NUMBER OF POSITIONS TO SHIFT
                                                                        ; DISPLAY
                                ;---------------------------------------------------------------
                                ;       BIT ASSIGNMETS FOR KB_FLAG_2                            :
                                ;---------------------------------------------------------------
```

Appendix A.

```
= 0080                  FN_FLAG         EQU     80H
= 0040                  FN_BREAK        EQU     40H
= 0020                  FN_PENDING      EQU     20H
= 0010                  FN_LOCK         EQU     10H
= 0008                  TYPE_OFF        EQU     08H
= 0004                  HALF_RATE       EQU     04H
= 0002                  INIT_DELAY      EQU     02H
= 0001                  PUTCHAR         EQU     01H
0089  ??                HORZ_POS        DB      ?               ; CURRENT VALUE OF HORIZONTAL
                                                                ; START PARM
008A  ??                PAGDAT          DB      ?               ; IMAGE OF DATA WRITTEN TO PAGREG
                        ;-----------------------------------------------------------------
                        ;       KANJI DOS WORK AREA                                      :
                        ;-----------------------------------------------------------------
                                ORG     0F0H
00F0                    GAIJI_ADDR      DW      2  DUP(0)       ; RESERVED FOR KANJI DOS
00F0    02 [
            0000
                    ]
                        ;-----------------------------------------------------------------
                        ;       CONTROL FLAGS                                            :
                        ;-----------------------------------------------------------------
                                ORG     300H            ;
0300                    J_EXT_STATUS    DW      ?               ; EXTENSION CARTRIDGE CHARACTERISTICS
0300  ????              JEQUIP_FLAG     DW      ?               ; EQUIPMENT FLAG EXTENSION
0302  ????              ;-----------------------------------------------------------------
                        ;       JAPAN KEYBOARD DATA AREA                                 :
                        ;-----------------------------------------------------------------
                        ;       KEYBOARD_BUFFER
= 006B                  KANJI_KEY       EQU     107             ; SCAN CODE FOR KANJI KEY
= 006C                  MUHEN_KEY       EQU     108             ; SCAN CODE FOR MUHENKAN KEY
= 006D                  HENKAN_KEY      EQU     109             ; SCAN CODE FOR HENKAN KEY
= 003A                  ALPHA_KEY       EQU      58             ; SCAN CODE FOR ALPHA STATE KEY
= 006E                  KATAKANA_KEY    EQU     110             ; SCAN CODE FOR KATAKANA STATE KEY
= 006F                  HIRAGANA_KEY    EQU     111             ; SCAN CODE FOR HIRAGANA STATE KEY
                                                                ;
                        ;       BIT ASSIGNMETS FOR JKB_FLAG
= 0004                  HIRAGANA_STATE  EQU     04H             ; HIRAGANA SHIFT ACTIVE
= 0002                  KATAKANA_STATE  EQU     02H             ; KATAKANA SHIFT ACTIVE
= 0001                  ZENKAKU_STATE   EQU     01H             ; ZENKAKU MODE
= 0006                  NOT_ALPHA_STATE EQU     06H             ;
= 00F9                  ALPHA_STATE     EQU     0F9H            ; ALPHA_STATE MASK
= 0005                  ZENKAKU_CHAR    EQU     05H             ; ZENKAKU CHARACTER
                                                                ;
                        ;       BIT ASSIGNMETS FOR JKB_FLAG_1
= 0010                  HANKAKU_SHIFT   EQU     10H             ; HANKAKU  SHIFT KEY DEPRESSED
= 0008                  ZENKAKU_SHIFT   EQU     08H             ; ZENKAKU  SHIFT KEY DEPRESSED
= 0004                  HIRAGANA_SHIFT  EQU     04H             ; HIRAGANA SHIFT KEY DEPRESSED
= 0002                  KATAKANA_SHIFT  EQU     02H             ; KATAKANA SHIFT KEY DEPRESSED
= 0001                  ALPHA_SHIFT     EQU     01H             ; ALPHA    SHIFT KEY DEPRESSED
                                                                ;
                        ;       BIT ASSIGNMETS FOR JKB_FLAG_2
= 0080                  KANJI_SHIFT     EQU     80H             ; KANJI    SHIFT KEY DEPRESSED
= 0040                  KNUM_SHIFT      EQU     40H             ; KNUMBER  SHIFT KEY DEPRESSED
= 0020                  MUHEN_SHIFT     EQU     20H             ; MUHEN    SHIFT KEY DEPRESSED
= 0010                  HENKAN_SHIFT    EQU     10H             ; HEN      SHIFT KEY DEPRESSED
= 0004                  NMI_FLG         EQU     04H             ; NMI ACTIVE FLAG
= 0002                  INDICATOR_OFF   EQU     02H             ; INDICATOR ON/OFF SWITCH
= 0001                  KANAKAN_OFF     EQU     01H             ; KANAKAN ON/OFF SWITCH
0304    19 [            KB_BUFFER_J     DW      25 DUP(?)       ; ROOM FOR 24 ENTRIES
            ????
                    ]
0336  ??                JKB_FLAG        DB      ?               ; 4TH BYTE OF KEYBOARD FLAGS
0337  ??                JKB_FLAG_1      DB      ?               ; 5TH BYTE OF KEYBOARD FLAGS
0338  ??                JKB_FLAG_2      DB      ?               ; 6TH BYTE OF KEYBOARD FLAGS
0339  ????              FIRST_PTR       DW      ?               ; USED BY BUFFER QUEING (INT 41H)
                        ;-----------------------------------------------------------------
                        ;       NEW VIDEO DATA AREA                                      :
                        ;-----------------------------------------------------------------
= B800                  REGEN_START     EQU     0B800H  ; SEGMENT ADDRESS OF REGEN
= 0000                  DEBUG           EQU     0       ; DEBUG FLAG
= 0010                  NO_ACT_PAGE     EQU     16      ; NUMBER OF ACTIVE PAGE
033B  ??                CRT_MODE2       DB      ?       ; CURRENT CRT MODE OF VIDEO PROCESSOR 2
033C  ??                PAGDAT2         DB      ?       ; IMAGE OF DATA WRITTEN TO PAGREG 2
033D  ??                CRT_MODE_SET2   DB      ?
033E  ????              K_1ST_CHAR      DW      ?       ; 1ST CHAR. CODE/ATTR. AT WRITE A/C IN KANA-KAN
0340  ????              W_1ST_CHAR      DW      ?       ; 1ST CHAR. CODE/ATTR. AT WRITE A/C
0342  ????              TTY_1ST_CHAR    DW      ?       ; 1ST CHAR AT WRITE TTY
0344  ????              K_TTY_1ST_CHAR  DW      ?       ; 1ST CHAR AT WRITE RRY IN KANA-KAN
0346  ??                CRT_ROWS        DB      ?       ; CURRENT CRT COLUMN SIZE
0347  ??                SUPIPCR         DB      ?       ; LAST VALUE OF SUPERIMPOSE CONTROL REGISTER
0348  ??                AC_PRESENT      DB      ?       ; ALTERNATE CURSOR PRESENT
0349  ??                GC_PRESENT      DB      ?       ; GRAPHICS  CURSOR PRESENT
034A  ????              ALT_CURSOR_POSN DW      ?       ; ALTERNATE CURSOR POSITION
034C  ??                CPU_PAGE        DB      ?       ; ACTIVE CPU PAGE
034D  ??                CRT_PAGE        DB      ?       ; ACTIVE CRT PAGE
034E  ????              GCURSOR_MODE    DW      ?       ; GRAPHICS CURSOR MODE
0350  ????              ACURSOR_MODE    DW      ?       ; ALTERNATE CURSOR MODE
0352  ??                PALETTE_MASK    DB      ?       ; VALUE OF PALETTE MASK REGISTER
0353  ??                KJROM_STAT      DB      ?       ; KANJI ROM STATUS (0:OFF, 1:ON)
0354  ??                VG_STAT         DB      ?       ; VIDEO GENERATER STATUS (0:VG2, 1:VG1)
0355  ????              IEP_CTRL        DW      ?       ; INTERRUPT ENABLE PROHIBIT FLAG
0357  ??                VSTACKL         DB      ?       ; USED LEVEL OF VIDEO STACK
0358  ????              SS_SAVE         DW      ?       ; SS SAVE AREA
035A  ????              SP_SAVE         DW      ?       ; SP SAVE AREA
035C    10 [            CURSOR_POSN     DW      16 DUP (?);
            ????
                    ]
                        ;
03F0                            ORG     3F0H            ;
03F0    10 [            BASIC_WORK      DB      16 DUP (?); TEMPORARY RESERVED FOR BASIC
            ??
                    ]
```

```
03FB                            ORG     3FBH
03FB  ??                        VIO_PROCESS     DB      ?       ; VIDEO I/O IS PROCESSING
03FC  ??                        SUPRESS_PAL     DB      ?       ; SUPRESS PALETTE SET DURING MODE SET
03FD                            DATA    ENDS
                                ;--------------------------------------------------------------------
                                ;       EXTRA DATA AREA                                             :
                                ;--------------------------------------------------------------------
0000                            XXDATA  SEGMENT AT 50H
0000  ??                        STATUS_BYTE     DB      ?
                                                                        ;
                                ;       THE FOLLOWING AREA IS USED ONLY DURING DIAGNOSTICS
0001  ??                        DCP_MENU_PAGE   DB      ?               ; TO CURRENT PAGE FOR DIAG. MENU
0002  ????                      DCP_ROW_COL     DW      ?               ; CURRENT ROW/COLUMN COORDINATES
                                                                        ; FOR DIAG MENU
0004  ??                        WRAP_FLAG       DB      ?               ; INTERNAL/EXTERNAL 8250 WRAP
                                                                        ; INDICATOR
0005  ??                        MFG_TST         DB      ?               ; INITIALIZATION FLAG
0006  ????                      MEM_TOT         DW      ?               ; WORD EQUIV. TO HIGHEST SEGMENT IN
                                                                        ; MEMORY
0008  ????                      MEM_DONES       DW      ?               ; CURRENT SEGMENT VALUE FOR
                                                                        ; BACKGROUND MEM TEST
000A  ????                      MEM_DONEO       DW      ?               ; CURRENT OFFSET VALUE FOR
                                                                        ; BACKGROUND MEM TEST
000C  ????                      INT1CO          DW      ?               ; SAVE AREA FOR INTERRUPT 1C
                                                                        ; ROUTINE
000E  ????                      INT1CS          DW      ?
0010  ??                        MENU_UP         DB      ?               ; FLAG TO INDICATE WHETHER MENU IS
                                                                        ; ON SCREEN (FF=YES, 0=NO)
0011  ??                        DONE128         DB      ?               ; COUNTER TO KEEP TRACK OF 128 BYTE
                                                                        ; BLOCKS TESTED BY BGMEM
0012  ????                      KBDONE          DW      ?               ; TOTAL K OF MEMORY THAT HAS BEEM
                                                                        ; TESTED BY BACKGROUND MEM TEST
                                ;--------------------------------------------------------------------
                                ;       POST DATA AREA                                              :
                                ;--------------------------------------------------------------------
0014  ????                      IO_ROM_INIT     DW      ?               ; POINTR TO OPTIONAL I/O ROM INIT
                                                                        ; ROUTINE
0016  ????                      IO_ROM_SEG      DW      ?               ; POINTER TO IO ROM SEGMENT
0018  ??                        POST_ERR        DB      ?               ; FLAG TO INDICATE ERROR OCCURRED
                                                                        ; DURING POST
0019     09 [                   MODEM_BUFFER    DB      9 DUP(?)        ; MODEM RESPONSE BUFFER
            ??
          ]
                                                                        ; (MAX 9 CHARS)
0022  ????                      MFG_RTN         DW      ?               ; POINTER TO MFG. OUTPUT ROUTINE
0024  ????                                      DW      ?
                                ;--------------------------------------------------------------------
                                ;       SERIAL PRINTER DATA                                         :
                                ;--------------------------------------------------------------------
0026  ????                      SP_FLAG         DW      ?
0028  ??                        SP_CHAR         DB      ?
                                                                        ; THE FOLLOWING SIX ENTRIES ARE
                                                                        ; DATA PERTAINING TO NEW STICK
0029  ????                      NEW_STICK_DATA  DW      ?               ; RIGHT STICK DELAY
002B  ????                                    DW      ?               ; RIGHT BUTTON A DELAY
002D  ????                                    DW      ?               ; RIGHT BUTTON B DELAY
002F  ????                                    DW      ?               ; LEFT STICK DELAY
0031  ????                                    DW      ?               ; LEFT BUTTON A DELAY
0033  ????                                    DW      ?               ; LEFT BUTTON B DELAY
0035  ????                                    DW      ?               ; RIGHT STICK LOCATION
0037  ????                                    DW      ?               ; UNUSED
0039  ????                                    DW      ?               ; UNUSED
003B  ????                                    DW      ?               ; LEFT STICK POSTITON
                                                                        ;
003D                            XXDATA  ENDS
                                ;--------------------------------------------------------------------
                                ;       BIOS LOCAL STACK AREA                                       :
                                ;--------------------------------------------------------------------
0000                            VSTACK  SEGMENT AT 1A0H
0000  0600 [                                  DB      1536    DUP(?)
            ??
          ]
0600                            VSTACK_TOP      LABEL   WORD
0600                            VSTACK  ENDS
```

```
                        PAGE ,121
0000                    DSEGM       SEGMENT AT      120H
                        ;-------------------------------------------------------------------------;
                        ;                                                                         ;
                        ;       DATA SEGMENT FOR INT 17 , INT 5                                   ;
                        ;-------------------------------------------------------------------------;
                        ;       ### PORT ASSIGN ###
= 0378                  PR_DATA_PORT    EQU     0378H                   ;PRINTER I/O PORT
= 0379                  PR_STATUS_PORT  EQU     0379H
= 037A                  PR_CMD_PORT     EQU     037AH
                        ;       ### STATUS ###
= 0080                  PR_BUSY         EQU     80H                     ;PRINTER STATUS
= 0040                  PR_PE           EQU     20H
= 0020                  PR_SELECT       EQU     10H
= 0010                  PR_ERROR        EQU     08H
= 0008                  PR_ATTACH       EQU     04H
= 0004                  PR_TIMEOUT      EQU     01H
= 0001                  ;       ### PRINTER ID ###
                        NO_PRINTER      EQU     0                       ;PRINTER TYPE-I
= 0000                  PRT1            EQU     1                       ;PRINTER TYPE-II
= 0001                  PRT2            EQU     2
= 0002                  ;       ### PRINT MODE ###
                        EVEN_PR_FLG     EQU     1                       ;CHARACTER LINE MODE
= 0001                  LOW_PR_FLG      EQU     2                       ;GRAPHIC IMAGE LINE MODE
= 0002                  ;       ### CHARACTER SIZE ###
                        NOR             EQU     0                       ;NORMAL SIZE
= 0000                  BAI             EQU     1                       ;DOUBLE SIZE
= 0001                  ;       ### FLAG 1 ###
                        TWO_BYTE_FLG    EQU     80H                     ;TWO BYTES CHARACTER CODE INDICATION
= 0080                  PRT_CHK_FLG     EQU     10H                     ;PRINTER-ID CHECK FLAG
= 0010                  F_FLG           EQU     04H                     ;ESC-F PROCESS FLAG
= 0004                  X_FLG           EQU     02H                     ;ESC-X PROCESS FLAG
= 0002                  ESC_FLG         EQU     01H                     ;ESC PROCESS FLAG
= 0001                  ;       ### FLAG 2 ###
                        IGN_FLG         EQU     40H                     ;COMMAND IGNORE INDICATION FLAG
= 0040                  X9_FLG          EQU     20H                     ;ESC-X9 PROCESS FLAG
= 0020                  X6_FLG          EQU     10H                     ;ESC-X6 PROCESS FLAG
= 0010                  X5_FLG          EQU     08H                     ;ESC-X5 PROCESS FLAG
= 0008                  X3_FLG          EQU     04H                     ;ESC-X3 PROCESS FLAG
= 0004                  X2_FLG          EQU     02H                     ;ESC-X2 PROCESS FLAG
= 0002                  X1_FLG          EQU     01H                     ;ESC-X1 PROCESS FLAG
= 0001                  ;       ### FLAG 3 ###
                        CHG_LPI_FLG     EQU     80H                     ;TEMPORARY LPI CHANGE INDICATION
= 0080                  BAI_FUL_FLG     EQU     20H                     ;BAIKAKU SLICE FULL INDICATION
= 0020                  SL_FUL_FLG      EQU     10H                     ;SLICE FULL INDICATION
= 0010                  ;       ### MAX SLICE VALUE ###
                        MAX             EQU     1120                    ;MAX NUMBER OF SLICES
= 0460                  ;-------------------------------------------------------------------------;
                        ;                                                                         ;
                        ;       WORK AREA FOR INT 17                                              ;
                        ;-------------------------------------------------------------------------;
0000    01 [            PRINTER_ID      DW      1       DUP     (1)     ;1:PTR1 2:PTR2
            0001 ]
0002    01 [            RETURN_CODE     DB      1       DUP     ('Z')   ;PRINTER STATUS
            5A ]
0003    01 [            CPI             DB      1       DUP     ('Z')   ;CHAR / INCH
            5A ]
0004    01 [            LPI             DB      1       DUP     ('Z')   ;LINE / INCH
            5A ]
0005    01 [            CPL             DW      1       DUP     ('ZZ')  ;CHAR / LINE
            5A5A ]
0007    01 [            LPP             DW      1       DUP     ('ZZ')  ;LINE / PAGE
            5A5A ]
0009    01 [            PRINT_MODE      DB      1       DUP     ('Z')   ;0:ELSE 1:CHAR 2:IMAGE
            5A ]
000A    01 [            CHAR_TYPE       DW      1       DUP     ('Z')   ;1:LARGE 2:SMALL
            005A ]
000C    01 [            SIZE_ESC        DB      1       DUP     ('Z')   ;0:NORMAL 1:DOUBLE (ESC+[,])
            5A ]
000D    01 [            SIZE_AH         DB      1       DUP     ('Z')   ;0:NORMAL 1:DOUBLE (AH=0/5)
            5A ]
000E    01 [            CCP             DW      1       DUP     ('ZZ')  ;CURRENT CHAR  POSITION
            5A5A ]
0010    01 [            CSP             DW      1       DUP     ('ZZ')  ;CURRENT SLICE POSITION
            5A5A ]
0012    01 [            CSPMAX          DW      1       DUP     ('ZZ')  ;CURRENT SLICE MAXIMUM POSITION
            5A5A ]
```

```
0014    01 [                SPSAVE          DW      1       DUP     ('ZZ')  ;SP SAVE AREA
                5A5A
                    ]
0016    01 [                PR_TIME1        DW      1       DUP     ('ZZ')  ;BEEP TIMER 1
                5A5A
                    ]
0018    01 [                PR_TIME2        DW      1       DUP     ('ZZ')  ;BEEP TIMER 2
                5A5A
                    ]
001A    01 [                PR_TIME3        DW      1       DUP     ('ZZ')  ;WAIT TIMER
                5A5A
                    ]
001C    01 [                CPIMASK         DW      1       DUP     ('ZZ')  ;13.5 CPI ADJUST MASK
                5A5A
                    ]
001E    01 [                STATUS17        DB      1       DUP     ('Z')   ;0:NOT PROGRESS 1:IN PROGRESS
                5A
                    ]
001F    01 [                SYSTEM_ID       DB      1       DUP     ('Z')   ;00:NATIVE 20:EXTENSION
                5A
                    ]
0020    01 [                CODEN           DW      1       DUP     ('ZZ')  ;THE NUMBER OF CODE
                5A5A
                    ]
0022    01 [                N1              DB      1       DUP     ('Z')   ;ESCAPE PARM N1
                5A
                    ]
0023    01 [                N2              DB      1       DUP     ('Z')   ;ESCAPE PARM N2
                5A
                    ]
0024    01 [                N1N2            DW      1       DUP     ('ZZ')  ;ESCAPE PARM N1N2
                5A5A
                    ]
0026    01 [                LF_CT           DB      1       DUP     ('Z')   ;LINE FEED COUNT
                5A
                    ]
0027    01 [                SP_VALUE        DB      1       DUP     ('Z')   ;SPACING VALUE
                5A
                    ]
0028    01 [                IM_MOD          DB      1       DUP     ('Z')   ;IMAGE MODE
                5A
                    ]
0029    01 [                FS_N            DW      1       DUP     ('ZZ')  ;FONT SLICE NUMBER
                5A5A
                    ]
002B    01 [                BYTE_ONE        DB      1       DUP     ('Z')   ;FIRST BYTE OF TWO-BYTED CODE
                5A
                    ]
002C    01 [                FLG1            DB      1       DUP     ('Z')   ;
                5A
                    ]
002D    01 [                FLG2            DB      1       DUP     ('Z')   ;
                5A
                    ]
002E    01 [                FLG3            DB      1       DUP     ('Z')   ;
                5A
                    ]
002F    01 [                CSIZE           DW      1       DUP     ('ZZ')  ;CHARACTER SIZE WORK
                5A5A
                    ]
0031    0F [                                DB      15      DUP     ('Z')   ; - RESERVED -
                5A
                    ]
0040    18 [                GVALUE          DB      24      DUP     ('Z')   ;GRID LINE CONTROL VALUE
                5A
                    ]
= 0001                      V_SOLID         EQU     1                       ; VERTICAL SOLID LINE
= 0004                      V_DASH          EQU     4                       ; VERTICAL DASHED LINE
= 0007                      V_UNDER         EQU     7                       ; UNDERSCORE IMAGE
= 0008                      V_SIZE          EQU     8                       ; VERTICAL SKIP SIZE
= 0012                      H_KEY           EQU     18                      ; KEY OF HORIZONTAL VALUE
= 0013                      H_DASH          EQU     19                      ; UPPER VERTICAL SOLID LINE
= 0016                      H_SIZE          EQU     22                      ; HORIZONTAL SKIP SIZE
= 0017                      H_SPACE         EQU     23                      ; SPACE BETWEEN CHARACTERS
0058    02 [                TVALUE          DW      2       DUP     ('ZZ')  ;CHAR TYPE CONTROL VALUE
                5A5A
                    ]
= 0000                      UPPER           EQU     0                       ; UPPER IMAGE FLAG
= 0002                      LOWER           EQU     2                       ; LOWER IMAGE FLAG
```

```
                                                DB      4       DUP     ('Z')   ; - RESERVED -
005C    04 [
            5A
                ]
                                ;-----------------------------------------------------------------;
                                ;       WORK AREA FOR INT 5                                       ;
                                ;-----------------------------------------------------------------;
0060    01 [                    HSIZE           DW      1       DUP     ('ZZ')  ;HORIZONTAL DOT SIZE
            5A5A
                ]
0062    01 [                    VSIZE           DW      1       DUP     ('ZZ')  ;VERTICAL   DOT SIZE
            5A5A
                ]
0064    01 [                    HRATIO          DW      1       DUP     ('ZZ')  ;HORIZONTAL ENLARGE RATIO
            5A5A
                ]
0066    01 [                    VRATIO          DW      1       DUP     ('ZZ')  ;VERTICAL   ENLARGE RATIO
            5A5A
                ]
0068    01 [                    PAGENO          DB      1       DUP     ('Z')   ;ACTIVE PAGE
            5A
                ]
0069    01 [                    SPSAVE5         DW      1       DUP     ('ZZ')  ;SP SAVE AREA FOR INT 5
            5A5A
                ]
006B    01 [                    COLORTB         DW      1       DUP     ('ZZ')  ;COLOR TABLE OFFSET
            5A5A
                ]
006D    01 [                    VIDEO_MODE      DB      1       DUP     ('Z')   ;>0:GRAPHIC <0:CHARACTER
            5A
                ]
006E    01 [                    SLICE           DW      1       DUP     ('ZZ')  ;SLICE SIZE (16 OR 24)
            5A5A
                ]
0070    01 [                    SHIFT           DB      1       DUP     ('Z')   ;SHIFT VALUE
            5A
                ]
0071    0F [                                    DB      15      DUP     ('Z')   ; - RESERVED -
            5A
                ]
0080    18 [                    DOT             DB      24      DUP     ('Z')   ;COLOR DOT PATTERN AREA
            5A
                ]
0098    08 [                                    DB      8       DUP     ('Z')   ; - RESERVED -
            5A
                ]
                                ;-----------------------------------------------------------------;
                                ;       BUFFER                                                    ;
                                ;-----------------------------------------------------------------;
00A0    F0 [                    CODE_BUFFER     DB      240     DUP     ('Z')   ;CODE       SAVE AREA
            5A
                ]
0190    F0 [                    ATTR_BUFFER     DB      240     DUP     ('Z')   ;ATTRIBUTE SAVE AREA
            5A
                ]
0280    20 [                    FONT            DB      32      DUP     ('Z')   ;FONT IMAGE WORK AREA
            5A
                ]
02A0    36 [                    GRID            DB      54      DUP     ('Z')   ;GRID LINE IMAGE AREA
            5A
                ]
02D6    FE [                                    DB      254     DUP     ('Z')   ; - RESERVED -
            5A
                ]
03D4    A0 [                    CODE_INT5       DB      160     DUP     ('Z')   ;CODE       AREA FOR INT 5
            5A
                ]
0474    A0 [                    ATTR_INT5       DB      160     DUP     ('Z')   ;ATTRIBUTE AREA FOR INT 5
            5A
                ]
00A0                                            ORG     OFFSET  CODE_BUFFER
00A0  0474 [                    IMAGE_BUFFER    DB      1140    DUP     ('Z')   ;EVEN DOT  SAVE AREA
            5A
                ]
0514                            DSEGM           ENDS
                                ;-----------------------------------------------------------------;
                                ;       EXTRA SEGMENT                                             ;
                                ;-----------------------------------------------------------------;
0000                            XXDATA  SEGMENT AT      50H
0000  00                        STATUS_BYTE     DB      0
0001                            XXDATA          ENDS
```

This page intentionally left blank.

```
XXXXXXXXXXXXX
XXXXXXXXXXXXX
X           X
X MODULE 1  X
X           X
XXXXXXXXXXXXX
XXXXXXXXXXXXX
```

```
0000


= 0000
0000   35 36 30 31 4A 46
       42 20 43 4F 50 52
       2E 20 49 42 4D 20
       31 39 38 34
0016
0030
0030
```

```
;----------------------------------------
;       ROM RESIDENT CODE               :
;----------------------------------------
CODE    SEGMENT PUBLIC
.LIST
        ASSUME  CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK
        BEGIN = $
        DB      '5601JFB COPR. IBM 1984'    ; COPYRIGHT NOTICE


        ORG     $                                   ; REAL SIZE
        ORG     BEGIN+30H
CODE    ENDS
```

```
;--- INT 10 --------------------------------------------------------
;
;       VIDEO_IO
;       --------
;
;       THESE ROUTINES PROVIDE THE CRT INTERFACE
;       THE FOLLOWING FUNCTIONS ARE PROVIDED:
;
;       (AH)=0  SET MODE (AL) CONTAINS MODE VALUE
;               (AL)= 0  40X25 COLOR ANK
;               (AL)= 1  40X25 COLOR ANK
;               (AL)= 2  80X25 COLOR ANK
;               (AL)= 3  80X25 COLOR ANK
;               GRAPHICS MODES
;               (AL)= 4  320X200  4 COLOR (40X25 ANK)
;               (AL)= 5  320X200  4 COLOR (40X25 ANK)
;               (AL)= 6  640X200  2 COLOR (80X25 ANK)
;               (AL)= 7  NOT VALID
;        **** EXTENDED MODES ***
;               (AL)= 8  160X200 16 COLOR (20X25 ANK)
;               (AL)= 9  320X200 16 COLOR (40X25 ANK)
;               (AL)= A  640X200  4 COLOR (80X25 ANK)
;               (AL)= B  NOT VALID
;               (AL)= C  NOT VALID
;               (AL)= D  NOT VALID
;               (AL)= E  NOT VALID
;               (AL)= F  NOT VALID
;        **** KANJI EXTENSION MODE ****
;               (AL)=10  40X11 COLOR
;               (AL)=11  40X11 COLOR
;               (AL)=12  80X11 COLOR
;               (AL)=13  80X11 COLOR
;               (AL)=14  320X200  4 COLOR (20X11 KJ)
;               (AL)=15  320X200  4 COLOR (20X11 KJ)
;               (AL)=16  640X200  2 COLOR (40X11 KJ)
;               (AL)=17  NOT VALID
;               (AL)=18  160X200 16 COLOR (10X11 KJ)
;               (AL)=19  320X200 16 COLOR (20X11 KJ)
;               (AL)=1A  640X200  4 COLOR (40X11 KJ)
;               (AL)=1B  640X200 16 COLOR (40X11 KJ)
;
;                *** NOTE IF HIGH ORDER BIT IN AL IS SET, THE REGEN
;                        BUFFER IS NOT CLEARED.
;
;       (AH)=1  SET CURSOR TYPE
;               (CH) =   BITS 4-0 = START LINE FOR CURSOR
;               ** HARDWARE WILL ALWAYS CAUSE BLINK
;               ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC
;                  BLINKING OR NO CURSOR AT ALL
;               ** IN ANK GRAPHICS MODES, BIT 5 IS FORCED ON TO
;                  DISABLE THE CURSOR
;               (CL) =   BITS 4-0 = END LINE FOR CURSOR
;
;       (AH)=2  SET CURSOR POSITION
;               (DH,DL) = ROW,COLUMN  (0,0) IS UPPER LEFT
;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
;
;       (AH)=3  READ CURSOR POSITION
;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
;               ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
;                       (CH,CL) = CURSOR MODE CURRENTLY SET
;
;       (AH)=4  READ LIGHT PEN POSITION
;               ON EXIT:
;            (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
;            (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
;                        (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN
;                        (CH) = RASTER LINE (0-199)
;                        (BX) = PIXEL COLUMN (0-319,639)
;
;       (AH)=5  SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)
;               (AL)=NEW PAGE VALUE (0- 7 FOR MODES  0&1 , 0-3 FOR MODES  2&3
;                                    0-15 FOR MODES 10&11, 0-7 FOR MODES 12&13)
;
;               IF BIT 7 (80H) OF AL=1
;                  READ/WRITE CRT/CPU PAGE REGISTERS
;               (AL) = 80H READ CRT/CPU PAGE REGISTERS
;               (AL) = 81H SET CPU PAGE REGISTER
;                        (BL) = VALUE TO SET
;                        (CL) = CRT MODE TO SET
;               (AL) = 82H SET CRT PAGE REGISTER
;                        (BH) = VALUE TO SET
;               (AL) = 83H SET BOTH CRT AND CPU PAGE REGISTERS
;                        (BH) = VALUE TO SET IN CRT PAGE REGISTER
;                        (BL) = VALUE TO SET IN CPU PAGE REGISTER
;                        (CL) = CRT MODE TO SET
;               IF BIT 7 (80H) OF AL=1
;                  ALWAYS  RETURNS (BH) = CONTENTS OF CRT PAGE REG
;                                  (BL) = CONTENTS OF CPU PAGE REG
;                *** NOTE CRT/CPU PAGE 0-8 MEANS MAIN RAM, AND
;                    9-A MEANS VIDEO RAM.
;
;       (AH)=6  SCROLL ACTIVE PAGE UP
;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT
;                      BOTTOM OF WINDOW, AL = 0 MEANS BLANK
;                      ENTIRE WINDOW
;               (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
;               (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
;               (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
;
;       (AH)=7  SCROLL ACTIVE PAGE DOWN
;               (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP
;                      OF WINDOW, AL=0 MEANS BLANK ENTIRE WINDOW
```

```
;         (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL
;         (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL
;         (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
;
;       CHARACTER HANDLING ROUTINES
;       (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODE)
;               ON EXIT:
;               (AL) = CHAR READ
;               (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES AND
;                                                   KANJI GRAPHICS MODE)
;
;        ** NOTE **
;                IN KANJI GRAPHICS MODE, CODE AND ATTRIBUTE ARE ASSURED
;                        ONLY WHEN CPU PAGE IS NOT SWITCHED.
;                IN ANK GRAPHICS MODE, CHARACTER CODE 81H-9FH,E0H-FCH IS READ
;                        AS 81H.
;
;       (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;               (CX) = COUNT OF CHARACTERS TO WRITE
;               (AL) = CHAR TO WRITE
;               (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF
;                      CHARACTER (GRAPHICS). SEE NOTE ON WRITE
;                      DOT FOR BIT 7 OF BL = 1.
;
;       (AH) = 10 (0AH) WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
;               (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;               (CX) = COUNT OF CHARACTERS TO WRITE
;               (AL) = CHAR TO WRITE
;               (BL) = COLOR OF CHAR (GRAPHICS)
;                       SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
;
;       FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE,
;               THE CHARACTERS ARE FORMED FROM A CHARACTER
;               GENERATOR IMAGE MAINTAINED IN THE CHARACTER GENERATER ROM.
;       FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE
;               REPLICATION FACTOR CONTAINED IN (CX) ON ENTRY WILL
;               PRODUCE VALID RESULTS ONLY FOR CHARACTERS
;               CONTAINED ON THE SAME ROW.  CONTINUATION TO
;               SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
;
;       GRAPHICS INTERFACE
;       (AH) = 11 (0BH) SET COLOR PALETTE
;               (BH) = PALETTE COLOR ID BEING SET (0-127)
;               (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
;                       COLOR ID = 0 SELECTS THE BACKGROUND
;                                    COLOR (0-15)
;                       COLOR ID = 1 SELECTS THE PALETTE TO BE
;                          USED:
;                          2 COLOR MODE:
;                               0 = WHITE FOR COLOR 1
;                               1 = BLACK FOR COLOR 1
;                          4 COLOR MODES:
;                               0 = GREEN, RED, BROWN FOR
;                                   COLORS 1,2,3
;                               1 = CYAN, MAGENTA, WHITE FOR
;                                   COLORS 1,2,3
;                          8  COLOR MODES:
;                               ALWAYS SETS UP PALETTE AS:
;                               BLUE          FOR COLOR  1
;                               GREEN         FOR COLOR  2
;                               CYAN          FOR COLOR  3
;                               RED           FOR COLOR  4
;                               MAGENTA       FOR COLOR  5
;                               BROWN         FOR COLOR  6
;                               LIGHT GRAY    FOR COLOR  7
;                          16 COLOR MODES:
;                               ALWAYS SETS UP PALETTE AS:
;                               BLUE          FOR COLOR  1
;                               GREEN         FOR COLOR  2
;                               CYAN          FOR COLOR  3
;                               RED           FOR COLOR  4
;                               MAGENTA       FOR COLOR  5
;                               YELLOW        FOR COLOR  6
;                               LIGHT GRAY    FOR COLOR  7
;                               DARK GRAY     FOR COLOR  8
;                               LIGHT BLUE    FOR COLOR  9
;                               LIGHT GREEN   FOR COLOR 10
;                               LIGHT CYAN    FOR COLOR 11
;                               LIGHT RED     FOR COLOR 12
;                               LIGHT MAGENTA FOR COLOR 13
;                               LIGHT YELLOW  FOR COLOR 14
;                               WHITE         FOR COLOR 15
;                       IN 40X25 OR 80X25 ALPHA MODES OR 40X11 OR 80X11
;                          KANJI MODES, THE VALUE SET
;                          FOR PALETTE COLOR 0 INDICATES THE BORDER
;                          COLOR TO BE USED. IN GRAPHIC MODES, IT
;                          INDICATES THE BORDER COLOR AND THE
;                          BACKGROUND COLOR.
;
;       (AH) = 12 (0CH) WRITE DOT
;               (DX) = ROW NUMBER
;               (CX) = COLUMN NUMBER
;               (AL) = COLOR VALUE
;                  IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS
;                  EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF THE DOT
;
;       (AH) = 13 (0DH) READ DOT
;               (DX) = ROW NUMBER
;               (CX) = COLUMN NUMBER
;               (AL) RETURNS THE DOT READ
```

```
; ASCII TELETYPE ROUTINE FOR OUTPUT
;       (AH) = 14 (0EH) WRITE TELETYPE TO ACTIVE PAGE
;               (AL) = CHAR TO WRITE
;               (BL) = FOREGROUND COLOR IN GRAPHICS MODE
;               NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS
;                       MODE SET
;
;       (AH) = 15 (0FH) CURRENT VIDEO STATE
;               RETURNS THE CURRENT VIDEO STATE
;               (AL) = MODE CURRENTLY SET (SEE AH=0 FOR EXPLANATION)
;               (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
;               (BH) = CURRENT ACTIVE DISPLAY PAGE
;
;       (AH) = 16 (10H) SET PALETTE REGISTERS
;               (AL) = 0 SET PALETTE REGISTER
;                       (BL) = PALETTE REGISTER TO SET (00H - 0FH)
;                       (BH) = VALUE TO SET
;               (AL) = 1 SET BORDER COLOR REGISTER
;                       (BH) = VALUE TO SET
;               (AL) = 2 SET ALL PALETTE REGISTERS AND BORDER
;                       REGISTER
;                       ES:DX POINTS TO A 17 BYTE LIST
;                       BYTES 0 THRU 15 ARE VALUES FOR PALETTE
;                               REGISTERS 0 THRU 15
;                       BYTE 16 IS THE VALUE FOR THE BORDER
;                               REGISTER
;
;       (AH) = 17 (11H) RESERVED
;       (AH) = 18 (12H) RESERVED
;
;       (AH) = 19 (13H) REQUEST FONT PATTERN
;               RETURNS THE REQUESTED FONT PATTERN IN USER AREA
;               (AL)=0   REQUEST BASE-FONT
;               (AL)=80H REQUEST BASE-FONT WITH FULL CHARACTER BOX
;
;               (AL)=40H WRITE FONT PATTERN FROM USER AREA TO GAIJI RAM
;               (CX)  =  INTERNAL CODE FOR REQUESTED FONT
;                        FOR HANKAKU-FONT (CH)=0
;               (ES:BX) = DATA AREA FOR FONT
;                               NORMAL-BOX      FULL-BOX
;                       -16X16  ; 32 BYTE       36 BYTE
;                       -16X8   ; 16 BYTE       18 BYTE
;
;       (AH) = 20 (14H) SUPERIMPOSE SCREEN
;               (AL) = 0 SET MODE
;                       (BH)= 0-3 NOT VALID
;                       (BH)= 4   320X200   4 COLOR (40X25 ANK)
;                       (BH)= 5   320X200   4 COLOR (40X25 ANK)
;                       (BH)= 6   640X200   2 COLOR (80X25 ANK)
;                       (BH)= 7   NOT VALID
;                       (BH)= 8   160X200 16 COLOR (20X25 ANK)
;                       (BH)= 9   320X200 16 COLOR (40X25 ANK)
;                       (BH)= A   640X200   4 COLOR (80X25 ANK)
;                       (BH)= B-13 NOT VALID
;                       (BH)=14   320X200   4 COLOR (20X11 KJ)
;                       (BH)=15   320X200   4 COLOR (20X11 KJ)
;                       (BH)=16   640X200   2 COLOR (40X11 KJ)
;                       (BH)=17   NOT VALID
;                       (BH)=18   160X200 16 COLOR (10X11 KJ)
;                       (BH)=19   320X200 16 COLOR (20X11 KJ)
;                       (BH)=1A   640X200   4 COLOR (40X11 KJ)
;
;                   *** NOTE IF HIGH ORDER BIT IN AL IS SET, THE REGEN
;                           BUFFER IS NOT CLEARED.
;
;               (AL) = 1 SET SUPERIMPOSE
;                       (BH)= 0  OFF
;                       (BH)= 1  ON
;               (AL) = 2 SET FORGROUND PAGE
;                       (BH)= 0  VRAM-1
;               (AL) = 3 SET TRANSPARENT PALETTE
;                       (BH) = PALETTE REGISTER NUMBER
;               (AL) = 4 SET SUPERIMPOSE MODE
;                       (BH) = 0 PRI (PRIORITY)
;                       (BH) = 1 XOR
;                       (BH) = 2 AND
;                       (BH) = 3 OR
;
;       CS,SS,DS,ES,BP,DI,SI,BX,CX,DX PRESERVED DURING CALL
;       AX IS DESTROYED
;-------------------------------------------------------
; VIDEO GATE ARRAY REGISTERS
;
;  PORT 3DA OUTPUT
;
;   * VIDEO PROCESSOR 1 (MAIN RAM:VRAM1)  * VIDEO PROCESSOR 2 (VIDEO RAM:VRAM2)
;
;       REG 0   MODE CONTROL 1 REGISTER         MODE CONTROL 1 REGISTER
;       01H                     +HI BANDWIDTH/-LOW BANDWIDTH
;       02H     +GRAPHICS/-ALPHA                +GRAPHICS/-ALPHA
;       04H     RESERVED                        RESERVED
;       08H     +VIDEO ENABLE                   +VIDEO ENABLE
;       10H     +16 COLOR GRAPHICS              +16 COLOR GRAPHICS
;       20H     RESERVED                        RESERVED
;       40H     RESERVED                        +KANJI MODE
;       80H     RESERVED                        +640X200 16 COLOR GRAPHICS
;
;       REG 1   PALETTE MASK REISTER            PALETTE MASK REISTER
;       01H     PALETTE MASK 0                  PALETTE MASK 0
;       02H     PALETTE MASK 1                  PALETTE MASK 1
;       04H     PALETTE MASK 2                  PALETTE MASK 2
;       08H     PALETTE MASK 3                  PALETTE MASK 3
;       10H     RESERVED                        +VRAM-1  ENABLE
```

```
;       20H     RESERVED                        +VRAM-2  ENABLE
;       40H     RESERVED                        +HIGH RESOLUTION ENABLE
;       80H     RESERVED                        +HIGH/-LOW FREQUENCY DISPLAY
;
;       REG 2   BORDER COLOR REGISTER           BORDER COLOR REGISTER
;       01H                     BLUE
;       02H                     GREEN
;       04H                     RED
;       08H                     INTENSITY
;
;       REG 3   MODE CONTROL 2 REGISTER         MODE CONTROL 2 REGISTER
;       01H     RESERVED -- MUST BE ZERO        RESERVED -- MUST BE ZERO
;       02H     +ENABLE BLINK                   +ENABLE BLINK
;       04H     RESERVED -- MUST BE ZERO        RESERVED -- MUST BE ZERO
;       08H     +2 COLOR GRAPHICS               +2 COLOR GRAPHICS
;               (640X200 2 COLOR ONLY)                  (640X200 2 COLOR ONLY)
;       10H     RESERVED                        RESERVED
;       20H     RESERVED                        RESERVED
;       40H     RESERVED                        RESERVED
;       80H     RESERVED                        RESERVED
;
;       REG 4   RESET REGISTER                  RESET REGISTER
;       01H                     +ASYNCHRONOUS RESET
;       02H                     +SYNCHRONOUS RESET
;
;       REG 5   TRANSPARENT PALETTE
;       01H     SELECT 0                        RESERVED
;       02H     SELECT 1                        RESERVED
;       04H     SELECT 2                        RESERVED
;       05H     SELECT 3                        RESERVED
;
;       REG 6   SUPERIMPOSE CONTROL REGISTER
;       01H     +FORE=V-RAM,BACK=MAIN-RAM       RESERVED
;       02H     +TRANSPARENT ON                 RESERVED
;       04H     MODE CONTROL 1                  RESERVED
;       08H     MODE CONTROL 2                  RESERVED
;
;       REGS 10 TO 1F   PALETTE REGISTERS       PALETTE REGISTERS
;       01H                     BLUE
;       02H                     GREEN
;       04H                     RED
;       08H                     INTENSITY
;
;  VIDEO GATE ARRAY STATUS
;    PORT 3DA INPUT
;       01H     +DISPLAY ENABLE
;       02H     +LIGHT PEN TRIGGER SET
;       04H     -LIGHT PEN SWITCH MADE
;       08H     +VERTICAL RETRACE
;       10H     +VIDEO DOTS
;
;---------------------------------------------------------------

= 000D          CR              EQU     0DH     ; CARIDGE RETURN
= 000A          LF              EQU     0AH     ; LINE FEED
= 0007          BELL            EQU     7       ; BEEP
= 0008          BS              EQU     8       ; BACK SPACE
= 00A0          HALFTONE        EQU     0A0H    ; CODE OF HALF TONE
= 2A55          HT_FONT         EQU     0AA55H AND 7F7FH ; FONT PATTERN OF HALF TONE
= FFFF          TRUE            EQU     0FFFFH
= 0000          FALSE           EQU     0
= 0004          GRAPHICS        EQU     4       ; GRAPHICS MODE
= 0014          KJGRAPH         EQU     14H     ; KANJI GRAPHICS MODE
= 0010          KJ_MODE         EQU     10H     ; KANJI MODE
= 0010          VIDEO           EQU     10H     ; VIDEO INTERRUPT
= 0016          KEYBORD         EQU     16H     ; KEYBORD INTERRUPT
=               DEFAULT_MODE    EQU     KJ_MODE ; DEFAULT MODE

= 0004          PORT_B_ALPHA    EQU     04H     ; PORT_B ALPHA MODE
= 0008          EXP64K          EQU     08H     ; 64K MAIN RAM EXPANTION CARD INSTALLED

= 0000          PCMODE1         EQU     00H     ; PC-J MODE CONTROL 1
= 0008          VIDEOENB        EQU     08H     ; VIDEO ENABLE

= 0001          PCPALETM        EQU     01H     ; PC-J PALETTE MASK
= 0002          PCBORD          EQU     02H     ; PC-J BORDER COLOR
= 0008          INTSEL          EQU     08H     ; INTENSITY BIT FOR PALETTE
= 0003          PCMODE2         EQU     03H     ; PC-J MODE CONTROL 2
= 0004          PCRESET         EQU     04H     ; PC-J RESET
= 0002          SYNCRST         EQU     02H     ; SYNCHRONOUSE RESET
= 0005          PCTRPALT        EQU     05H     ; PC-J TRANSPARENT PALETTE
= 0006          PCSUPER         EQU     06H     ; PC-J SUPERIMPOSE REGISTER
= 0010          PCPALET         EQU     10H     ; PC-J PALETTE REGISTER

= 0001          FOREVRAM        EQU     01H     ; V-RAM IS FOREGROUND
= 0002          TRANSON         EQU     02H     ; TRANSPARENT ON

= 0000          SX2STAT         EQU     00H     ; PC-J SX-02 STATUS REGISTER
= 0008          VERTRET         EQU     08H     ; VERTICAL RETRACE
= 0004          LPENSW          EQU     04H     ; -LIGHT PEN SWITCH MODE
= 0002          LPENTRG         EQU     02H     ; LIGHT PEN TRIGGER SET

= 0000          IXMODE1         EQU     00H     ; PC-J MODE CONTROL 1
= 0001          IXPALETM        EQU     01H     ; PC-J PALETTE MASK
= 0010          VRAM1ENB        EQU     10H     ; VRAM 1 ENABLE                 03/05
= 0002          IXBORD          EQU     02H     ; PC-J BORDER COLOR
= 0003          IXMODE2         EQU     03H     ; PC-J MODE CONTROL 2
= 0004          IXRESET         EQU     04H     ; PC-J RESET
= 0010          IXPALET         EQU     10H     ; PC-J PALETTE REGISTER

= 0007          S8KJROM         EQU     07H     ; SX-08 KANJI ROM AND GAIJI RAM
= 0009          S8VRAM1         EQU     09H     ; SX-08 VIDEO RAM (SHARED)
= 000A          S8VRAM2         EQU     0AH     ; SX-08 VIDEO RAM (SEPARATED)
```

```
= 008C                  S8SX02A         EQU     8CH     ; SX-08 SX-02A PC-J VIDEO
= 008D                  S8SX02B         EQU     8DH     ; SX-08 SX-02B JX   VIDEO

= 0080                  ON              EQU     80H     ; ENABLE BIT
= 0020                  MEM             EQU     20H     ; MEMORY BIT
= 8000                  AKJROM          EQU     08000H  ; KANJI ROM ADDRESS
= B800                  AVRAM1          EQU     0B800H  ; VIDEO RAM 1 ADDRESS
= B800                  AVRAM2          EQU     0B800H  ; VIDEO RAM 2 ADDRESS

= 0000                  M32K            EQU     NOT 1FH AND 1FH ;  32K MEMORY RANGE SELECT
= 0001                  M64K            EQU     NOT 1EH AND 1FH ;  64K MEMORY RANGE SELECT
= 0003                  M128K           EQU     NOT 1CH AND 1FH ; 128K MEMORY RANGE SELECT
= 0007                  M256K           EQU     NOT 18H AND 1FH ; 256K MEMORY RANGE SELECT

= 0040                  WR              EQU     40H     ; WRITE ENABLE BIT
= 0020                  RD              EQU     20H     ; READ  ENABLE BIT
= 0000                  FULL            EQU     7FH AND 0; FULL DECODE

= 03D4                  A6845           EQU     03D4H   ; I/O ADDRESS OF 6845
= 01FF                  SX08BASE        EQU     01FFH   ; I/O ADDRESS OF I/O ADDRESS CONTROLLER
= 03D8                  APOCON          EQU     03D8H   ; I/O ADDRESS OF PALETTE AND SUPERIMPOSE CONT.
= 03DA                  S2ABASE         EQU     03DAH   ; I/O ADDRESS OF VIDEO PROCESSOR 1
= 03DA                  S2BBASE         EQU     03DAH   ; I/O ADDRESS OF VIDEO PROCESSOR 2

= 0030                  KJROM_OFF       EQU     MEM OR  AKJROM/800H     ; TURN OFF KANJI ROM
= 0080                  KJROM_ON        EQU     ON  OR  KJROM_OFF       ; TURN ON  KANJI ROM
= 0037                  VRAM1_OFF       EQU     MEM OR  AVRAM1/800H     ; TURN OFF VIDEO RAM 1
= 00B7                  VRAM1_ON        EQU     ON  OR  VRAM1_OFF       ; TURN ON  VIDEO RAM 1
= 0037                  VRAM2_OFF       EQU     MEM OR  AVRAM2/800H     ; TURN OFF VIDEO RAM 2
= 00B7                  VRAM2_ON        EQU     ON  OR  VRAM2_OFF       ; TURN ON  VIDEO RAM 2

= 007B                  SX02A_OFF       EQU             S2ABASE/8       ; TURN OFF SX-02 A
= 00FB                  SX02A_ON        EQU     ON  OR  SX02A_OFF       ; TURN ON  SX-02 A
= 007B                  SX02B_OFF       EQU             S2BBASE/8       ; TURN OFF SX-02 B
= 00FB                  SX02B_ON        EQU     ON  OR  SX02B_OFF       ; TURN ON  SX-02 B

= 007F                  KJMASKL         EQU     007FH   ; MASK PATTERN FOR LEFT PART OF 16X16 FONT
= B0E8                  WHITE_BOX EQU (9874H * 2) AND 3FFFH OR 8000H ;KJ-ROM SEG.ADDRESS OF WHITE BOX CHA

= 8140                  JIS1_L          EQU     08140H  ; LOW BOUND OF JIS 1 CHARACTER
= 9873                  JIS1_H          EQU     09872H+1; HIGH BOUND OF JIS 1 CHARACTER
                                                        ; +1 FOR SPECIAL CHARACTER FOR KANA-KANJI CONV.
= 8440                  ROSS_L_LOW      EQU     8440H   ; LOW  BOUND OF ROSSIAN LOWER CHARACTER
= 8460                  ROSS_L_HIGH     EQU     8460H   ; HIGH BOUND OF ROSSIAN LOWER CHARACTER
= 8470                  ROSS_U_LOW      EQU     8470H   ; LOW  BOUND OF ROSSIAN UPPER CHARACTER
= 8491                  ROSS_U_HIGH     EQU     8491H   ; HIGH BOUND OF ROSSIAN UPPER CHARACTER

= 8100                  RROSS_L_LOW     EQU     8100H   ; LOW  BOUND OF ROSSIAN LOWER REGEN CODE
= 8120                  RROSS_L_HIGH    EQU     8120H   ; HIGH BOUND OF ROSSIAN LOWER REGEN CODE
= 8200                  RROSS_U_LOW     EQU     8200H   ; LOW  BOUND OF ROSSIAN UPPER REGEN CODE
= 8221                  RROSS_U_HIGH    EQU     8221H   ; HIGH BOUND OF ROSSIAN UPPER REGEN CODE

= 0800                  S_RAM           EQU     2048    ; SIZE OF STATIC RAM (FOR EXTERNAL CHARACTER)
= F040                  EXT_CHAR_L      EQU     0F040H  ; LOW BOUND OF EXTERNAL CHARACTER
= F07E                  EXT_CHAR_H      EQU     EXT_CHAR_L+S_RAM/32-2 ; HIGH BOUND OF EXTERNAL CHARACTER

= 0080                  ZENBIT          EQU     80H                     ; ZENKAKU BIT
= 0008                  ZEN2BIT         EQU     08H                     ; ZENKAKU 2ND BIT
= 0077                  HAN_MASK        EQU     (NOT ZENBIT) AND (NOT ZEN2BIT) AND 0FFH ; HANKAKU MASK
= 0088                  ZEN2_MASK       EQU     ZENBIT OR ZEN2BIT       ; MASK OF 2ND BYTE OF ZENKAKU
= 0080                  ZEROCOL         EQU     80H     ; ZERO COLUMN FLAG USED IN W_1ST_CHAR
                                                        ; BIT 8 OF 1ST BYTE OF 2 BYTE CODE IS ALWAYS 1

= 0080                  XOR_BIT         EQU     80H     ; X'OR WRITE IN GRAPHICS WRITE
= 0702                  KKN_OFF         EQU     0702H   ; KANA-KAN DISABLE
= 0703                  KKNI_OFF        EQU     0703H   ; KANA-KAN & INDICATOR DISABLE
= 0700                  KKNI_ON         EQU     0700H   ; KANA-KAN & INDICATOR ENABLE
= 0701                  INDICATOR_ON    EQU     0701H   ; INDICATOR ENABLE
= 853F                  KKN_TERM        EQU     853FH   ; KANA-KAN TERMINATE

= 0010                  DISABLE_NMI     EQU     10H     ; DISABLE NMI
= 0080                  ENABLE_NMI      EQU     80H     ; ENABLE NMI

= 0008                  VRAM2_PAGE      EQU     8       ; PAGE NUMBER OF V-RAM 2

= 07FF                  PCB_MASK        EQU     07FFH   ; MASK FOR PESUDO CODE BUFFER POINTER

= 0009                  ROW_KJ          EQU     11-1-1  ; ROW NUMBER OF KJ MODE
= 0018                  ROW_ANK         EQU     25 - 1  ; ROW NUMBER OF AN MODE

= 0012                  CBOX_ROW        EQU     18      ; ROW NUMBER OF KANJI CHARACTER BOX

= 0011                  BLOCK_CURSOR    EQU     CBOX_ROW-1                      ; BLOCK CURSOR
= 1011                  UNDER_CURSOR    EQU     (CBOX_ROW-2)*256+(CBOX_ROW-1)   ; UNDER CURSOR

= 0004                  F_BASE          EQU     4       ; STACK FRAME BASE FOR SET RETURN PARAMETER
= 0004                  F_DI            EQU     F_BASE + 0      ; FRAME OFFSET  OF DI
= 0006                  F_SI            EQU     F_BASE + 2      ; FRAME OFFSET  OF SI
= 0008                  F_BX            EQU     F_BASE + 4      ; FRAME OFFSET  OF BX
= 000A                  F_CX            EQU     F_BASE + 6      ; FRAME OFFSET  OF CX
= 000C                  F_DX            EQU     F_BASE + 8      ; FRAME OFFSET  OF DX
= 000E                  F_DS            EQU     F_BASE +10      ; FRAME OFFSET  OF DS
= 0010                  F_ES            EQU     F_BASE +12      ; FRAME OFFSET  OF ES

= 000A                  VIO_LOCAL       EQU     10              ; LOCAL AREA SIZE OF VIDEO IO
= 0000                  VACT_PG         EQU     0               ; ACTIVE PAGE
= 0002                  VCRS_POS        EQU     2               ; CURRENT CURSOR POSITION
= 0004                  VKK_FLAG        EQU     4               ; KANA KANJI CONV. FLAG
= 0005                  VKJ_STAT        EQU     5               ; KJ-ROM STATUS (0:OFF, 1:ON)
= 0006                  VVG_STAT        EQU     6               ; VIDEO GENETATER STATUS(0:VG2, 1:VG1)
= 0007                  VGC_ON          EQU     7               ; GRAPHICS CURSOR STATUS
= 0008                  VVIO_ON         EQU     8               ; VIDEO I/O STATUS
```

```
= 0006                  SUP_LOCAL       EQU     6               ; LOCAL AREA SIZE OF SCROLL UP
= 0000                  SUC_MODE        EQU     0               ; CRT MODE
= 0001                  SU_ULR          EQU     1               ; ROW OF UPPER LEFT
= 0002                  SU_TSR          EQU     2               ; TOP OF SOURCE ROW
= 0004                  SU_ES1          EQU     4               ; SEGMENT ADDRESS OF VRAM-1

= 0006                  SDN_LOCAL       EQU     6               ; LOCAL AREA SIZE OF SCROLL DOWN
= 0000                  SDC_MODE        EQU     0               ; CRT MODE
= 0001                  SD_LRR          EQU     1               ; ROW OF LOWER RIGHT
= 0002                  SD_BSR          EQU     2               ; BOTTOM OF SOURCE ROW
= 0004                  SD_ES1          EQU     4               ; SEGMENT ADDRESS OF VRAM-1

= 0010                  RAC_LOCAL       EQU     16              ; LOCAL AREA SIZE OF READ AC CURRENT

= 002E                  W_LOCAL         EQU     46              ; LOCAL AREA SIZE OF WRITE AC/C CURRENT
= 0012                  GW_LOCAL        EQU     18              ; LOCAL AREA SIZE OF GRAPHICS WRITE
= 001C                  SAC_LOCAL       EQU     28              ; LOCAL AREA SIZE OF SET ALT CURSOR
= 0000                  WGPOSN          EQU     0               ; GRAPHICS WRITE POSITION
= 0002                  WPOSN           EQU     2               ; WRITE POSITION
= 0004                  W2CODE          EQU     4               ; 2ND BYTE CODE
= 0005                  W2ATTR          EQU     5               ; 2ND BYTE ATTRIBUTE
= 0006                  WR_SEG          EQU     6               ; REGEN SEGMENT
= 0008                  WGMODE          EQU     8               ; GRAPHICS MODE FLAG
= 0009                  WC_MODE         EQU     9               ; CRT_MODE
= 000A                  WFONT           EQU     10              ; FONT PATTERN

= 000F                  KJ_OFF          EQU     00FH            ; MASK FOR CRT MODE
= 0020                  CURSOR_DISABLE  EQU     20H             ; CURSOR DISABLE BIT
= 3FFF                  GCURSOR_MASK    EQU     3FFFH           ; MASK FOR GRAPHICS CURSOR

= 0001                  VG1_ON          EQU     1               ; VIDEO GENERATER 1 IS ON
= 0000                  VG2_ON          EQU     0               ; VIDEO GENERATER 2 IS ON
= 0002                  VG12_ON         EQU     2               ; VIDEO GENERATER 1 AND 2 ARE ON

= 0001                  PS_PROCESS      EQU     1               ; PRINT SCREEN IS PROCESSING

= 8000                  REGEN_SIZE      EQU     08000H  ; SIZE OF REGEN

= 00A0                  P_CODE_START    EQU     000A0H  ; SEGMNT ADDRESS OF PESUDO REGEN
= 0800                  P_CODE_SIZE     EQU     2048    ; SIZE OF PESUDO REGEN
                        ;------------------------------- VIDEO RAM ---------------------------

0000                    VIDEO_RAM       SEGMENT AT REGEN_START
0000  8000 [                   DB      REGEN_SIZE DUP(?)
            ??
                ]
8000                    VIDEO_RAM       ENDS
                        ;------------------------------- PESUDO CODE BUFFER ------------------

0000                    P_CODE_BUFFER   SEGMENT AT P_CODE_START
0000  0800 [                   DB      P_CODE_SIZE DUP(?)
            ??
                ]
0800                    P_CODE_BUFFER   ENDS
                        ;------------------------------- INDETERMINATE DS, ES ----------------

0000                    INDETERMINATE   SEGMENT
0000                    INDETERMINATE   ENDS

                                ASSUME  CS:CODE, DS:INDETERMINATE, ES:INDETERMINATE

0000                    VF_TABLE        LABEL   WORD            ;  TABLE OF ROUTINES WITHIN VIDEO I/O
0000  02D1 R                    DW      OFFSET  SET_MODE                ; AH =  0
0002  05CC R                    DW      OFFSET  SET_CTYPE               ; AH =  1
0004  0689 R                    DW      OFFSET  SET_CPOS                ; AH =  2
0006  06E6 R                    DW      OFFSET  READ_CURSOR             ; AH =  3
0008  0716 R                    DW      OFFSET  READ_LPEN               ; AH =  4
000A  07FF R                    DW      OFFSET  ACT_DISP_PAGE           ; AH =  5
000C  097B R                    DW      OFFSET  SCROLL_UP               ; AH =  6
000E  0B70 R                    DW      OFFSET  SCROLL_DOWN             ; AH =  7
0010  0D8D R                    DW      OFFSET  READ_AC_CURRENT         ; AH =  8
0012  0F32 R                    DW      OFFSET  WRITE_AC_CURRENT        ; AH =  9
0014  0F56 R                    DW      OFFSET  WRITE_C_CURRENT         ; AH =  A
0016  16C1 R                    DW      OFFSET  SET_COLOR               ; AH =  B
0018  174D R                    DW      OFFSET  WRITE_DOT               ; AH =  C
001A  185B R                    DW      OFFSET  READ_DOT                ; AH =  D
001C  18B0 R                    DW      OFFSET  WRITE_TTY               ; AH =  E
001E  19CA R                    DW      OFFSET  VIDEO_STATE             ; AH =  F
0020  19DD R                    DW      OFFSET  SET_PALLETTE            ; AH = 10
0022  1A52 R                    DW      OFFSET  NO_OPERATION            ; AH = 11
0024  1A52 R                    DW      OFFSET  NO_OPERATION            ; AH = 12
0026  1A53 R                    DW      OFFSET  FONT_PATTERN            ; AH = 13
0028  1C00 R                    DW      OFFSET  SUPERIMPOSE             ; AH = 14

002A  1E2A R                    DW      OFFSET  SET_ALT_CTYPE           ; AH = 81
002C  1E62 R                    DW      OFFSET  SET_ALT_CPOS            ; AH = 82
002E  1ED8 R                    DW      OFFSET  READ_ALT_CURSOR         ; AH = 83
0030  1A52 R                    DW      OFFSET  NO_OPERATION            ; AH = 84
0032  1A52 R                    DW      OFFSET  NO_OPERATION            ; AH = 85
0034  1A52 R                    DW      OFFSET  NO_OPERATION            ; AH = 86
0036  1A52 R                    DW      OFFSET  NO_OPERATION            ; AH = 87
0038  0D8D R                    DW      OFFSET  READ_AC_CURRENT         ; AH = 88
003A  1EEB R                    DW      OFFSET  K_WRITE_AC_CURRENT      ; AH = 89
003C  1EFD R                    DW      OFFSET  K_WRITE_C_CURRENT       ; AH = 8A
003E  1A52 R                    DW      OFFSET  NO_OPERATION            ; AH = 8B
0040  1A52 R                    DW      OFFSET  NO_OPERATION            ; AH = 8C
0042  1A52 R                    DW      OFFSET  NO_OPERATION            ; AH = 8D
0044  1F0F R                    DW      OFFSET  K_WRITE_TTY             ; AH = 8E
```

```
= 0046                          VFT_END EQU     $-VF_TABLE

0046                            VIDEO_IO        PROC    NEAR
0046  FB                                STI                             ; INTERRUPTS BACK ON
0047  FC                                CLD                             ; SET DIRECTION FORWARD

0048  55                                PUSH    BP                      ; SAVE BP
0049  1E                                PUSH    DS                      ; SAVE DS

004A  E8 0000 E                           CALL    DDS                     ; POINT BIOS DATA AREA
                                                  ASSUME  DS:DATA

004D  E8 1BE4 R                           CALL    DISABLE_INT             ; DISABLE INTERRUPT

0050  F6 06 0357 R FF                     TEST    VSTACKL,TRUE            ; VSTACK IS ALREADY USED ?
0055  75 12                               JNZ     VI00                    ; YES

0057  8C D5                               MOV     BP,SS                   ; SAVE STACK SEGMENT
0059  8C 16 0358 R                        MOV     SS_SAVE,SS              ;
005D  89 26 035A R                        MOV     SP_SAVE,SP              ; SAVE STACK POINTER

0061  BD ---- R                           MOV     BP,VSTACK               ; SETUP NEW STACK SEGMENT
0064  8E D5                               MOV     SS,BP                   ;
0066  BC 0600 R                           MOV     SP,OFFSET VSTACK_TOP    ; SETUP NEW STACK POINTER
0069                            VI00:
0069  FE 06 0357 R                        INC     BYTE PTR VSTACKL        ; INCREMENT VIDEO STACK LEVEL

006D  E8 1BEC R                           CALL    ENABLE_INT              ; ENABLE INTERRUPT

0070  83 EC 0A                            SUB     SP,VIO_LOCAL            ; ALLOCATE LOCAL WORK AREA
0073  8B EC                               MOV     BP,SP                   ; ASSIGN BP AS FRAME POINTER

0075  06                                  PUSH    ES
0076  1E                                PUSH    DS                      ; SAVE SEGMENT REGISTERS
0077  52                                PUSH    DX
0078  51                                PUSH    CX
0079  53                                PUSH    BX
007A  56                                PUSH    SI
007B  57                                PUSH    DI
007C  50                                PUSH    AX                      ; SAVE AX VALUE

007D  50                                PUSH    AX
007E  C6 46 04 00                         MOV     BYTE PTR [BP+VKK_FLAG],FALSE  ; CLEAR KANA KANJI FUNCTION FLAG

0082  A0 0353 R                           MOV     AL,KJROM_STAT                 ;
0085  88 46 05                            MOV     [BP+VKJ_STAT],AL              ; SET CURRENT KJROM STATUS

0088  A0 0354 R                           MOV     AL,VG_STAT                    ;
008B  88 46 06                            MOV     [BP+VVG_STAT],AL              ; SET CURRENT VG STATUS

008E  A0 03FB R                           MOV     AL,VIO_PROCESS                ;
0091  88 46 08                            MOV     [BP+VVIO_ON],AL               ; SET CURRENT VIDEO I/O STATUS
0094  80 0E 03FB R FF                     OR      VIO_PROCESS,TRUE              ; SET VIDEO I/O PROCESSING FLAG
0099  58                                  POP     AX

009A  80 FC 81                            CMP     AH,81H                  ; FUNCTION FOR KANA-KANJI CONVERSION ?
009D  72 3F                               JB      VI2                     ; NO

                                                                          ;--- SET DATA FOR KANA-KANJI CONVERSION
009F  80 EC 6C                            SUB     AH,81H-14H-1            ; ADJUST FOR JUMP TABLE

00A2  80 FC 18                            CMP     AH,84H - (81H-14H-1)    ; ALTERNATE CURSOR FUNCTION ?
00A5  72 33                               JB      VI1                     ; YES, SKIP WORK SWAP

00A7  80 FC 22                            CMP     AH,8EH - (81H-14H-1)    ; WRITE TTY FUNCTION ?
00AA  74 2E                               JE      VI1                     ; YES, SKIP WORK SWAP

                                                                          ; VIDEO FUNCTION IS WRITE A/C
00AC  50                                  PUSH    AX                      ; SAVE AX
00AD  53                                  PUSH    BX                      ; SAVE BX

00AE  32 FF                               XOR     BH,BH                   ;
00B0  8A 1E 0062 R                        MOV     BL,ACTIVE_PAGE          ; SET CURRENT ACTIVE PAGE TO BX
00B4  D1 E3                               SAL     BX,1                    ; TIMES 2 FOR WORD OFFSET
00B6  89 5E 00                            MOV     [BP+VACT_PG],BX         ; SAVE ACTIVE PAGE * 2

00B9  A1 034A R                           MOV     AX,ALT_CURSOR_POSN      ; SET CURSOR POSITION TO AX
00BC  87 87 035C R                        XCHG    AX,[BX+OFFSET CURSOR_POSN]      ; SWAP CURSOR POSITION
00C0  89 46 02                            MOV     [BP+VCRS_POS],AX                ; SAVE IT

00C3  A1 0340 R                           MOV     AX,W_1ST_CHAR           ; GET 1ST BYTE CHARACTER AT WRITE A/C
00C6  87 06 033E R                        XCHG    AX,K_1ST_CHAR           ; SWAP 1ST BYTE CHAR FOR KANA-KAN
00CA  A3 0340 R                           MOV     W_1ST_CHAR,AX           ;


00CD  8A 26 0349 R                        MOV     AH,GC_PRESENT           ; SET GRAPHICS CURSOR FLAG
00D1  88 66 07                            MOV     [BP+VGC_ON],AH          ;

00D4  C6 46 04 FF                         MOV     BYTE PTR [BP+VKK_FLAG],TRUE  ; SET KANA-KANJI FUNCTION FLAG

00D8  5B                                  POP     BX                      ; RESTORE BX
00D9  58                                  POP     AX                      ; RESTORE AX
00DA                            VI1:
00DA  8A 3E 0062 R                        MOV     BH,ACTIVE_PAGE          ; SET ACTIVE PAGE TO BH
00DE                            VI2:
00DE  8A C4                               MOV     AL,AH                   ; GET INTO LOW BYTE
00E0  32 E4                               XOR     AH,AH                   ; ZERO TO HIGH BYTE
00E2  D1 E0                               SAL     AX,1                    ; *2 FOR TABLE LOOKUP
00E4  8B F0                               MOV     SI,AX                   ; PUT INTO SI FOR BRANCH

00E6  3D 0046                             CMP     AX,VFT_END              ; TEST FOR WITHIN RANGE
00E9  72 03                               JB      VI3                     ; BRANCH AROUND BRANCH
```

```
00EB  58                        POP     AX                      ; THROW AWAY THE PARAMETER
00EC  EB 44                     JMP     SHORT VIDEO_RETURN      ; DO NOTHING IF NOT IN RANGE

00EE                    VI3:                                    ;--- DETERMINE SEGMENT ADDRESS OF REGEN
00EE  8A 26 0049 R              MOV     AH,CRT_MODE             ; GET CURRENT CRT MODE
00F2  80 E4 DF                  AND     AH,KJ_OFF               ; MASK KJ-BIT OFF
00F5  80 FC 09                  CMP     AH,9                    ; IN MODE USING 32K REGEN ?
00F8  B8 B800                   MOV     AX,REGEN_START          ; SEGMENT FOR COLOR CARD
00FB  72 0C                     JB      VI31                    ; NO,JUMP

00FD  80 3E 034C R 08           CMP     CPU_PAGE,VRAM2_PAGE     ; IN MODE USING V-RAM1 ?
0102  73 1B                     JAE     VI4                     ; NO

                                ;---- SETUP SEGMENT VALUE OF 32K REGEN ACCORDING TO MEMORY SIZE
0104  E8 1D95 R                                                ; ASSUME AL = 0
0107  EB 16                     CALL    GET_DIRSEG              ; GET DIRECT SEGMENT OF MAIN RAM
                                JMP     SHORT   VI4

0109                    VI31:
0109  80 FC 04                  CMP     AH,GRAPHICS             ; GRAPHICS MODE ?
010C  72 11                     JB      VI4                     ; NO
010E  80 3E 034C R 08           CMP     CPU_PAGE,VRAM2_PAGE     ; IN MODE USING V-RAM2 ?
0113  72 0A                     JB      VI4                     ; NO

0115  F6 06 034C R 01           TEST    CPU_PAGE,1              ; PAGE 9 OR B ?
011A  74 03                     JZ      VI4                     ; NO

011C  05 0400                   ADD     AX,400H                 ; SET BIAS FOR ACCESS HIGH 16K OF REGEN

011F                    VI4:
011F  8E C0                     MOV     ES,AX                   ; SET UP TO POINT AT VIDEO RAM AREA
                                        ASSUME  ES: VIDEO_RAM

0121  58                        POP     AX                      ; RECOVER VALUE
0122  8A 26 0049 R              MOV     AH,CRT_MODE             ; GET CURRENT MODE INTO AH
0126  80 E4 0F                  AND     AH,KJ_OFF               ; MASK KANJI MODE FLAG OFF

0129  BF 0132 R                 MOV     DI,OFFSET VIDEO_RETURN  ; GET RETURN ADDRESS
012C  57                        PUSH    DI                      ; STACK IT
012D  2E: FF A4 0000 R          JMP     WORD PTR CS:[SI+OFFSET VF_TABLE]; JUMP TO VIDEO ROUTINE

0132                                    ASSUME  DS:INDETERMINATE, ES:INDETERMINATE
0132                    VIDEO_RETURN:
0132  F6 46 05 FF               TEST    BYTE PTR [BP+VKJ_STAT],TRUE; KJ-ROM ENABLED AT ENTERANCE ?
0136  74 03                     JZ      VI5                     ; NO

0138  E8 1B88 R                 CALL    ENABLE_KJROM            ; ENABLE KJ-ROM
013B                    VI5:
013B  F6 46 06 FF               TEST    BYTE PTR [BP+VVG_STAT],TRUE; VG-1 ENABLED AT ENTERANCE ?
013F  74 0C                     JZ      VI6                     ; NO

0141  E8 1DB0 R                 CALL    ENABLE_VG1              ; ENABLE VIDEO GENERATER 1
0144  80 7E 06 02               CMP     BYTE PTR [BP+VVG_STAT],VG12_ON ; VG-1&2 ENABLED ?
0148  75 03                     JNE     VI6                     ; NO

014A  E8 1DFC R                 CALL    ENABLE_VG12             ; ENABLE VIDEO GENERATER 1&2
014D                    VI6:
014D  E8 0000 E                 CALL    DDS                     ; POINT BIOS DATA SEGMENT
                                        ASSUME  DS:DATA

0150  F6 46 04 FF               TEST    BYTE PTR [BP+VKK_FLAG],TRUE; FUNCTION OF KANA-KANJI CONVERSION?
0154  74 1D                     JZ      VI7                     ; NO
                                                                ;---- RESTORE THE SAVED PARAMETER
0156  8B 4E 02                  MOV     CX,[BP+VCRS_POS]        ; GET SAVED CURSOR POSITION
0159  8B 5E 00                  MOV     BX,[BP+VACT_PG]         ; GET ACTIVE PAGE
015C  89 8F 035C R              MOV     [BX+OFFSET CURSOR_POSN],CX ; RESTORE IT

0160  8B 0E 0340 R              MOV     CX,W_1ST_CHAR           ; GET 1ST BYTE CHAR IN KANA-KAN
0164  87 0E 033E R              XCHG    CX,K_1ST_CHAR           ; SWAP CX AND K_1ST_CHAR
0168  89 0E 0340 R              MOV     W_1ST_CHAR,CX           ; SET W_1ST_CHAR

016C  8A 7E 07                  MOV     BH,[BP+VGC_ON]          ;
016F  88 3E 0349 R              MOV     GC_PRESENT,BH           ; RESTORE GRAPHICS CURSOR FLAG
0173                    VI7:
0173  8A 5E 08                  MOV     BL,[BP+VVIO_ON]         ; RESTORE VIDEO I/O PROCESSING FLAG
0176  88 1E 03FB R              MOV     VIO_PROCESS,BL          ;

017A  5F                        POP     DI
017B  5E                        POP     SI
017C  5B                        POP     BX
017D  59                        POP     CX
017E  5A                        POP     DX
017F  1F                        POP     DS
0180  07                        POP     ES                      ; RECOVER SEGMENTS

0181  83 C4 0A                  ADD     SP,VIO_LOCAL            ; DEALLOCATE LOCAL WORK AREA

0184  E8 1BE4 R                 CALL    DISABLE_INT             ; DISABLE INTERRUPT

0187  FE 0E 0357 R              DEC     BYTE PTR VSTACKL        ; DECREMENT STACK LEVEL = 0 ?
018B  75 0A                     JNZ     VI01                    ; NO

018D  8B 2E 0358 R              MOV     BP,SS_SAVE              ; RESTORE OLD STACK SEGMENT
0191  8E D5                     MOV     SS,BP                   ;
0193  8B 26 035A R              MOV     SP,SP_SAVE              ; RESTORE OLD STACK POINTER
0197                    VI01:
0197  E8 1BEC R                 CALL    ENABLE_INT              ; ENABLE INTERRUPT

019A  1F                        POP     DS                      ; RESTORE DS
019B  5D                        POP     BP                      ; RESTORE BP

019C  CF                        IRET                            ; ALL DONE
```

```
019D                            VIDEO_IO        ENDP
                                ;--------------------------------------------
                                ;
                                ;       SET_MODE                INT 10H, AH = 0
                                ;       --------
                                ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO
                                ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.
                                ;
                                ;       INPUT   AL = MODE SELECTED
                                ;
                                ;       OUTPUT  NONE
                                ;
                                ;       VOLATILE        AX,BX,CX,DX,SI,DI,ES
                                ;
                                ;--------------------------------------------
                                        ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

                                ;------------------------------ TABLES FOR USE IN SETTING OF CRT MODE

019D                            VIDEO_PARMS     LABEL   BYTE                    ;-- INIT_TABLE FOR 6845
                                ;----0---1---2---3---4---5---6---7---8---9--10--11--12..15

019D  38 28 2F 06 1F 06          DB 38H,28H,2FH,06H,1FH, 6H,19H,1CH, 2H, 7H, 6H, 7H,0,0,0,0 ; SETUP FOR 40X25
      19 1C 02 07 06 07
      00 00 00 00
= 0010                          VPARML                                          EQU     $-VIDEO_PARMS

01AD  71 50 5C 0C 1F 06          DB 71H,50H,5CH,0CH,1FH, 6H,19H,1CH, 2H, 7H, 6H, 7H,0,0,0,0 ; 80X25
      19 1C 02 07 06 07
      00 00 00 00

01BD  38 28 2E 06 7F 06          DB 38H,28H,2EH,06H,7FH, 6H,64H,70H, 2H, 1H,26H, 7H,0,0,0,0 ; GRAPHICS
      64 70 02 01 26 07
      00 00 00 00

01CD  71 50 5B 0C 3F 06          DB 71H,50H,5BH,0CH,3FH, 6H,32H,38H, 2H, 3H,26H, 7H,0,0,0,0 ; GRAPHICS 32K REGEN
      32 38 02 03 26 07
      00 00 00 00

01DD                            VIDEO_PARMS_RAS LABEL   BYTE
01DD  38 28 2F 06 0D 0A          DB 38H,28H,2FH,06H,0DH,0AH,0BH,0CH, 2H,11H,10H,11H,0,0,0,0 ; 40x11
      0B 0C 02 11 10 11
      00 00 00 00

01ED  71 50 5C 0C 0D 0A          DB 71H,50H,5CH,0CH,0DH,0AH,0BH,0CH, 2H,11H,10H,11H,0,0,0,0 ; 80x11
      0B 0C 02 11 10 11
      00 00 00 00

01FD                            REGEN_L LABEL   WORD            ;--- TABLE OF REGEN LENGTHS
01FD  0800                              DW      2048            ; MODE  0 40X25  (BW)
01FF  0800                              DW      2048            ; MODE  1 40X25  COLOR
0201  1000                              DW      4096            ; MODE  2 80X25  (BW)
0203  1000                              DW      4096            ; MODE  3 80X25  COLOR
0205  4000                              DW      16384           ; MODE  4 320X200 4 COLOR
0207  4000                              DW      16384           ; MODE  5 320X200 4 (SHADE)
0209  4000                              DW      16384           ; MODE  6 640X200 2 (SHADE)
020B  0000                              DW      0               ; MODE  7 INVALID
020D  4000                              DW      16384           ; MODE  8 160X200 16 COLOR
020F  8000                              DW      32768           ; MODE  9 320X200 16 COLOR
0211  8000                              DW      32768           ; MODE  A 640X200 4 COLOR
0213  0000                              DW      0               ; MODE  B INVALID
0215  0000                              DW      0               ; MODE  C INVALID
0217  0000                              DW      0               ; MODE  D INVALID
0219  0000                              DW      0               ; MODE  E INVALID
021B  0000                              DW      0               ; MODE  F INVALID
021D  0400                              DW      1024            ; MODE 10 40X11  (BW)
021F  0400                              DW      1024            ; MODE 11 40X11  COLOR
0221  0800                              DW      2048            ; MODE 12 80X11  (BW)
0223  0800                              DW      2048            ; MODE 13 80X11  COLOR
0225  4000                              DW      16384           ; MODE 14 320X200 4 COLOR
0227  4000                              DW      16384           ; MODE 15 320X200 4 (SHADE)
0229  4000                              DW      16384           ; MODE 16 640X200 2 (SHADE)
022B  0000                              DW      0               ; MODE 17 INVALID
022D  4000                              DW      16384           ; MODE 18 160X200 16 COLOR
022F  8000                              DW      32768           ; MODE 19 320X200 16 COLOR
0231  8000                              DW      32768           ; MODE 1A 640X200 4 COLOR
0233  8000                              DW      32768           ; MODE 1B 640X200 16 COLOR

0235                            COL_L   LABEL   BYTE            ;--- COLUMNS
                                ; MODE---0--1--2--3--4--5--6--7--8--9--A--B--C--D--E--F
                                ; MODE--10-11-12-13-14-15-16-17-18-19-1A
0235  28 28 50 50 28 28           DB    40,40,80,80,40,40,80, 0,20,40,80,80; 0, 0, 0, 0
      50 00 14 28 50 50

0241                            GAPARM  LABEL   BYTE    ;--- TABLE OF GATE ARRAY PARAMETERS FOR MODE SETTING

0241  0C 2F 00 02                       DB      0CH,2FH,0,2     ;------- SET UP FOR 40X25    (BW)       MODE 0
= 0004                          GAPARML                         EQU     $-GAPARM
0245  08 2F 00 02                       DB      08H,2FH,0,2     ;------- SET UP FOR 40X25    COLOR      MODE 1
0249  0D 2F 00 02                       DB      0DH,2FH,0,2     ;------- SET UP FOR 80X25    (BW)       MODE 2
024D  09 2F 00 02                       DB      09H,2FH,0,2     ;------- SET UP FOR 80X25    COLOR      MODE 3
0251  0A 23 00 00                       DB      0AH,23H,0,0     ;------- SET UP FOR 320X200  4 COLOR    MODE 4
0255  0E 23 00 00                       DB      0EH,23H,0,0     ;------- SET UP FOR 320X200  4 (SHADE) MODE 5
0259  0E 21 00 08                       DB      0EH,21H,0,8     ;------- SET UP FOR 640X200  2 (SHADE) MODE 6
025D  00 00 00 00                       DB      00H,00H,0,0     ;------- INVALID                        MODE 7
0261  1A 2F 00 00                       DB      1AH,2FH,0,0     ;------- SET UP FOR 160X200  16 COLOR   MODE 8
0265  1B 2F 00 00                       DB      1BH,2FH,0,0     ;------- SET UP FOR 320X200  16 COLOR   MODE 9
0269  0B 23 00 00                       DB      0BH,23H,0,0     ;------- SET UP FOR 640X200   4 COLOR   MODE A
026D  00 00 00 00                       DB      00H,00H,0,0     ;------- INVALID                        MODE B
0271  00 00 00 00                       DB      00H,00H,0,0     ;------- INVALID                        MODE C
0275  00 00 00 00                       DB      00H,00H,0,0     ;------- INVALID                        MODE D
0279  00 00 00 00                       DB      00H,00H,0,0     ;------- INVALID                        MODE E
027D  00 00 00 00                       DB      00H,00H,0,0     ;------- INVALID                        MODE F
```

```
0281  4C 27 00 00                DB      4CH,27H,0,0     ;------ SET UP FOR 40X11    (BW)       MODE 10
0285  48 27 00 00                DB      48H,27H,0,0     ;------ SET UP FOR 40X11    COLOR      MODE 11
0289  4D 27 00 00                DB      4DH,27H,0,0     ;------ SET UP FOR 80X11    (BW)       MODE 12
028D  49 27 00 00                DB      49H,27H,0,0     ;------ SET UP FOR 80X11    COLOR      MODE 13
0291  0A 23 00 00                DB      0AH,23H,0,0     ;------ SET UP FOR 320X200  4 COLOR    MODE 14
0295  0E 23 00 00                DB      0EH,23H,0,0     ;------ SET UP FOR 320X200  4 (SHADE)  MODE 15
0299  0E 21 00 08                DB      0EH,21H,0,8     ;------ SET UP FOR 640X200  2 (SHADE)  MODE 16
029D  00 00 00 00                DB      00H,00H,0,0     ;------ INVALID                        MODE 17
02A1  1A 2F 00 00                DB      1AH,2FH,0,0     ;------ SET UP FOR 160X200  16 COLOR   MODE 18
02A5  1B 2F 00 00                DB      1BH,2FH,0,0     ;------ SET UP FOR 320X200  16 COLOR   MODE 19
02A9  0B 23 00 00                DB      0BH,23H,0,0     ;------ SET UP FOR 640X200   4 COLOR   MODE 1A
02AD  8B 23 00 00                DB      8BH,23H,0,0     ;------ SET UP FOR 640X200  16 COLOR   MODE 1B

                        ;--------------- TABLES OF PALETTE COLORS FOR 2 AND 4 COLOR MODES

02B1                    PLTC20  LABEL   BYTE            ;-----  2 COLOR, SET 0
02B1  00 0F 00 00               DB      0,0FH,0,0
= 0004                  PLTC20L EQU     $-PLTC20        ;ENTRY LENGTH
02B5  0F 00 00 00       PLTC40  DB      0FH,0,0,0       ;-----  2 COLOR, SET 1
02B9                            LABEL   BYTE
02B9  00 02 04 06       PLTC41  DB      0,2,4,6         ;-----  4 COLOR, SET 0
02BD                            LABEL   BYTE
02BD  00 03 05 0F               DB      0,3,5,0FH       ;-----  4 COLOR, SET 1

02C1  0A 0B 08 09 0E 0F PLTCS16 DB      10,11,8,9,14,15,12,13,2,3,0,1,6,7,4,5   ; SUPER 16 COLOR STD PLT
      0C 0D 02 03 00 01
      06 07 04 05

02D1                    SET_MODE        PROC    NEAR

02D1  FF 36 0355 R              PUSH    WORD PTR IEP_CTRL       ; SAVE INTERRUPT ENABLE PROHIBIT FLAG
02D5  80 0E 0355 R FF           OR      BYTE PTR IEP_CTRL,TRUE  ; PROHIBIT INTERRUPT ENABLE
02DA  E8 1BE4 R                 CALL    DISABLE_INT             ; DISABLE HARDWARE INTERRUPT

02DD  80 3E 0049 R 10           CMP     CRT_MODE,KJ_MODE        ; KANJI MODE ?
02E2  72 11                     JB      SETM0                   ; NO

02E4  50                        PUSH    AX                      ;
02E5  B8 0702                   MOV     AX,KKN_OFF              ;
02E8  CD 16                     INT     KEYBORD                 ;
02EA  B8 853F                   MOV     AX,KKN_TERM             ; TERMINATE KANA-KAN
02ED  CD 16                     INT     KEYBORD                 ;
02EF  B8 0700                   MOV     AX,KKNI_ON              ;
02F2  CD 16                     INT     KEYBORD                 ;
02F4  58                        POP     AX                      ;
02F5                    SETM0:
02F5  50                        PUSH    AX                      ; SAVE INPUT MODE ON STACK
02F6  24 7F                     AND     AL,7FH                  ; REMOVE CLEAR REGEN SWITCH

02F8  3C 07                     CMP     AL,7                    ; CHECK FOR VALID MODES
02FA  74 10                     JE      SETM1                   ; MODE 7 IS INVALID
02FC  3C 17                     CMP     AL,17H                  ; CHECK FOR VALID MODES
02FE  74 0C                     JE      SETM1                   ; MODE 17H IS INVALID

0300  3C 0A                     CMP     AL,0AH                  ; UNDER 0AH ?
0302  76 0A                     JBE     SETM2                   ; YES, OK
0304  3C 0F                     CMP     AL,0FH                  ; UNDER 0FH ?
0306  76 04                     JBE     SETM1                   ; YES, INVALID

0308  3C 1B                     CMP     AL,1BH                  ;
030A  76 02                     JBE     SETM2                   ; GREATER THAN 1BH IS INVALID
030C                    SETM1:
030C  B0 10                     MOV     AL,DEFAULT_MODE ; DEFAULT TO DEFAULT_MODE
030E                    SETM2:
030E  E8 0940 R                 CALL    ERASE_SCURSOR   ; ERASE CURSOR

0311  BA 03D4                   MOV     DX,A6845                ; ADDRESS OF COLOR CARD
0314  8A E0                     MOV     AH,AL                   ; SAVE MODE IN AH
0316  A2 0049 R                 MOV     CRT_MODE,AL             ; SAVE IN GLOBAL VARIABLE
0319  A2 033B R                 MOV     CRT_MODE2,AL            ; SAVE MODE OF VIDEO PROCESSOR 2
031C  89 16 0063 R              MOV     ADDR_6845,DX            ; SAVE ADDRESS OF BASE
0320  24 0F                     AND     AL,KJ_OFF               ; MASK KANJI MODE FLAG OFF
0322  8B F8                     MOV     DI,AX                   ; SAVE MODE IN DI

0324  E8 1DFC R                 CALL    ENABLE_VG12             ; ENABLE BOTH VG1 AND VG2
0327  E8 1A44 R                 CALL    WAIT_VERTRET            ; WAIT UNTIL VERTICAL RETRACE

032A  BA 03DA                   MOV     DX,VGA_CTL              ; POINT TO CONTROL REGISTER
032D  EC                        IN      AL,DX                   ; SYNC CONTROL REG TO ADDRESS
032E  32 C0                     XOR     AL,AL                   ; SET VGA REG 0
0330  EE                        OUT     DX,AL                   ; SELECT IT

0331  A0 0065 R                 MOV     AL,CRT_MODE_SET ; GET LAST MODE SET
0334  24 F7                     AND     AL,NOT VIDEOENB ; TURN OFF VIDEO
0336  EE                        OUT     DX,AL           ; SET IN GATE ARRAY
                                                        ;----- SET DEFAULT PALETTES
0337  F6 06 03FC R FF           TEST    SUPRESS_PAL,TRUE; SET PALETTE IS SUPRESSED ?
033C  75 37                     JNZ     SETM5           ; YES, SKIP SET PALETTE

033E  B0 06                     MOV     AL,PCSUPER      ;
0340  EE                        OUT     DX,AL           ; CLEAR SUPERIMPOSE CONTROL REGISTER
0341  32 C0                     XOR     AL,AL           ;     FOR SET PALETTE
0343  EE                        OUT     DX,AL           ;

0344  8B C7                     MOV     AX,DI           ; GET MODE
0346  B4 10                     MOV     AH,IXPALET      ; SET PALETTE REG 0

0348  BB 02B1 R                 MOV     BX,OFFSET PLTC20; POINT TO TABLE ENTRY
034B  3C 06                     CMP     AL,6            ; 2 COLOR MODE?
034D  74 0F                     JE      SETM3           ; YES, JUMP

034F  BB 02BD R                 MOV     BX,OFFSET PLTC41; POINT TO TABLE ENTRY
0352  3C 05                     CMP     AL,5            ; CHECK FOR 4 COLOR MODE
```

```
0354  74 08                          JE      SETM3           ; YES, JUMP
0356  3C 04                          CMP     AL,4            ; CHECK FOR 4 COLOR MODE
0358  74 04                          JE      SETM3           ; YES JUMP
035A  3C 0A                          CMP     AL,0AH          ; CHECK FOR 4 COLOR MODE
035C  75 05                          JNE     SETM4           ; NO, JUMP
035E                          SETM3:
035E  E8 0535 R                      CALL    SET_PALETTE4    ; SET PALETTES FOR DEFAULT 4 COLOR
0361  EB 12                          JMP     SHORT SETM5

0363                          SETM4:
0363  3C 0B                          CMP     AL,0BH          ; SUPER 16 COLOR ?
0365  74 05                          JE      SETM41          ; YES

0367  E8 0545 R                      CALL    SET_PALETTE16   ; SET PALETTES FOR DEFAULT 16 COLOR
036A  EB 09                          JMP     SHORT SETM5

036C                          SETM41:
036C  BB 02C1 R                      MOV     BX,OFFSET PLTCS16;GET ADDRESS OF SUPER 16 COLOR PALETTE TABLE
036F  B9 0010                        MOV     CX,16           ; SET NUMBER OF PALETTE
0372  E8 0538 R                      CALL    SPAL41          ; SET PALETTE

0375                          SETM5:                          ;----- SET UP M0 & M1 IN PAGREG for V-RAM1
0375  8B C7                          MOV     AX,DI           ; GET CURRENT MODE
0377  32 DB                          XOR     BL,BL           ; SET UP FOR ALPHA MODE

0379  3C 04                          CMP     AL,GRAPHICS     ; IN ALPHA MODE
037B  72 08                          JC      SETM6           ; YES, JUMP

037D  B3 40                          MOV     BL,40H          ; SET UP FOR 16K REGEN
037F  3C 09                          CMP     AL,09H          ; MODE USE 16K
0381  72 02                          JC      SETM6           ; YES, JUMP

0383  B3 C0                          MOV     BL,0C0H         ; SET UP FOR 32K REGEN
0385                          SETM6:
0385  BA 03DF                        MOV     DX,PAGREG       ; SET PORT ADDRESS OF PAGREG
0388  A0 008A R                      MOV     AL,PAGDAT       ; GET LAST DATA OUTPUT
038B  24 3F                          AND     AL,3FH          ; CLEAR M0 & M1 BITS
038D  0A C3                          OR      AL,BL           ; SET NEW BITS
038F  EE                             OUT     DX,AL           ; STUFF BACK IN PORT
0390  A2 008A R                      MOV     PAGDAT,AL       ; SAVE COPY IN RAM
                                                             ;--- ENABLE VIDEO AND CORRECT PORT SETTING
0393  E8 0551 R                      CALL    VGA_RESET       ; RESET VIDEO GATE ARRAY
0396  EB 07                          JMP     SHORT SETM8

0398                          SETM7:
0398  8A C4                          MOV     AL,AH           ; GET VGA REG NUMBER
039A  EE                             OUT     DX,AL           ; SELECT REG
039B  2E: 8A 07                      MOV     AL,CS:[BX]      ; GET TABLE VALUE
039E  EE                             OUT     DX,AL           ; PUT IN VGA REG
039F                          SETM8:
039F  43                             INC     BX              ; NEXT IN TABLE
03A0  FE C4                          INC     AH              ; NEXT REG
03A2  E2 F4                          LOOP    SETM7           ; DO ENTIRE ENTRY

03A4  2E: 8A 47 FD                   MOV     AL,CS:[BX-(GAPARML-1)]  ; GET VALUE OF PALETTE MASK REG.
03A8  A2 0352 R                      MOV     PALETTE_MASK,AL ; SAVE IT FOR SUPERIMPOSE

03AB  B0 05                          MOV     AL,PCTRPALT     ; POINT SUPERIMPOSE TRANSPARENT REGISTER
03AD  EE                             OUT     DX,AL           ;
03AE  B0 00                          MOV     AL,0            ; SET TRANSPARENT PALETTER NUMBER AS 0
03B0  EE                             OUT     DX,AL           ;

03B1  B0 06                          MOV     AL,PCSUPER      ; POINT SUPERIMPOSE CONTROL REGISTER
03B3  EE                             OUT     DX,AL           ;
03B4  B0 01                          MOV     AL,FOREVRAM     ; MAKE VRAM 2 DISPLAYD IN FOREGROUND
03B6  EE                             OUT     DX,AL           ;
03B7  A2 0347 R                      MOV     SUPIPCR,AL      ; SAVE IT TO RAM

                                     ;----   SET UP CRT AND CPU PAGE REGS ACCORDING TO MODE & MEMORY SIZE
                                     ;       FOR V-RAM1
03BA  BA 03DF                        MOV     DX,PAGREG       ; SET IO ADDRESS OF PAGREG
03BD  A0 008A R                      MOV     AL,PAGDAT       ; GET LAST DATA OUTPUT
03C0  24 C0                          AND     AL,0C0H         ; CLEAR REG BITS
03C2  B3 36                          MOV     BL,36H          ; SET UP FOR GRAPHICS MODE WITH 32K REGEN
03C4  A8 80                          TEST    AL,80H          ; IN THIS MODE?
03C6  75 0C                          JNZ     SETM9           ; YES, JUMP

03C8  B3 3F                          MOV     BL,3FH          ; SET UP FOR 16K REGEN AND 128K MEMORY
03CA  50                             PUSH    AX              ; SAVE AX
03CB  E4 62                          IN      AL,PORT_C       ; READ 8255 PORT C
03CD  A8 08                          TEST    AL,EXP64K       ; 64K CARD INSTALLED ?
03CF  58                             POP     AX              ; RESTORE AX
03D0  74 02                          JZ      SETM9           ; YES

03D2  B3 1B                          MOV     BL,1BH          ; SET UP FOR 16K REGEN AND 64K MEMORY
03D4                          SETM9:
03D4  0A C3                          OR      AL,BL           ; COMBINE MODE BITS AND REG VALUES
03D6  EE                             OUT     DX,AL           ; SET PORT

03D7  A2 008A R                      MOV     PAGDAT,AL       ; SAVE COPY IN RAM


                                     ;----   SET UP CRT AND CPU PAGE REGS ACCORDING TO MODE & MEMORY SIZE
                                     ;       FOR V-RAM2
03DA  BA 03D9                        MOV     DX,PAGREG2
03DD  32 C0                          XOR     AL,AL           ; SET CPU/CRT PAGE 0 IN V-RAM2
03DF  EE                             OUT     DX,AL           ; SET PORT

03E0  A2 033C R                      MOV     PAGDAT2,AL      ; SAVE COPY IN RAM
03E3  8B C6                          MOV     AX,SI           ; PUT MODE SET & PALETTE IN RAM
03E5  88 26 0065 R                   MOV     CRT_MODE_SET,AH ; SAVE MODE CONTROL REG VALUE
03E9  88 26 033D R                   MOV     CRT_MODE_SET2,AH; SAVE MODE CONTROL REG VALUE FOR SUPERIMPOSE
03ED  A2 0066 R                      MOV     CRT_PALLETTE,AL ; SAVE BORDER COLOR
```

```
03F0  1E                                        ;---- SET UP 6845
03F1  8B C7                     PUSH    DS      ; SAVE DATA SEGMENT VALUE
03F3                            MOV     AX,DI   ; GET CURRENT MODE IN AX
                        SETM10:
03F3  33 DB                     XOR     BX,BX   ; SET UP FOR ABS0 SEGMENT
03F5  8E DB                     MOV     DS,BX   ; ESTABLISH VECTOR TABLE ADDRESSING
                                        ASSUME  DS: ABS0

03F7  C5 1E 0074 R              LDS     BX,PARM_PTR     ; GET POINTER TO VIDEO PARMS
                                        ASSUME  DS: CODE

03FB  B9 0010                   MOV     CX,VPARML       ; LENGTH OF EACH ROW OF TABLE

03FE  80 FC 02                  CMP     AH,2            ; DETERMINE WHICH TO USE
0401  72 28                     JC      SETM11          ; MODE IS 0 OR 1

0403  03 D9                     ADD     BX,CX           ; MOVE TO NEXT ROW OF INIT TABLE
0405  80 FC 04                  CMP     AH,4
0408  72 21                     JC      SETM11          ; MODE IS 2 OR 3

040A  03 D9                     ADD     BX,CX           ; MOVE TO GRAPHICS ROW OF INIT_TABLE
040C  80 FC 09                  CMP     AH,9
040F  72 1A                     JC      SETM11          ; MODE IS 4, 5, 6, 8

0411  03 D9                     ADD     BX,CX           ; MOVE TO NEXT GRAPHICS ROW OF INIT_TABLE
0413  80 FC 0B                  CMP     AH,0BH
0416  76 13                     JBE     SETM11          ; MODE IS 9, A OR B

0418  03 D9                     ADD     BX,CX           ; MOVE TO KANJI ROW OF INIT_TABLE
041A  80 FC 12                  CMP     AH,12H
041D  72 0C                     JC      SETM11          ; MODE IS 10, 11

041F  03 D9                     ADD     BX,CX           ; MOVE TO NEXT ROW OF INIT_TABLE
0421  80 FC 14                  CMP     AH,14H
0424  72 05                     JC      SETM11          ; MODE IS 12,13

0426  80 E4 0F                  AND     AH,KJ_OFF       ; MASK KJ BIT OFF
0429  EB C8                     JMP     SHORT   SETM10  ; SERCH AGAIN FOR MODE 14H-1AH

                                ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
042B                    SETM11:
042B  8B F7                     MOV     SI,DI           ; SAVE MODE IN SI
042D  8A 47 02                  MOV     AL,DS:[BX+2]    ; GET HORZ. SYNC POSITION
0430  8B 57 0A                  MOV     DX,WORD PTR DS:[BX+10]  ; GET CURSOR TYPE
0433  86 F2                     XCHG    DH,DL                   ; SWAP FOR HIGH-LOW REVERSE

0435  1E                        PUSH    DS
0436  E8 0000 E                CALL    DDS
                                        ASSUME  DS:DATA
0439  A2 0089 R                 MOV     HORZ_POS,AL     ; SAVE HORZ. SYNC POSITION VARIABLE
043C  89 16 0060 R              MOV     CURSOR_MODE,DX  ; SAVE CURSOR MODE

0440  A0 0086 R                 MOV     AL,VAR_DELAY    ; SET DEFAULT OFFSET
0443  24 0F                     AND     AL,0FH
0445  A2 0086 R                 MOV     VAR_DELAY,AL
                                        ASSUME  DS:CODE
0448  1F                       POP     DS

0449  32 E4                     XOR     AH,AH           ; AH WILL SERVE AS REGISTER NUMBER DURING LOOP
044B  BA 03D4                   MOV     DX,A6845        ; POINT TO 6845
                                ;---LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
044E                    SETM12:
044E  8A C4                     MOV     AL,AH           ; GET 6845 REGISTER NUMBER
0450  EE                        OUT     DX,AL

0451  42                       INC     DX              ; POINT TO DATA PORT
0452  FE C4                     INC     AH              ; NEXT REGISTER VALUE
0454  8A 07                     MOV     AL,[BX]         ; GET TABLE VALUE
0456  EE                        OUT     DX,AL           ; OUT TO CHIP

0457  43                       INC     BX              ; NEXT IN TABLE
0458  4A                       DEC     DX              ; BACK TO POINTER REGISTER
0459  E2 F3                    LOOP    SETM12          ; DO THE WHOLE TABLE

045B  8B C6                     MOV     AX,SI           ; GET MODE BACK
045D  1F                        POP     DS              ; RECOVER SEGMENT VALUE
                                        ASSUME  DS:DATA
                                ;--- FILL REGEN AREA WITH BLANK
045E  33 FF                     XOR     DI,DI           ; SET UP POINTER FOR REGEN
0460  89 3E 004E R              MOV     CRT_START,DI    ; START ADDRESS SAVED IN GLOBAL

0464  C6 06 0062 R 00           MOV     ACTIVE_PAGE,0   ; SET PAGE VALUE
0469  C6 06 036C R 08           MOV     CPU_PAGE,VRAM2_PAGE  ; SET CURRENT ACTIVE PAGE AS 8
046E  C6 06 036D R 08           MOV     CRT_PAGE,VRAM2_PAGE  ; SET CURRENT ACTIVE PAGE AS 8
0473  E8 1B88 R                 CALL    ENABLE_KJROM    ; DISABLE BOTH V-RAM 1 & 2 ONCE
0476  E8 1BB4 R                 CALL    ENABLE_VRAM     ; ENABLE V-RAM 2

0479  8B D8                     MOV     BX,AX           ; SET MODE TO BX
047B  5A                        POP     DX              ; GET ORIGINAL INPUT BACK

047C  8A F2                     MOV     DH,DL           ;
047E  80 E6 7F                  AND     DH,07FH         ;
0481  80 FE 1B                  CMP     DH,1BH          ; SUPER 16 COLOR ?
0484  75 29                     JNE     SETM121         ; NO

0486  50                        PUSH    AX              ; SAVE AX
0487  53                        PUSH    BX              ; SAVE BX
0488  52                        PUSH    DX              ; SAVE DX
0489  57                        PUSH    DI              ; SAVE DI
048A  56                        PUSH    SI              ; SAVE SI

048B  E8 1DB0 R                 CALL    ENABLE_VG1      ;
048E  8A C2                     MOV     AL,DL           ;
0490  2C 11                     SUB     AL,11H          ; SETUP VIDEO PROCESSOR-1
```

```
0492  E8 1C7D R                       CALL    SUP_SET_MODE    ;
0495  C6 06 0049 R 1B                 MOV     CRT_MODE,1BH    ;
049A  E8 1DFC R                       CALL    ENABLE_VG12     ;

049D  BA 03DA                         MOV     DX,S2ABASE      ;
04A0  EC                              IN      AL,DX           ;
04A1  B0 06                           MOV     AL,PCSUPER      ;
04A3  EE                              OUT     DX,AL           ; SETUP SUPERIMPOSE LOGIC
04A4  B0 06                           MOV     AL,0110B        ;
04A6  EE                              OUT     DX,AL           ;
04A7  A2 0347 R                       MOV     SUPIPCR,AL      ;
04AA  5E                              POP     SI              ;
04AB  5F                              POP     DI              ;
04AC  5A                              POP     DX              ; RESTORE REGISTERS
04AD  5B                              POP     BX              ;
04AE  58                              POP     AX              ;
04AF                            SETM121:
04AF  80 E2 80                        AND     DL,80H          ; NO CLEAR OF REGEN ?
04B2  75 4E                           JNZ     SETM17          ; SKIP CLEARING REGEN

04B4  B9 2000                         MOV     CX,8192         ; NUMBER OF WORDS TO CLEAR

04B7  3C 09                           CMP     AL,09H          ; REQUIRE 32K BYTE REGEN ?
04B9  72 02                           JC      SETM13          ; NO, JUMP

04BB  D1 E1                           SHL     CX,1            ; SET 16K WORDS TO CLEAR

04BD                            SETM13:
04BD  BA B800                         MOV     DX,REGEN_START  ; SET UP SEGMENT FOR V-RAM2
04C0  8E C2                           MOV     ES,DX           ; SET REGEN SEGMENT

04C2  B8 0F20                         MOV     AX,' '+15*256   ; FILL CHAR FOR ALPHA
04C5  80 FF 10                        CMP     BH,KJ_MODE      ; KJ MODE ?
04C8  72 03                           JB      SETM14          ; NO

04CA  B8 0720                         MOV     AX,' '+ 7*256   ; FILL CHAR FOR KANJI
04CD                            SETM14:
04CD  80 FB 04                        CMP     BL,GRAPHICS     ; TEST FOR GRAPHICS
04D0  72 1A                           JC      SETM16          ; NO_GRAPHICS_INIT

04D2  80 FF 10                        CMP     BH,KJ_MODE      ; KJ GRAPHICS MODE ?
04D5  72 13                           JB      SETM15          ; NO
                                                              ;--- CLEAR PESUDO CODE BUFFER
04D7  06                              PUSH    ES              ; SAVE CURRENT ES
04D8  57                              PUSH    DI              ; SAVE CURRENT DI
04D9  51                              PUSH    CX              ; SAVE CURRENT CX

04DA  B9 00A0                         MOV     CX,P_CODE_START ; SET PESUDO CODE BUFFER SEGMENT
04DD  8E C1                           MOV     ES,CX           ;    TO ES
                                              ASSUME  ES:P_CODE_BUFFER

04DF  B9 0400                         MOV     CX,P_CODE_SIZE/2; SET BUFFER SIZE IN WORD

04E2  B8 0720                         MOV     AX,' '+ 7*256   ; FILL CHAR FOR ALPHA IN ATTRIBUTE TYPE 2
04E5  F3/ AB                          REP     STOSW           ; FILL THE PESUDO CODE BUFFER

04E7  59                              POP     CX              ; RESTORE CX
04E8  5F                              POP     DI              ; RESTORE DI
04E9  07                              POP     ES              ; RESTORE ES
                                              ASSUME  ES: VIDEO_RAM

04EA                            SETM15:
04EA  33 C0                           XOR     AX,AX           ; FILL FOR GRAPHICS MODE
04EC                            SETM16:
04EC  F3/ AB                          REP     STOSW           ; FILL THE REGEN BUFFER WITH BLANKS

04EE  33 C0                           XOR     AX,AX           ;
04F0  A2 0348 R                       MOV     AC_PRESENT,AL   ; CLEAR ALTERNATE CURSOR FLAG
04F3  A2 0349 R                       MOV     GC_PRESENT,AL   ; CLEAR GRAPHICS  CURSOR FLAG
04F6  C7 06 034E R 1011               MOV     GCURSOR_MODE,UNDER_CURSOR       ; SETUP GRAPHICS CURSOR MODE
04FC  C7 06 0350 R 2011               MOV     ACURSOR_MODE,CURSOR_DISABLE+100H OR BLOCK_CURSOR ;SETUP ALT CSR

0502                            SETM17:                       ;----- ENABLE VIDEO
0502  BA 03DA                         MOV     DX,VGA_CTL      ; SET PORT ADDRESS OF VGA
0505  32 C0                           XOR     AL,AL           ;
0507  EE                              OUT     DX,AL           ; SELECT VGA REG 0
0508  A0 0065 R                       MOV     AL,CRT_MODE_SET ; GET MODE SET VALUE
050B  EE                              OUT     DX,AL           ; SET MODE

050C  E8 1DD6 R                       CALL    ENABLE_VG2      ; ENABLE VIDEO GRNARATER 2

                                      ;----- DETERMINE NUMBER OF COLUMNS AND ROWS, BOTH FOR ENTIRE
                                      ;----- DISPLAY AND THE NUMBER TO BE USED FOR TTY INTERFACE
050F  8B DE                           MOV     BX,SI           ; GET CURRENT CRT MODE
0511  E8 059C R                       CALL    SMODE_SET       ; SET SCREEN PARAMETERS
                                                              ;----- SET CURSOR POSITIONS
0514  32 FF                           XOR     BH,BH           ;
0516  D1 E3                           SHL     BX,1            ; WORD OFFSET INTO CLEAR LENGTH TABLE
0518  2E: 8B 8F 01FD R                MOV     CX,CS:[BX + OFFSET REGEN_L]  ; LENGTH TO CLEAR
051D  89 0E 004C R                    MOV     CRT_LEN,CX      ; SAVE LENGTH OF CRT

0521  B9 0010                         MOV     CX,NO_ACT_PAGE  ; CLEAR ALL CURSOR POSITIONS
0524  BF 035C R                       MOV     DI,OFFSET CURSOR_POSN

0527  1E                              PUSH    DS              ; ESTABLISH SEGMENT
0528  07                              POP     ES              ;   ADDRESSING
                                              ASSUME  ES:DATA

0529  33 C0                           XOR     AX,AX
052B  F3/ AB                          REP     STOSW           ; FILL WITH ZEROES
```

```
052D  8F 06 0355 R                      POP     WORD PTR IEP_CTRL;RESTORE INTERRUPT ENABLE PROHIBIT FLAG
0531  E8 1BEC R                         CALL    ENABLE_INT      ; ENABLE INTERRPUT IF IT IS NOT PROHIBITED

0534  C3                                RET

0535                                SET_MODE        ENDP

                                    ;--------------------------------------------------
                                    ;
                                    ;       SET_PALETTE4
                                    ;
                                    ;       THIS ROUTINE SET PALETTES FOR 4 COLOR
                                    ;
                                    ;       INPUT   AH =    PALETTE REGISTER NUMBER
                                    ;               BX =    ADDRESS OF SETUP PARAMETERS
                                    ;
                                    ;       OUTPUT  NONE
                                    ;
                                    ;       VOLATILE        AX,CX
                                    ;
                                    ;--------------------------------------------------
                                                    ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

0535                                SET_PALETTE4    PROC    NEAR

0535  B9 0004                           MOV     CX,4            ; NUMBER OF REGS TO SET
0538                                SPAL41:
0538  8A C4                             MOV     AL,AH           ; GET REG NUMBER
053A  EE                                OUT     DX,AL           ; SELECT IT
053B  2E: 8A 07                         MOV     AL,CS:[BX]      ; GET DATA
053E  EE                                OUT     DX,AL           ; SET IT

053F  FE C4                             INC     AH              ; NEXT REG
0541  43                                INC     BX              ; NEXT TABLE VALUE
0542  E2 F4                             LOOP    SPAL41

0544  C3                                RET

0545                                SET_PALETTE4    ENDP

                                    ;--------------------------------------------------
                                    ;
                                    ;       SET_PALETTE16
                                    ;
                                    ;       THIS ROUTINE SET PALETTES FOR 16 COLOR
                                    ;
                                    ;       INPUT   AH =    PALETTE REGISTER NUMBER
                                    ;
                                    ;       OUTPUT  NONE
                                    ;
                                    ;       VOLATILE        AX,CX
                                    ;
                                    ;--------------------------------------------------

0545                                SET_PALETTE16   PROC    NEAR

0545  B9 0010                           MOV     CX,16           ; NUMBER OF PALETTES, AH IS REG COUNTER
0548                                SPAL1:
0548  8A C4                             MOV     AL,AH           ; GET REG NUMBER
054A  EE                                OUT     DX,AL           ; SELECT IT
054B  EE                                OUT     DX,AL           ; SET PALETTE VALUE

054C  FE C4                             INC     AH              ; NEXT REG
054E  E2 F8                             LOOP    SPAL1

0550  C3                                RET

0551                                SET_PALETTE16   ENDP

                                    ;--------------------------------------------------
                                    ;
                                    ;       VGA_RESET
                                    ;
                                    ;       THIS ROUTINE RESET VIDEO GATE ARRAY
                                    ;
                                    ;       INPUT   DI =    CRT MODE
                                    ;
                                    ;       OUTPUT  SI =    CONTENTS OF MODE CONTROL 1, BORDER COLOR REG.
                                    ;               CX =    NUMBER OF GATE ARRAY PARAMETERS
                                    ;
                                    ;       VOLATILE        AX,BX,DX
                                    ;
                                    ;--------------------------------------------------

0551                                VGA_RESET       PROC    NEAR

0551  8B C7                             MOV     AX,DI           ; GET CURRENT MODE
0553  8A C4                             MOV     AL,AH
0555  32 E4                             XOR     AH,AH           ;   INTO AX REG
0557  B9 0004                           MOV     CX,GAPARML      ; SET TABLE ENTRY LENGTH
055A  F7 E1                             MUL     CX              ; TIMES MODE FOR OFFSET INTO TABLE
055C  8B D8                             MOV     BX,AX           ; TABLE OFFSET IN BX
055E  81 C3 0241 R                      ADD     BX,OFFSET GAPARM; ADD TABLE START TO OFFSET

0562  2E: 8A 27                         MOV     AH,CS:[BX+IXMODE1];SAVE MODE SET AND PALETTE
0565  2E: 8A 47 02                      MOV     AL,CS:[BX+IXBORD] ; TILL WE CAN PUT THEM IN RAM
0569  8B F0                             MOV     SI,AX

056B  E8 058C R                         CALL    MODE_ALIVE      ; KEEP MEMORY DATA VALID

056E  BA 03DA                           MOV     DX,VGA_CTL      ;
0571  B0 04                             MOV     AL,IXRESET      ; POINT TO RESET REG
```

```
0573  EE                        OUT     DX,AL           ; SEND TO GATE ARRAY
0574  B0 02                     MOV     AL,SYNCRST      ; SET SYNCHRONOUS RESET
0576  EE                        OUT     DX,AL           ; DO IT

                                ; WHILE THE GATE ARRAY IS IN RESET STATE, WE CANNOT ACCESS RAM
0577  8B C6                     MOV     AX,SI           ; RESTORE NEW MODE SET
0579  80 E4 F7                  AND     AH,NOT VIDEOENB ; TURN OFF VIDEO ENABLE

057C  32 C0                     XOR     AL,AL           ; SET UP TO SELECT VGA REG 0
057E  EE                        OUT     DX,AL           ; SELECT IT
057F  86 E0                     XCHG    AH,AL           ; AH IS VGA REG COUNTER
0581  EE                        OUT     DX,AL           ; SET MODE

0582  B0 04                     MOV     AL,IXRESET      ; SET UP TO SELECT VGA REG 4
0584  EE                        OUT     DX,AL           ; SELECT IT
0585  32 C0                     XOR     AL,AL           ;
0587  EE                        OUT     DX,AL           ; REMOVE RESET FROM VGA
                                ; NOW OKAY TO ACCESS RAM AGAIN

0588  E8 058C R                 CALL    MODE_ALIVE      ; KEEP MEMORY DATA VALID

058B  C3                        RET

058C                    VGA_RESET       ENDP

                        ;-----------------------------------------
                        ;
                        ;       MODE_ALIVE
                        ;
                        ;       THIS ROUTINE READS 256 LOCATIONS IN MEMORY AS EVERY OTHER
                        ;       LOCATION IN 512 LOCATIONS.  THIS IS TO INSURE THE DATA
                        ;       INTEGRITY OF MEMORY DURING MODE CHANGES.
                        ;
                        ;       INPUT           NONE
                        ;       OUTPUT          NONE
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------------

058C                    MODE_ALIVE      PROC    NEAR

058C  50                        PUSH    AX              ;SAVE USED REGS
058D  56                        PUSH    SI
058E  51                        PUSH    CX
058F  33 F6                     XOR     SI,SI
0591  B9 0100                   MOV     CX,256
0594                    MALIVE1:
0594  AC                        LODSB
0595  46                        INC     SI
0596  E2 FC                     LOOP    MALIVE1

0598  59                        POP     CX
0599  5E                        POP     SI
059A  58                        POP     AX
059B  C3                        RET

059C                    MODE_ALIVE      ENDP

                        ;-----------------------------------------
                        ;
                        ;       SMODE_SET       SOFTWARE MODE SET
                        ;
                        ;       THIS ROUTINE INITIALIZES KANA-KAN AND
                        ;               SET ROW,COLUMN NUMBER OF SCREEN
                        ;
                        ;       INPUT           BH = CRT MODE
                        ;                       BL = CRT MODE (MASKED)
                        ;                       DS = DATA SEGMENT
                        ;       OUTPUT          NONE
                        ;       VOLATILE        BH
                        ;
                        ;-----------------------------------------
                        ;
059C                    SMODE_SET       PROC    NEAR

059C  50                        PUSH    AX              ;----- DETEMINE NUMBER OF ROWS
059D  53                        PUSH    BX              ; SAVE AX
                                                        ; SAVE MODE
059E  32 FF                     XOR     BH,BH           ; CLEAR BH FOR CONVERT BYTE TO WORD
05A0  2E: 8A 87 0235 R          MOV     AL,CS:[BX + OFFSET COL_L]
05A5  32 E4                     XOR     AH,AH
05A7  A3 004A R                 MOV     CRT_COLS,AX     ; NUMBER OF COLUMNS IN THIS SCREEN

05AA  5B                        POP     BX              ; RESTORE MODE
05AB  B8 0701                   MOV     AX,INDICATOR_ON ; ENABLE INDICATOR
05AE  80 FF 18                  CMP     BH,18H          ; 20X11 GRAPHICS ?
05B1  74 10                     JE      SSET1           ; YES

05B3  C6 06 0346 R 18           MOV     CRT_ROWS,ROW_ANK; SET ROW NUMBER OF ANK MODE
05B8  B8 0703                   MOV     AX,KKNI_OFF     ; DISABLE KANA-KAN
05BB  80 FF 10                  CMP     BH,KJ_MODE      ; KJ MODE ?
05BE  72 08                     JB      SSET2           ; NO

05C0  B8 0700                   MOV     AX,KKNI_ON      ; ENABLE KANA-KAN
05C3                    SSET1:
05C3  C6 06 0346 R 09           MOV     CRT_ROWS,ROW_KJ ; SET ROW NUMBER OF KJ MODE
05C8                    SSET2:
05C8  CD 16                     INT     KEYBORD         ; KANA-KAN ON/OFF

05CA  58                        POP     AX              ; RESTORE AX
05CB  C3                        RET
```

```
05CC                                    SMODE_SET       ENDP
                                        ;-----------------------------------------------
                                        ;       SET_CTYPE                       INT 10H, AH = 1
                                        ;       ---------
                                        ;       THIS ROUTINE SETS THE CURSOR VALUE
                                        ;
                                        ;       INPUT   AH =    CURRENT CRT MODE ( MASKED )
                                        ;               CX =    CURSOR VALUE CH-START LINE, CL-STOP LINE
                                        ;
                                        ;       OUTPUT  NONE
                                        ;
                                        ;       VOLATILE        AL,CX,DX
                                        ;
                                        ;-----------------------------------------------
                                                ASSUME  CS:CODE, DS:DATA

05CC                                    SET_CTYPE       PROC    NEAR
05CC  80 FC 04                                  CMP     AH,GRAPHICS     ; IN GRAPHICS MODE?
05CF  72 27                                     JC      SCT2            ; NO, JUMP

05D1  8B 16 035C R                              MOV     DX,CURSOR_POSN  ; GET CURRENT CURSOR POSITION

05D5  F6 06 0349 R FF                           TEST    GC_PRESENT,TRUE ; GRAPHICS CURSOR PRESENT ?
05DA  74 09                                     JZ      SCT1            ; NO

05DC  51                                        PUSH    CX              ; SAVE NEW CURSOR MODE
05DD  8B 0E 034E R                              MOV     CX,GCURSOR_MODE ; GET OLD CURSOR MODE
05E1  E8 0618 R                                 CALL    WRITE_GCURSOR   ; ERASE CURRENT GRAPHICS CURSOR
05E4  59                                        POP     CX              ; RESTORE NEW CURSOR MODE
05E5                                    SCT1:
05E5  89 0E 034E R                              MOV     GCURSOR_MODE,CX ; SAVE NEW GRAPHICS CURSOR MODE
05E9  E8 0618 R                                 CALL    WRITE_GCURSOR   ; WRITE NEW GRAPHICS CURSOR
05EC  C6 06 0349 R FF                           MOV     GC_PRESENT,TRUE ; SET GRAPHICS CURSOR FLAG ON

05F1  81 E1 3FFF                                AND     CX,GCURSOR_MASK ; MASK FOR GRAPHICS CURSOR
05F5  80 CD 20                                  OR      CH,CURSOR_DISABLE; DISABLE CURSOR OF 6845
05F8                                    SCT2:
05F8  B4 0A                                     MOV     AH,10           ; 6845 REGISTER FOR CURSOR SET
05FA  89 0E 0060 R                              MOV     CURSOR_MODE,CX  ; SAVE IN DATA AREA
05FE  E8 0602 R                                 CALL    OUT6845         ; OUTPUT CX REG

0601  C3                                        RET

0602                                    SET_CTYPE       ENDP
                                        ;-----------------------------------------------
                                        ;       OUT6845
                                        ;
                                        ;       THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
                                        ;
                                        ;       INPUT   AH =    6845 REGISTER ADDRESS
                                        ;               CH =    DATA SHOULD BE WRITE TO (AH)
                                        ;               CL =    DATA SHOULD BE WRITE TO (AH+1)
                                        ;
                                        ;       OUTPUT  NONE
                                        ;
                                        ;       VOLATILE        AL, DX
                                        ;
                                        ;-----------------------------------------------

0602                                    OUT6845 PROC    NEAR

0602  8B 16 0063 R                              MOV     DX,ADDR_6845    ; ADDRESS REGISTER

0606  8A C4                                     MOV     AL,AH           ; GET VALUE
0608  EE                                        OUT     DX,AL           ; REGISTER SET
0609  42                                        INC     DX              ; DATA REGISTER
060A  8A C5                                     MOV     AL,CH           ; DATA
060C  EE                                        OUT     DX,AL
060D  4A                                        DEC     DX

060E  8A C4                                     MOV     AL,AH
0610  FE C0                                     INC     AL              ; POINT TO OTHER DATA REGISTER
0612  EE                                        OUT     DX,AL           ; SET FOR SECOND REGISTER
0613  42                                        INC     DX
0614  8A C1                                     MOV     AL,CL           ; SECOND DATA VALUE
0616  EE                                        OUT     DX,AL

0617  C3                                        RET                     ; ALL DONE

0618                                    OUT6845 ENDP
                                        ;-----------------------------------------------
                                        ;       WRITE_GCURSOR
                                        ;
                                        ;       THIS ROUTINE WRITES GRAPHICS CURSOR
                                        ;
                                        ;       INPUT   AH = CRT MODE ( MASKED )
                                        ;               CX = GRAPHICS CURSOR MODE
                                        ;               DX = ROW,COLUMN POSITION TO WRITE
                                        ;               DS = DATA SEGMENT
                                        ;
                                        ;       OUTPUT          NONE
                                        ;       VOLATILE        NONE
                                        ;
                                        ;-----------------------------------------------
```

```
0618                                    WRITE_GCURSOR           PROC    NEAR

0618  80 3E 0049 R 14                          CMP     CRT_MODE,KJGRAPH; KJ GRAPHICS MODE ?
061D  72 69                                   JB      WGC2            ; NO

061F  F6 C5 20                                TEST    CH,CURSOR_DISABLE; CURSOR DISABLED ?
0622  75 64                                   JNZ     WGC2            ; YES

0624  81 E1 3FFF                              AND     CX,GCURSOR_MASK ; MASK FOR GRAPHICS CURSOR MODE
0628  80 F9 11                                CMP     CL,CBOX_ROW - 1 ; END CURSOR EXCEED CHARACTER BOX ?
062B  76 02                                   JBE     WGC1            ; NO

062D  B1 11                                   MOV     CL,CBOX_ROW - 1 ; SET MAX ROW OF CHARACTER BOX
062F                                    WGC1:
062F  3A E9                                   CMP     CH,CL           ; CURSOR START > END ?
0631  77 55                                   JA      WGC2            ; YES, CURSOR DOES NOT APPEARE

0633  55                                      PUSH    BP              ; SAVE BP
0634  83 EC 1C                                SUB     SP,SAC_LOCAL    ; ALLOCATE LOCAL WORK AREA
0637  8B EC                                   MOV     BP,SP           ; ASSIGN BP AS FRAME POINTER

0639  50                                      PUSH    AX              ;
063A  53                                      PUSH    BX              ;
063B  51                                      PUSH    CX              ;
063C  52                                      PUSH    DX              ; SAVE REGISTERS
063D  56                                      PUSH    SI              ;
063E  57                                      PUSH    DI              ;
063F  1E                                      PUSH    DS              ;
0640  06                                      PUSH    ES              ;

0641  88 66 09                                MOV     [BP+WC_MODE],AH ; SET CRT MODE
0644  89 56 02                                MOV     [BP+WPOSN],DX   ; SAVE ROW/COLUMN POSIITON

0647  8B C2                                   MOV     AX,DX           ; SET IT TO AX FOR GET LOCATION
0649  E8 15E8 R                               CALL    GRAPH_POSN      ; DETERMINE LOCATION IN REGEN BUFFER
064C  89 46 00                                MOV     [BP+WGPOSN],AX  ; SAVE WRITE POSITION
064F  8B D9                                   MOV     BX,CX           ; SET GRAPHICS CURSOR MODE TO BX

0651  06                                      PUSH    ES              ; SAVE ES

0652  16                                      PUSH    SS              ;
0653  1F                                      POP     DS              ; POINT TO STACK SEGMENT
0654  16                                      PUSH    SS              ;
0655  07                                      POP     ES              ; POINT TO STACK SEGMENT
                                                        ASSUME DS:STACK, ES:STACK

0656  8D 76 0A                                LEA     SI,[BP+WFONT]   ; SET ADDRESS OF FONT BUFFER AREA TO SI
0659  8B FE                                   MOV     DI,SI           ; SET IT TO DI

065B  B9 0009                                 MOV     CX,CBOX_ROW/2   ;
065E  33 C0                                   XOR     AX,AX           ; CLEAR 18 BYTE FOR CURSOR
0660  F3/ AB                                  REP     STOSW           ;

0662  33 D2                                   XOR     DX,DX           ;
0664  8A D7                                   MOV     DL,BH           ;
0666  8B FE                                   MOV     DI,SI           ; SET CURSOR START LINE TO DI
0668  03 FA                                   ADD     DI,DX           ;

066A  32 FF                                   XOR     BH,BH           ;
066C  2B DA                                   SUB     BX,DX           ;
066E  43                                      INC     BX              ; SET CURSOR LINE NUMBER TO CX.
066F  8B CB                                   MOV     CX,BX           ;

0671  48                                      DEC     AX              ;
0672  F3/ AA                                  REP     STOSB           ; SET CURSOR

0674  07                                      POP     ES              ; RESTORE REGEN SEGMENT
                                                        ASSUME  ES:VIDEO_RAM

0675  B3 8F                                   MOV     BL,XOR_BIT OR 0FH;SET COLOR 16 AND X'OR WRITE FUNCTION
0677  32 D2                                   XOR     DL,DL           ; SET 0 TO DISPLACEMENT

0679  E8 1265 R                               CALL    G_WRT1          ; WRITE CURSOR

067C  07                                      POP     ES              ;
067D  1F                                      POP     DS              ;
067E  5F                                      POP     DI              ;
067F  5E                                      POP     SI              ; RESTORE REGISTERS
0680  5A                                      POP     DX              ;
0681  59                                      POP     CX              ;
0682  5B                                      POP     BX              ;
0683  58                                      POP     AX              ;

0684  83 C4 1C                                ADD     SP,SAC_LOCAL    ; DEALLOCATE WORK AREA
0687  5D                                      POP     BP              ; RESTORE BP
0688                                    WGC2:
0688  C3                                     RET

0689                                    WRITE_GCURSOR   ENDP
                                        ;-----------------------------------------
                                        ;
                                        ;       SET_CPOS                INT 10H, AH = 2
                                        ;       --------
                                        ;       THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
                                        ;       NEW X-Y VALUES PASSED
                                        ;
                                        ;       INPUT   AH = CRT MODE ( MASKED )
                                        ;               DX = ROW,COLUMN OF NEW CURSOR
                                        ;               BH = DISPLAY PAGE OF CURSOR
                                        ;
                                        ;       OUTPUT          NONE
                                        ;       VOLATILE        AX,CX,SI,DI
                                        ;
```

```
                        ;       CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
                        ;
                        ;--------------------------------------------
                                ASSUME  CS:CODE, DS:DATA

0689                    SET_CPOS        PROC    NEAR

                                MOV     CL,BH           ;
0689  8A CF                     XOR     CH,CH           ; ESTABLISH LOOP COUNT
068B  32 ED                     SAL     CX,1            ; WORD OFFSET
068D  D1 E1                     MOV     SI,CX           ; USE INDEX REGISTER
068F  8B F1                     MOV     DI,[SI+OFFSET CURSOR_POSN] ; GET OLD CURSOR POSITION
0691  8B BC 035C R              MOV     [SI+OFFSET CURSOR_POSN],DX ; SAVE THE POINTER
0695  89 94 035C R

                                CMP     ACTIVE_PAGE,BH
0699  38 3E 0062 R              JNZ     SCP3            ; SET_CPOS_RETURN
069D  75 24

                                CMP     AH,GRAPHICS     ; GRAPHIS MODE ?
069F  80 FC 04                  JB      SCP2            ; NO
06A2  72 1A

                                MOV     CX,GCURSOR_MODE ; SET GRAPHICS CURSOR MODE TO CX
06A4  8B 0E 034E R              TEST    GC_PRESENT,TRUE ; GRAPHCIS CURSOR PRESENT ?
06A8  F6 06 0349 R FF           JZ      SCP1            ; NO
06AD  74 07

                                PUSH    DX              ; SAVE NEW CURSOR POSITION
06AF  52

                                MOV     DX,DI           ; SET OLD CURSOR POSITION TO DX
06B0  8B D7                     CALL    WRITE_GCURSOR   ; ERASE CURRENT GRAPHCIS CURSOR
06B2  E8 0618 R

                                POP     DX              ; RESTORE NEW CURSOR POSITION
06B5  5A                SCP1:
06B6                            CALL    WRITE_GCURSOR   ; WRITES NEW GRAPHIS CURSOR
06B6  E8 0618 R                 MOV     GC_PRESENT,TRUE ; SET GRAPHICS CURSOR FLAG ON
06B9  C6 06 0349 R FF   SCP2:
06BE                            MOV     AX,DX           ; GET ROW/COLUMN TO AX
06BE  8B C2                     CALL    SET_CURSOR      ; CURSOR_SET
06C0  E8 06C4 R         SCP3:
06C3                            RET
06C3  C3

06C4                    SET_CPOS        ENDP

                        ;--------------------------------------------
                        ;
                        ;       SET_CURSOR
                        ;
                        ;       THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE 6845
                        ;
                        ;       INPUT   AX = ROW,COLUMN VALUE TO SET
                        ;
                        ;       OUTPUT          NONE
                        ;       VOLATILE        AX,CX
                        ;
                        ;--------------------------------------------

06C4                    SET_CURSOR      PROC    NEAR

                                CALL    POSITION        ; DETERMINE LOCATION IN REGEN BUFFER
06C4  E8 06D5 R                 MOV     CX,AX
06C7  8B C8                     ADD     CX,CRT_START    ; ADD IN THE START ADDRESS FOR THIS PAGE
06C9  03 0E 004E R

                                SAR     CX,1            ; DIVIDE BY 2 FOR CHAR ONLY COUNT
06CD  D1 F9                     MOV     AH,14           ; REGISTER NUMBER FOR CURSOR
06CF  B4 0E                     CALL    OUT6845         ; OUTPUT THE VALUE TO THE 6845
06D1  E8 0602 R

                                RET
06D4  C3

06D5                    SET_CURSOR      ENDP

                        ;-----------------------------------
                        ;
                        ;       POSITION
                        ;
                        ;       THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                        ;       OF A CHARACTER IN THE ALPHA MODE
                        ;
                        ;       INPUT   AX = ROW, COLUMN POSITION
                        ;
                        ;       OUTPUT  AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                        ;
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------

06D5                    POSITION        PROC    NEAR

                                PUSH    BX              ; SAVE REGISTER
06D5  53

                                MOV     BX,AX
06D6  8B D8                     MOV     AL,AH           ; ROWS TO AL
06D8  8A C4                     MUL     BYTE PTR CRT_COLS       ; DETERMINE BYTES TO ROW
06DA  F6 26 004A R

                                XOR     BH,BH
06DE  32 FF                     ADD     AX,BX           ; ADD IN COLUMN VALUE
06E0  03 C3                     SAL     AX,1            ; * 2 FOR ATTRIBUTE BYTES
06E2  D1 E0                     POP     BX
06E4  5B                        RET
06E5  C3
```

```
06E6                                    POSITION        ENDP
                                        ;-----------------------------------------------
                                        ;
                                        ;       READ_CURSOR             INT 10H, AH = 3
                                        ;       -----------
                                        ;       THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                                        ;       6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
                                        ;
                                        ;       INPUT   BH = PAGE OF CURSOR
                                        ;
                                        ;       OUTPUT  DX = ROW, COLUMN OF THE CURRENT CURSOR POSITION
                                        ;               CX = CURRENT CURSOR MODE
                                        ;
                                        ;       VOLATILE        BX
                                        ;
                                        ;-----------------------------------------
                                                ASSUME  CS:CODE, DS:DATA

06E6                                    READ_CURSOR     PROC    NEAR

06E6  55                                        PUSH    BP              ; SAVE BP

06E7  8A DF                                     MOV     BL,BH
06E9  32 FF                                     XOR     BH,BH
06EB  D1 E3                                     SAL     BX,1            ; WORD OFFSET
06ED  8B 97 035C R                              MOV     DX,[BX+OFFSET CURSOR_POSN]
06F1  8B 0E 0060 R                              MOV     CX,CURSOR_MODE

06F5  80 3E 0049 R 14                           CMP     CRT_MODE,KJGRAPH; KJ GRAPHICS MODE ?
06FA  72 04                                     JB      RCSR1           ; NO

06FC  8B 0E 034E R                              MOV     CX,GCURSOR_MODE ; SET GRAPHICS CURSOR MODE
0700                                    RCSR1:
0700  8B EC                                     MOV     BP,SP           ; SET FRAME POINTER
0702  89 56 0C                                  MOV     [BP+F_DX],DX    ; SET RETURN DX
0705  89 4E 0A                                  MOV     [BP+F_CX],CX    ; SET RETURN CX

0708  5D                                        POP     BP              ; RESTORE BP
0709  C3                                        RET

070A                                    READ_CURSOR     ENDP
                                        ;--------------------------------------------------------------------------
                                        ;
                                        ;       LIGHT PEN               INT 10H, AH = 4
                                        ;       ---------
                                        ;       THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
                                        ;       PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
                                        ;       PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION
                                        ;       IS MADE.
                                        ;
                                        ;       INPUT   AH =    CURRENT CRT MODE ( MASKED )
                                        ;
                                        ;       ON EXIT:
                                        ;               AH = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
                                        ;                       BX,CX,DX ARE DESTROYED
                                        ;               AH = 1 IF LIGHT PEN IS AVAILABLE
                                        ;                       DH,DL = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
                                        ;                       CH =    RASTER POSITION
                                        ;                       BX =    BEST GUESS AT PIXEL HORIZONTAL POSITION
                                        ;
                                        ;--------------------------------------------------------------------
                                                ASSUME  CS:CODE,DS:DATA

                                        ;-----------------------------------------SUBTRACT_TABLE
070A                                    SUBTBL  LABEL   BYTE
                                        ; MODE--0----1----2----3----4----5----6----7----8----9----A----B---C--D--E--F
                                        ; MODE-10---11---12---13---14---15---16---17---18---19---1A---1B
070A  06 06 09 09 05 05                 DB      06H, 06H, 09H, 09H, 05H, 05H, 05H,  0,  05H, 08H, 08H, 08H; 0, 0, 0, 0
      05 00 05 08 08 08

0716                                    READ_LPEN       PROC    NEAR

0716  55                                        PUSH    BP              ; SAVE BP
0717  E8 1DB0 R                               CALL    ENABLE_VG1      ; ENABLE VIDEO GENERATER 1

                                                                        ;--- WAIT FOR LIGHT PEN TO BE DEPRESSED
071A  32 E4                                     XOR     AH,AH           ; SET NO LIGHT PEN RETURN CODE
071C  BA 03DA                                   MOV     DX,VGA_CTL      ; GET ADDRESS OF VGA CONTROL REG

071F  EC                                        IN      AL,DX           ; GET STATUS REGISTER
0720  A8 04                                     TEST    AL,LPENSW       ; TEST LIGHT PEN SWITCH
0722  74 03                                     JZ      RLPEN1

0724  E9 07E5 R                                 JMP     RLPEN14         ; NOT SET, RETURN

0727                                    RLPEN1:
                                                                        ;--- NOW TEST FOR LIGHT PEN TRIGGER
0727  A8 02                                     TEST    AL,LPENTRG      ; TEST LIGHT PEN TRIGGER
0729  75 03                                     JNZ     RLPEN2          ; RETURN WITHOUT RESETTING TRIGGER

072B  E9 07EF R                                 JMP     RLPEN15

072E                                                                    ;--- TRIGGER HAS BEEN SET, READ THE VALUE IN
072E                                    RLPEN2:
072E  B4 10                                     MOV     AH,16           ; LIGHT PEN REGISTERS ON 6845
                                                                        ;---INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
0730  8B 16 0063 R                              MOV     DX,ADDR_6845    ; ADDRESS REGISTER FOR 6845
0734  8A C4                                     MOV     AL,AH           ; REGISTER TO READ
0736  EE                                        OUT     DX,AL           ; SET IT UP
0737  42                                        INC     DX              ; DATA REGISTER
0738  EC                                        IN      AL,DX           ; GET THE VALUE
0739  8A E8                                     MOV     CH,AL           ; SAVE IN CX
073B  4A                                        DEC     DX              ; ADDRESS REGISTER
073C  FE C4                                     INC     AH
```

```
073E  8A C4                     MOV     AL,AH           ; SECOND DATA REGISTER
0740  EE                        OUT     DX,AL
0741  42                        INC     DX              ; POINT TO DATA REGISTER
0742  EC                        IN      AL,DX           ; GET SECOND DATA VALUE
0743  8A E5                     MOV     AH,CH           ; AX HAS INPUT VALUE
                                                        ;--- AX HAS THE VALUE READ IN FROM THE 6845
0745  8A 1E 0049 R              MOV     BL,CRT_MODE
0749  81 E3 000F                AND     BX,KJ_OFF       ; MODE VALUE TO BX
074D  2E: 8A 9F 070A R          MOV     BL,CS:SUBTBL[BX]; DETERMINE AMOUNT TO SUBTRACT
0752  2B C3                     SUB     AX,BX           ; TAKE IT AWAY

0754  3D 0FA0                   CMP     AX,4000         ; IN TOP OR BOTTOM BORDER?
0757  72 02                     JB      RLPEN3          ; NO, OKAY
0759  33 C0                     XOR     AX,AX           ; YES, SET TO ZERO
075B                    RLPEN3:
075B  8B 1E 004E R              MOV     BX,CRT_START
075F  D1 EB                     SHR     BX,1
0761  2B C3                     SUB     AX,BX           ; CONVERT TO CORRECT PAGE ORIGIN
0763  79 02                     JNS     RLPEN4          ; IF POSITIVE, DETERMINE MODE
0765  2B C0                     SUB     AX,AX           ; <0 PLAYS AS 0
                                                        ;--- DETERMINE MODE OF OPERATION
0767                    RLPEN4:                         ; DETERMINE_MODE
0767  B1 03                     MOV     CL,3            ; SET *8 SHIFT COUNT

0769  8A 36 0049 R              MOV     DH,CRT_MODE     ; SET CRT MODE TO DH
076D  80 E6 0F                  AND     DH,KJ_OFF       ; STRIP KJ BIT OFF
0770  80 FE 04                  CMP     DH,GRAPHICS     ; GRAPHICS MODE ?
0773  72 4D                     JB      RLPEN11         ; ALPHA_PEN
                                                        ;--- GRAPHICS MODE
0775  B2 28                     MOV     DL,40           ; DIVISOR FOR GRAPHICS
0777  80 FE 09                  CMP     DH,9            ; USING 32K REGEN?
077A  72 02                     JB      RLPEN5          ; NO, JUMP
077C  B2 50                     MOV     DL,80           ; YES, SET RIGHT DIVSOR
077E                    RLPEN5:
077E  F6 F2                     DIV     DL              ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                        ;  AL RANGE 0-99, AH RANGE 0-39
                                                        ;--- DETERMINE GRAPHIC ROW POSITION
0780  8A E8                     MOV     CH,AL           ; SAVE ROW VALUE IN CH
0782  02 ED                     ADD     CH,CH           ; *2 FOR EVEN/ODD FIELD
0784  80 FE 09                  CMP     DH,9            ; USING 32K REGEN?
0787  72 06                     JB      RLPEN6          ; NO, JUMP

0789  D0 EC                     SHR     AH,1            ; ADJUST ROW & COLUMN
078B  D0 E0                     SHL     AL,1
078D  02 ED                     ADD     CH,CH           ; *4 FOR 4 SCAN LINES
078F                    RLPEN6:
078F  8A DC                     MOV     BL,AH           ; COLUMN VALUE TO BX
0791  2A FF                     SUB     BH,BH           ; MULTIPLY BY 8 FOR MEDIUM RES
0793  80 FE 06                  CMP     DH,6            ; DETERMINE MEDIUM OR HIGH RES
0796  72 13                     JB      RLPEN9          ; MODE 4 OR 5
0798  77 06                     JA      RLPEN8          ; MODE 8, 9, OR A,B
079A                    RLPEN7:
079A  B1 04                     MOV     CL,4            ; SHIFT VALUE FOR HIGH RES
079C  D0 E4                     SAL     AH,1            ; COLUMN VALUE TIMES 2 FOR HIGH RES
079E  EB 0B                     JMP     SHORT RLPEN9

07A0                    RLPEN8:
07A0  80 FE 09                  CMP     DH,9            ; CHECK MODE
07A3  77 F5                     JA      RLPEN7          ; MODE A,B
07A5  74 04                     JE      RLPEN9          ; MODE 9

07A7  B1 02                     MOV     CL,2            ; MODE 8 SHIFT VALUE
07A9  D0 EC                     SHR     AH,1
07AB                    RLPEN9:                         ; NOT_HIGH_RES
07AB  D3 E3                     SHL     BX,CL           ; MULTIPLY *16 FOR HIGH RES
                                                        ;--- DETERMINE ALPHA CHAR POSITION
07AD  8A D4                     MOV     DL,AH           ; COLUMN VALUE FOR RETURN

07AF  32 E4                     XOR     AH,AH           ; CLEAT TO CONVERT TO WORD

07B1  B6 04                     MOV     DH,8/2          ; DIVISOR FOR ANK (8 ROW/ 2 SCAN)
07B3  80 3E 0049 R 10           CMP     CRT_MODE,KJ_MODE; KANJI MODE ?
07B8  72 02                     JB      RLPEN10         ; NO
07BA  B6 09                     MOV     DH,CBOX_ROW/2   ; DIVISOR FOR KJ  (18 ROW/ 2 SCAN)
07BC                    RLPEN10:
07BC  F6 F6                     DIV     DH
07BE  8A F0                     MOV     DH,AL           ; DIVIDE FOR CHAR POSITION
                                                        ; SET ROW POSITION TO DH
07C0  EB 21                     JMP     SHORT RLPEN13   ; LIGHT_PEN_RETURN_SET

07C2                    RLPEN11:
07C2  F6 36 004A R              DIV     BYTE PTR CRT_COLS ;--- ALPHA MODE ON LIGHT PEN
07C6  8A F0                     MOV     DH,AL           ; DETERMINE ROW,COLUMN VALUE
07C8  8A D4                     MOV     DL,AH           ; ROWS TO DH
                                                        ; COLS TO DL
07CA  8A E8                     MOV     CH,AL           ; SET AL*1 TO CH
07CC  D2 E0                     SAL     AL,CL           ; MULTIPLY ROWS * 8
07CE  80 3E 0049 R 10           CMP     CRT_MODE,KJ_MODE; KANJI MODE ?
07D3  72 06                     JB      RLPEN12         ; NO
                                                        ;--- MULTIPLY ROWS * 18
07D5  D0 E0                     SAL     AL,1            ; MULTIPLY ROWS * 16
07D7  D0 E5                     SAL     CH,1            ; CH HAS CL*2
07D9  02 C5                     ADD     AL,CH           ; AL HAS CL*18
07DB                    RLPEN12:
07DB  8A E8                     MOV     CH,AL           ; GET RASTER VALUE TO RETURN REG
07DD  8A DC                     MOV     BL,AH           ; COLUMN VALUE
07DF  32 FF                     XOR     BH,BH           ;  TO BX
07E1  D3 E3                     SAL     BX,CL
07E3                    RLPEN13:
07E3  B4 01                     MOV     AH,1            ; LIGHT_PEN_RETURN_SET
07E5                    RLPEN14:                        ; INDICATE EVERYTHING SET
07E5  52                        PUSH    DX              ; LIGHT_PEN_RETURN
07E6  8B 16 0063 R              MOV     DX,ADDR_6845    ; SAVE RETURN VALUE (IN CASE)
07EA  83 C2 07                  ADD     DX,7            ; GET BASE ADDRESS
                                                        ; POINT TO RESET PARM
```

```
07ED  EE                        OUT     DX,AL           ; ADDRESS, NOT DATA, IS IMPORTANT
07EE  5A                        POP     DX              ; RECOVER VALUE
07EF                    RLPEN15:                        ; RETURN_NO_RESET
07EF  8B EC                     MOV     BP,SP           ; SET FRAME POINTER
07F1  89 5E 08                  MOV     [BP+F_BX],BX    ; SET RETURN BX
07F4  89 4E 0A                  MOV     [BP+F_CX],CX    ; SET RETURN CX
07F7  89 56 0C                  MOV     [BP+F_DX],DX    ; SET RETURN DX

07FA  E8 1DD6 R                 CALL    ENABLE_VG2      ; ENABLE VIDEO GENERATER 2

07FD  5D                        POP     BP              ; RESTORE BP
07FE  C3                        RET

07FF                    READ_LPEN       ENDP
                        ;--------------------------------------------
                        ;
                        ;       ACT_DISP_PAGE           INT 10H, AH = 5
                        ;       ------------
                        ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                        ;       THE FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT
                        ;
                        ;       INPUT   AH =    CRT MODE (MASKED)
                        ;               AL =    NEW ACTIVE DISPLAY PAGE
                        ;
                        ;       OUTPUT  NONE
                        ;
                        ;       THE 6845 IS RESET TO DISPLAY THAT PAGE
                        ;
                        ;--------------------------------------------
                                ASSUME  CS:CODE, DS:DATA

07FF                    ACT_DISP_PAGE   PROC    NEAR

07FF  55                        PUSH    BP              ; SAVE BP

0800  A8 80                     TEST    AL,080H         ; CRT/CPU PAGE REG FUNCTION
0802  75 2A                     JNZ     SET_CRTCPU      ; YES, GO HANDLE IT

0804  80 FC 04                  CMP     AH,GRAPHICS     ; GRAPHICS MODE ?
0807  73 23                     JAE     ACTDP1          ; YES, SKIP

0809  50                        PUSH    AX              ; SAVE CRT MODE

080A  A2 0062 R                 MOV     ACTIVE_PAGE,AL  ; SAVE ACTIVE PAGE VALUE
080D  8B 0E 004C R              MOV     CX,CRT_LEN      ; GET SAVED LENGTH OF REGEN BUFFER
0811  98                        CBW                     ; CONVERT AL TO WORD
0812  50                        PUSH    AX              ; SAVE PAGE VALUE

0813  F7 E1                     MUL     CX              ; DISPLAY PAGE TIMES REGEN LENGTH
0815  A3 004E R                 MOV     CRT_START,AX    ; SAVE START ADDRESS FOR LATER USE
0818  8B C8                     MOV     CX,AX           ; START ADDRESS TO CX
081A  D1 F9                     SAR     CX,1            ; DIVIDE BY 2 FOR 6845 HANDLING
081C  B4 0C                     MOV     AH,12           ; 6845 REGISTER FOR START ADDRESS
081E  E8 0602 R                 CALL    OUT6845

0821  5B                        POP     BX              ; RECOVER PAGE VALUE
0822  D1 E3                     SAL     BX,1            ; *2 FOR WORD OFFSET
0824  8B 87 035C R              MOV     AX,[BX + OFFSET CURSOR_POSN] ; GET CURSOR FOR THIS PAGE
0828  E8 06C4 R                 CALL    SET_CURSOR      ; SET THE CURSOR POSITION

082B  58                        POP     AX              ; RESTORE CRT MODE
082C                    ACTDP1:
082C  5D                        POP     BP              ; RESTORE BP
082D  C3                        RET

082E                    ACT_DISP_PAGE   ENDP

                        ;--------------------------------------------------------
                        ;
                        ;       SET_CRTCPU
                        ;
                        ;       THIS ROUTINE READS OR WRITES THE CRT/CPU PAGE REGISTERS
                        ;
                        ;       INPUT   AH =    CRT MODE (MASKED)
                        ;               AL = 83H        SET BOTH CRT AND CPU PAGE REGS
                        ;                       BH =    VALUE TO SET IN CRT PAGE REG
                        ;                       BL =    VALUE TO SET IN CPU PAGE REG
                        ;                       CL =    CRT MODE FOR CPU PAGE
                        ;               AL = 82H        SET CRT PAGE REG
                        ;                       BH =    VALUE TO SET IN CRT PAGE REG
                        ;               AL = 81H        SET CPU PAGE REG
                        ;                       BL =    VALUE TO SET IN CPU PAGE REG
                        ;                       CL =    CRT MODE FOR CPU PAGE
                        ;               AL = 80H        READ CURRENT VALUE OF CRT/CPU PAGE REGS
                        ;
                        ;       OUTPUT  ALL FUNCTIONS RETURN
                        ;               BH = CURRENT CONTENTS OF CRT PAGE REG
                        ;               BL = CURRENT CONTENTS OF CPU PAGE REG
                        ;
                        ;--------------------------------------------------------

082E                    SET_CRTCPU      PROC    NEAR

082E  FF 36 0355 R              PUSH    WORD PTR IEP_CTRL       ; SAVE INTERRUPT ENABLE PROHIBIT FLAG
0832  80 0E 0355 R FF           OR      BYTE PTR IEP_CTRL,TRUE  ; PROHIBIT INTERRUPT ENABLE
0837  E8 1BE4 R                 CALL    DISABLE_INT             ; DISABLE ALL HARDWARE INTERRUPT

083A  80 FC 04                  CMP     AH,GRAPHICS     ; GRAPHICS MODE ?
083D  73 10                     JAE     SETCC00         ; YES

083F  50                        PUSH    AX              ; SAVE AX
0840  51                        PUSH    CX              ; SAVE CX
```

```
0841  8B 0E 0060 R          MOV     CX,CURSOR_MODE  ; GET CURSOR MODE OF TEXT
0845  80 CD 20             OR      CH,CURSOR_DISABLE; DISABLE CURSOR
0848  B4 0A                MOV     AH,10           ; SET 6845 CURSOR REGISTER
084A  E8 0602 R            CALL    OUT6845         ; OUT DATA TO 6845
084D  59                   POP     CX              ; RESTORE CX
084E  58                   POP     AX              ; RESTORE AX
084F                    SETCC00:
084F  E8 0940 R            CALL    ERASE_SCURSOR   ; ERASE SOFTWARE CURSOR

0852  80 3E 0049 R 10      CMP     CRT_MODE,KJ_MODE; KANJI MODE ?
0857  72 07                JB      SCC0            ; NO

0859  50                   PUSH    AX              ;
085A  B8 853F              MOV     AX,KKN_TERM     ; TERMINATE KANA-KAN
085D  CD 16                INT     KEYBORD         ;
085F  58                   POP     AX              ;
0860                    SCC0:                      ;----- WAIT VERTICAL RETRACE
0860  E8 1DB0 R            CALL    ENABLE_VG1      ; ENABLE VIDEO GENERATER 1

0863  8A E0                MOV     AH,AL           ; SAVE REQUEST IN AH
0865  BA 03DA              MOV     DX,VGA_CTL      ; SET ADDRESS OF GATE ARRAY
0868                    SCC1:
0868  EC                   IN      AL,DX           ; GET STATUS
0869  24 08                AND     AL,VERTRET      ; VERTICAL RETRACE?
086B  74 FB                JZ      SCC1            ; NO, WAIT FOR IT

                                                   ;----- SELECT VIDEO RAM 1 OR 2
086D  80 FB 08             CMP     BL,VRAM2_PAGE   ; VIDEO RAM 2 ?
0870  73 08                JAE     SCC2            ; YES

0872  BA 03DF              MOV     DX,PAGREG       ; SET IO ADDRESS OF PAGE REG
0875  A0 008A R            MOV     AL,PAGDAT       ; GET DATA LAST OUTPUT TO REG
0878  EB 06                JMP     SHORT SCC3

087A                    SCC2:
087A  BA 03D9              MOV     DX,PAGREG2      ; SET IO ADDRESS OF PAGE REG
087D  A0 033C R            MOV     AL,PAGDAT2      ; GET DATA LAST OUTPUT TO REG
0880                    SCC3:

0880  80 FC 80             CMP     AH,80H          ;----- CHECK FUNCTION
0883  74 3E                JZ      SCC32           ; READ FUNCTION REQUESTED?
0885  80 FC 84             CMP     AH,84H          ; YES, DON'T SET ANYTHING
0888  73 39                JNC     SCC32           ; VALID REQUEST?
088A  F6 C4 01             TEST    AH,1            ; NO, PRETEND IT WAS A READ REQUEST
088D  74 39                JZ      SCC5            ; SET CPU REG?
088F  53                   PUSH    BX              ; NO, GO SEE ABOUT CRT REG
0890  8B 0E 0049 R         MOV     CRT_MODE,CL     ; SAVE PAGE NO.
0894  8A F9                MOV     BH,CL           ; SET MODE
0896  8A D9                MOV     BL,CL           ; SET UNMASKED MODE TO BH
0898  80 E3 0F             AND     BL,KJ_OFF       ; SET MASKED MODE TO BL
089B  E8 059C R            CALL    SMODE_SET       ;
089E  5B                   POP     BX              ; SET SOFTWARE MODE

089F  88 1E 034C R         MOV     CPU_PAGE,BL     ; SET NEW CPU PAGE

08A3  E8 1B88 R            CALL    ENABLE_KJROM    ; DISABLE BOTH VRAM1 AND VRAM2 ONCE
08A6  E8 1BB4 R            CALL    ENABLE_VRAM     ; SELECT V-RAM ACCORDING TO CPU PAGE

08A9  D0 E3                SHL     BL,1            ;-- NEW CPU PAGE IS IN CURRENT V-RAM
08AB  D0 E3                SHL     BL,1            ; SHIFT VALUE TO RIGHT BIT POSITION
08AD  D0 E3                SHL     BL,1
08AF  24 C7                AND     AL,NOT CPUREG   ; CLEAR OLD CPU VALUE
08B1  80 E3 38             AND     BL,CPUREG       ; BE SURE UNRELATED BITS ARE ZERO
08B4  0A C3                OR      AL,BL           ; OR IN NEW VALUE
08B6  EE                   OUT     DX,AL           ; SET NEW VALUE

08B7  80 3E 034C R 08      CMP     CPU_PAGE,VRAM2_PAGE  ; VIDEO RAM 2 ?
08BC  73 07                JAE     SCC4            ; YES

08BE  A2 008A R            MOV     PAGDAT,AL       ; SAVE COPY IN RAM
08C1  EB 05                JMP     SHORT SCC5

08C3                    SCC32:
08C3  EB 43                JMP     SHORT SCC11     ; INTERMEDIATE POINT TO JUMP

08C5                    SCC4:
08C5  A2 033C R            MOV     PAGDAT2,AL      ; SAVE COPY IN RAM
08C8                    SCC5:
08C8  F6 C4 02             TEST    AH,2            ; SET CRT REG?

08CB  74 3B                JZ      SCC11           ; NO, GO RETURN CURRENT SETTINGS

                                                   ;----- SET CRT PAGE
08CD  88 3E 034D R         MOV     CRT_PAGE,BH     ; SAVE CRT PAGE TO RAM

08D1  80 FF 08             CMP     BH,VRAM2_PAGE   ; VIDEO RAM 2 ?
08D4  73 08                JAE     SCC7            ; YES

08D6  BA 03DF              MOV     DX,PAGREG       ;
08D9  A0 008A R            MOV     AL,PAGDAT       ;
08DC  EB 06                JMP     SHORT SCC8

08DE                    SCC7:
08DE  BA 03D9              MOV     DX,PAGREG2      ;
08E1  A0 033C R            MOV     AL,PAGDAT2      ;
08E4                    SCC8:
08E4  24 F8                AND     AL,NOT CRTREG   ; CLEAR OLD CRT VALUE
08E6  80 E7 07             AND     BH,CRTREG       ; BE SURE UNRELATED BITS ARE ZERO
08E9  0A C7                OR      AL,BH           ; OR IN NEW VALUE
08EB  EE                   OUT     DX,AL           ; SET NEW VALUES

08EC  8A 26 0347 R         MOV     AH,SUPIPCR      ; GET LAST DATA OF SUPERIMPOSE CONTROL REG.
```

```
08F0  80 3E 034D R 08                   CMP     CRT_PAGE,VRAM2_PAGE  ; VIDEO RAM 2 ?
08F5  73 08                             JAE     SCC9                 ; YES

08F7  A2 008A R                         MOV     PAGDAT,AL            ;
08FA  80 E4 FE                          AND     AH,NOT FOREVRAM      ; SET V-RAM AS BACKGROUND
08FD  EB 06                             JMP     SHORT SCC10

08FF                                SCC9:
08FF  A2 033C R                         MOV     PAGDAT2,AL           ;
0902  80 CC 01                          OR      AH,FOREVRAM          ; SET V-RAM AS FOREGROUND
0905                                SCC10:
0905  E8 0929 R                         CALL    SET_SUPREG           ; SET AH TO SUPERIMPOSE REGISTER
0908                                SCC11:
0908  8A 3E 034D R                      MOV     BH,CRT_PAGE          ; GET CURRENT CRT PAGE
090C  8A 1E 034C R                      MOV     BL,CPU_PAGE          ; GET CURRENT CPU PAGE

0910  E8 1DD6 R                         CALL    ENABLE_VG2           ; ENABLE VIDEO GENERATER 2

0913  B4 03                             MOV     AH,3                 ; GET CURSOR TYPE
0915  CD 10                             INT     VIDEO                ;

0917  B4 01                             MOV     AH,1                 ; SET CURSOR TYPE
0919  CD 10                             INT     VIDEO                ;
091B                                SCC12:
091B  8F 06 0355 R                      POP     WORD PTR IEP_CTRL;RESTORE INTERRUPT ENABLE PROHIBIT FLAG
091F  E8 1BEC R                         CALL    ENABLE_INT           ; ENABLE INTERRUPT IF IT IS NOT PROHIBITED

0922  8B EC                             MOV     BP,SP                ; SET FRAME POINTER
0924  89 5E 08                          MOV     [BP+F_BX],BX         ; SET RETURN BX

0927  5D                                POP     BP                   ; RESTORE BP
0928  C3                                RET

0929                                SET_CRTCPU      ENDP

                                    ;--------------------------------------------------------
                                    ;
                                    ;       SET_SUPREG              SET SUPERIMPOSE REGISTER
                                    ;       ----------
                                    ;
                                    ;       THIS ROUTINE SET SUPERIMPOSE REGISTER
                                    ;
                                    ;       INPUT           AH =    VALUE TO SET
                                    ;                       DS =    DATA SEGMENT
                                    ;       OUTPUT          NONE
                                    ;       VOLATILE        AL,BX,DX
                                    ;
                                    ;--------------------------------------------------------

0929                                SET_SUPREG      PROC    NEAR

0929  8A DC                             MOV     BL,AH                ; SET OUT DATA TO BL
092B  F6 C3 02                          TEST    BL,TRANSON           ; SET SUPERIMPOSE ON ?
092E  75 03                             JNE     SUPREG1              ; YES
0930  80 E3 01                          AND     BL,FOREVRAM          ; MASK MODE CONTROL BITS FOR SUPERIMPOSE OFF
0933                                SUPREG1:
0933  B7 06                             MOV     BH,PCSUPER           ; SET SUPERIMPOSE CONTROL REG
0935  BA 03DA                          MOV     DX,S2ABASE           ; GET ADDRESS OF SX-02A
0938  E8 1E22 R                         CALL    OUT_GA               ; OUT BX TO GATE ARRAY
093B  88 26 0347 R                      MOV     SUPIPCR,AH           ; SAVE NEW VALUE TO RAM

093F  C3                                RET

0940                                SET_SUPREG      ENDP

                                    ;--------------------------------------------------------
                                    ;
                                    ;       ERASE_SCURSOR           ERASE SOFTWARE CURSOR
                                    ;       -------------
                                    ;
                                    ;       THIS ROUTINE ERASE SOFTWARE CURSOR IF IT PRESENTS
                                    ;
                                    ;       INPUT           AH =    CRT MODE  (MASKED)
                                    ;                       DS =    DATA SEGMENT
                                    ;       OUTPUT          NONE
                                    ;       VOLATILE        NONE
                                    ;
                                    ;--------------------------------------------------------

0940                                ERASE_SCURSOR   PROC    NEAR

0940  51                                PUSH    CX                          ; SAVE CX
0941  52                                PUSH    DX                          ; SAVE DX

0942  80 FC 04                          CMP     AH,GRAPHICS                 ; GRAPHICS MODE ?
0945  72 17                             JB      ERACRS1                     ; NO,

0947  F6 06 0349 R FF                   TEST    GC_PRESENT,TRUE             ; GRAPHICS CURSOR PRESENT ?
094C  74 10                             JZ      ERACRS1                     ; NO

094E  8B 0E 034E R                      MOV     CX,GCURSOR_MODE             ; SET CURSOR MODE
0952  8B 16 035C R                      MOV     DX,CURSOR_POSN              ; SET CURSOR POSITION IN GRAPHICS
0956  E8 0618 R                         CALL    WRITE_GCURSOR               ; WRITE GRAPHICS CURSOR

0959  C6 06 0349 R 00                   MOV     GC_PRESENT,FALSE            ; FLAG OFF
095E                                ERACRS1:
095E  5A                                POP     DX                          ; RESTORE DX
095F  59                                POP     CX                          ; RESTORE CX

0960  C3                                RET
```

```
0961                            ERASE_SCURSOR   ENDP

                                ;--------------------------------------------------------
                                ;
                                ;       APPEAR_SCURSOR          APPEAR SOFTWARE CURSOR
                                ;       --------------
                                ;
                                ;       THIS ROUTINE APPEAR SOFTWARE CURSOR
                                ;
                                ;       INPUT           AH =    CRT MODE
                                ;                       DS =    DATA SEGMENT
                                ;       OUTPUT          NONE
                                ;       VOLATILE        NONE
                                ;
                                ;--------------------------------------------------------

0961                            APPEAR_SCURSOR  PROC    NEAR

0961  51                               PUSH    CX                      ; SAVE REGISTERS
0962  52                               PUSH    DX                      ;

0963  80 FC 04                          CMP     AH,GRAPHICS             ; GRAPHICS MODE ?
0966  72 10                             JB      APPCRS1                 ; NO,

0968  8B 0E 034E R                      MOV     CX,GCURSOR_MODE         ; SET CURSOR MODE
096C  8B 16 035C R                      MOV     DX,CURSOR_POSN          ; SET CURSOR POSITION IN GRAPHICS
0970  E8 0618 R                         CALL    WRITE_GCURSOR           ; WRITE GRAPHICS CURSOR

0973  C6 06 0349 R FF                   MOV     GC_PRESENT,TRUE         ; FLAG ON
0978                            APPCRS1:
0978  5A                                POP     DX                      ; RESTORE REGISTERS
0979  59                               POP     CX                      ;

097A  C3                                RET

097B                            APPEAR_SCURSOR  ENDP
                                ;----------------------------------------
                                ;
                                ;       SCROLL_UP               INT 10H, AH = 6
                                ;       ---------
                                ;       THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                                ;       ON THE SCREEN
                                ;
                                ;
                                ;       INPUT   AH = CURRENT CRT MODE ( MASKED )
                                ;               AL = NUMBER OF ROWS TO SCROLL
                                ;               CX = ROW/COLUMN OF UPPER LEFT CORNER
                                ;               DX = ROW/COLUMN OF LOWER RIGHT CORNER
                                ;               BH = ATTRIBUTE TO BE USED ON BLANKED LINE
                                ;               DS = DATA SEGMENT
                                ;               ES = REGEN BUFFER SEGMENT
                                ;
                                ;       OUTPUT  NONE -- THE REGEN BUFFER IS MODIFIED
                                ;
                                ;       WORK    [BP+SUC_MODE] =  CURRENT CRT MODE ( MASKED )
                                ;               [BP+SU_ULR]   =  ROW OF UPPER LEFT CORNER
                                ;               [BP+SU_TSR]   =  TOP OF SOURCE ROW
                                ;               [BP+SU_ES1]   =  SEGMENT OF V-RAM1
                                ;
                                ;----------------------------------------
                                        ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM
097B                            SCROLL_UP       PROC    NEAR

097B  55                                PUSH    BP              ; SAVE BP
097C  83 EC 06                          SUB     SP,SUP_LOCAL    ; ALLOCATE LOCAL WORK AREA
097F  8B EC                             MOV     BP,SP           ; SET BP AS FRAME POINTER

0981  50                                PUSH    AX              ; SAVE CRT MODE
0982  1E                                PUSH    DS              ; SAVE DS
0983  E8 0940 R                         CALL    ERASE_SCURSOR   ; ERASE SOFTWARE CURSOR

0986  8A D8                             MOV     BL,AL           ; SAVE LINE COUNT IN BL
0988  80 E7 77                          AND     BH,HAN_MASK     ; STRIP KANJI BITS OFF FOR SPACE CODE

098B  88 66 00                          MOV     BYTE PTR [BP+SUC_MODE],AH; SET MASKED CRT MODE

098E  80 FC 04                          CMP     AH,GRAPHICS     ; TEST FOR GRAPHICS MODE
0991  73 05                             JAE     SCRUP1          ; YES, HANDLE SEPARATELY

0993  E8 09C9 R                         CALL    TEXT_UP         ; SCROLL TEXT UP
0996  EB 27                             JMP     SHORT SCRUP3    ; GO TO END

0998                            SCRUP1:
0998  80 3E 0049 R 10                   CMP     CRT_MODE,KJ_MODE; KJ GRAPHICS MODE ?
099D  72 1D                             JB      SCRUP2          ; NO

099F  88 6E 01                          MOV     BYTE PTR [BP+SU_ULR],CH ; SET UPPER LEFT ROW POSITION
09A2  88 6E 02                          MOV     BYTE PTR [BP+SU_TSR],CH ;
09A5  00 46 02                          ADD     BYTE PTR [BP+SU_TSR],AL ; SET TOP OF SOURCE ROW POSITION

09A8  50                                PUSH    AX              ;
09A9  53                                PUSH    BX              ;
09AA  51                                PUSH    CX              ; SAVE REGISTERS
09AB  52                                PUSH    DX              ;
09AC  1E                                PUSH    DS              ;
09AD  06                                PUSH    ES              ;

09AE  B8 00A0                           MOV     AX,P_CODE_START ; SET PESUDO CODE BUFFER SEGMENT
09B1  8E C0                             MOV     ES,AX           ;   TO ES

09B3  E8 09C9 R                         CALL    TEXT_UP         ; SCROLL TEXT UP
```

```
09B6  07                        POP     ES                      ;
09B7  1F                        POP     DS                      ;
09B8  5A                        POP     DX                      ;
09B9  59                        POP     CX                      ; RESTORE REGISTERS
09BA  5B                        POP     BX                      ;
09BB  58                        POP     AX                      ;
09BC                    SCRUP2:
09BC  E8 0A2B R                 CALL    GRAPHICS_UP             ; SCROLL IN GRAPHICS MODE
09BF                    SCRUP3:
                                        ASSUME DS: DATA
09BF  1F                        POP     DS                      ; RESTORE DS
09C0  58                        POP     AX                      ; RESTORE CRT MODE
09C1  E8 0961 R                 CALL    APPEAR_SCURSOR          ; WRITES SOFTWARE CURSOR

09C4  83 C4 06                  ADD     SP,SUP_LOCAL            ; DEALLOCATE LOCAL WORK AREA
09C7  5D                        POP     BP                      ; RESTORE BP
09C8  C3                        RET                             ; RETURN TO CALLER

09C9                    SCROLL_UP       ENDP

                        ;----------------------------------------
                        ;
                        ;       TEXT_UP
                        ;
                        ;       THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                        ;       ON THE TEXT SCREEN
                        ;
                        ;       INPUT   BH = ATTRIBUTE TO BE USED ON BLANKED LINE
                        ;               BL = NUMBER OF ROWS TO SCROLL
                        ;               CX = ROW/COLUMN OF UPPER LEFT CORNER
                        ;               DX = ROW/COLUMN OF LOWER RIGHT CORNER
                        ;               DS = DATA SEGMENT
                        ;               ES = REGEN BUFFER SEGMENT
                        ;
                        ;       OUTPUT  NONE
                        ;
                        ;       VOLATILE        AX,BL,CX,DX,SI,DI,DS
                        ;
                        ;----------------------------------------

09C9                    TEXT_UP PROC    NEAR

09C9  55                        PUSH    BP                      ; SAVE BP
09CA  53                        PUSH    BX                      ; SAVE FILL ATTRIBUTE IN BH

09CB  8B C1                     MOV     AX,CX                   ; UPPER LEFT POSITION
09CD  E8 09F5 R                 CALL    SCROLL_POSITION         ; DO SETUP FOR SCROLL
09D0  74 1F                     JZ      TUP4                    ; BLANK_FIELD
                                        ASSUME  DS:VIDEO_RAM

09D2  03 F0                     ADD     SI,AX                   ; FROM ADDRESS
09D4  8A E6                     MOV     AH,DH                   ; # ROWS IN BLOCK
09D6  2A E3                     SUB     AH,BL                   ; # ROWS TO BE MOVED
09D8                    TUP1:
09D8  E8 0A1B R                 CALL    MOVE_ROW                ; MOVE ONE ROW
09DB  03 F5                     ADD     SI,BP
09DD  03 FD                     ADD     DI,BP                   ; POINT TO NEXT LINE IN BLOCK
09DF  FE CC                     DEC     AH                      ; COUNT OF LINES TO MOVE
09E1  75 F5                     JNZ     TUP1                    ; ROW_LOOP

09E3                    TUP2:
09E3  58                        POP     AX                      ; RECOVER ATTRIBUTE IN AH
09E4  B0 20                     MOV     AL,' '                  ; FILL WITH BLANKS
09E6                    TUP3:
09E6  E8 0A24 R                 CALL    CLEAR_ROW               ; CLEAR THE ROW
09E9  03 FD                     ADD     DI,BP                   ; POINT TO NEXT LINE
09EB  FE CB                     DEC     BL                      ; COUNTER OF LINES TO SCROLL
09ED  75 F7                     JNZ     TUP3                    ; CLEAR_LOOP

09EF  5D                        POP     BP                      ; RESTORE BP
09F0  C3                        RET                             ; RETURN TO CALLER

09F1                    TUP4:
09F1  8A DE                     MOV     BL,DH                   ; GET ROW COUNT
09F3  EB EE                     JMP     TUP2                    ; GO CLEAR THAT AREA

09F5                    TEXT_UP ENDP

                        ;--------------------------------------------------
                        ;
                        ;       SCROLL_POSITION
                        ;
                        ;       THIS ROUTINE SET UP SOME PARAMETERS FOR SCROLL FUNCTION
                        ;
                        ;       INPUT   AX =    ROW/COLUMN
                        ;               BL =    SCROLL ROW NUMBER
                        ;
                        ;       OUTPUT  AX =    BL * CRT_COLS * 2
                        ;               CH =    0
                        ;               DX =    DIFFERENCE OF ROW/COLUMN
                        ;               BP =    CRT_COLS * 2
                        ;               DI,SI = REGEN ADDRESS CORRESPONDING TO ROW/COLUMN
                        ;               FLAG =  CONTENT OF BL
                        ;               DS =    ES
                        ;
                        ;--------------------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM
```

```
09F5                    SCROLL_POSITION PROC    NEAR

09F5  E8 06D5 R                 CALL    POSITION        ; CONVERT TO REGEN POINTER
09F8  03 06 004E R              ADD     AX,CRT_START    ; OFFSET OF ACTIVE PAGE
09FC  8B F8                     MOV     DI,AX           ; TO ADDRESS FOR SCROLL
09FE  8B F0                     MOV     SI,AX           ; FROM ADDRESS FOR SCROLL

0A00  2B D1                     SUB     DX,CX           ; DX = #ROWS, #COLS IN BLOCK
0A02  FE C6                     INC     DH
0A04  FE C2                     INC     DL              ; INCREMENT FOR 0 ORIGIN
0A06  32 ED                     XOR     CH,CH           ; SET HIGH BYTE OF COUNT TO ZERO

0A08  8B 2E 004A R              MOV     BP,CRT_COLS     ; GET NUMBER OF COLUMNS IN DISPLAY
0A0C  03 ED                     ADD     BP,BP           ; TIMES 2 FOR ATTRIBUTE BYTE
0A0E  8A C3                     MOV     AL,BL           ; GET LINE COUNT
0A10  F6 26 004A R              MUL     BYTE PTR CRT_COLS ; DETERMINE OFFSET TO FROM ADDRESS
0A14  03 C0                     ADD     AX,AX           ; *2 FOR ATTRIBUTE BYTE

0A16  06                        PUSH    ES              ; ESTABLISH ADDRESSING TO REGEN BUFFER
0A17  1F                        POP     DS              ;  FOR BOTH POINTERS

0A18  0A DB                     OR      BL,BL           ; 0 SCROLL MEANS BLANK FIELD
0A1A  C3                        RET                     ; RETURN WITH FLAGS SET

0A1B                    SCROLL_POSITION ENDP

                        ;--------------------------------------------------
                        ;
                        ;       MOVE_ROW
                        ;
                        ;       THIS ROUTINE MOVES ONE ROW IN TEXT MODE
                        ;
                        ;       INPUT   DL =    NUMBER OF CHARACTERS TO MOVE
                        ;               DS:SI = SOURCE TOP ADDRESS
                        ;               ES:DI = DESTINATION TOP ADDRESS
                        ;
                        ;       OUTPUT  NOTHING
                        ;
                        ;       VOLATILE        CL
                        ;
                        ;--------------------------------------------------
                                ASSUME  CS:CODE, DS:VIDEO_RAM, ES:VIDEO_RAM

0A1B                    MOVE_ROW PROC   NEAR

0A1B  8A CA                     MOV     CL,DL           ; GET # OF COLS TO MOVE
0A1D  56                        PUSH    SI
0A1E  57                        PUSH    DI              ; SAVE START ADDRESS
0A1F  F3/ A5                    REP     MOVSW           ; MOVE THAT LINE ON SCREEN
0A21  5F                        POP     DI
0A22  5E                        POP     SI              ; RECOVER ADDRESSES
0A23  C3                        RET

0A24                    MOVE_ROW ENDP
                        ;--------------------------------------------------
                        ;
                        ;       CLEAR_ROW
                        ;
                        ;       THIS ROUTINE MOVES ONE ROW IN TEXT MODE
                        ;
                        ;       INPUT   AX =    ATTRIBUTE/CHARACTER TO FILL
                        ;               ES:DI = DESTINATION TOP ADDRESS
                        ;
                        ;       OUTPUT  NOTHING
                        ;
                        ;       VOLATILE        CL
                        ;
                        ;--------------------------------------------------

0A24                    CLEAR_ROW       PROC NEAR

0A24  8A CA                     MOV     CL,DL           ; GET # COLUMNS TO CLEAR
0A26  57                        PUSH    DI
0A27  F3/ AB                    REP     STOSW           ; STORE THE FILL CHARACTER
0A29  5F                        POP     DI
0A2A  C3                        RET

0A2B                    CLEAR_ROW       ENDP


                        ;--------------------------------------------------
                        ;
                        ;       SCROLL UP (GRAPHICS)
                        ;
                        ;       THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
                        ;
                        ;       INPUT   [BP+SUC_MODE] =  CURRENT CRT MODE ( MASKED )
                        ;               [BP+SU_ULR]   =  ROW OF UPPER LEFT CORNER
                        ;               [BP+SU_TSR]   =  TOP OF SOURCE ROW
                        ;               CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                        ;               DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                        ;           BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                        ;
                        ;               BH = FILL VALUE FOR BLANKED LINES
                        ;               AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                        ;               DS = DATA SEGMENT
                        ;               ES = REGEN SEGMENT
                        ;
                        ;       OUTPUT  NOTHING, THE SCREEN IS SCROLLED
                        ;
                        ;       VOLATILE        AX,BL,CX,DX,SI,DI,DS
                        ;
                        ;--------------------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM
```

```
0A2B                    GRAPHICS_UP     PROC    NEAR

0A2B  8A D8                             MOV     BL,AL           ; SAVE LINE COUNT IN BL
0A2D  8B C1                             MOV     AX,CX           ; GET UPPER LEFT POSITION INTO AX REG

                                        ;--- USE CHARACTER SUBROUTINE FOR POSITIONING
0A2F  E8 15E8 R                         CALL    GRAPH_POSN      ;-ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
0A32  8B F8                             MOV     DI,AX           ; SAVE RESULT AS DESTINATION
                                                                ; ADDRESS
                                                                ;--- DETERMINE SIZE OF WINDOW
0A34  2B D1                             SUB     DX,CX
0A36  81 C2 0101                        ADD     DX,101H         ; ADJUST VALUES

0A3A  8A E6                             MOV     AH,DH           ; SAVE VALUE OF DH*1 TO AH
0A3C  D0 E6                             SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT DOTS/CHAR
0A3E  D0 E6                             SAL     DH,1            ;  AND EVEN/ODD ROWS

0A40  80 3E 0049 R 14                   CMP     CRT_MODE,KJGRAPH; KANJI GRAPHICS MODE ?
0A45  72 04                             JB      GRUP1           ; NO

0A47  D0 E6                             SAL     DH,1            ; MULTIPLY # ROWS BY 9 SINCE 18 VERT DOTS/CHAR
0A49  02 F4                             ADD     DH,AH           ;
0A4B                    GRUP1:

0A4B  E8 1D95 R                         call    get_dirseg      ; get direct access segment of v-ram1
0A4E  89 46 04                          mov     [bp+su_esl],ax  ; save it

                                                                ;--- DETERMINE CRT MODE
0A51  8A 66 08                          MOV     AH,[BP+SUC_MODE] ; SET CRT MODE TO AH
0A54  80 FC 06                          CMP     AH,6            ; TEST FOR HIGH RES
0A57  74 17                             JE      GRUP2           ; FIND_SOURCE
                                                                ;--- MEDIUM RES UP
0A59  D0 E2                             SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
0A5B  D1 E7                             SAL     DI,1            ; OFFSET *2 SINCE 2 BYTES/CHAR

0A5D  80 FC 04                          CMP     AH,4            ; TEST FOR MEDIUM RES
0A60  74 0E                             JE      GRUP2
0A62  80 FC 05                          CMP     AH,5            ; TEST FOR MEDIUM RES
0A65  74 09                             JE      GRUP2
0A67  80 FC 0A                          CMP     AH,0AH          ; TEST FOR MEDIUM RES
0A6A  73 04                             JAE     GRUP2

0A6C  D0 E2                             SAL     DL,1            ;--- LOW RES UP
0A6E  D1 E7                             SAL     DI,1            ; # COLUMNS * 2 AGAIN, SINCE 4 BYTES/CHAR
                                                                ; OFFSET *2 AGAIN, SINCE 4 BYTES/CHAR
0A70                                    ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
0A70                    GRUP2:          ; FIND_SOURCE
0A70  2A ED                             SUB     CH,CH           ; ZERO TO HIGH OF COUNT REG

0A72  0A DB                             OR      BL,BL           ; # LINES TO SCROLL IS ZERO ?
0A74  75 16                             JNZ     GRUP4           ; IF ZERO, THEN BLANK ENTIRE FIELD

0A76  8A DE                             MOV     BL,DH           ; SET BLANK COUNT TO EVERYTHING IN FIELD
0A78  80 3E 0049 R 19                   CMP     CRT_MODE,19H    ; KJ GRAPHICS 32K REGEN ?
0A7D  72 0A                             JB      GRUP3           ; NO

0A7F  F6 46 01 01                       TEST    BYTE PTR [BP+SU_ULR],1  ; ODD ROW ?
0A83  74 04                             JZ      GRUP3           ; NO
0A85  81 C7 3FB0                        ADD     DI,4000H-80     ; ADJUST POINTER
0A89                    GRUP3:
0A89  E9 0B19 R                         JMP     GRUP12

0A8C                    GRUP4:
0A8C  8A C3                             MOV     AL,BL           ; SAVE VALUE OF BL*1 TO AL
0A8E  D0 E3                             SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 4
0A90  D0 E3                             SAL     BL,1

0A92  80 3E 0049 R 14                   CMP     CRT_MODE,KJGRAPH; KJ GRAPHICS MODE ?
0A97  72 04                             JB      GRUP5           ; NO

0A99  D0 E3                             SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 9
0A9B  02 D8                             ADD     BL,AL           ;
0A9D                    GRUP5:
0A9D  8A C3                             MOV     AL,BL           ; GET NUMBER OF LINES IN AL
0A9F  B4 50                             MOV     AH,80           ; 80 BYTES/ROW
0AA1  F6 E4                             MUL     AH              ; DETERMINE OFFSET TO SOURCE
0AA3  8B F7                             MOV     SI,DI           ; SET UP SOURCE
0AA5  03 F0                             ADD     SI,AX           ;  ADD IN OFFSET TO IT
0AA7  8A E6                             MOV     AH,DH           ; NUMBER OF ROWS IN FIELD
0AA9  2A E3                             SUB     AH,BL           ; DETERMINE NUMBER TO MOVE

0AAB  A0 0049 R                         MOV     AL,CRT_MODE     ; SET GRAPHICS MODE TO AL
0AAE  06                                PUSH    ES              ; GET SEGMENTS BOTH POINTING TO REGEN
0AAF  1F                                POP     DS
                                                ASSUME  DS:VIDEO_RAM

                                        ;---- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD
                                                                ; FIELDS
0AB0  80 7E 00 09                       CMP     BYTE PTR [BP+SUC_MODE],9 ; MODE USES 32K REGEN?
0AB4  73 11                             JAE     GRUP7           ; YES
0AB6                    GRUP6:
0AB6  E8 0D63 R                         CALL    G_MOVE_ROW      ; ROW LOOP
                                                                ; MOVE ONE ROW
0AB9  81 EE 1FB0                        SUB     SI,2000H-80     ; MOVE TO NEXT ROW
0ABD  81 EF 1FB0                        SUB     DI,2000H-80

0AC1  FE CC                             DEC     AH              ; NUMBER OF ROWS TO MOVE
0AC3  75 F1                             JNZ     GRUP6           ; CONTINUE TILL ALL MOVED

0AC5  EB 52                             JMP     SHORT   GRUP12  ; MOVE END
```

```
0AC7
0AC7  3C 14                GRUP7:                          ;--- 32K REGEN
0AC9  72 14                        CMP     AL,KJGRAPH      ; KJ GRAPHICS MODE ?
                                   JB      GRUP9           ; NO

0ACB  F6 46 01 01                  TEST    BYTE PTR [BP+SU_ULR],1  ; ODD ROW ?
0ACF  74 04                        JZ      GRUP8           ; NO
0AD1  81 C7 3FB0                   ADD     DI,4000H-80     ; ADJUST POINTER
0AD5                       GRUP8:
0AD5  F6 46 02 01                  TEST    BYTE PTR [BP+SU_TSR],1  ; ODD ROW ?
0AD9  74 04                        JZ      GRUP9           ; NO
0ADB  81 C6 3FB0                   ADD     SI,4000H-80     ; ADJUST POINTER
0ADF                       GRUP9:

0ADF  80 7E 00 0B                  CMP     BYTE PTR [BP+SUC_MODE],0BH ; 640 X 200 X 16 COLOR ?
0AE3  75 11                        JNE     GRUP91          ; NO
                                                           ;--- SCROLL V-RAM 1
0AE5  57                           PUSH    DI              ;
0AE6  56                           PUSH    SI              ; SAVE REGISTERS
0AE7  1E                           PUSH    DS              ;
0AE8  06                           PUSH    ES              ;

0AE9  8E 5E 04                     MOV     DS,[BP+SU_ES1]  ; GET SEGMENT OF V-RAM 1
0AEC  8E 46 04                     MOV     ES,[BP+SU_ES1]  ;
0AEF  E8 0D63 R                    CALL    G_MOVE_ROW      ; MOVE ROW
0AF2  07                           POP     ES              ;
0AF3  1F                           POP     DS              ; RESTORE REGISTERS
0AF4  5E                           POP     SI              ;
0AF5  5F                           POP     DI              ;

0AF6                       GRUP91:

0AF6  E8 0D63 R                    CALL    G_MOVE_ROW      ; MOVE ONE ROW

0AF9  81 C6 2000                   ADD     SI,2000H        ; NEXT ROW
0AFD  81 FE 8000                   CMP     SI,8000H        ; IN 32K REGEN ?
0B01  72 04                        JB      GRUP10          ; YES

0B03  81 EE 7F60                   SUB     SI,8000H-160    ; NO, WRAP
0B07                       GRUP10:
0B07  81 C7 2000                   ADD     DI,2000H        ; NEXT ROW
0B0B  81 FF 8000                   CMP     DI,8000H        ; IN 32K REGEN ?
0B0F  72 04                        JB      GRUP11          ; YES

0B11  81 EF 7F60                   SUB     DI,8000H-160    ; NO, WRAP
0B15                       GRUP11:
0B15  FE CC                        DEC     AH              ; NUMBER OF ROWS TO MOVE
0B17  75 C6                        JNZ     GRUP9           ; CONTINUE TILL ALL MOVED


0B19                               ;--- FILL IN THE VACATED LINE(S)
0B19  8A C7                GRUP12: MOV     AL,BH           ; CLEAR_ENTRY
0B1B                       GRUP13:                         ; ATTRIBUTE TO FILL WITH

0B1B  80 7E 00 0B                  CMP     BYTE PTR [BP+SUC_MODE],0BH ; 640 X 200 X 16 COLOR ?
0B1F  75 0A                        JNE     GRUP131         ; NO

0B21  57                           PUSH    DI              ; SAVE REGISTERS
0B22  06                           PUSH    ES              ;

0B23  8E 46 04                     MOV     ES,[BP+SU_ES1]  ; GET SEGMENT OF V-RAM 1
0B26  E8 0D7C R                    CALL    G_CLEAR_ROW     ; CLEAR ROW

0B29  07                           POP     ES              ; RESTORE REGISTERS
0B2A  5F                           POP     DI              ;

0B2B                       GRUP131:

0B2B  E8 0D7C R                    CALL    G_CLEAR_ROW     ; CLEAR THAT ROW

0B2E  80 7E 00 09                  CMP     BYTE PTR [BP+SUC_MODE],9 ; MODE USES 32K REGEN?
0B32  72 33                        JC      GRUP15          ; NO, JUMP

0B34  FE CB                        DEC     BL              ; ADJUST COUNT
0B36  74 37                        JZ      GRUP16          ; IF ZERO, THEN DONE

0B38  81 C7 2000                   ADD     DI,2000H
0B3C  81 FF 8000                   CMP     DI,8000H        ; IN 32K REGEN RANGE ?
0B40  72 04                        JB      GRUP14          ; YES

0B42  81 EF 7F60                   SUB     DI,8000H-160    ; ADJUST POINTER
0B46                       GRUP14:
0B46  80 7E 00 0B                  CMP     BYTE PTR [BP+SUC_MODE],0BH ; 640 X 200 X 16 COLOR ?
0B4A  75 0A                        JNE     GRUP141         ; NO
0B4C  57                           PUSH    DI              ; SAVE REGISTERS
0B4D  06                           PUSH    ES              ;

0B4E  8E 46 04                     MOV     ES,[BP+SU_ES1]  ; GET SEGMENT OF V-RAM 1
0B51  E8 0D7C R                    CALL    G_CLEAR_ROW     ; CLEAR ROW

0B54  07                           POP     ES              ; RESTORE REGISTERS
0B55  5F                           POP     DI              ;

0B56                       GRUP141:

0B56  E8 0D7C R                    CALL    G_CLEAR_ROW     ; CLEAR 2 MORE ROWS

0B59  81 C7 3FB0                   ADD     DI,2000H + (2000H-80)
0B5D  81 FF 9FB0                   CMP     DI,8000H + (2000H-80); IN 32K REGEN RANGE ?
0B61  72 04                        JB      GRUP15          ; YES

0B63  81 EF 7F60                   SUB     DI,8000H-160    ; BACK UP POINTERS
0B67                       GRUP15:
```

```
0B67  81 EF 1FB0                SUB     DI,2000H-80     ; POINT TO NEXT LINE
0B6B  FE CB                     DEC     BL              ; NUMBER OF LINES TO FILL
0B6D  75 AC                     JNZ     GRUP13          ; CLEAR_LOOP
0B6F                    GRUP16:
0B6F  C3                        RET                     ; EVERYTHING DONE

0B70                    GRAPHICS_UP     ENDP
                        ;--------------------------------------
                        ;
                        ;       SCROLL_DOWN             INT 10H, AH = 7
                        ;       -----------
                        ;       THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                        ;       BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
                        ;       WITH A DEFINED CHARACTER
                        ;
                        ;       INPUT   AH = CURRENT CRT MODE ( MASKED )
                        ;               AL = NUMBER OF LINES TO SCROLL
                        ;               CX = UPPER LEFT CORNER OF REGION
                        ;               DX = LOWER RIGHT CORNER OF REGION
                        ;               BH = ATTRIBUTE TO BE USED ON BLANKED LINE
                        ;               DS = DATA SEGMENT
                        ;               ES = REGEN SEGMENT
                        ;
                        ;       OUPUT   NONE -- SCREEN IS SCROLLED
                        ;
                        ;       WORK    [BP+SDC_MODE] = CURRENT CRT MODE ( MASKED )
                        ;               [BP+SD_LRR]   = LOWER RIGHT ROW POSITION
                        ;               [BP+SD_BSR]   = BOTTOM OF SOURCE ROW POSITION
                        ;               [BP+SD_ES1]   = SEGMENT OF V-RAM 1
                        ;
                        ;--------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

0B70                    SCROLL_DOWN     PROC    NEAR

0B70  55                        PUSH    BP              ; SAVE BP
0B71  83 EC 06                  SUB     SP,SDN_LOCAL    ; ALLOCATE 2 BYTE FOR LOCAL WORK AREA
0B74  8B EC                     MOV     BP,SP           ; SET BP AS FRAME POINTER

0B76  50                        PUSH    AX              ; SAVE CRT MODE
0B77  1E                        PUSH    DS              ; SAVE DS
0B78  E8 0940 R                 CALL    ERASE_SCURSOR   ; ERASE SOFTWARE CURSOR

0B7B  FD                        STD                     ; DIRECTION FOR SCROLL DOWN
0B7C  8A D8                     MOV     BL,AL           ; LINE COUNT TO BL
0B7E  80 E7 77                  AND     BH,HAN_MASK     ; STRIP KANJI BITS OFF FOR SPACE CODE

0B81  88 66 00                  MOV     BYTE PTR [BP+SDC_MODE],AH; SET MASKED CRT MODE
0B84  80 FC 04                  CMP     AH,GRAPHICS     ; TEST FOR GRAPHICS
0B87  73 05                     JAE     SCRDN1          ; YES, HANDLE SEPARATELY

0B89  E8 0BBF R                 CALL    TEXT_DOWN       ; SCROLL TEXT DOWN
0B8C  EB 27                     JMP     SHORT SCRDN3    ; END

0B8E                    SCRDN1:
0B8E  80 3E 0049 R 14           CMP     CRT_MODE,KJGRAPH; KJ GRAPHICS MODE ?
0B93  72 1D                     JB      SCRDN2          ; NO

0B95  88 76 01                  MOV     BYTE PTR [BP+SD_LRR],DH ; SAVE LOWER RIGHT ROW POSITION
0B98  88 76 02                  MOV     BYTE PTR [BP+SD_BSR],DH ;
0B9B  28 46 02                  SUB     BYTE PTR [BP+SD_BSR],AL ; SAVE BOTTOM OF SOURCE ROW POSITION

0B9E  50                        PUSH    AX              ;
0B9F  53                        PUSH    BX              ;
0BA0  51                        PUSH    CX              ; SAVE REGISTERS
0BA1  52                        PUSH    DX              ;
0BA2  1E                        PUSH    DS              ;
0BA3  06                        PUSH    ES              ;

0BA4  B8 00A8                   MOV     AX,P_CODE_START ; SET PEDUDO CODE BUFFER
0BA7  8E C0                     MOV     ES,AX           ;    TO ES
                                        ASSUME  ES:P_CODE_BUFFER

0BA9  E8 0BBF R                 CALL    TEXT_DOWN       ; SCROLL TEXT DOWN

0BAC  07                        POP     ES              ;
0BAD  1F                        POP     DS              ;
0BAE  5A                        POP     DX              ;
0BAF  59                        POP     CX              ; RESRORE REGISTERS
0BB0  5B                        POP     BX              ;
0BB1  58                        POP     AX              ;
0BB2                    SCRDN2:
0BB2  E8 0BEB R                 CALL    GRAPHICS_DOWN   ; SCROLL DOWN IN GRAPHICS MODE
0BB5                    SCRDN3:
                                        ASSUME  DS: DATA
0BB5  1F                        POP     DS              ; RESTORE DS
0BB6  58                        POP     AX              ; RESTORE CRT MODE
0BB7  E8 0961 R                 CALL    APPEAR_SCURSOR  ; WRITES SOFTWARE CURSOR

0BBA  83 C4 06                  ADD     SP,SDN_LOCAL    ; DEALLOCATE LOCAL WORK AREA
0BBD  5D                        POP     BP              ; RESTORE BP
0BBE  C3                        RET                     ; SCROLL_END

0BBF                    SCROLL_DOWN     ENDP

                        ;--------------------------------------
                        ;
                        ;       TEXT_DOWN
                        ;
                        ;       THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                        ;       BLOCK DOWN ON THE TEXT SCREEN, FILLING THE TOP LINES
                        ;       WITH A DEFINED CHARACTER
```

```
                        ;
                        ;       INPUT   BH = ATTRIBUTE TO BE USED ON BLANKED LINE
                        ;               BL = NUMBER OF LINES TO SCROLL
                        ;               CX = UPPER LEFT CORNER OF REGION
                        ;               DX = LOWER RIGHT CORNER OF REGION
                        ;               DS = DATA SEGMENT
                        ;               ES = REGEN SEGMENT
                        ;
                        ;       OUPUT   NONE
                        ;
                        ;       VOLATILE        AX,BL,CX,DX,SI,DI,DS
                        ;
                        ;--------------------------------------

0BBF                    TEXT_DOWN       PROC    NEAR

0BBF  55                        PUSH    BP              ; SAVE BP
0BC0  53                        PUSH    BX              ; SAVE ATTRIBUTE IN BH

0BC1  8B C2                     MOV     AX,DX           ; LOWER RIGHT CORNER
0BC3  E8 09F5 R                 CALL    SCROLL_POSITION ; GET REGEN LOCATION
0BC6  74 1F                     JZ      TDOWN4
                                        ASSUME  DS:VIDEO_RAM

0BC8  2B F0                     SUB     SI,AX           ; SI IS FROM ADDRESS
0BCA  8A E6                     MOV     AH,DH           ; GET TOTAL # ROWS
0BCC  2A E3                     SUB     AH,BL           ; COUNT TO MOVE IN SCROLL
0BCE                    TDOWN1:
0BCE  E8 0A1B R                 CALL    MOVE_ROW        ; MOVE ONE ROW
0BD1  2B F5                     SUB     SI,BP
0BD3  2B FD                     SUB     DI,BP
0BD5  FE CC                     DEC     AH
0BD7  75 F5                     JNZ     TDOWN1
0BD9                    TDOWN2:
0BD9  58                        POP     AX              ; RECOVER ATTRIBUTE IN AH
0BDA  B0 20                     MOV     AL,' '
0BDC                    TDOWN3:
0BDC  E8 0A24 R                 CALL    CLEAR_ROW       ; CLEAR ONE ROW
0BDF  2B FD                     SUB     DI,BP           ; GO TO NEXT ROW
0BE1  FE CB                     DEC     BL
0BE3  75 F7                     JNZ     TDOWN3

0BE5  5D                        POP     BP              ; RESTORE BP
0BE6  C3                        RET                     ; SCROLL_END

0BE7                    TDOWN4:
0BE7  8A DE                     MOV     BL,DH
0BE9  EB EE                     JMP     SHORT TDOWN2

0BEB                    TEXT_DOWN       ENDP

                        ;--------------------------------------------------
                        ;
                        ;       SCROLL DOWN (GRAPHICS)
                        ;
                        ;       THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                        ;
                        ;       INPUT   [BP+SDC_MODE] = CURRENT CRT MODE ( MASKED )
                        ;               [BP+SD_LRR]   = LOWER RIGHT ROW POSITION
                        ;               [BP+SD_BSR]   = BOTTOM OF SOURCE ROW POSITION
                        ;               CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                        ;               DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                        ;            BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                        ;               BH = FILL VALUE FOR BLANKED LINES
                        ;               AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                        ;               DS = DATA SEGMENT
                        ;               ES = REGEN SEGMENT
                        ;
                        ;       OUTPUT  NOTHING, THE SCREEN IS SCROLLED
                        ;
                        ;       VOLATILE        AX,BL,CX,DX,SI,DI,DS
                        ;
                        ;--------------------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM
0BEB                    GRAPHICS_DOWN   PROC    NEAR

0BEB  FD                        STD             ; SET DIRECTION
0BEC  8A D8                     MOV     BL,AL   ; SAVE LINE COUNT IN BL
0BEE  8B C2                     MOV     AX,DX   ; GET LOWER RIGHT POSITION INTO AX REG
                                ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                                ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
0BF0  E8 15E8 R                 CALL    GRAPH_POSN
0BF3  8B F8                     MOV     DI,AX           ; SAVE RESULT AS DESTINATION
                                                        ; ADDRESS
0BF5  2B D1                                             ;--- DETERMINE SIZE OF WINDOW
0BF7  81 C2 0101                SUB     DX,CX
                                ADD     DX,101H         ; ADJUST VALUES

0BFB  8A E6                     MOV     AH,DH           ; SAVE VALUE OF DH+1 TO AH
0BFD  D0 E6                     SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT DOTS/CHAR
0BFF  D0 E6                     SAL     DH,1            ;  AND EVEN/ODD ROWS

0C01  80 3E 0049 R 14           CMP     CRT_MODE,KJGRAPH; KANJI GRAPHICS MODE ?
0C06  72 04                     JB      GRDN1           ; NO

0C08  D0 E6                     SAL     DH,1            ; MULTIPLY # ROWS BY 9 SINCE 18 VERT DOTS/CHAR
0C0A  02 F4                     ADD     DH,AH           ;
0C0C                    GRDN1:
```

```
0C0C  E8 1D95 R                       CALL    GET_DIRSEG      ; GET DIRECT ACCESS SEGMENT OF V-RAM1
0C0F  89 46 04                        MOV     [BP+SD_ES1],AX  ; SAVE IS

                                              ;--- DETERMINE CRT MODE
0C12  8A 66 00                        MOV     AH,[BP+SDC_MODE]; SET CRT MODE TO AH
0C15  80 FC 06                        CMP     AH,6            ; TEST FOR HIGH RES
0C18  74 21                           JZ      GRDN2           ; FIND_SOURCE_DOWN
                                              ;--- MEDIUM RES DOWN
0C1A  D0 E2                           SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
                                              ; (OFFSET OK)
0C1C  D1 E7                           SAL     DI,1            ; OFFSET *2 SINCE 2 BYTES/CHAR
0C1E  47                              INC     DI              ; POINT TO LAST BYTE

0C1F  80 FC 04                        CMP     AH,4            ; TEST FOR MEDIUM RES
0C22  74 17                           JZ      GRDN2           ; FIND_SOURCE_DOWN
0C24  80 FC 05                        CMP     AH,5            ; TEST FOR MEDIUM RES
0C27  74 12                           JZ      GRDN2           ; FIND_SOURCE_DOWN
0C29  80 FC 0A                        CMP     AH,0AH          ; TEST FOR MEDIUM RES
0C2C  74 0D                           JZ      GRDN2           ; FIND_SOURCE_DOWN

0C2E  80 FC 0B                        cmp     ah,0bh          ; test for 640 x 200 x 16 color
0C31  74 08                           jz      grdn2           ; find_source_down

0C33  4F                              DEC     DI
0C34  D0 E2                           SAL     DL,1            ; # COLUMNS * 2 AGAIN, SINCE 4 BYTES/CHAR
                                              ; (OFFSET OK)
0C36  D1 E7                           SAL     DI,1            ; OFFSET *2 AGAIN, SINCE 4 BYTES/CHAR
0C38  83 C7 03                        ADD     DI,3            ; POINT TO LAST BYTE
                                      ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
0C3B                          GRDN2:                          ; FIND_SOURCE_DOWN
0C3B  2A ED                           SUB     CH,CH           ; ZERO TO HIGH OF COUNT REG

0C3D  80 3E 0049 R 14                 CMP     CRT_MODE,KJGRAPH; KJ GRAPHICS MODE ?
0C42  73 0D                           JAE     GRDN3           ; YES
                                              ;--- ANK GRAPHICS
0C44  80 FC 09                        CMP     AH,9            ; USING 32K REGEN?
0C47  B8 00F0                         MOV     AX,80*(8/2-1)   ; OFFSET TO LAST ROW OF PIXELS IF 16K REGEN
0C4A  72 08                           JC      GRDN4           ; NO, JUMP
0C4C  B8 00A0                         MOV     AX,160*(8/4-1)  ; OFFSET TO LAST ROW OF PIXELS IF 32K REGEN

0C4F  EB 03                           JMP     SHORT   GRDN4

0C51                          GRDN3:                          ;--- KJ GRAPHICS
0C51  B8 0280                         MOV     AX,80*(18/2-1)  ; OFFSET TO LAST ROW OF PIXELS
0C54                          GRDN4:
0C54  03 F8                           ADD     DI,AX           ; POINT TO LAST ROW OF PIXELS

0C56  0A DB                           OR      BL,BL           ; # LINES TO SCROLL IS ZERO ?
0C58  75 19                           JNZ     GRDN7

0C5A  8A DE                           MOV     BL,DH           ; SET ENTIRE FIELD
0C5C  80 3E 0049 R 19                 CMP     CRT_MODE,19H    ; KJ GRAPHICS 32K REGEN ?
0C61  72 0D                           JB      GRDN6

0C63  F6 46 01 01                     TEST    BYTE PTR [BP+SD_LRR],1  ; ODD ROW ?
0C67  74 04                           JZ      GRDN5           ; YES
0C69  81 C7 3FB0                      ADD     DI,4000H-80     ; ADJUST POINTER
0C6D                          GRDN5:
0C6D  E9 0D19 R                       JMP     GRDN16
0C70                          GRDN6:
0C70  E9 0D41 R                       JMP     GRDN19

0C73                          GRDN7:
0C73  8A C3                           MOV     AL,BL           ; SAVE VALUE OF BL*1 TO AL
0C75  D0 E3                           SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 4
0C77  D0 E3                           SAL     BL,1

0C79  80 3E 0049 R 14                 CMP     CRT_MODE,KJGRAPH; KANJI GRAPHICS MODE ?
0C7E  72 04                           JB      GRDN8           ; NO

0C80  D0 E3                           SAL     BL,1            ; MULTIPLY NUMBER OF LINES BY 9
0C82  02 D8                           ADD     BL,AL           ;
0C84                          GRDN8:
0C84  8A C3                           MOV     AL,BL           ; GET NUMBER OF LINES IN AL
0C86  B4 50                           MOV     AH,80           ; 80 BYTES/ROW
0C88  F6 E4                           MUL     AH              ; DETERMINE OFFSET TO SOURCE
0C8A  8B F7                           MOV     SI,DI           ; SET UP SOURCE
0C8C  2B F0                           SUB     SI,AX           ;  SUBTRACT THE OFFSET
0C8E  8A E6                           MOV     AH,DH           ; NUMBER OF ROWS IN FIELD
0C90  2A E3                           SUB     AH,BL           ; DETERMINE NUMBER TO MOVE

0C92  80 3E 0049 R 14                 CMP     CRT_MODE,KJGRAPH; KJ GRAPHICS MODE ?
0C97  06                              PUSH    ES              ; BOTH SEGMENTS TO REGEN
0C98  1F                              POP     DS
                                              ASSUME  DS:VIDEO_RAM
0C99  72 06                           JB      GRDN9           ; NO
                                              ;--- LOOP THROUGH, MOVING ONE ROW
0C9B  80 7E 00 09                     CMP     BYTE PTR [BP+SDC_MODE],9 ; MODE USES 32K REGEN ?
0C9F  73 2C                           JAE     GRDN11                   ; YES
0CA1                          GRDN9:                          ; ROW_LOOP_DOWN
0CA1  E8 0D63 R                       CALL    G_MOVE_ROW      ; MOVE ONE ROW

0CA4  80 7E 00 09                     CMP     BYTE PTR [BP+SDC_MODE],9 ; MODE USES 32K REGEN?
0CA8  72 15                           JC      GRDN10          ; NO, JUMP

0CAA  81 C6 2000                      ADD     SI,2000H        ; ADJUST POINTERS
0CAE  81 C7 2000                      ADD     DI,2000H
0CB2  E8 0D63 R                       CALL    G_MOVE_ROW      ; MOVE 2 MORE ROWS

0CB5  81 EE 4050                      SUB     SI,4000H+80     ; BACK UP POINTERS
0CB9  81 EF 4050                      SUB     DI,4000H+80     ;
0CBD  FE CC                           DEC     AH              ; ADJUST COUNT
```

```
0CBF
0CBF  81 EE 2050          GRDN10:
0CC3  81 EF 2050                  SUB     SI,2000H+80     ; MOVE TO NEXT ROW
                                  SUB     DI,2000H+80

0CC7  FE CC                       DEC     AH              ; NUMBER OF ROWS TO MOVE
0CC9  75 D6                       JNZ     GRDN9           ; CONTINUE TILL ALL MOVED
0CCB  EB 74                       JMP     SHORT GRDN19
0CCD                      GRDN11:                         ;--- 32K REGEN  KJ MODE
0CCD  F6 46 01 01                 TEST    BYTE PTR [BP+SD_LRR],1  ; ODD ROW ?
0CD1  74 04                       JZ      GRDN12          ; YES
0CD3  81 C7 3FB0                  ADD     DI,4000H-80     ; ADJUST POINTER
0CD7                      GRDN12:
0CD7  F6 46 02 01                 TEST    BYTE PTR [BP+SD_BSR],1  ; ODD ROW ?
0CDB  74 04                       JZ      GRDN13          ; YES
0CDD  81 C6 3FB0                  ADD     SI,4000H-80     ; ADJUST POINTER
0CE1                      GRDN13:

0CE1  80 7E 00 0B                 CMP     BYTE PTR [BP+SDC_MODE],0BH ; 640 X 200 X 16 COLOR ?
0CE5  75 11                       JNE     GRDN131         ; NO
                                                          ;--- SCROLL V-RAM 1
0CE7  57                          PUSH    DI              ;
0CE8  56                          PUSH    SI              ; SAVE REGISTERS
0CE9  1E                          PUSH    DS              ;
0CEA  06                          PUSH    ES              ;

0CEB  8E 5E 04                    MOV     DS,[BP+SD_ES1]  ; GET SEGMENT OF V-RAM 1
0CEE  8E 46 04                    MOV     ES,[BP+SD_ES1]  ;
0CF1  E8 0D63 R                   CALL    G_MOVE_ROW      ; MOVE ROW

0CF4  07                          POP     ES              ;
0CF5  1F                          POP     DS              ; RESTORE REGISTERS
0CF6  5E                          POP     SI              ;
0CF7  5F                          POP     DI              ;
0CF8                      GRDN131:
0CF8  E8 0D63 R                   CALL    G_MOVE_ROW      ; MODE ONE ROW

0CFB  81 EE 6000                  SUB     SI,6000H        ; NEXT ROW
0CFF  83 FE 00                    CMP     SI,0000H        ; IN 32K REGEN ?
0D02  7D 04                       JGE     GRDN14          ; YES
0D04  81 C6 7F60                  ADD     SI,8000H-160    ; NO, WRAP
0D08                      GRDN14:
0D08  81 EF 6000                  SUB     DI,6000H        ; NEXT ROW
0D0C  83 FF 00                    CMP     DI,0000H        ; IN 32K REGEN ?
0D0F  7D 04                       JGE     GRDN15          ; YES
0D11  81 C7 7F60                  ADD     DI,8000H-160    ; NO, WRAP
0D15                      GRDN15:
0D15  FE CC                       DEC     AH              ; ADJUST COUNT
0D17  75 C8                       JNZ     GRDN13          ; CONTINUE TILL ALL MOVED
0D19                      GRDN16:
0D19  8A C7                       MOV     AL,BH           ; ATTRIBUTE TO FILL WITH
0D1B                      GRDN17:

0D1B  80 7E 00 0B                 CMP     BYTE PTR [BP+SDC_MODE],0BH ; 640 X 200 X 16 COLOR ?
0D1F  75 0A                       JNE     GRDN171         ; NO

0D21  57                          PUSH    DI              ; SAVE REGISTERS
0D22  06                          PUSH    ES              ;

0D23  8E 46 04                    MOV     ES,[BP+SD_ES1]  ; GET SEGMENT OF V-RAM 1
0D26  E8 0D7C R                   CALL    G_CLEAR_ROW     ; CLEAR ROW

0D29  07                          POP     ES              ; RESTORE REGISTERS
0D2A  5F                          POP     DI              ;
0D2B                      GRDN171:
0D2B  E8 0D7C R                   CALL    G_CLEAR_ROW     ; CLEAR A ROW

0D2E  81 EF 6000                  SUB     DI,6000H        ; NEXT ROW
0D32  83 FF 00                    CMP     DI,000H         ; IN 32K REGEN ?
0D35  7D 04                       JGE     GRDN18          ; YES
0D37  81 C7 7F60                  ADD     DI,8000H-160    ; NO, WRAP
0D3B                      GRDN18:
0D3B  FE CB                       DEC     BL              ; NUMBER OF LINES TO FILL
0D3D  75 DC                       JNZ     GRDN17          ; CLEAR ROW LOOP DOWN BY 2 ROW

0D3F  EB 20                       JMP     SHORT   GRDN22  ; DONE


0D41                      GRDN19:                         ;------ FILL IN THE VACATED LINE(S)
0D41  8A C7                       MOV     AL,BH           ; CLEAR_ENTRY_DOWN
0D43                      GRDN20:                         ; ATTRIBUTE TO FILL WITH
0D43  E8 0D7C R                   CALL    G_CLEAR_ROW     ; CLEAR_LOOP_DOWN
                                                          ; CLEAR A ROW
0D46  80 7E 00 09                 CMP     BYTE PTR [BP+SDC_MODE],9 ; MODE USES 32K REGEN?
0D4A  72 0D                       JC      GRDN21          ; NO, JUMP

0D4C  81 C7 2000                  ADD     DI,2000H        ; NEXT ROW
0D50  E8 0D7C R                   CALL    G_CLEAR_ROW     ; CLEAR 2 MORE ROWS

0D53  81 EF 4050                  SUB     DI,4000H+80     ; BACK UP POINTERS
0D57  FE CB                       DEC     BL              ; ADJUST COUNT
0D59                      GRDN21:
0D59  81 EF 2050                  SUB     DI,2000H+80     ; POINT TO NEXT LINE
0D5D  FE CB                       DEC     BL              ; NUMBER OF LINES TO FILL
0D5F  75 E2                       JNZ     GRDN20          ; CLEAR_LOOP_DOWN
0D61                      GRDN22:
0D61  FC                          CLD                     ; RESET THE DIRECTION FLAG
0D62  C3                          RET                     ; EVERYTHING DONE

0D63                      GRAPHICS_DOWN   ENDP

                          ;-----------------------------------------------
                          ;
                          ;       MOVE ONE ROW ( GRAPHICS )
```

```
                        ;
                        ;       THIS ROUTINE MOVES ONE GRAPHICS ROW
                        ;
                        ;       INPUT           DL =    MOVE COUNT IN BYTE
                        ;                       DS:SI = SOURCE ROW ADDRESS
                        ;                       ES:DI = DESTINATION ROW ADDRESS
                        ;
                        ;       OUTPUT          SI =    SI + 2000H
                        ;                       DI =    DI + 2000H
                        ;
                        ;       VOLATILE        CL
                        ;
                        ;-----------------------------------------------

0D63                    G_MOVE_ROW      PROC    NEAR

0D63  8A CA                     MOV     CL,DL           ; NUMBER OF BYTES IN THE ROW
0D65  56                        PUSH    SI
0D66  57                        PUSH    DI              ; SAVE POINTERS
0D67  F3/ A4                    REP     MOVSB           ; MOVE THE EVEN FIELD
0D69  5F                        POP     DI
0D6A  5E                        POP     SI
0D6B  81 C6 2000                ADD     SI,2000H
0D6F  81 C7 2000                ADD     DI,2000H        ; POINT TO THE ODD FIELD

0D73  56                        PUSH    SI
0D74  57                        PUSH    DI              ; SAVE THE POINTERS
0D75  8A CA                     MOV     CL,DL           ; COUNT BACK
0D77  F3/ A4                    REP     MOVSB           ; MOVE THE ODD FIELD
0D79  5F                        POP     DI
0D7A  5E                        POP     SI              ; POINTERS BACK

0D7B  C3                        RET                     ; RETURN TO CALLER

0D7C                    G_MOVE_ROW      ENDP

                        ;-----------------------------------------------
                        ;
                        ;       CLEAR ONE ROW ( GRAPHICS )
                        ;
                        ;       THIS ROUTINE MOVES ONE GRAPHICS ROW
                        ;
                        ;       INPUT           DL =    CLEAR COUNT IN BYTE
                        ;                       ES:DI = DESTINATION ROW ADDRESS
                        ;
                        ;       OUTPUT          DI =    DI + 2000H
                        ;
                        ;       VOLATILE        CL
                        ;
                        ;-----------------------------------------------

0D7C                    G_CLEAR_ROW     PROC    NEAR

0D7C  8A CA                     MOV     CL,DL           ; NUMBER OF BYTES IN FIELD
0D7E  57                        PUSH    DI              ; SAVE POINTER
0D7F  F3/ AA                    REP     STOSB           ; STORE THE NEW VALUE
0D81  5F                        POP     DI              ; POINTER BACK

0D82  81 C7 2000                ADD     DI,2000H        ; POINT TO ODD FIELD

0D86  57                        PUSH    DI
0D87  8A CA                     MOV     CL,DL
0D89  F3/ AA                    REP     STOSB           ; FILL THE ODD FILELD
0D8B  5F                        POP     DI

0D8C  C3                        RET                     ; RETURN TO CALLER

0D8D                    G_CLEAR_ROW     ENDP
                        ;-----------------------------------------
                        ;
                        ;       READ_AC_CURRENT                 INT 10H, AH = 8
                        ;       ---------------
                        ;       THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE
                        ;       CURRENT CURSOR POSITION AND RETURNS THEM TO THE CALLER
                        ;
                        ;       INPUT   AH =    CURRENT CRT MODE (MASKED)
                        ;               BH =    DISPLAY PAGE ( ALPHA MODES ONLY )
                        ;               DS =    DATA SEGMENT
                        ;               ES =    REGEN SEGMENT
                        ;
                        ;       OUTPUT  AL =    CHAR READ
                        ;               AH =    ATTRIBUTE READ
                        ;
                        ;-----------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

0D8D                    READ_AC_CURRENT PROC    NEAR

0D8D  80 3E 0049 R 04           CMP     CRT_MODE,GRAPHICS;IS THIS ANK TEXT MODE ?
0D92  72 43                     JC      RDACC4          ; YES

0D94  80 3E 0049 R 14           CMP     CRT_MODE,KJGRAPH; KANJI GRAPHICS MODE ?
0D99  73 0C                     JAE     RDACC1          ; YES

0D9B  80 3E 0049 R 10           CMP     CRT_MODE,KJ_MODE; KANJI TEXT MODE ?
0DA0  73 0C                     JAE     RDACC2          ; YES

0DA2  E8 0DFC R                 CALL    GRAPHICS_READ   ; READ ANK GRAPHICS
0DA5  EB 38                     JMP     SHORT RDACC5    ; END

0DA7                    RDACC1:
0DA7  B9 00A0                   MOV     CX,P_CODE_START ; SET PEDUDO CODE BUFFER SEGMENT
0DAA  8E C1                     MOV     ES,CX           ;   TO ES
```

```
0DAC  32 FF                     XOR     BH,BH           ; CLEAR DISPLAY PAGE
0DAE                    RDACC2:                         ; READ_AC_CONTINUE
0DAE  E8 0DE0 R                 CALL    FIND_POSITION
0DB1  8B F3                     MOV     SI,BX           ; ESTABLISH ADDRESSING IN SI
0DB3  06                        PUSH    ES              ;
0DB4  1F                        POP     DS              ; GET SEGMENT FOR QUICK ACCESS
                                        ASSUME  DS:VIDEO_RAM

0DB5  AD                        LODSW                   ; GET THE CHAR/ATTR

0DB6  F6 C4 80                  TEST    AH,ZENBIT       ; 2 BYTE CODE ?
0DB9  74 24                     JZ      RDACC5          ; NO

0DBB  F6 C4 08                  TEST    AH,ZEN2BIT      ; 2ND BYTE CODE ?
0DBE  75 0B                     JNZ     RDACC3          ; YES
                                                        ;--- 1ST BYTE
0DC0  8A E8                     MOV     CH,AL           ; SET 1ST CHARACTER'S CODE TO CH
0DC2  8A 0C                     MOV     CL,DS:[SI]      ; SET 2ND CHARACTER'S CODE TO CL
0DC4  E8 1B42 R                 CALL    CHECK_ROSS_CODE ; CHECK AND CONVERT ROSSIAN CHARACTER CODE

0DC7  8A C5                     MOV     AL,CH           ; SET 1ST BYTE OF CONVERTED CODE
0DC9  EB 14                     JMP     SHORT RDACC5    ; END

0DCB                    RDACC3:                         ;--- 2ND BYTE
0DCB  8A C8                     MOV     CL,AL           ; SET 2ND CHARACTER'S CODE TO CL
0DCD  8A 6C FC                  MOV     CH,DS:[SI-4]    ; SET 1ST CHARACTER'S CODE TO CH
0DD0  E8 1B42 R                 CALL    CHECK_ROSS_CODE ; CHECK AND CONVERT ROSSIAN CHARACTER CODE

0DD3  8A C1                     MOV     AL,CL           ; SET 2ND BYTE OF CONVERTED CODE
0DD5  EB 08                     JMP     SHORT RDACC5

0DD7                                    ASSUME  DS:DATA
0DD7                    RDACC4:                         ; READ_AC_CONTINUE
0DD7  E8 0DE0 R                 CALL    FIND_POSITION
0DDA  8B F3                     MOV     SI,BX           ; ESTABLISH ADDRESSING IN SI
0DDC  06                        PUSH    ES              ;
0DDD  1F                        POP     DS              ; GET SEGMENT FOR QUICK ACCESS
0DDE  AD                        LODSW                   ; GET THE CHAR/ATTR
0DDF                    RDACC5:
0DDF  C3                        RET

0DE0                    READ_AC_CURRENT ENDP

                        ;-----------------------------------------
                        ;
                        ;       FIND_POSITION
                        ;
                        ;       THIS ROUTINE DETERMINES THE REGEN ADDRESS FORM
                        ;       CURRENT CURSOR POSITION
                        ;
                        ;       INPUT   BH =    DISPLAY PAGE
                        ;
                        ;       OUTPUT  BX =    REGEN ADDRESS CORRESPONDING TO
                        ;                               CURRENT CURSOR POSITION
                        ;
                        ;       VOLATILE        AX,CX
                        ;
                        ;-----------------------------------------
                                        ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM
0DE0                    FIND_POSITION   PROC    NEAR

0DE0  8A CF                     MOV     CL,BH           ; DISPLAY PAGE TO CX
0DE2  32 ED                     XOR     CH,CH
0DE4  8B F1                     MOV     SI,CX
0DE6  D1 E6                     SAL     SI,1            ; MOVE TO SI FOR INDEX
0DE8  8B 84 035C R              MOV     AX,[SI+ OFFSET CURSOR_POSN] ; * 2 FOR WORD OFFSET
                                                        ; GET ROW/COLUMN OF THAT PAGE
0DEC                    FIND_POSN       LABEL   NEAR    ; CALLED FROM WRITE_ALT_CURSOR
0DEC  33 DB                     XOR     BX,BX           ; SET START ADDRESS TO ZERO
0DEE  E3 06                     JCXZ    FPOS2           ; NO_PAGE
0DF0                    FPOS1:                          ; PAGE_LOOP
0DF0  03 1E 004C R              ADD     BX,CRT_LEN      ; LENGTH OF BUFFER
0DF4  E2 FA                     LOOP    FPOS1
0DF6                    FPOS2:
0DF6  E8 06D5 R                 CALL    POSITION        ; NO_PAGE
0DF9  03 D8                     ADD     BX,AX           ; DETERMINE LOCATION IN REGEN
                                                        ; ADD TO START OF REGEN
0DFB  C3                        RET

0DFC                    FIND_POSITION   ENDP

                        ;-----------------------------------------
                        ;
                        ;       GRAPHICS READ
                        ;
                        ;       THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
                        ;       POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO
                        ;       THE CHARACTER GENERATOR CODE POINTS
                        ;
                        ;       INPUT   AH =    CRT MODE ( MASKED )
                        ;
                        ;       INPUT   NONE  (0 IS ASSUMED AS THE BACKGROUND COLOR)
                        ;
                        ;       OUTPUT  AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
                        ;
                        ;       NOTE    0 IS ASSUMED AS THE BACKGROUND COLOR
                        ;
                        ;       VOLATILE        BX,CX,DX,SI,DI,DS,ES
                        ;
                        ;-----------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM
```

```
0DFC                    GRAPHICS_READ   PROC    NEAR

0DFC  55                        PUSH    BP                      ; SAVE BP
0DFD  83 EC 10                  SUB     SP,RAC_LOCAL            ; ALLOCATE SPACE TO SAVE THE READ CODE POINT
0E00  8B EC                     MOV     BP,SP                   ; POINTER TO SAVE AREA

0E02  50                        PUSH    AX                      ; SAVE CRT MODE

0E03  E8 15E5 R                 CALL    GRAPH_POSITION          ; CONVERTED TO OFFSET IN REGEN
0E06  8B F0                     MOV     SI,AX                   ; SAVE IN SI
                                                                ;------ DETERMINE GRAPHICS MODES
0E08  58                        POP     AX                      ; RESTORE CRT MODE
0E09  06                        PUSH    ES
0E0A  1F                        POP     DS                      ; POINT TO REGEN SEGMENT
                                        ASSUME DS:VIDEO_RAM
0E0B  B6 04                     MOV     DH,4                    ; NUMBER OF PASSES

0E0D  80 FC 06                  CMP     AH,6
0E10  74 11                     JZ      GREAD1                  ; HIGH RESOLUTION

0E12  80 FC 04                  CMP     AH,4
0E15  74 54                     JZ      GREAD5                  ; MEDIUM RESOLUTION

0E17  80 FC 05                  CMP     AH,5
0E1A  74 4F                     JZ      GREAD5                  ; MEDIUM RESOLUTION

0E1C  80 FC 0A                  CMP     AH,0AH
0E1F  74 4A                     JZ      GREAD5                  ; MEDIUM RESOLUTION

0E21  EB 17                     JMP     SHORT GREAD2            ; LOW RESOLUTION

                                                                ;------ HIGH RESOLUTION READ
0E23                                    ;-GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
0E23                    GREAD1:
0E23  8A 04                     MOV     AL,[SI]                 ; GET FIRST BYTE
0E25  88 46 00                  MOV     [BP],AL                 ; SAVE IN STORAGE AREA
0E28  45                        INC     BP                      ; NEXT LOCATION

0E29  8A 84 2000                MOV     AL,[SI+2000H]           ; GET LOWER REGION BYTE
0E2D  88 46 00                  MOV     [BP],AL                 ; ADJUST AND STORE
0E30  45                        INC     BP

0E31  83 C6 50                  ADD     SI,80                   ; POINTER INTO REGEN
0E34  FE CE                     DEC     DH                      ; LOOP CONTROL
0E36  75 EB                     JNZ     GREAD1                  ; DO IT SOME MORE

0E38  EB 5E                     JMP     SHORT GREAD8            ; GO MATCH THE SAVED CODE POINTS

0E3A                                                            ;------ LOW RESOLUTION READ
0E3A                    GREAD2:
0E3A  D1 E6                     SAL     SI,1                    ; OFFSET*4 SINCE 4 BYTES/CHAR
0E3C  D1 E6                     SAL     SI,1
0E3E                    GREAD3:
0E3E  E8 0F03 R                 CALL    LOW_READ_BYTE           ; GET 4 BYTES FROM REGEN INTO SINGLE SAVE
0E41  81 C6 2000                ADD     SI,2000H                ; GOTO LOWER REGION
0E45  E8 0F03 R                 CALL    LOW_READ_BYTE           ; GET 4 BYTES FROM REGEN INTO SINGLE SAVE

0E48  80 FC 09                  CMP     AH,9                    ; DO WE HAVE A 32K REGEN AREA?
0E4B  75 14                     JNE     GREAD4                  ; NO, JUMP

0E4D  81 C6 2000                ADD     SI,2000H                ; GOTO LOWER REGION
0E51  E8 0F03 R                 CALL    LOW_READ_BYTE           ; GET 4 BYTES FROM REGEN INTO SINGLE SAVE

0E54  81 C6 2000                ADD     SI,2000H                ; GOTO LOWER REGION
0E58  E8 0F03 R                 CALL    LOW_READ_BYTE           ; GET 4 BYTES FROM REGEN INTO SINGLE SAVE

0E5B  81 EE 3FB0                SUB     SI,4000H-80             ; ADJUST POINTER
0E5F  FE CE                     DEC     DH
0E61                    GREAD4:
0E61  81 EE 1FB0                SUB     SI,2000H-80             ; ADJUST POINTER BACK TO UPPER
0E65  FE CE                     DEC     DH
0E67  75 D5                     JNZ     GREAD3                  ;DO IT SOME MORE

0E69  EB 2D                     JMP     SHORT GREAD8            ; GO MATCH THE SAVED CODE POINTS

0E6B                                                            ;------ MEDIUM RESOLUTION READ
0E6B                    GREAD5:
0E6B  D1 E6                     SAL     SI,1                    ; MED_RES_READ
0E6D                                                            ; OFFSET*2 SINCE 2 BYTES/CHAR
0E6D                    GREAD6:
0E6D  E8 0ECF R                 CALL    MID_READ_BYTE           ; GET PAIR BYTES FROM REGEN INTO SINGLE SAVE
0E70  81 C6 2000                ADD     SI,2000H                ; GO TO LOWER REGION
0E74  E8 0ECF R                 CALL    MID_READ_BYTE           ; GET THIS PAIR INTO SAVE

0E77  80 FC 0A                  CMP     AH,0AH                  ; DO WE HAVE A 32K REGEN AREA?
0E7A  75 14                     JNE     GREAD7                  ; NO, JUMP

0E7C  81 C6 2000                ADD     SI,2000H                ; GOTO LOWER REGION
0E80  E8 0ECF R                 CALL    MID_READ_BYTE           ; GET PAIR BYTES FROM REGEN INTO SINGLE SAVE

0E83  81 C6 2000                ADD     SI,2000H                ; GOTO LOWER REGION
0E87  E8 0ECF R                 CALL    MID_READ_BYTE           ; GET PAIR BYTES FROM REGEN INTO SINGLE SAVE

0E8A  81 EE 3FB0                SUB     SI,4000H-80             ; ADJUST POINTER
0E8E  FE CE                     DEC     DH
0E90                    GREAD7:
0E90  81 EE 1FB0                SUB     SI,2000H-80             ; ADJUST POINTER BACK INTO UPPER
0E94  FE CE                     DEC     DH

0E96  75 D5                     JNZ     GREAD6                  ; KEEP GOING UNTIL ALL 8 DONE

0E98                                                            ;--- SAVE AREA HAS CHARACTER IN IT, MATCH IT
0E98                    GREAD8:                                 ; FIND CHAR
0E98  8B DD                     MOV     BX,BP                   ; SET POINTER TO FONT PATTERN
0E9A  83 ED 08                  SUB     BP,8                    ; ADJUST POINTER TO BEGINNING OF SAVE AREA
```

```
0E9D  8B F5                   MOV     SI,BP
0E9F  FC                      CLD                     ; ENSURE DIRECTION
0EA0  33 C0                   XOR     AX,AX           ; CURRENT CODE POINT BEING MATCHED

0EA2  16                      PUSH    SS              ; ESTABLISH ADDRESSING TO STACK
0EA3  1F                      POP     DS              ; FOR THE STRING COMPARE
0EA4  16                      PUSH    SS              ; ESTABLISH ADDRESSING TO STACK
0EA5  07                      POP     ES              ; FOR THE STRING COMPARE
                                      ASSUME  DS:STACK, ES:STACK
0EA6  BA 0100                 MOV     DX,256          ; NUMBER TO TEST AGAINST
0EA9                  GREAD9:
0EA9  56                      PUSH    SI              ; SAVE AREA POINTER

0EAA  50                      PUSH    AX              ; SAVE AX
0EAB  56                      PUSH    SI              ; SAVE SI

0EAC  32 ED                   XOR     CH,CH           ; CLEAR FOR HANKAKU FONT
0EAE  8A C8                   MOV     CL,AL           ; SET CHARACTER CODE
0EB0  32 C0                   XOR     AL,AL           ; SET REQUEST BASE FONT FUNCTION

0EB2  52                      PUSH    DX              ; SAVE DX
0EB3  B6 00                   MOV     DH,0            ; INDICATES ANK MODE
0EB5  E8 1A63 R               CALL    FONT            ; DO IT
0EB8  5A                      POP     DX              ; RESTORE DX
0EB9  8B FB                   MOV     DI,BX           ; SET FONT TOP ADDRESS TO DI

0EBB  5E                      POP     SI              ; RESTORE SI
0EBC  58                      POP     AX              ; RESTORE AX

0EBD  B9 0008                 MOV     CX,8            ; NUMBER OF BYTES TO MATCH
0EC0  F3/ A6                  REPE    CMPSB           ; COMPARE THE 8 BYTES

0EC2  5E                      POP     SI
0EC3  74 05                   JZ      GREAD10         ; IF ZERO FLAG SET, THEN MATCH OCCURRED

0EC5  FE C0                   INC     AL
0EC7  4A                      DEC     DX              ; NO MATCH, MOVE ON TO NEXT
0EC8  75 DF                   JNZ     GREAD9          ; LOOP CONTROL
                                                      ; DO ALL OF THEM
0ECA                  GREAD10:                        ;---CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
0ECA  83 C4 10                ADD     SP,RAC_LOCAL    ; READJUST THE STACK, THROW AWAY WORK AREA
0ECD  5D                      POP     BP              ; RESTORE BP

0ECE  C3                      RET                     ; ALL DONE

0ECF                  GRAPHICS_READ   ENDP

                      ;-----------------------------------------------
                      ;
                      ;       MED_READ_BYTE
                      ;
                      ;       THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
                      ;        COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
                      ;        THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                      ;        POSITION IN THE SAVE AREA
                      ;
                      ;       INPUT           AH     = CRT MODE ( MASKED )
                      ;                       SI,DS = POINTER TO REGEN AREA OF INTEREST
                      ;                       BX = EXPANDED FOREGROUND COLOR
                      ;                       BP = POINTER TO SAVE AREA
                      ;
                      ;       OUTPUT          BP = BP+1
                      ;
                      ;       VOLATILE        CX,DL
                      ;
                      ;-----------------------------------------------

0ECF
0ECF  50              MID_READ_BYTE   PROC    NEAR
0ED0  80 FC 0A                PUSH    AX
                              CMP     AH,0AH          ; SAVE CURRENT AX
                                                      ; IN 640X200 4 COLOR MODE?
0ED3  8A 24                   MOV     AH,[SI]
0ED5  8A 44 01                MOV     AL,[SI+1]       ; GET FIRST BYTE
                                                      ; GET SECOND BYTE
0ED8  75 11                   JNE     MRBYTE2         ; NO, JUMP

                      ; IN 640X200 4 COLOR MODE, ALL THE c0 BITS ARE IN ONE BYTE, AND ALL
                      ; THE c1 BITS ARE IN THE NEXT BYTE. HERE WE CHANGE THEM BACK TO
                      ; NORMAL c1c0 ADJACENT PAIRS.

0EDA  53                      PUSH    BX              ; SAVE REG
0EDB  B9 0008                 MOV     CX,8            ; SET LOOP COUNTER
0EDE                  MRBYTE1:
0EDE  D0 FC                   SAR     AH,1            ; c0 BIT INTO CARRY
0EE0  D1 DB                   RCR     BX,1            ;  AND INTO BX

0EE2  D0 F8                   SAR     AL,1            ; c1 BIT INTO CARRY
0EE4  D1 DB                   RCR     BX,1            ;  AND INTO BX
0EE6  E2 F6                   LOOP    MRBYTE1         ; REPEAT

0EE8  8B C3                   MOV     AX,BX           ; RESULT INTO AX
0EEA  5B                      POP     BX              ; RESTORE BX
0EEB                  MRBYTE2:
0EEB  B9 C000                 MOV     CX,0C000H       ; 2 BIT MASK TO TEST THE ENTRIES
0EEE  32 D2                   XOR     DL,DL           ; RESULT REGISTER
0EF0                  MRBYTE3:
0EF0  85 C1                   TEST    AX,CX           ; IS THIS SECTION BACKGROUND?
0EF2  74 01                   JZ      MRBYTE4         ; IF ZERO, IT IS BACKGROUND

0EF4  F9                      STC                     ; WASN'T, SO SET CARRY
0EF5                  MRBYTE4:
0EF5  D0 D2                   RCL     DL,1            ; MOVE THAT BIT INTO THE RESULT
```

```
0EF7  D1 E9                     SHR     CX,1
0EF9  D1 E9                     SHR     CX,1            ; MOVE THE MASK TO THE RIGHT BY 2 BITS
0EFB  73 F3                     JNC     MRBYTE3         ; DO IT AGAIN IF MASK DIDN'T FALL OUT

0EFD  88 56 00                  MOV     [BP],DL         ; STORE RESULT IN SAVE AREA
0F00  45                        INC     BP              ; ADJUST POINTER

0F01  58                        POP     AX              ; RESTORE AX
0F02  C3                        RET                     ; ALL DONE

0F03                    MID_READ_BYTE   ENDP

                        ;--------------------------------------------------
                        ;
                        ;       LOW_READ_BYTE
                        ;
                        ;       THIS ROUTINE WILL TAKE 4 BYTES FROM THE REGEN BUFFER,
                        ;        COMPARE FOR BACKGROUND COLOR, AND PLACE
                        ;        THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                        ;        POSITION IN THE SAVE AREA
                        ;
                        ;       INPUT           SI,DS = POINTER TO REGEN AREA OF INTEREST
                        ;                       BP = POINTER TO SAVE AREA
                        ;
                        ;       OUTPUT          BP = BP + 1
                        ;
                        ;       VOLATILE        CX,DL
                        ;
                        ;--------------------------------------------------

0F03                    LOW_READ_BYTE   PROC    NEAR

0F03  50                        PUSH    AX              ; SAVE CURRENT AX

0F04  8A 24                     MOV     AH,[SI]         ;GET FIRST 2 BYTES
0F06  8A 44 01                  MOV     AL,[SI+1]

0F09  32 D2                     XOR     DL,DL
0F0B  E8 0F1D R                 CALL    BUILD_NIBBLE    ;BUILD HIGH NIBBLE

0F0E  8A 64 02                  MOV     AH,[SI+2]       ;GET SECOND 2 BYTES
0F11  8A 44 03                  MOV     AL,[SI+3]
0F14  E8 0F1D R                 CALL    BUILD_NIBBLE    ;BUILD LOW NIBBLE

0F17  88 56 00                  MOV     [BP],DL         ;STORE RESULT IN SAVE AREA
0F1A  45                        INC     BP              ;ADJUST POINTER

0F1B  58                        POP     AX              ; RESTORE AX
0F1C  C3                        RET

0F1D                    LOW_READ_BYTE   ENDP

                        ;--------------------------------------------------
                        ;
                        ;       BUILD_NIBBLE
                        ;
                        ;       THIS ROUTINE WILL TAKE 1 WORD FROM THE REGEN BUFFER,
                        ;        AND MAKE 4 BIT PATTERN FROM THAT.
                        ;
                        ;       INPUT           AX =    ANY WORD OF REGEN BUFFER
                        ;
                        ;       OUTPUT          DL =    DOT PATTERN
                        ;                               (LOW NIBBLE, HIGH IS LOW NIBBLE OF INPUT DL)
                        ;
                        ;       VOLATILE        CX
                        ;
                        ;--------------------------------------------------

0F1D                    BUILD_NIBBLE    PROC    NEAR

0F1D  B9 F000           BLDN1:  MOV     CX,0F000H       ; 4 BIT MASK TO TEST THE ENTRIES
0F20  85 C1                     TEST    AX,CX           ; IS THIS SECTION BACKGROUND?
0F22  74 01                     JZ      BLDN2           ; IF ZERO, IT IS BACKGROUND

0F24  F9                        STC                     ; WASN'T, SO SET CARRY
0F25  D0 D2             BLDN2:  RCL     DL,1            ;MOVE THAT BIT INTO RESULT
0F27  D1 E9                     SHR     CX,1            ;MOVE MASK RIGH 4 BITS
0F29  D1 E9                     SHR     CX,1
0F2B  D1 E9                     SHR     CX,1
0F2D  D1 E9                     SHR     CX,1
0F2F  73 EF                     JNC     BLDN1           ;DO IT AGAIN IF MASK DID'T FALL OUT
0F31  C3                        RET

0F32                    BUILD_NIBBLE    ENDP
                        ;----------------------------------------
                        ;
                        ;       WRITE_AC_CURRENT        INT 10H, AH = 9
                        ;       ----------------
                        ;       THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER AT
                        ;       THE CURRENT CURSOR POSITION
                        ;
                        ;       INPUT
                        ;               AH = CURRENT CRT MODE (MASKED)
                        ;               BH = DISPLAY PAGE
                        ;               CX = COUNT OF CHARACTERS TO WRITE
                        ;               AL = CHAR TO WRITE
                        ;               BL = ATTRIBUTE OF CHAR TO WRITE
                        ;               DS = DATA SEGMENT
                        ;               ES = REGEN SEGMENT
                        ;
```

```
                        ;       OUTPUT
                        ;               NONE
                        ;
                        ;       CALL    (GRAPHICS_WRITE)
                        ;               FIND_POSITION
                        ;               WRITE_AC
                        ;
                        ;       VOLATILE        AX,BX,CX,DX,SI,DI
                        ;
                        ;----------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

0F32                    WRITE_AC_CURRENT        PROC    NEAR

0F32  80 3E 0049 R 10           CMP     CRT_MODE,KJ_MODE; KJ DISPLAY MODE ?
0F37  73 17                     JAE     WRTACC3         ; YES

0F39  80 FC 04                  CMP     AH,GRAPHICS     ; IS THIS GRAPHICS?
0F3C  72 03                     JC      WRTACC1         ; NO

0F3E  E9 1625 R                 JMP     GRAPHICS_WRITE  ; GO TO WRITE ANK IN GRAPHICS

0F41                    WRTACC1:                        ; WRITE_AC_CONTINUE
0F41  8A E3                     MOV     AH,BL           ; GET ATTRIBUTE TO AH
0F43  50                        PUSH    AX              ; SAVE ON STACK
0F44  51                        PUSH    CX              ; SAVE WRITE COUNT
0F45  E8 0DE0 R                 CALL    FIND_POSITION
0F48  8B FB                     MOV     DI,BX           ; ADDRESS TO DI REGISTER
0F4A  59                        POP     CX              ; WRITE COUNT
0F4B  58                        POP     AX              ; CHARACTER IN AX REG
0F4C                    WRTACC2:                        ; WRITE_LOOP
0F4C  AB                        STOSW                   ; PUT THE CHAR/ATTR
0F4D  E2 FD                     LOOP    WRTACC2         ;  AS MANY TIMES AS REQUESTED

0F4F  C3                        RET

0F50                    WRTACC3:
0F50  B2 00                     MOV     DL,FALSE        ; RESET WRITE CHAR ONLY FLAG
0F52  E8 0F7B R                 CALL    WRITE_AC        ; WRITE ATTRIBUTE AND CHARACTER

0F55  C3                        RET
0F56                    WRITE_AC_CURRENT        ENDP

                        ;----------------------------------------
                        ;
                        ;       WRITE_C_CURRENT         INT 10H, AH = 10 (0AH)
                        ;       ---------------
                        ;       THIS ROUTINE WRITES THE CHARACTER AT
                        ;       THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
                        ;
                        ;       INPUT
                        ;               AH = CURRENT CRT MODE (MASKED)
                        ;               BH = DISPLAY PAGE
                        ;               CX = COUNT OF CHARACTERS TO WRITE
                        ;               AL = CHAR TO WRITE
                        ;               BL = COLOR OF CHAR (GRAPHICS)
                        ;               DS = DATA SEGMENT
                        ;               ES = REGEN SEGMENT
                        ;
                        ;       OUTPUT
                        ;               NONE
                        ;
                        ;       CALL    (GRAPHICS_WRITE)
                        ;               FIND_POSITION
                        ;               WRITE_AC
                        ;
                        ;       VOLATILE        AX,BX,CX,DX,SI,DI
                        ;
                        ;----------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

0F56                    WRITE_C_CURRENT PROC    NEAR

0F56  80 3E 0049 R 10           CMP     CRT_MODE,KJ_MODE; KJ DISPLAY MODE ?
0F5B  73 18                     JAE     WRTCC3          ; YES

0F5D  80 FC 04                  CMP     AH,GRAPHICS     ; IS THIS GRAPHICS?
0F60  72 03                     JC      WRTCC1

0F62  E9 1625 R                 JMP     GRAPHICS_WRITE  ; GO TO WRITE ANK IN GRAPHICS

0F65                    WRTCC1:
0F65  50                        PUSH    AX
0F66  51                        PUSH    CX              ; SAVE ON STACK
0F67  E8 0DE0 R                 CALL    FIND_POSITION   ; SAVE WRITE COUNT
0F6A  8B FB                     MOV     DI,BX
0F6C  59                        POP     CX              ; ADDRESS TO DI
0F6D  58                        POP     BX              ; WRITE COUNT
0F6E                    WRTCC2:                         ; BL HAS CHAR TO WRITE
0F6E  8A C3                     MOV     AL,BL           ; WRITE_LOOP
0F70  AA                        STOSB                   ; RECOVER CHAR
0F71  47                        INC     DI              ; PUT THE CHAR/ATTR
0F72  E2 FA                     LOOP    WRTCC2          ; BUMP POINTER PAST ATTRIBUTE
                                                        ;  AS MANY TIMES AS REQUESTED
0F74  C3                        RET

0F75                    WRTCC3:
0F75  B2 FF                     MOV     DL,TRUE
0F77  E8 0F7B R                 CALL    WRITE_AC        ; SET WRITE CHAR ONLY FLAG
                                                        ; WRITE ATTRIBUTE AND CHARACTER
```

```
0F7A  C3                                RET
0F7B                                    WRITE_C_CURRENT ENDP

                                        ;----------------------------------------
                                        ;
                                        ;       WRITE_AC
                                        ;
                                        ;       THIS ROUTINE WRITES THE ATTRIBUTE(IF DL=0) AND CHARACTER AT
                                        ;       THE CURRENT CURSOR POSITION
                                        ;
                                        ;       INPUT
                                        ;               AH = CURRENT CRT MODE (MASKED)
                                        ;               BH = DISPLAY PAGE
                                        ;               CX = COUNT OF CHARACTERS TO WRITE
                                        ;               AL = CHAR TO WRITE
                                        ;               BL = ATTRIBUTE OF CHAR TO WRITE
                                        ;               DL = FLAG OF WRITE CHARACTER ONLY
                                        ;               DS = DATA SEGMENT
                                        ;               ES = REGEN SEGMENT
                                        ;
                                        ;       WORK    [BP+WGPOSN] = REGEN ADDRESS TO WRITE IN GRAPHICS
                                        ;               [BP+WPOSN]  = ROW/COLUMN POSITION TO WRITE
                                        ;               [BP+W2CODE] = ATTRIBUTE/CODE OF 2ND BYTE OF 2 BYTE CODE
                                        ;               [BP+WR_SEG] = REGEN SEGMENT
                                        ;               [BP+WGMODE] = FLAG OF GRAPHICS MODE
                                        ;               [BP+WC_MODE]= CURRENT CRT MODE (MASKED)
                                        ;               [BP+WFONT - WFONT+35] =  FONT PATTERN
                                        ;
                                        ;       OUTPUT          NONE
                                        ;
                                        ;       CALL            FINT_POSITION
                                        ;                       GRAPH_POSITION
                                        ;                       WRITE_ONE_CHAR
                                        ;                       WRITE_TWO_CHAR
                                        ;
                                        ;       VOLATILE        AX,BX,CX,DI,SI
                                        ;
                                        ;----------------------------------------

0F7B                                    WRITE_AC        PROC    NEAR

0F7B  55                                        PUSH    BP                      ; SAVE BP
0F7C  83 EC 2E                                  SUB     SP,W_LOCAL              ; ALLOCATE LOCAL WORK AREA
0F7F  8B EC                                     MOV     BP,SP                   ; ASSIGN BP AS FRAME POINTER

0F81  50                                        PUSH    AX                      ; SAVE MODE
0F82  53                                        PUSH    BX                      ; SAVE DISPLAY PAGE AND ATTRIBUTE
0F83  51                                        PUSH    CX                      ; SAVE COUNTER
0F84  E8 0DE0 R                                 CALL    FIND_POSITION           ; GET REGEN ADDRESS TO WRITE
0F87  8B FB                                     MOV     DI,BX                   ;       IN DI
0F89  59                                        POP     CX                      ; RESTORE COUNTER
0F8A  5B                                        POP     BX                      ; RESTORE DISPLAY PAGE AND ATTRIBUTE
0F8B  58                                        POP     AX                      ; RESTORE MODE

0F8C  C6 46 08 00                               MOV     BYTE PTR [BP+WGMODE],FALSE      ; SET NO GRAPHICS MODE
0F90  88 66 09                                  MOV     BYTE PTR [BP+WC_MODE],AH        ; SET CRT MODE
0F93  80 FC 04                                  CMP     AH,GRAPHICS             ; IS THIS GRAPHICS?
0F96  72 18                                     JB      WRTAC1                  ; NO
                                                                                ;--- GRAPHICS MODE
0F98  50                                        PUSH    AX                      ; SAVE MODE AND CHARACTER CODE
0F99  E8 15E5 R                                 CALL    GRAPH_POSITION          ; GET GRAPHICS REGEN ADDRESS TO WRITE
0F9C  89 46 00                                  MOV     [BP+WGPOSN],AX          ;       IN [BP]

0F9F  8C 46 06                                  MOV     [BP+WR_SEG],ES          ; SAVE REGEN SEGMENT
0FA2  C6 46 08 FF                               MOV     BYTE PTR [BP+WGMODE],TRUE       ; SET GRAPHICS MODE

0FA6  B8 00A0                                   MOV     AX,P_CODE_START         ; SET PESUDO CODE BUFFER SEGMENT
0FA9  8E C0                                     MOV     ES,AX                   ;  TO ES

0FAB  81 E7 07FF                                AND     DI,PCB_MASK             ; MASK POINTER TO INSURE RANGE
0FAF  58                                        POP     AX                      ; RESTORE AX
0FB0                                    WRTAC1:
0FB0  53                                        PUSH    BX                      ; SAVE DISPLAY PAGE AND ATTRIBUTE

0FB1  8A DF                                     MOV     BL,BH                   ; SET PAGE NUMBER TO BL
0FB3  32 FF                                     XOR     BH,BH                   ; CLEAR FOR WORD
0FB5  D1 E3                                     SAL     BX,1                    ; CONVERT TO WORD OFFSET

0FB7  8B B7 035C R                              MOV     SI,[BX+OFFSET CURSOR_POSN]      ; GET CURSOR POSITION
0FBB  89 76 02                                  MOV     [BP+WPOSN],SI           ; SET IT TO LOCAL WORK AREA

0FBE  5B                                        POP     BX                      ; RESTORE DISPLAY PAGE AND ATTRIBUTE

0FBF  F7 06 0340 R FFFF                         TEST    W_1ST_CHAR,TRUE         ; 1ST BYTE OF 2 BYTE CODE HAS BEEN SET ?
0FC5  75 21                                     JNZ     WRTAC4                  ; YES

0FC7  3C 80                                     CMP     AL,080H                 ; 1ST BYTE OF 2 BYTE CODE ?
0FC9  76 2F                                     JBE     WRTAC6                  ; NO, GO TO WRITE ONE CHARACTER
0FCB  3C FD                                     CMP     AL,0FDH                 ;
0FCD  73 2B                                     JAE     WRTAC6                  ; NO, GO TO WRITE ONE CHARACTER
0FCF  3C A0                                     CMP     AL,0A0H                 ;
0FD1  72 04                                     JB      WRTAC2                  ; YES
0FD3  3C DF                                     CMP     AL,0DFH                 ;
0FD5  76 23                                     JBE     WRTAC6                  ; NO, GO TO WRITE ONE CHARACTER

0FD7                                    WRTAC2:
0FD7  8A E3                                     MOV     AH,BL                   ;--CHARACTER IS 1ST BYTE OF 2 BYTE CODE
                                                                                ; SET ATTRIBUTE TO AH
                                                                                ;-SET ZERO COLUMN FLAG FOR AVOLD
                                                                                ;   DESTRUCT OF PREVOUS ROW
0FD9  24 7F                                     AND     AL,NOT ZEROCOL          ; SET ZERO COLUMN BIT OFF
0FDB  F7 C6 00FF                                TEST    SI,00FFH                ; CURRENT COLUMN IS ZERO ?
```

```
0FDF  75 02                    JNZ     WRTAC3                  ; NO
0FE1  0C 80                    OR      AL,ZEROCOL              ; YES, SET ZERO COLUMN BIT ON
0FE3                    WRTAC3:
0FE3  A3 0340 R                MOV     W_1ST_CHAR,AX           ; SAVE 1ST BYTE OF 2 BYTE CODE

0FE6  EB 4D                    JMP     SHORT WRTAC10           ; RETURN; WRITE PROCESS IS PENDING

0FE8                    WRTAC4:                                ;--- CHARACTER MUST BE 2ND BYTE
0FE8  3C 40                    CMP     AL,040H                 ; 2ND BYTE OF 2 BYTE CODE ?
0FEA  72 08                    JB      WRTAC5                  ; NO
0FEC  3C FC                    CMP     AL,0FCH                 ;
0FEE  77 04                    JA      WRTAC5                  ; NO
0FF0  3C 7F                    CMP     AL,07FH                 ;
0FF2  75 0D                    JNE     WRTAC7                  ; YES
0FF4                    WRTAC5:                                ;--- IGNORE 1ST BYTE
0FF4  C7 06 0340 R 0000        MOV     W_1ST_CHAR,0            ; RESET FLAG OF 1ST BYTE
0FFA                    WRTAC6:                                ;--- WRITE ONE BYTE CODE
0FFA  E8 103A R                CALL    WRITE_ONE_CHAR          ; WRITE ONE BYTE CHARACTER
0FFD  E2 FB                    LOOP    WRTAC6                  ; REPEAT CX TIMES

0FFF  EB 34                    JMP     SHORT WRTAC10           ;--- END

1001                    WRTAC7:
1001  88 46 04                MOV     [BP+W2CODE],AL          ; SAVE CODE OF 2ND BYTE IN LOCAL
1004  88 5E 05                MOV     [BP+W2ATTR],BL          ; SAVE ATTR OF 2ND BYTE IN LOCAL

1007  8A 3E 0340 R            MOV     BH,BYTE PTR W_1ST_CHAR  ; GET 1ST BYTE CODE TO SEE IF ZEROCOL

100B  80 0E 0340 R 80          OR      BYTE PTR W_1ST_CHAR,ZEROCOL ; RESTORE CHARACTER CODE

1010  8B C6                    MOV     AX,SI                   ; GET CURSOR POSITION
1012  0A C0                    OR      AL,AL                   ; COLUMN IS ZERO ?
1014  75 0A                    JNZ     WRTAC8                  ; NO

1016  F6 C7 80                 TEST    BH,ZEROCOL              ; ZERO COLUMN FLAG ON ?
1019  75 0F                    JNZ     WRTAC9                  ; YES, SKIP CORRECT CURSOR POSN.

101B  FE CC                    DEC     AH                      ; DECREMENT ROW
101D  A0 004A R                MOV     AL,BYTE PTR CRT_COLS    ; SET COLUMN NUMBER AT END OF LINE
1020                    WRTAC8:
1020  FE C8                    DEC     AL                      ; DECREMENT ROW FOR 2 BYTE CODE
1022  89 46 02                MOV     [BP+WPOSN],AX           ; SET IT TO LOCAL WORK AREA

1025  4F                      DEC     DI                      ; DECREMENT WRITE POSITION 2 (CODE/ATTR)
1026  4F                      DEC     DI                      ;

1027  FF 4E 00                DEC     WORD PTR [BP+WGPOSN]    ; DECREMENT GRAPHICS WRITE POSITION
102A                    WRTAC9:
102A  E8 10E1 R                CALL    WRITE_TWO_CHAR          ; WRITE TWO BYTE CHARACTER
102D  E2 FB                    LOOP    WRTAC9                  ; REPEAT CX TIMES

102F  C7 06 0340 R 0000        MOV     W_1ST_CHAR,0            ; RESET FLAG OF 1ST BYTE
1035                    WRTAC10:
1035  83 C4 2E                ADD     SP,W_LOCAL              ; DEALLOCATE LOCAL WORK AREA
1038  5D                      POP     BP                      ; RESTORE BP

1039  C3                      RET                             ; RETURN TO CALLER

103A                    WRITE_AC ENDP


                        ;--------------------------------------------------------
                        ;
                        ;       WRITE_ONE_CHAR
                        ;
                        ;       THIS ROUTINE WRITES ONE CHARACTER
                        ;
                        ;       INPUT   AL =    CHARACTER CODE
                        ;               BL =    ATTRIBUTE
                        ;               DL =    FLAG OF WRITE CHARACTER ONLY
                        ;               [BP+WGPOSN] =   GRAPHICS WRITE POSITION
                        ;               [BP+WPOSN]= ROW/COLUMN POSITION TO WRITE
                        ;               [BP+WGMODE]= FLAG OF GRAPHICS MODE
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT          ES:DI = REGEN ADDRESS TO WRITE NEXT CHAR
                        ;
                        ;       CALL            G_WRITE1
                        ;                       GRAPH_POSN
                        ;
                        ;       VOLATILE        SI
                        ;
                        ;--------------------------------------------------------

103A                    WRITE_ONE_CHAR  PROC    NEAR

103A  50                        PUSH    AX
103B  53                        PUSH    BX                      ; SAVE CHARACTER CODE
103C  51                        PUSH    CX                      ; SAVE ATTRIBUTE
103D  52                        PUSH    DX                      ; SAVE NUMBER OF CHARACTER TO WRITE
                                                                ; SAVE FLAG OF WRITE CHARACTER ONLY
103E  26: 8A 65 01              MOV     AH,ES:[DI+1]
1042  F6 C4 80                 TEST    AH,ZENBIT               ; GET ATTRIBUTE AT CURRENT POSITION
1045  75 17                    JNZ     WOC3                    ; IS IT HANKAKU ?
                                                                ; NO
                        ;---------------------------- OVERRIDE HANKAKU
1047  0A D2                    OR      DL,DL                   ; WRITE CHARACTER ONLY ?
1049  75 02                    JNZ     WOC1                    ; YES

1048  8A E3                    MOV     AH,BL
104D                    WOC1:                                   ; SET ATTRIBUTE
104D  80 E4 77                  AND     AH,HAN_MASK
1050  AB                        STOSW                           ; MASK OFF ZEN,1ST/2ND BIT
                                                                ; WRITE CHAR/ATTRIBUTE
```

```
1051  F6 46 08 FF                TEST    BYTE PTR [BP+WGMODE],TRUE    ; GRAPHICS MODE ?
1055  74 05                      JZ      WOC2                         ; NO

1057  32 D2                      XOR     DL,DL                        ; SET 0 TO DISPLACEMENT
1059  E8 1213 R                  CALL    G_WRITE1                     ; WRITE ONE CHAR IN GRAPHICS
105C                     WOC2:
105C  EB 5E                      JMP     SHORT WOC8                   ;--- END

105E                     WOC3:
105E  F6 C4 08                   TEST    AH,ZEN2BIT                   ; SECOND BYTE OF ZENKAKU ?
1061  75 2D                      JNZ     WOC6                         ; YES

                                 ;---------------------------- OVERRIDE 1ST BYTE OF ZENKAKU
1063  0A D2                      OR      DL,DL                        ; WRITE CHARACTER ONLY ?
1065  75 02                      JNZ     WOC4                         ; YES

1067  8A E3                      MOV     AH,BL                        ; SET ATTRIBUTE
1069                     WOC4:
1069  80 E4 77                   AND     AH,HAN_MASK                  ; MASK OFF ZEN,1ST/2ND BIT
106C  AB                         STOSW                                ; WRITE CHAR/ATTRIBUTE

106D  26: C6 05 20               MOV     BYTE PTR ES:[DI],' '         ; ERASE 2ND BYTE OF ZENKAKU
1071  26: 80 65 01 77            AND     BYTE PTR ES:[DI+1],HAN_MASK; MASK OFF ZEN,1ST/2ND BIT

1076  F6 46 08 FF               TEST    BYTE PTR [BP+WGMODE],TRUE    ; GRAPHICS MODE ?
107A  74 12                      JZ      WOC5                         ; NO

107C  8A E3                      MOV     AH,BL                        ; SAVE ATTRIBUTE
107E  50                         PUSH    AX                           ;    AND CHARACTER

107F  B0 20                      MOV     AL,' '                       ; SET SPACE
1081  B2 01                      MOV     DL,1                         ; SET +1 TO DISPLACEMENT
1083  E8 1213 R                  CALL    G_WRITE1                     ; ERASE 2ND BYTE OF ZENKAKU

1086  58                         POP     AX                           ; RESTORE CHARACTER
1087  8A DC                      MOV     BL,AH                        ;     AND ATTRIBUTE
1089  32 D2                      XOR     DL,DL                        ; SET 0 TO DISPLACEMENT
108B  E8 1213 R                  CALL    G_WRITE1                     ; WRITE ONE CHAR IN GRAPHICS
108E                     WOC5:
108E  EB 2C                      JMP     SHORT WOC8                   ;--- END

1090                     WOC6:   ;---------------------------- OVERRIDE 2ND BYTE OF ZENKAKU
1090  0A D2                      OR      DL,DL                        ; WRITE CHARACTER ONLY ?
1092  75 02                      JNZ     WOC7                         ; YES

1094  8A E3                      MOV     AH,BL                        ; SET ATTRIBUTE
1096                     WOC7:
1096  80 E4 77                   AND     AH,HAN_MASK                  ; MASK OFF ZEN,1ST/2ND BIT

1099  26: C6 45 FE 20            MOV     BYTE PTR ES:[DI-2],' '       ; ERASE 1ST BYTE OF ZENKAKU
109E  26: 80 65 FF 77            AND     BYTE PTR ES:[DI-1],HAN_MASK; MASK OFF ZEN,1ST/2ND BIT

10A3  AB                         STOSW                                ; WRITE CHAR/ATTRIBUTE

10A4  F6 46 08 FF                TEST    BYTE PTR [BP+WGMODE],TRUE; GRAPHICS MODE ?
10A8  74 12                      JZ      WOC8                         ; NO

10AA  8A E3                      MOV     AH,BL                        ; SAVE ATTRIBUTE
10AC  50                         PUSH    AX                           ;       AND CHARACTER

10AD  B0 20                      MOV     AL,' '                       ; SET SPACE
10AF  B2 FF                      MOV     DL,-1                        ; SET -1 TO DISPLACEMENT
10B1  E8 1213 R                  CALL    G_WRITE1                     ; ERASE 2ND BYTE OF ZENKAKU

10B4  58                         POP     AX                           ; RESTORE CHARACTER
10B5  8A DC                      MOV     BL,AH                        ;       AND ATTRIBUTE
10B7  32 D2                      XOR     DL,DL                        ; SET 0 TO DISPLACEMENT
10B9  E8 1213 R                  CALL    G_WRITE1                     ; WRITE ONE CHAR IN GRAPHICS
10BC                     WOC8:
10BC  8B 46 02                   MOV     AX,[BP+WPOSN]                ; GET COLUMN COUNT
10BF  FE C0                      INC     AL                           ; INCREMENT COLUMN

10C1  3A 06 004A R               CMP     AL,BYTE PTR CRT_COLS         ; EXCEED END OF LINE ?
10C5  72 12                      JB      WOC9                         ; NO

10C7  32 C0                      XOR     AL,AL                        ; CLEAR SINCE MODULO OF CRT_COLS
10C9  FE C4                      INC     AH                           ; INCREMENT ROW

10CB  F6 46 08 FF                TEST    BYTE PTR [BP+WGMODE],TRUE    ; GRAPHICS MODE ?
10CF  74 08                      JZ      WOC9                         ; NO

10D1  50                         PUSH    AX                           ; SAVE AX
10D2  E8 15E8 R                  CALL    GRAPH_POSN                   ; SET NEW GRAPHICS WRITE POSITION
10D5  89 46 00                   MOV     [BP+WGPOSN],AX               ;  TO [BP]
10D8  58                         POP     AX                           ; RESTORE AX
10D9                     WOC9:
10D9  89 46 02                   MOV     [BP+WPOSN],AX                ; SET NEW ROW/COLUMN POSITION

10DC  5A                         POP     DX                           ; RESTORE FLAG OF WRITE CHARACTER ONLY
10DD  59                         POP     CX                           ; RESTORE NUMBER OF CHARACTER
10DE  5B                         POP     BX                           ; RESTORE ATTRIBUTE
10DF  58                         POP     AX                           ; RESTORE CHARACTER CODE
10E0  C3                         RET

10E1                     WRITE_ONE_CHAR  ENDP

                         ;--------------------------------------------------------------
                         ;
                         ;       WRITE_TWO_CHAR
                         ;
                         ;       THIS ROUTINE WRITES TWO BYTE CHARACTER
                         ;
```

```
10E1                    ;       INPUT   [BP+WPOSN] = ROW/COLUMN POSITION TO WRITE
                        ;               [BP+W2CODE] = 2ND BYTE OF 2 BYTE CODE TO WRITE
                        ;               [BP+W2ATTR] = ATTRIBUTE FOR 2ND BYTE
                        ;               [BP+WGMODE] = FLAG OF GRAPHICS MODE
                        ;                DL =     FLAG OF WRITE CHARACTER ONLY
                        ;                ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT          ES:DI   REGEN ADDRESS TO WRITE NEXT CHARACTER
                        ;
                        ;       CALL            WRITE_TWO
                        ;                       G_WRITE1
                        ;                       G_WRITE2
                        ;                       GRAPH_POSN
                        ;
                        ;       VOLATILE        AX,BX,SI
                        ;
                        ;-------------------------------------------------------------

10E1                    WRITE_TWO_CHAR  PROC    NEAR

10E1  51                        PUSH    CX                      ; SAVE NUMBER OF CHARACTER TO WRITE
10E2  52                        PUSH    DX                      ; SAVE FLAG OF WRITE CHARACTER ONLY

10E3  8B 46 02                  MOV     AX,[BP+WPOSN]           ; GET ROW/COLUMN POSITION
10E6  FE C0                     INC     AL                      ; ADJUST FOR COMPARE

10E8  3A 06 004A R              CMP     AL,BYTE PTR CRT_COLS    ; AT LINE END BOUNDARY ?
10EC  72 2C                     JB      WTC1                    ; NO

10EE  FE C4                     INC     AH                      ; INCREMENT ROW
10F0  32 C0                     XOR     AL,AL                   ; SET 0 TO COLUMN
10F2  89 46 02                  MOV     [BP+WPOSN],AX           ; SET IT TO WORK

10F5  26: C6 05 20              MOV     BYTE PTR ES:[DI],' '    ; WRITE SPACE
10F9  47                        INC     DI                      ;
10FA  26: 80 25 77              AND     BYTE PTR ES:[DI],HAN_MASK;MASK KJ-BIT OFF
10FE  47                        INC     DI                      ; ADVANCE POINTER

10FF  F6 46 08 FF               TEST    BYTE PTR [BP+WGMODE],TRUE    ; GRAPHICS MODE ?
1103  74 15                     JZ      WTC1                    ; NO

1105  50                        PUSH    AX                      ; SAVE ROW/COLUMN
1106  52                        PUSH    DX                      ; SAVE FLAG OF WRITE CHAR ONLY

1107  B0 20                     MOV     AL,' '                  ; WRITE SPACE
1109  8A 1E 0341 R              MOV     BL,BYTE PTR W_1ST_CHAR+1; GET ATTRIBUTE OF 1ST BYTE
110D  32 D2                     XOR     DL,DL                   ; WRITE AT CURRENT CURSOR POSITON +0
110F  E8 1213 R                 CALL    G_WRITE1                ; WRITE ONE BYTE CHAR IN GRAPHICS
1112  5A                        POP     DX                      ; RESTORE FLAG
1113  58                        POP     AX                      ; RESTORE ROW/COLUMN
1114  E8 15E8 R                 CALL    GRAPH_POSN              ; SET NEW GRAPHICS WRITE POSITION
1117  89 46 00                  MOV     [BP+WGPOSN],AX          ;    TO AX
111A                    WTC1:
111A  26: 8A 7D 01              MOV     BH,ES:[DI+1]            ; GET ATTR. AT CURRENT CURSOR POSN
111E  26: 8A 5D 03              MOV     BL,ES:[DI+3]            ; GET ATTR. AT CURRENT CURSOR POSN+1

1122  F6 C7 80                 TEST    BH,ZENBIT               ; HANKAKU ?
1125  74 05                     JZ      WTC2                    ; YES
1127  F6 C7 08                 TEST    BH,ZEN2BIT              ; 2ND BYTE OF 2 BYTE CODE ?
112A  75 3A                     JNZ     WTC7                    ; YES
112C                    WTC2:                           ;--- HANKAKU OR 1ST BYTE IS PRESENT AT CURRENT
112C  F6 C3 80                  TEST    BL,ZENBIT               ; HANKAKU ?
112F  74 05                     JZ      WTC3                    ; YES
1131  F6 C3 08                  TEST    BL,ZEN2BIT              ; 2ND BYTE OF 2 BYTE CODE ?
1134  74 0E                     JZ      WTC5                    ; NO

1136                    WTC3:   ;---------------------------- OVERRIDE TWO HANKAKU OR 1ST,2ND BYTE
1136  E8 11E4 R                 CALL    WRITE_TWO               ; WRITE TWO BYTE CHARACTER IN REGEN

1139  F6 46 08 FF               TEST    BYTE PTR [BP+WGMODE],TRUE    ; GRAPHICS MODE ?
113D  74 03                     JZ      WTC4                    ; NO

113F  E8 1335 R                 CALL    G_WRITE2
1142                    WTC4:
1142  EB 7D                     JMP     SHORT WTC10             ; WRITE 2 BYTE CHARACTER IN GRAPHICS

1144                    WTC5:   ;---------------------------- OVERRIDE HANKAKU,1ST BYTE OF 2 BYTE CODE
1144  E8 11E4 R                 CALL    WRITE_TWO               ; WRITE TWO BYTE CHARACTER IN REGEN

1147  26: C6 05 20              MOV     BYTE PTR ES:[DI  ],' '  ; ERASE 2ND BYTE OF 2 BYTE CODE
114B  26: 80 65 01 77           AND     BYTE PTR ES:[DI+1],HAN_MASK; MASK OFF KJ-BIT

1150  F6 46 08 FF               TEST    BYTE PTR [BP+WGMODE],TRUE; GRAPHICS MODE ?
1154  74 0E                     JZ      WTC6                    ; NO

1156  B0 20                     MOV     AL,' '
1158  8A 1E 0341 R              MOV     BL,BYTE PTR W_1ST_CHAR+1; ERASE 2ND BYTE OF 2 BYTE CODE
115C  B2 02                     MOV     DL,2                    ; GET ATTRIBUTE OF 1ST BYTE
115E  E8 1213 R                 CALL    G_WRITE1                ; WRITE AT CURRENT CURSOR POSITON +2
                                                                ; WRITE ONE BYTE CHAR IN GRAPHICS
1161  E8 1335 R                 CALL    G_WRITE2
1164                    WTC6:
1164  EB 5B                     JMP     SHORT WTC10             ; WRITE 2 BYTE CHARACTER IN GRAPHICS

1166                    WTC7:                                   ;--- END
1166  F6 C3 80                  TEST    BL,ZENBIT
1169  75 23                     JNZ     WTC9                    ; ATTRIBUTE AT CURRENT +1 IS HANKAKU ?
                                                                ; NO
                                ;---------------------------- OVERRIDE 2ND BYTE, HANKAKU
116B  26: C6 45 FE 20           MOV     BYTE PTR ES:[DI-2],' '  ; ERASE 1ST BYTE OF 2 BYTE CODE
1170  26: 80 65 FF 77           AND     BYTE PTR ES:[DI-1],HAN_MASK; MASK OFF KJ-BIT

1175  E8 11E4 R                 CALL    WRITE_TWO
                                                                ; WRITE TWO BYTE CHARACTER IN REGEN
```

```
1178  F6 46 08 FF                    TEST    BYTE PTR [BP+WGMODE],TRUE; GRAPHICS MODE ?
117C  74 0E                          JZ      WTC8                    ; NO
117E  B0 20                          MOV     AL,' '                  ; ERASE 1ST BYTE OF 2 BYTE CODE
1180  8A 1E 0341 R                    MOV     BL,BYTE PTR W_1ST_CHAR+1; GET ATTRIBUTE OF 1ST BYTE
1184  B2 FF                          MOV     DL,-1                   ; WRITE AT CURRENT CURSOR POSITON -1
1186  E8 1213 R                      CALL    G_WRITE1                ; WRITE ONE BYTE CHAR IN GRAPHICS

1189  E8 1335 R                      CALL    G_WRITE2                ; WRITE 2 BYTE CHARACTER IN GRAPHICS
118C                           WTC8:
118C  EB 33                          JMP     SHORT WTC10             ;--- END

118E                           WTC9:  ;---------------------------- OVERRIDE 2ND,1ST BYTE OF 2 BYTE CODE
118E  26: C6 45 FE 20                MOV     BYTE PTR ES:[DI-2],' '  ; ERASE 1ST BYTE OF 2 BYTE CODE
1193  26: 80 65 FF 77                AND     BYTE PTR ES:[DI-1],HAN_MASK; MASK OFF KJ-BIT

1198  E8 11E4 R                      CALL    WRITE_TWO               ; WRITE TWO BYTE CHARACTER IN REGEN

119B  26: C6 05 20                   MOV     BYTE PTR ES:[DI  ],' '  ; ERASE 1ST BYTE OF 2 BYTE CODE
119F  26: 80 65 01 77                AND     BYTE PTR ES:[DI+1],HAN_MASK; MASK OFF KJ-BIT

11A4  F6 46 08 FF                    TEST    BYTE PTR [BP+WGMODE],TRUE    ; GRAPHICS MODE ?
11A8  74 17                          JZ      WTC10                   ; NO

11AA  B0 20                          MOV     AL,' '                  ; ERASE 1ST BYTE OF 2 BYTE CODE
11AC  8A 1E 0341 R                    MOV     BL,BYTE PTR W_1ST_CHAR+1; GET ATTRIBUTE OF 1ST BYTE
11B0  50                             PUSH    AX                      ; SAVE AX
11B1  53                             PUSH    BX                      ; SAVE BX

11B2  B2 FF                          MOV     DL,-1                   ; WRITE AT CURRENT CURSOR POSITON -1
11B4  E8 1213 R                      CALL    G_WRITE1                ; WRITE ONE BYTE CHAR IN GRAPHICS

11B7  5B                             POP     BX                      ; RESTORE BX
11B8  58                             POP     AX                      ; RESTORE AX
11B9  B2 02                          MOV     DL,2                    ; WRITE CURRENT CURSOR POSITON +2
11BB  E8 1213 R                      CALL    G_WRITE1                ; WRITE ONE BYTE CHAR IN GRAPHICS

11BE  E8 1335 R                      CALL    G_WRITE2                ; WRITE 2 BYTE CHARACTER IN GRAPHICS
11C1                           WTC10:
11C1  8B 46 02                       MOV     AX,[BP+WPOSN]           ; GET COLUMN COUNT
11C4  04 02                          ADD     AL,2                    ; ADVANCE 2 BY 2 BYTE CHAR WRITING

11C6  3A 06 004A R                   CMP     AL,BYTE PTR CRT_COLS    ; EXCEED END OF LINE ?
11CA  72 12                          JB      WTC11                   ; NO

11CC  32 C0                          XOR     AL,AL                   ; CLEAR SINCE MODULO OF CRT_COLS
11CE  FE C4                          INC     AH                      ; INCREMENT ROW

11D0  F6 46 08 FF                    TEST    BYTE PTR [BP+WGMODE],TRUE; GRAPHICS MODE ?
11D4  74 08                          JZ      WTC11                   ; NO

11D6  50                             PUSH    AX                      ; SAVE ROW/COLUMN
11D7  E8 15E8 R                      CALL    GRAPH_POSN              ; SET NEW GRAPHICS WRITE POSITION
11DA  89 46 00                       MOV     [BP+WGPOSN],AX          ;    TO [BP+WGPOSN]
11DD  58                             POP     AX                      ; RESTORE ROW/COLUMN
11DE                           WTC11:
11DE  89 46 02                       MOV     [BP+WPOSN],AX           ; SET NEW ROW/COLUMN POSITION

11E1  5A                             POP     DX                      ; RESTORE FLAG OF WRITE CHARACTER ONLY
11E2  59                             POP     CX                      ; RESTORE NUMBER OF CHARECTER
11E3  C3                             RET

11E4                           WRITE_TWO_CHAR  ENDP
                               ;--------------------------------------------------------------
                               ;
                               ;       WRITE_TWO
                               ;
                               ;       THIS ROUTINE WRITES TWO BYTE CHARACTER IN REGEN
                               ;
                               ;       INPUT  [BP+W2CODE] = 2ND BYTE OF 2 BYTE CODE TO WRITE
                               ;              [BP+W2ATTR] = ATTRIBUTE FOR 2ND BYTE
                               ;                BH =    ATTRIBUTE AT CURRENT CURSOR POSITION
                               ;                BL =    ATTRIBUTE AT CURRENT CURSOR POSITION + 1
                               ;                DL =    FLAG OF WRITE CHARACTER ONLY
                               ;                ES:DI = REGEN ADDRESS TO WRITE
                               ;
                               ;       OUTPUT  ES:DI = REGEN ADDRESS TO WRITE NEXT CHARACTER
                               ;
                               ;       CALL    CHECK_ROSS_CHAR
                               ;
                               ;       VOLATILE        AX
                               ;
                               ;--------------------------------------------------------------

11E4                           WRITE_TWO       PROC    NEAR

11E4  51                             PUSH    CX                      ; SAVE CX

11E5  8A 2E 0340 R                   MOV     CH,BYTE PTR W_1ST_CHAR  ; GET 1ST CHARACTER'S CODE
11E9  8A 4E 04                       MOV     CL,BYTE PTR [BP+W2CODE] ; GET 2ND CHARACTER'S CODE
11EC  E8 1B65 R                      CALL    CHECK_ROSS_CHAR         ; CHECK AND CONVERT ROSSIAN CHARACTER

11EF  8A C5                          MOV     AL,CH                   ; GET 1ST CHARACTER'S CODE
11F1  8A 26 0341 R                   MOV     AH,BYTE PTR W_1ST_CHAR+1; GET 1ST CHARACTER'S ATTRIBUTE

11F5  0A D2                          OR      DL,DL                   ; WRITE CHARACTER ONLY ?
11F7  74 02                          JZ      WRTTWO1                 ; NO

11F9  8A E7                          MOV     AH,BH                   ; GET ATTRIBUTE AT CURRENT
11FB                           WRTTWO1:
11FB  80 E4 77                       AND     AH,HAN_MASK             ; STRIP OFF ZEN,1ST/2ND BIT
11FE  80 CC 80                       OR      AH,ZENBIT               ; SET ZENKAKU BIT
1201  AB                            STOSW                           ; WRITE TO REGEN
```

```
1202  8A C1                   MOV     AL,CL                   ; GET 2ND CHARACTER'S CODE
1204  8A 66 05                MOV     AH,[BP+W2ATTR]          ; GET 2ND CHARACTER'S ATTR/CODE

1207  0A D2                   OR      DL,DL                   ; WRITE CHARACTER ONLY ?
1209  74 02                   JZ      WRTTWO2                 ; NO

1208  8A E3                   MOV     AH,BL                   ; GET ATTRIBUTE AT CURRENT + 1
120D                  WRTTWO2:
120D  80 CC 88                OR      AH,ZEN2_MASK            ; SET ZENKAKU,2ND_BYTE BIT
1210  AB                      STOSW                           ; WRITE TO REGEN

1211  59                      POP     CX                      ; RESTORE CX
1212  C3                      RET

1213                  WRITE_TWO       ENDP
                      ;---------------------------------------------------------------
                      ;
                      ;       G_WRITE1
                      ;
                      ;       THIS ROUTINE WRITES ONE CHARACTER IN GRAPHICS
                      ;
                      ;       INPUT   AL =    CHARACTER CODE TO WRITE
                      ;               BL =    ATTRIBUTE
                      ;               DL =    DISPLACEMENT FOR WRITE POSITION
                      ;               [BP+WGPOSN] =  WRITE POSITION
                      ;               [BP+WPOSN]= ROW/COLUMN POSITION TO WRITE
                      ;               [BP+WR_SEG]= REGEN SEGMENT FOR GRAPHICS
                      ;               [BP+WC_MODE]= CRT MODE (MASKED)
                      ;       WORK    [BP+WFONT - WFONT+17] =  FONT PATTERN
                      ;
                      ;       OUTPUT          [BP+WGPOSN] =  NEXT WRITE POSITION (IF DL = 0)
                      ;
                      ;       CALL            ENABLE_VRAM
                      ;                       FONT
                      ;                       G_WRT1
                      ;
                      ;       VOLATILE        AX,BX,CX,DX,SI
                      ;
                      ;---------------------------------------------------------------

1213                  G_WRITE1 PROC    NEAR

1213  1E                      PUSH    DS              ; SAVE CURRENT DS
1214  57                      PUSH    DI              ; SAVE CURRENT DI
1215  06                      PUSH    ES              ; SAVE CURRENT ES

1216  F6 C3 80                TEST    BL,XOR_BIT              ; XOR WRITE BIT ON ?
1219  75 05                   JNZ     G_WRT11                 ; YES

121B  C6 06 0349 R 00         MOV     GC_PRESENT,FALSE        ; GRAPHICS CURSOR FLAG OFF
1220                  G_WRT11:
1220  53                      PUSH    BX              ; SAVE COLOR ATTRIBUTE

1221  16                      PUSH    SS              ; SET DESTINATION SEGMENT FOR FONT
1222  07                      POP     ES              ;   TO ES
                                      ASSUME  ES:STACK

1223  8D 5E 0A                LEA     BX,[BP+WFONT]   ; SET DESTINATION OFFSET FOR FONT TO BX

1226  32 E4                   XOR     AH,AH           ; CLEAR FOR 1 BYTE CODE
1228  8B C8                   MOV     CX,AX           ; SET CHARACTER CODE TO CX

122A  80 F9 A0                CMP     CL,HALFTONE     ; HALF TONE CHARACTER ?
122D  B8 2A55                 MOV     AX,HT_FONT      ; SET FONT PATTERN OF HALF TONE
1230  74 07                   JE      G_WRT12         ; YES

1232  80 F9 20                CMP     CL,' '          ; SPACE ?
1235  75 18                   JNE     G_WRT13         ; NO

                                      ;--- SPACE & HALF TONE CODE IS HANDLED SPECIALLY FOR
                                      ;---   IMPROVE THROUGHPUT OF "BASIC" AND KANA-KAN
1237  33 C0                   XOR     AX,AX
1239                  G_WRT12:        ; CLEAR FOR FONT PATTERN OF SPACE
1239  8B FB                   MOV     DI,BX           ; SET DESTINATION ADDRESS
123B  B9 0008                 MOV     CX,(CBOX_ROW-2)/2; SET CLEAR COUNT BY WORD
123E  F3/ AB                  REP     STOSW           ; SET FONT PATTERN
1240  33 C0                   XOR     AX,AX           ;
1242  AB                      STOSW                   ; CLEAR BOTTOM 2 ROW

1243  F6 06 0353 R FF         TEST    KJROM_STAT,TRUE ;--- INSURE V-RAM IS ON FOR KANA-KAN INTERFACE
1248  74 0C                   JZ      G_WRT14         ; KJ-ROM IS ON ?
                                                      ; NO
124A  E8 1BB4 R               CALL    ENABLE_VRAM     ; MUST ENABLE VRAM FOR WRITE GRAPHICS PATTERN

124D  EB 07                   JMP     SHORT   G_WRT14
124F                  G_WRT13:

124F  B0 80                   MOV     AL,80H          ; SET FUNCTION OF REQUEST FONT WITH FULL BOX
1251  B6 10                   MOV     DH,KJ_MODE      ; INDICATES KJ MODE
1253  E8 1A63 R               CALL    FONT            ; GET FONT PATTERN
1256                  G_WRT14:
1256  06                      PUSH    ES
1257  1F                      POP     DS              ; SET SEGMENT FOR FONT PATTERN
                                                      ;   TO DS
                                      ASSUME  DS:STACK
1258  8B F3                   MOV     SI,BX           ; SET TOP ADDRESS OF FONT PATTERN

125A  5B                      POP     BX              ; RESTORE COLOR ATTRIBUTE
```

```
125B  8E 46 06                        MOV     ES,[BP+WR_SEG]  ; GET ACTUAL REGEN SEGMENT FOR GRAPHICS
                                              ASSUME  ES:VIDEO_RAM
125E  E8 1265 R                       CALL    G_WRT1          ; WRITE FONT PATTERN

1261  07                              POP     ES              ; RESTORE ES
1262  5F                              POP     DI              ; RESTORE DI
1263  1F                              POP     DS              ; RESTORE DS
1264  C3                              RET

1265                                  G_WRITE1 ENDP

                                      ;--------------------------------------------------------
                                      ;
                                      ;       G_WRT1
                                      ;
                                      ;       THIS ROUTINE WRITES GRAPHICS PATTERN
                                      ;
                                      ;       INPUT   BL          = ATTRIBUTE
                                      ;               DL          = DISPLACEMENT
                                      ;               [BP+WGPOSN] = WRITE POSITION
                                      ;               [BP+WPOSN]  = ROW/COLUMN POSITION TO WRITE
                                      ;               [BP+WC_MODE]= CRT MODE (MASKED)
                                      ;               DS:SI       = FONT PATTERN ADDRESS
                                      ;               ES          = REGEN SEGMENT
                                      ;
                                      ;       OUTPUT  [BP+WGPOSN] =   NEXT WRITE POSITION (IF DL=0)
                                      ;
                                      ;       CALL            G_W_1
                                      ;                       G_W_10
                                      ;                       G_W_2
                                      ;                       G_W_4
                                      ;                       G_W_40
                                      ;                       GET_DIRSEG
                                      ;
                                      ;       VOLATILE        AX,BX,DX,SI,DI,ES
                                      ;
                                      ;--------------------------------------------------------
                                              ASSUME  CS:CODE, DS:INDETERMINATE, ES:VIDEO_RAM
1265                                  G_WRT1  PROC    NEAR

1265  8A C2                           MOV     AL,DL           ; SET DISPLACEMENT
1267  98                              CBW                     ;       TO AX
1268  50                              PUSH    AX              ; SAVE IT
1269  03 46 00                        ADD     AX,[BP+WGPOSN]  ; GET POSITION TO WRITE
126C  8B F8                           MOV     DI,AX           ;       IN DI

126E  8A 76 09                        MOV     DH,[BP+WC_MODE] ; GET CRT MODE
1271  80 FE 09                        CMP     DH,9            ; 320X200 16 COLOR ?
1274  74 25                           JE      GWRT11          ; YES

1276  80 FE 0B                        CMP     DH,0BH          ; 640X200 16 COLOR ?
1279  74 3B                           JE      GWRT121         ; YES

127B  80 FE 0A                        CMP     DH,0AH          ; 640X200 4 COLOR ?
127E  74 68                           JE      GWRT13          ; YES

1280  80 FE 04                        CMP     DH,4            ; 320X200 4 COLOR ?
1283  74 7C                           JE      GWRT15          ; YES
1285  80 FE 05                        CMP     DH,5            ; 320X200 4 SHADE ?
1288  74 77                           JE      GWRT15          ; YES

128A  80 FE 06                        CMP     DH,6            ; 640X200 2 SHADE ?
128D  74 7B                           JE      GWRT16          ; YES

                                                              ;--- 160X200 16 COLOR: MODE 8
128F  D1 E7                           SAL     DI,1            ;
1291  D1 E7                           SAL     DI,1            ; OFFSET*4 SINCE 4 BYTES/CHAR

1293  B6 09                           MOV     DH,CBOX_ROW/2   ; SET LOOP COUNT ( 18 ROW / 2 SCAN )
1295  E8 14F3 R                       CALL    G_W_4           ; WRITE ONE CHAR

1298  E9 132C R                       JMP     GWRT110         ;--- END

129B                                  GWRT11:                 ;--- 320X200 16 COLOR
129B  D1 E7                           SAL     DI,1            ;
129D  D1 E7                           SAL     DI,1            ; OFFSET*4 SINCE 4 BYTES/CHAR

129F  B6 09                           MOV     DH,CBOX_ROW/(4/2); SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC. )

12A1  8B 46 02                        MOV     AX,[BP+WPOSN]   ; GET ROW COLUMN POSITION
12A4  D0 EC                           SHR     AH,1            ; ODD ROW ?
12A6  72 05                           JC      GWRT12          ; YES
                                                              ;--- WRITE EVEN ROW CHARACTER
12A8  E8 14F3 R                       CALL    G_W_4           ; WRITE ONE CHAR

12AB  EB 7F                           JMP     SHORT GWRT110

12AD                                  GWRT12:                 ;--- WRITE ODD ROW CHARACTER
12AD  81 C7 3FB0                      ADD     DI,4000H-80     ; ADJUST WRITE ADDRESS FOR ODD ROW
12B1  E8 1527 R                       CALL    G_W_40          ; WRITE ONE CHAR

12B4  EB 76                           JMP     SHORT GWRT110

12B6                                  GWRT121:                ;--- 640X200 16 COLOR
                                                              ;--- WRITE V-RAM 2
12B6  53                              PUSH    BX              ; SAVE ATTRIBUTE
12B7  57                              PUSH    DI              ; SAVE REGEN POINTER
12B8  56                              PUSH    SI              ; SAVE CHARACTER PATTERN ADDRESS

12B9  8A FB                           MOV     BH,BL           ; SAVE ATTRIBUTE TO BH
12BB  80 E7 80                        AND     BH,XOR_BIT      ; GET XOR BIT
12BE  D0 EB                           SHR     BL,1            ; SHIFT ATTRIBUTE FOR WRITE TO V-RAM2
12C0  D0 EB                           SHR     BL,1            ;
```

```
12C2  0A DF                     OR      BL,BH           ; RESTORE XOR BIT
12C4  D1 E7                     SAL     DI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR

12C6  B6 09                     MOV     DH,CBOX_ROW/(4/2); SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC.)

12C8  8B 46 02                  MOV     AX,[BP+WPOSN]   ; GET ROW COLUMN POSITION
12CB  D0 EC                     SHR     AH,1            ; ODD ROW ?
12CD  72 05                     JC      GWRT122         ; YES
                                                        ;--- WRITE EVEN ROW CHARACTER
12CF  E8 1498 R                 CALL    G_W_1           ; WRITE ONE CAHAR

12D2  EB 07                     JMP     SHORT GWRT123

12D4                    GWRT122:                        ;--- WRITE ODD ROW CHARACTER
12D4  81 C7 3FB0                ADD     DI,4000H-80     ; ADJUST WRITE ADDRESS FOR ODD ROW
12D8  E8 14C2 R                 CALL    G_W_10          ; WRITE ONE CHAR
12DB                    GWRT123:
12DB  5E                        POP     SI              ; RESTORE CHARACTER PATTERN ADDRESS
12DC  5F                        POP     DI              ; RESTORE REGEN POINTER
12DD  5B                        POP     BX              ; RESTORE ATTRIBUTE

12DE  1E                        PUSH    DS              ; SAVE DS
12DF  E8 0000 E                 CALL    DDS             ; POINT TO DATA AREA

12E2  E8 1D95 R                 CALL    GET_DIRSEG      ; GET DIRECT ACCESS SEGENT OF V-RAM 1
12E5  8E C0                     MOV     ES,AX           ; SET IT TO ES
12E7  1F                        POP     DS              ; RESTORE DS
                                                        ;--- WRITE V-RAM 1

12E8                    GWRT13:                         ;--- 640X200 4 COLOR
12E8  D1 E7                     SAL     DI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR

12EA  B6 09                     MOV     DH,CBOX_ROW/(4/2); SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC.)

12EC  8B 46 02                  MOV     AX,[BP+WPOSN]   ; GET ROW COLUMN POSITION
12EF  D0 EC                     SHR     AH,1            ; ODD ROW ?
12F1  72 05                     JC      GWRT14          ; YES
                                                        ;--- WRITE EVEN ROW CHARACTER
12F3  E8 1498 R                 CALL    G_W_1           ; WRITE ONE CAHAR

12F6  EB 34                     JMP     SHORT GWRT110   ;--- END

12F8                    GWRT14:                         ;--- WRITE ODD ROW CHARACTER
12F8  81 C7 3FB0                ADD     DI,4000H-80     ; ADJUST WRITE ADDRESS FOR ODD ROW
12FC  E8 14C2 R                 CALL    G_W_10          ; WRITE ONE CHAR

12FF  EB 2B                     JMP     SHORT GWRT110


1301                    GWRT15:                         ;--- 320X200 4 COLOR/SHADE
1301  D1 E7                     SAL     DI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR

1303  B6 09                     MOV     DH,CBOX_ROW/2   ; SET LOOP COUNT ( 18 ROW / 2 SCAN )
1305  E8 14C8 R                 CALL    G_W_2           ; WRITE ONE CHAR

1308  EB 22                     JMP     SHORT GWRT110   ;--- END

130A                    GWRT16:                         ;--- 640X200 2 SHADE
130A  B6 09                     MOV     DH,CBOX_ROW/2   ; SET LOOP COUNT ( 16 ROW / 2 SCAN )
130C                    GWRT17:
130C  AC                        LODSB                   ; GET WORD FROM CODE POINTS
130D  F6 C3 80                  TEST    BL,XOR_BIT      ; SHOULD WE USE THE FUNCTION
1310  75 04                     JNZ     GWRT18          ;  TO PUT CHAR IN?

1312  AA                        STOSB                   ; STORE IN REGEN BUFFER
1313  AC                        LODSB                   ; GET NEXT ROW
1314  EB 0A                     JMP SHORT GWRT19

1316                    GWRT18:
1316  26: 32 05                 XOR     AL,ES:[DI]      ; EXCLUSIVE OR WITH CURRENT DATA
1319  AA                        STOSB                   ; STORE THE CODE POINT
131A  AC                        LODSB                   ; AGAIN FOR ODD FIELD
131B  26: 32 85 1FFF            XOR     AL,ES:[DI+2000H-1]
1320                    GWRT19:
1320  26: 88 85 1FFF            MOV     ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
1325  83 C7 4F                  ADD     DI,79           ; MOVE TO NEXT ROW IN REGEN
1328  FE CE                     DEC     DH              ; DONE WITH LOOP ?
132A  75 E0                     JNZ     GWRT17          ; NO

132C                    GWRT110:                        ;--- END
132C  58                        POP     AX              ; RESTORE DISPLACEMENT
132D  0A C0                     OR      AL,AL           ; ZERO ?
132F  75 03                     JNE     GWRT111         ; NO

1331  FF 46 00                  INC     WORD PTR [BP+WGPOSN]    ; ADVANCE WRITE POSITION
1334                    GWRT111:
1334  C3                        RET

1335                    G_WRT1  ENDP


                        ;---------------------------------------------------------------
                        ;
                        ;       G_WRITE2
                        ;
                        ;       THIS ROUTINE WRITES TWO BYTE CHARACTER IN GRAPHICS
                        ;
                        ;       INPUT   [BP+WGPOSN]  =  WRITE POSITION
                        ;               [BP+WPOSN]   =  ROW/COLUMN POSITION TO WRITE
                        ;               [BP+W2CODE]  =  ATTRIBUTE/CODE OF 2ND BYTE
                        ;               [BP+WR_SEG]  =  REGEN SEGMENT FOR GRAPHICS
                        ;               [BP+WC_MODE] =  CURRENT CRT MODE (MASKED)
```

```
                        ;       OUTPUT  [BP+WGPOSN]  =  NEXT WRITE POSITION
                        ;       WORK    [BP+WFONT - WFONT+35] =  FONT PATTERN
                        ;
                        ;       CALL            CHECK_ROSS_CHAR
                        ;                       G_W_1
                        ;                       G_W_10
                        ;                       G_W_2
                        ;                       G_W_4
                        ;                       G_W_40
                        ;                       GET_DIRSEG
                        ;
                        ;       VOLATILE        AX,BX,CX,DX,SI
                        ;
                        ;-----------------------------------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

1335                    G_WRITE2 PROC   NEAR

1335  1E                        PUSH    DS              ; SAVE CURRENT DS
1336  57                        PUSH    DI              ; SAVE CURRENT DI
1337  06                        PUSH    ES              ; SAVE CURRENT ES

1338  F6 C3 80                  TEST    BL,XOR_BIT              ; XOR WRITE BIT ON ?
133B  75 05                     JNZ     GWRT21                  ; YES

133D  C6 06 0349 R 00           MOV     GC_PRESENT,FALSE        ; GRAPHICS CURSOR FLAG OFF
1342                    GWRT21:
1342  8B 1E 0340 R              MOV     BX,W_1ST_CHAR   ; SET ATTRIBUTE OF 1ST BYTE TO BH
1346  8A E3                    MOV     AH,BL           ; SET 1ST BYTE OF 2 BYTE CODE TO AH
1348  8A 46 04                 MOV     AL,[BP+W2CODE]  ; SET 2ND BYTE OF 2 BYTE CODE TO AL
134B  8A 5E 05                 MOV     BL,[BP+W2ATTR]  ; SET ATTRIBUTE OF 2ND BYTE TO BL

134E  53                        PUSH    BX              ; SAVE COLOR ATTRIBUTE

134F  16                        PUSH    SS              ; SET DESTINATION SEGMENT FOR FONT
1350  07                        POP     ES              ;   TO ES
                                        ASSUME  ES:STACK

1351  8D 5E 0A                 LEA     BX,[BP+WFONT]   ; SET DESTINATION OFFSET FOR FONT TO BX

1354  8B C8                     MOV     CX,AX           ; SET CHARACTER CODE TO CX
1356  E8 1B65 R                 CALL    CHECK_ROSS_CHAR ; CHECK AND CONVERT ROSSIAN CHARACTER

1359  B0 80                     MOV     AL,80H          ; SET FUNCTION OF REQUEST FONT WITH FULL BOX
135B  B6 10                     MOV     DH,KJ_MODE      ; INDICATES KJ MODE
135D  E8 1A63 R                 CALL    FONT            ; GET FONT PATTERN

1360  06                        PUSH    ES              ; SET SEGMENT FOR FONT PATTERN
1361  1F                        POP     DS              ;   TO DS
                                        ASSUME  DS:STACK

1362  8B F3                     MOV     SI,BX           ; SET TOP ADDRESS OF FONT PATTERN

1364  5B                        POP     BX              ; RESTORE COLOR ATTRIBUTE

1365  8E 46 86                 MOV     ES,[BP+WR_SEG]  ; GET ACTUAL REGEN SEGMENT FOR GRAPHICS
                                        ASSUME  ES:VIDEO_RAM

1368  8B 7E 00                 MOV     DI,[BP+WGPOSN]  ; SET WRITE POSITION TO DI

136B  8A 76 09                 MOV     DH,[BP+WC_MODE] ; GET CRT MODE
136E  80 FE 06                 CMP     DH,6            ; 640X200 2 SHADE ?
1371  75 2F                    JNE     GWRT26          ; NO, MODE IS 8

                                                        ;--- 640X200 2 SHADE
1373  B9 0002                  MOV     CX,2            ; REPEAT 2 TIMES FOR LEFT/RIGHT PART
1376                    GWRT22:
1376  57                        PUSH    DI              ; SAVE REGEN ADDRESS TO WRITE
1377  B6 09                     MOV     DH,CBOX_ROW/2   ; NUMBER OF TIMES THROUGH LOOP
1379                    GWRT23:
1379  AC                        LODSB                   ; GET WORD FROM CODE POINTS
137A  F6 C7 80                  TEST    BH,XOR_BIT      ; SHOULD WE USE THE FUNCTION
137D  75 04                     JNZ     GWRT24          ;  TO PUT CHAR IN?

137F  AA                        STOSB                   ; STORE IN REGEN BUFFER
1380  AC                        LODSB                   ; GET NEXT ROW
1381  EB 0A                     JMP SHORT GWRT25

1383                    GWRT24:
1383  26: 32 05                 XOR     AL,ES:[DI]      ; EXCLUSIVE OR WITH CURRENT DATA
1386  AA                        STOSB                   ; STORE THE CODE POINT
1387  AC                        LODSB                   ; AGAIN FOR ODD FIELD
1388  26: 32 85 1FFF            XOR     AL,ES:[DI+2000H-1]
138D                    GWRT25:
138D  26: 88 85 1FFF            MOV     ES:[DI+2000H-1],AL ; STORE IN SECOND HALF

1392  83 C7 4F                  ADD     DI,79           ; MOVE TO NEXT ROW IN REGEN
1395  FE CE                     DEC     DH              ; DONE WITH LOOP ?
1397  75 E0                     JNZ     GWRT23          ; NO

1399  5F                        POP     DI              ; RESTORE REGEN ADDRESS
139A  47                        INC     DI              ; NEXT POSITION
139B  8A FB                     MOV     BH,BL           ; SET 2ND ATTRIBUTE TO BH
139D  E2 D7                     LOOP    GWRT22

139F  E9 148E R                 JMP     GWRT212         ;--- END

13A2                    GWRT26:
13A2  80 FE 08                  CMP     DH,8            ; 160X200 16 COLOR ?
13A5  75 18                     JNE     GWRT27          ; NO

                                                        ;--- 160X200 16 COLOR: MODE 8
13A7  D1 E7                     SAL     DI,1            ;
```

```
13A9  D1 E7                     SAL     DI,1                    ; OFFSET*4 SINCE 4 BYTES/CHAR

13AB  53                        PUSH    BX                      ; SAVE ATTRIBUTE
13AC  8A DF                     MOV     BL,BH                   ; SET 1ST ATTRIBUTE

13AE  B6 09                     MOV     DH,CBOX_ROW/2           ; SET LOOP COUNT ( 18 ROW / 2 SCAN )
13B0  E8 14F3 R                 CALL    G_W_4                   ; WRITE LEFT PART

13B3  83 C7 04                  ADD     DI,4                    ; NEXT WRITE POSITION

13B6  5B                        POP     BX                      ; RESTORE ATTRIBUTE
13B7  B6 09                     MOV     DH,CBOX_ROW/2           ; SET LOOP COUNT ( 18 ROW / 2 SCAN )
13B9  E8 14F3 R                 CALL    G_W_4                   ; WRITE RIGHT PART

13BC  E9 148E R                 JMP     GWRT212                 ;--- END

13BF                    GWRT27:
13BF  80 FE 09                  CMP     DH,9                    ; 320X200 16 COLOR ?
13C2  75 32                     JNE     GWRT281                 ; NO


13C4  D1 E7                     SAL     DI,1                    ;--- 320X200 16 COLOR
13C6  D1 E7                     SAL     DI,1                    ;
13C8  53                        PUSH    BX                      ; OFFSET*4 SINCE 4 BYTES/CHAR
13C9  8A DF                     MOV     BL,BH                   ; SAVE ATTRIBUTE
                                                                ; SET 1ST ATTRIBUTE
13CB  B6 09                     MOV     DH,CBOX_ROW/(4/2)       ; SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC. )

13CD  8B 46 02                  MOV     AX,[BP+WPOSN]           ; GET ROW COLUMN POSITION
13D0  D0 EC                     SHR     AH,1                    ; ODD ROW ?
13D2  72 0F                     JC      GWRT28                  ; YES

13D4  E8 14F3 R                 CALL    G_W_4                   ;--- WRITE EVEN ROW CHARACTER
                                                                ; WRITE LEFT PART
13D7  83 C7 04                  ADD     DI,4                    ; NEXT WRITE POSITION

13DA  5B                        POP     BX                      ; RESTORE ATTRIBUTE
13DB  B6 09                     MOV     DH,CBOX_ROW/(4/2)       ; SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC. )
13DD  E8 14F3 R                 CALL    G_W_4                   ; WRITE RIGHT PART

13E0  E9 148E R                 JMP     GWRT212                 ;--- END

13E3                    GWRT28:
13E3  81 C7 3FB0                ADD     DI,4000H-80             ;--- WRITE ODD ROW CHARACTER
13E7  E8 1527 R                 CALL    G_W_40                  ; ADJUST FOR ODD ROW
                                                                ; WRITE LEFT PART
13EA  83 C7 04                  ADD     DI,4                    ; NEXT WRITE POSITION

13ED  5B                        POP     BX                      ; RESTORE ATTRIBUTE
13EE  B6 09                     MOV     DH,CBOX_ROW/(4/2)       ; SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC. )

13F0  E8 1527 R                 CALL    G_W_40                  ; WRITE RIGHT PART

13F3  E9 148E R                 JMP     GWRT212                 ;--- END

13F6                    GWRT281:
13F6  80 FE 0B                  CMP     DH,0BH
13F9  75 4D                     JNE     GWRT29                  ; 640X200 16 COLOR ?
                                                                ; NO


                                                                ;--- 640X200 16 COLOR
                                                                ;--- WRITE V-RAM 2
13FB  53                        PUSH    BX                      ; SAVE ATTRIBUTE
13FC  57                        PUSH    DI                      ; SAVE REGEN POINTER
13FD  56                        PUSH    SI                      ; SAVE CHARACTER PATTERN ADDRESS

13FE  8B C3                     MOV     AX,BX
1400  25 8080                  AND     AX,XOR_BIT*100H OR XOR_BIT      ; SAVE ATTRIBUTE TO BH
1403  81 E3 7C7C                AND     BX,7C7CH                        ; GET XOR BIT
1407  D1 EB                     SHR     BX,1                            ; MASK XOR BIT
1409  D1 EB                     SHR     BX,1                    ; SHIFT ATTRIBUTE FOR WRITE TO V-RAM2
140B  0B D8                     OR      BX,AX                   ;
140D  D1 E7                     SAL     DI,1                    ; RESTORE XOR BIT
                                                                ; OFFSET*2 SINCE 2 BYTES/CHAR
140F  53                        PUSH    BX
1410  8A DF                     MOV     BL,BH                   ; SAVE ATTRIBUTE
1412  B6 09                     MOV     DH,CBOX_ROW/(4/2)       ; SET 1ST ATTRIBUTE
                                                                ; SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC.)
1414  8B 46 02                  MOV     AX,[BP+WPOSN]           ; GET ROW COLUMN POSITION
1417  D0 EC                     SHR     AH,1                    ; ODD ROW ?
1419  72 0E                     JC      GWRT282                 ; YES

141B  E8 1498 R                 CALL    G_W_1                   ;--- WRITE EVEN ROW CHARACTER
141E  83 C7 02                  ADD     DI,2                    ; WRITE LEFT PART
                                                                ; NEXT WRITE POSITION
1421  5B                        POP     BX
1422  B6 09                     MOV     DH,CBOX_ROW/(4/2)       ; RESTORE ATTRIBUTE
1424  E8 1498 R                 CALL    G_W_1                   ; SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC.)
                                                                ; WRITE RIGHT PART
1427  EB 10                     JMP     SHORT GWRT283           ;--- END

1429                    GWRT282:
1429  81 C7 3FB0                ADD     DI,4000H-80             ;--- WRITE ODD ROW CHARACTER
142D  E8 14C2 R                 CALL    G_W_10                  ; ADJUST FOR ODD ROW
                                                                ; WRITE LEFT PART
1430  83 C7 02                  ADD     DI,2
                                                                ; NEXT WRITE POSITION
1433  5B                        POP     BX
1434  B6 09                     MOV     DH,CBOX_ROW/(4/2)       ; RESTORE ATTRIBUTE
1436  E8 14C2 R                 CALL    G_W_10                  ; SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC.)
1439                    GWRT283:                                ; WRITE RIGHT PART
1439  5E                        POP     SI
143A  5F                        POP     DI                      ; RESTORE CHARACTER PATTERN ADDRESS
143B  5B                        POP     BX                      ; RESTORE REGEN POINTER
                                                                ; RESTORE ATTRIBUTE
```

```
143C  1E                      PUSH    DS                ; SAVE DS
143D  E8 0000 E               CALL    DDS               ; POINT TO DATA AREA

1440  E8 1D95 R               CALL    GET_DIRSEG        ; GET DIRECT ACCESS SEGENT OF V-RAM 1
1443  8E C0                   MOV     ES,AX             ; SET IT TO ES
1445  1F                      POP     DS                ; RESTORE DS
1446  EB 05                   JMP     SHORT   GWRT291 ;--- WRITE V-RAM 1

1448                    GWRT29:
1448  80 FE 0A                CMP     DH,0AH            ; 640X200 4 COLOR ?
144B  75 2E                   JNE     GWRT211           ; NO, MODE = 4,5

144D                    GWRT291:                        ;--- 640X200 4 COLOR
144D  D1 E7                   SAL     DI,1              ; OFFSET*2 SINCE 2 BYTES/CHAR

144F  53                      PUSH    BX                ; SAVE ATTRIBUTE
1450  8A DF                   MOV     BL,BH             ; SET 1ST ATTRIBUTE
1452  B6 09                   MOV     DH,CBOX_ROW/(4/2); SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC.)

1454  8B 46 02                MOV     AX,[BP+WPOSN]     ; GET ROW COLUMN POSITION
1457  D0 EC                   SHR     AH,1              ; ODD ROW ?
1459  72 0E                   JC      GWRT210           ; YES
                                                        ;--- WRITE EVEN ROW CHARACTER
145B  E8 1498 R               CALL    G_W_1             ; WRITE LEFT PART

145E  83 C7 02                ADD     DI,2              ; NEXT WRITE POSITION

1461  5B                      POP     BX                ; RESTORE ATTRIBUTE
1462  B6 09                   MOV     DH,CBOX_ROW/(4/2); SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC.)
1464  E8 1498 R               CALL    G_W_1             ; WRITE RIGHT PART

1467  EB 25                   JMP     SHORT GWRT212   ;--- END

1469                    GWRT210:                        ;--- WRITE ODD ROW CHARACTER
1469  81 C7 3FB0              ADD     DI,4000H-80       ; ADJUST FOR ODD ROW
146D  E8 14C2 R               CALL    G_W_10            ; WRITE LEFT PART

1470  83 C7 02                ADD     DI,2              ; NEXT WRITE POSITION

1473  5B                      POP     BX                ; RESTORE ATTRIBUTE
1474  B6 09                   MOV     DH,CBOX_ROW/(4/2); SET LOOP COUNT ( 18 ROW / 4 SCAN / 2 DEC.)
1476  E8 14C2 R               CALL    G_W_10            ; WRITE RIGHT PART

1479  EB 13                   JMP     SHORT GWRT212   ;--- END

147B                    GWRT211:                        ;--- 320X200 4 COLOR/SHADE
147B  D1 E7                   SAL     DI,1              ; OFFSET*2 SINCE 2 BYTES/CHAR

147D  53                      PUSH    BX                ; SAVE ATTRIBUTE
147E  8A DF                   MOV     BL,BH             ; SET 1ST ATTRIBUTE
1480  B6 09                   MOV     DH,CBOX_ROW/2     ; SET LOOP COUNT ( 18 ROW / 2 SCAN )
1482  E8 14C8 R               CALL    G_W_2             ; WRITE LEFT PART

1485  83 C7 02                ADD     DI,2              ; NEXT WRITE POSITION

1488  5B                      POP     BX                ; RESTORE ATTRIBUTE
1489  B6 09                   MOV     DH,CBOX_ROW/2     ; SET LOOP COUNT ( 18 ROW / 2 SCAN )
148B  E8 14C8 R               CALL    G_W_2             ; WRITE RIGHT PART

148E                    GWRT212:                        ;--- END
148E  B8 0002                 MOV     AX,2              ;
1491  01 46 00                ADD     [BP+WGPOSN],AX  ; ADVANCE WRITE POSITION

1494  07                      POP     ES                ; RESTORE ES
1495  5F                      POP     DI                ; RESTORE DI
1496  1F                      POP     DS                ; RESTORE DS

1497  C3                      RET

1498                    G_WRITE2 ENDP


                        ;--------------------------------------------------------------
                        ;
                        ;       G_W_1
                        ;
                        ;       THIS ROUTINE WRITES ONE CHARACTER PATTERN INTO REGEN
                        ;               ( 640X200 4 COLOR )
                        ;
                        ;       INPUT   BL =    ATTRIBUTE
                        ;               DH =    ROW NUMBER OF ONE SCAN BANK
                        ;               DS:SI = CHARACTER PATTERN ADDRESS
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT          NONE
                        ;
                        ;       CALL            EX_2BIT
                        ;                       G_W_R1
                        ;
                        ;       VOLATILE        AX,BX,DH,SI
                        ;
                        ;--------------------------------------------------------------
                                ASSUME  CS:CODE, DS:STACK, ES:VIDEO_RAM

1498                    G_W_1   PROC    NEAR

1498  E8 152D R                 CALL    EX_2BIT         ; EXPAND LOW 2 BITS IN BX
                                                        ;--- WRITE CHARACTER
149B  57                        PUSH    DI              ; SAVE REGEN POINTER
149C                    GW11:
149C  E8 1554 R                 CALL    G_W_R1          ; DO FIRST DOT ROW
149F  81 C7 2000                ADD     DI,2000H        ; ADJUST REGEN POINTER
14A3  E8 1554 R                 CALL    G_W_R1          ; DO NEXT DOT ROW
14A6  81 C7 2000                ADD     DI,2000H        ; ADJUST REGEN POINTER
```

```
14AA  FE CE                     DEC     DH              ; ALL DONE ?
14AC  74 12                     JZ      GW13            ; NO
14AE                    GW12:
14AE  E8 1554 R                 CALL    G_W_R1          ; DO NEXT DOT ROW
14B1  81 C7 2000                ADD     DI,2000H        ; ADJUST REGEN POINTER
14B5  E8 1554 R                 CALL    G_W_R1          ; DO NEXT DOT ROW
14B8  81 EF 5F60                SUB     DI,6000H-160    ; ADJUST REGEN POINTER TO NEXT ROW

14BC  FE CE                     DEC     DH
14BE  75 DC                     JNZ     GW11            ; KEEP GOING
14C0                    GW13:
14C0  5F                        POP     DI              ; RECOVER REGEN POINTER
14C1  C3                        RET
14C2                    G_W_1   ENDP

                        ;--------------------------------------------------------------
                        ;
                        ;       G_W_10
                        ;
                        ;       THIS ROUTINE WRITES ONE CHARACTER PATTERN INTO REGEN ( FOR ODD ROW )
                        ;               ( 640X200 4 COLOR )
                        ;
                        ;       INPUT   BL =    ATTRIBUTE
                        ;               DH =    ROW NUMBER OF ONE SCAN BANK
                        ;               DS:SI = CHARACTER PATTERN ADDRESS
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT          NONE
                        ;
                        ;       CALL            EX_2BIT
                        ;                       (G_W_1)
                        ;
                        ;       VOLATILE        AX,BX,DH,SI
                        ;
                        ;--------------------------------------------------------------
                                ASSUME  CS:CODE, DS:STACK, ES:VIDEO_RAM
14C2                    G_W_10  PROC    NEAR
14C2  E8 152D R                 CALL    EX_2BIT         ; EXPAND LOW 2 COLOR BITS IN BL
                                                        ;--- WRITE CHARACTER
14C5  57                        PUSH    DI              ; SAVE REGEN POINTER
14C6  EB E6                     JMP     GW12
14C8                    G_W_10  ENDP

                        ;--------------------------------------------------------------------
                        ;
                        ;       G_W_2
                        ;
                        ;       THIS ROUTINE WRITES ONE CHARACTER PATTERN INTO REGEN
                        ;               ( 320X200 4 COLOR, 320X200 4 SHADE )
                        ;
                        ;       INPUT   BL =    ATTRIBUTE
                        ;               DH =    ROW NUMBER OF ONE SCAN BANK
                        ;               DS:SI = CHARACTER PATTERN ADDRESS
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT          NONE
                        ;
                        ;       CALL            G_W_R2
                        ;
                        ;       VOLATILE        AX,BX,DH,SI
                        ;
                        ;--------------------------------------------------------------------
14C8                    G_W_2   PROC    NEAR
14C8  8A D3                     MOV     DL,BL           ; COPY ATTRIBUTE TO DL
                                                        ;---EXPANDS THE LOW 2 BITS IN BL TO FILL THE BX
14CA  80 E3 03                  AND     BL,3            ; ISOLATE THE COLOR BITS
14CD  8A C3                     MOV     AL,BL           ; COPY TO AL
14CF  51                        PUSH    CX              ; SAVE REGISTER
14D0  B9 0003                   MOV     CX,3            ; NUMBER OF TIMES TO DO THIS
14D3                    GW21:
14D3  D0 E0                     SAL     AL,1
14D5  D0 E0                     SAL     AL,1            ; LEFT SHIFT BY 2
14D7  0A D8                     OR      BL,AL           ; ANOTHER COLOR VERSION INTO BL
14D9  E2 F8                     LOOP    GW21            ; FILL ALL OF BL

14DB  8A FB                     MOV     BH,BL           ; FILL UPPER PORTION
14DD  59                        POP     CX              ; REGISTER BACK
14DE  57                        PUSH    DI              ;--- WRITE CHARACTER
14DF                                                    ; SAVE REGEN POINTER
14DF  E8 1572 R         GW22:   CALL    G_W_R2          ; DO FIRST 2 BYTES
14E2  81 C7 2000                ADD     DI,2000H        ; NEXT SPOT IN REGEN
14E6  E8 1572 R                 CALL    G_W_R2          ; DO NEXT 2 BYTES
14E9  81 EF 1FB0                SUB     DI,2000H-80

14ED  FE CE                     DEC     DH
14EF  75 EE                     JNZ     GW22            ; KEEP GOING

14F1  5F                        POP     DI              ; RECOVER REGEN POINTER
14F2  C3                        RET
14F3                    G_W_2   ENDP
```

```
                        ;--------------------------------------------------------
                        ;
                        ;       G_W_4
                        ;
                        ;       THIS ROUTINE WRITES ONE CHARACTER PATTERN INTO REGEN
                        ;               ( 160X200 16 COLOR, 320X200 16 COLOR )
                        ;
                        ;       INPUT   BL =    ATTRIBUTE  (LOW 4 BITS)
                        ;               DH =    ROW NUMBER OF ONE SCAN BANK
                        ;               [BP+WC_MODE]= CURRENT CRT MODE (MASKED)
                        ;               DS:SI = CHARACTER PATTERN ADDRESS
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT          NONE
                        ;
                        ;       CALL            EX_4BIT
                        ;                       G_W_R4
                        ;
                        ;       VOLATILE        AX,BX,DH,SI
                        ;
                        ;--------------------------------------------------------------------

14F3                    G_W_4   PROC    NEAR

14F3  E8 1542 R                 CALL    EX_4BIT         ; EXPANDS THE LOW 4 BITS
                                                        ;--- WRITE CHARACTER
14F6  57                        PUSH    DI              ; SAVE REGEN POINTER
14F7                    GW41:
14F7  E8 158C R                 CALL    G_W_R4          ; EXPAND DOT ROW IN REGEN
14FA  81 C7 2000                ADD     DI,2000H        ; POINT TO NEXT REGEN ROW
14FE  E8 158C R                 CALL    G_W_R4          ; EXPAND DOT ROW IN REGEN

1501  80 7E 09 09              CMP     BYTE PTR [BP+WC_MODE],09H       ; USING 32K REGEN AREA?
1505  75 16                    JNE     GW43            ; JUMP IF 16K REGEN

1507  FE CE                    DEC     DH              ; DECREMENT COUNTER
1509  74 1A                    JZ      GW44            ; ALL DONE

150B  81 C7 2000                ADD     DI,2000H        ; POINT TO NEXT REGEN ROW
150F                    GW42:
150F  E8 158C R                 CALL    G_W_R4          ; EXPAND DOT ROW IN REGEN
1512  81 C7 2000                ADD     DI,2000H        ; POINT TO NEXT REGEN ROW
1516  E8 158C R                 CALL    G_W_R4          ; EXPAND DOT ROW IN REGEN
1519  81 EF 3FB0                SUB     DI,4000H-80     ; ADJUST REGEN POINTER
151D                    GW43:
151D  81 EF 1FB0                SUB     DI,2000H-80     ; ADJUST REGEN POINTER TO NEXT ROW

1521  FE CE                    DEC     DH
1523  75 D2                    JNZ     GW41            ; KEEP GOING
1525                    GW44:
1525  5F                        POP     DI              ; RECOVER REGEN POINTER
1526  C3                        RET

1527                    G_W_4   ENDP


                        ;--------------------------------------------------------
                        ;
                        ;       G_W_40
                        ;
                        ;       THIS ROUTINE WRITES ONE CHARACTER PATTERN INTO REGEN (ODD ROW)
                        ;               ( 160X200 16 COLOR, 320X200 16 COLOR )
                        ;
                        ;       INPUT   BL =    ATTRIBUTE  (LOW 4 BITS)
                        ;               DH =    ROW NUMBER OF ONE SCAN BANK
                        ;               [BP+WC_MODE]= CURRENT CRT MODE (NIBBLE)
                        ;               DS:SI = CHARACTER PATTERN ADDRESS
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT          NONE
                        ;
                        ;       CALL            EX_4BIT
                        ;                       (G_W_4)
                        ;
                        ;       VOLATILE        AX,BX,DH,SI
                        ;
                        ;--------------------------------------------------------------------

1527                    G_W_40  PROC    NEAR

1527  E8 1542 R                 CALL    EX_4BIT         ; EXPANDS THE LOW 4 BITS
                                                        ;--- WRITE CHARACTER
152A  57                        PUSH    DI              ; SAVE REGEN POINTER

152B  EB E2                     JMP     GW42            ;

152D                    G_W_40  ENDP

                        ;--------------------------------------------------------
                        ;
                        ;       EX_2BIT
                        ;
                        ;       THIS ROUTINE EXPANDS LOW 2 COLOR BITS IN BL
                        ;               ( 640X200 4 COLOR )
                        ;
                        ;       INPUT   BL =    ATTRIBUTE
                        ;
                        ;       OUTPUT  DL =    ATTRIBUTE
                        ;               BX =    EXPANDED COLOR
                        ;
                        ;       CALL            NONE
                        ;
                        ;       VOLATILE        AX,BX
```

```
                        ;
                        ;--------------------------------------------------------
                                ASSUME  CS:CODE, DS:STACK, ES:VIDEO_RAM

152D                    EX_2BIT PROC    NEAR

152D  8A D3                     MOV     DL,BL           ; COPY ATTRIBUTE TO DL
                                                        ;--- EXPAND LOW 2 COLOR BITS IN BL (C1C0)
152F  33 C0                     XOR     AX,AX           ; INTO BX (C0C0C0C0C0C0C0C0C1C1C1C1C1C1C1C1)
1531  F6 C3 01                  TEST    BL,1            ; C0 COLOR BIT ON?
1534  74 02                     JZ      EX2B1           ; NO, JUMP
1536  B4 FF                     MOV     AH,0FFH         ; YES, SET ALL C0 BITS ON
1538                    EX2B1:
1538  F6 C3 02                  TEST    BL,2            ; C1 COLOR BIT ON?
153B  74 02                     JZ      EX2B2           ; NO, JUMP
153D  B0 FF                     MOV     AL,0FFH         ; YES, SET ALL C1 BITS ON
153F                    EX2B2:
153F  8B D8                     MOV     BX,AX           ; COLOR MASK IN BX
1541  C3                       RET

1542                    EX_2BIT ENDP

                        ;--------------------------------------------------------
                        ;
                        ;       EX_4BIT
                        ;
                        ;       THIS ROUTINE EXPANDS LOW 4 COLOR BITS IN BL
                        ;               ( 160X200 16 COLOR, 320X200 16 COLOR )
                        ;
                        ;       INPUT   BL =    ATTRIBUTE  (LOW 4 BITS)
                        ;
                        ;       OUTPUT  DL =    ATTRIBUTE
                        ;               BX =    EXPANDED ATTRIBUTE
                        ;
                        ;       CALL            NONE
                        ;
                        ;       VOLATILE        AX
                        ;
                        ;--------------------------------------------------------

1542                    EX_4BIT PROC    NEAR

1542  8A D3                     MOV     DL,BL           ; COPY ATTRIBUTE TO DL
1544  51                       PUSH    CX              ;---EXPANDS THE LOW 4 BITS IN BL TO FILL BX

1545  80 E3 0F                  AND     BL,0FH          ; ISOLATE THE COLOR BITS
1548  8A FB                     MOV     BH,BL           ; COPY TO BH

154A  B1 04                     MOV     CL,4            ; MOVE TO HIGH NIBBLE
154C  D2 E7                     SHL     BH,CL           ;

154E  0A FB                     OR      BH,BL           ; MAKE BYTE FROM HIGH AND LOW NIBBLES
1550  8A DF                     MOV     BL,BH           ;
1552  59                       POP     CX

1553  C3                       RET

1554                    EX_4BIT ENDP

                        ;--------------------------------------------------------
                        ;
                        ;       G_W_R1
                        ;
                        ;       THIS ROUTINE WRITES ONE ROW OF CHARACTER PATTERN INTO REGEN
                        ;       EXPAND 1 DOT ROW OF A CHAR INTO 2 BYTES
                        ;               ( 640X200 4 COLOR )
                        ;
                        ;       INPUT   BX =    EXPANDED ATTRIBUTE
                        ;               DL =    ATTRIBUTE
                        ;               DS:SI = CHARACTER PATTERN ADDRESS
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT  SI = SI+1
                        ;
                        ;       CALL            NONE
                        ;
                        ;       VOLATILE        AX
                        ;
                        ;--------------------------------------------------------
                                ASSUME  CS:CODE, DS:STACK, ES:VIDEO_RAM

1554                    G_W_R1  PROC    NEAR

1554  AC                        LODSB                   ; GET CODE POINT
1555  8A E0                     MOV     AH,AL           ; COPY INTO AH
1557  23 C3                     AND     AX,BX           ; SET COLOR
1559  F6 C2 80                  TEST    DL,XOR_BIT      ; XOR FUNCTION?
155C  74 07                     JZ      GWR11           ; NO, JUMP
155E  26: 32 25                 XOR     AH,ES:[DI]      ; EXCLUSIVE OR WITH CURRENT DATA
1561  26: 32 45 01              XOR     AL,ES:[DI+1]
1565                    GWR11:
1565  26: 88 25                 MOV     ES:[DI],AH      ; STORE IN REGEN BUFFER
1568  26: 88 45 01              MOV     ES:[DI+1],AL

156C  C3                        RET

156D                    G_W_R1  ENDP

                        ;--------------------------------------------------------
                        ;
                        ;       EX_W_R2
                        ;
```

```
                        ;       THIS ROUTINE EXPANDS ONE ROW OF CHARACTER PATTERN
                        ;        AND WRITES IT INTO REGEN
                        ;
                        ;       INPUT   AL =    CHARACTER PATTERN
                        ;               BX =    EXPANDED ATTRIBUTE
                        ;               DL =    ATTRIBUTE
                        ;               DS:SI = CHARACTER PATTERN ADDRESS
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT  SI = SI+1
                        ;
                        ;       CALL            EX_NIBBLE
                        ;                       (G_W_R2)
                        ;
                        ;       VOLATILE        AX
                        ;
                        ;--------------------------------------------------------

156D                    EX_W_R2 PROC    NEAR

156D  E8 15C2 R                 CALL    EX_NIBBLE       ; QUAD UP THE LOW NIBBLE
1570  EB 04                     JMP     SHORT GWR21

1572                    EX_W_R2 ENDP

                        ;--------------------------------------------------------
                        ;
                        ;       G_W_R2
                        ;
                        ;       THIS ROUTINE WRITES ONE ROW OF CHARACTER PATTERN INTO REGEN
                        ;       EXPAND 1 DOT ROW OF A CHAR INTO 2 BYTES
                        ;               ( 320X200 4 COLOR, 320X200 4 SHADE )
                        ;
                        ;       INPUT   BX =    EXPANDED ATTRIBUTE
                        ;               DL =    ATTRIBUTE
                        ;               DS:SI = CHARACTER PATTERN ADDRESS
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT  SI = SI+1
                        ;
                        ;       CALL            EX_BYTE
                        ;
                        ;       VOLATILE        AX
                        ;
                        ;--------------------------------------------------------

1572                    G_W_R2  PROC    NEAR

1572  AC                        LODSB                   ; GET CODE POINT
1573  E8 15A0 R                 CALL    EX_BYTE         ; DOUBLE UP ALL THE BITS
1576                    GWR21:
1576  23 C3                     AND     AX,BX           ; CONVERT THEM TO FOREGROUND COLOR
                                                        ; ( 0 BACK )
1578  F6 C2 80                  TEST    DL,XOR_BIT      ; IS THIS XOR FUNCTION?
157B  74 07                     JZ      GWR22           ; NO, STORE IT IN AS IT IS

157D  26: 32 25                 XOR     AH,ES:[DI]      ; DO FUNCTION WITH HALF
1580  26: 32 45 01                XOR     AL,ES:[DI+1]    ;  AND WITH OTHER HALF
1584                    GWR22:
1584  26: 88 25                 MOV     ES:[DI],AH      ; STORE FIRST BYTE
1587  26: 88 45 01              MOV     ES:[DI+1],AL    ; STORE SECOND BYTE
158B  C3                        RET

158C                    G_W_R2  ENDP

                        ;--------------------------------------------------------
                        ;
                        ;       G_W_R4
                        ;
                        ;       THIS ROUTINE WRITES ONE ROW OF CHARACTER PATTERN INTO REGEN
                        ;       EXPAND 1 DOT ROW OF A CHAR INTO 4 BYTES
                        ;               ( 160X200 16 COLOR, 320X200 16 COLOR )
                        ;
                        ;       INPUT   BX =    EXPANDED ATTRIBUTE
                        ;               DL =    ATTRIBUTE
                        ;               DS:SI = CHARACTER PATTERN ADDRESS
                        ;               ES:DI = REGEN ADDRESS TO WRITE
                        ;
                        ;       OUTPUT  SI = SI+1
                        ;
                        ;       CALL            EX_W_R2
                        ;
                        ;       VOLATILE        AX
                        ;
                        ;--------------------------------------------------------

158C                    G_W_R4  PROC    NEAR

158C  AC                        LODSB                   ; GET CODE POINT
158D  50                        PUSH    AX              ; SAVE

158E  51                        PUSH    CX
158F  B1 04                     MOV     CL,4            ; MOV HIGH NIBBLE TO LOW
1591  D2 E8                     SHR     AL,CL
1593  59                       POP     CX

1594  E8 156D R                 CALL    EX_W_R2         ; EXPAND TO 2 BYTES & PUT IN REGEN
1597  58                        POP     AX              ; RECOVER CODE POINT

1598  47                        INC     DI              ; ADJUST REGEN POINTER
1599  47                        INC     DI
159A  E8 156D R                 CALL    EX_W_R2         ; EXPAND LOW NIBBLE & PUT IN REGEN
```

```
159D  4F                        DEC     DI              ; RESTORE REGEN POINTER
159E  4F                        DEC     DI

159F  C3                        RET

15A0                    G_W_R4  ENDP

                        ;-----------------------------------------
                        ;
                        ;       EX_BYTE         EXPAND BYTE
                        ;
                        ;       THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
                        ;        OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
                        ;        THE RESULT IS LEFT IN AX
                        ;
                        ;       INPUT   AL = BIT TO DOUBLE
                        ;
                        ;       OUTPUT  AX = DOUBLED BITS
                        ;
                        ;       CALL            NONE
                        ;
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------------

15A0                    EX_BYTE PROC    NEAR

15A0  52                        PUSH    DX              ; SAVE REGISTERS
15A1  51                        PUSH    CX
15A2  53                        PUSH    BX
15A3  2B D2                     SUB     DX,DX           ; RESULT REGISTER
15A5  B9 0001                   MOV     CX,1            ; MASK REGISTER
15A8                    EXBYTE1:
15A8  8B D8                     MOV     BX,AX           ; BASE INTO TEMP
15AA  23 D9                     AND     BX,CX           ; USE MASK TO EXTRACT A BIT
15AC  0B D3                     OR      DX,BX           ; PUT INTO RESULT REGISTER
15AE  D1 E0                     SHL     AX,1
15B0  D1 E1                     SHL     CX,1            ; SHIFT BASE AND MASK BY 1
15B2  8B D8                     MOV     BX,AX           ; BASE TO TEMP
15B4  23 D9                     AND     BX,CX           ; EXTRACT THE SAME BIT
15B6  0B D3                     OR      DX,BX           ; PUT INTO RESULT

15B8  D1 E1                     SHL     CX,1            ; SHIFT ONLY MASK NOW, MOVING TO NEXT BASE
15BA  73 EC                     JNC     EXBYTE1         ; USE MASK BIT COMING OUT TO TERMINATE

15BC  8B C2                     MOV     AX,DX           ; RESULT TO PARM REGISTER
15BE  5B                        POP     BX
15BF  59                        POP     CX              ; RECOVER REGISTERS
15C0  5A                        POP     DX
15C1  C3                        RET                     ; ALL DONE

15C2                    EX_BYTE ENDP

                        ;-----------------------------------------
                        ;
                        ;       EX_NIBBLE       EXPAND NIBBLE
                        ;
                        ;       THIS ROUTINE TAKES THE LOW NIBBLE IN AL AND QUADS ALL
                        ;        OF THE BITS, TURNING THE 4 BITS INTO 16 BITS.
                        ;        THE RESULT IS LEFT IN AX
                        ;
                        ;       INPUT   AL = NIBBLE DATA
                        ;.
                        ;       OUTPUT  AX = QUADED BITS
                        ;
                        ;       CALL            NONE
                        ;
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------------

15C2                    EX_NIBBLE PROC    NEAR

15C2  52                        PUSH    DX              ; SAVE REGISTERS
15C3  33 D2                     XOR     DX,DX           ; RESULT REGISTER
15C5  A8 08                     TEST    AL,8
15C7  74 03                     JZ      EXNBL1

15C9  80 CE F0                  OR      DH,0F0H
15CC                    EXNBL1:
15CC  A8 04                     TEST    AL,4
15CE  74 03                     JZ      EXNBL2

15D0  80 CE 0F                  OR      DH,0FH
15D3                    EXNBL2:
15D3  A8 02                     TEST    AL,2
15D5  74 03                     JZ      EXNBL3

15D7  80 CA F0                  OR      DL,0F0H
15DA                    EXNBL3:
15DA  A8 01                     TEST    AL,1
15DC  74 03                     JZ      EXNBL4

15DE  80 CA 0F                  OR      DL,0FH
15E1                    EXNBL4:
15E1  8B C2                     MOV     AX,DX
15E3  5A                        POP     DX              ; RESULT TO PARM REGISTER
15E4  C3                        RET                     ; RECOVER REGISTERS
                                                        ; ALL DONE
15E5                    EX_NIBBLE ENDP
```

```
                        ;----------------------------------------
                        ;
                        ;       GRAPH_POSITION
                        ;
                        ;       THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
                        ;        THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
                        ;        INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
                        ;        FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
                        ;        BE DOUBLED.
                        ;
                        ;       INPUT    NO REGISTERS,MEMORY LOCATION CURSOR_POSN IS USED
                        ;       OUTPUT   AX CONTAINS OFFSET INTO REGEN BUFFER
                        ;
                        ;       VOLATILE        NONE
                        ;
                        ;----------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

15E5                    GRAPH_POSITION  PROC    NEAR
15E5  A1 035C R                 MOV     AX,CURSOR_POSN  ; GET CURRENT CURSOR

15E8                    GRAPH_POSN      LABEL   NEAR

15E8  53                        PUSH    BX              ; SAVE REGISTER
15E9  51                        PUSH    CX              ; SAVE CX

15EA  8B D8                     MOV     BX,AX           ; SAVE A COPY OF CURRENT CURSOR
15EC  8A C4                     MOV     AL,AH           ; GET ROWS TO AL
15EE  F6 26 004A R              MUL     BYTE PTR CRT_COLS ; MULTIPLY BY BYTES/COLUMN

15F2  8A 3E 0049 R              MOV     BH,CRT_MODE     ; GET CRT MODE
15F6  80 E7 0F                  AND     BH,KJ_OFF       ; MASK VIDEO PROCESSOR NO. OFF

15F9  80 3E 0049 R 14           CMP     CRT_MODE,KJGRAPH; KANJI GRAPHICS MODE ?
15FE  73 0B                     JAE     GRPOS2          ; YES

                                                        ;--- AH/ANK GRAPHICS MODE
1600  80 FF 09                  CMP     BH,9            ; MODE USING 32K REGEN?
1603  73 02                     JNC     GRPOS1          ; YES, JUMP

1605  D1 E0                     SHL     AX,1            ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
1607                    GRPOS1:
1607  D1 E0                     SHL     AX,1
1609  EB 13                     JMP     SHORT GRPOS4

160B                    GRPOS2:                         ;--- KANJI GRAPHICS MODE
160B  8B C8                     MOV     CX,AX           ; SAVE AX VALUE FOR AFTER CALCULATION
160D  80 FF 09                  CMP     BH,9            ; MODE USING 32K REGEN?
1610  73 04                     JNC     GRPOS3          ; YES, JUMP

1612  D1 E0                     SAL     AX,1            ; MULTIPLY * 9  SINCE 18/2=9 ROWS/BYTE
1614  D1 E1                     SAL     CX,1            ;
1616                    GRPOS3:
1616  D1 E0                     SAL     AX,1            ;
1618  D1 E0                     SAL     AX,1            ; MULTIPLY * 4.5 SINCE 18/4=4.5 ROWS/BYTE
161A  D1 E9                     SHR     CX,1            ;
161C  03 C1                     ADD     AX,CX           ;
161E                    GRPOS4:
161E  2A FF                     SUB     BH,BH           ; ISOLATE COLUMN VALUE
1620  03 C3                     ADD     AX,BX           ; DETERMINE OFFSET

1622  59                        POP     CX              ; RESTORE CX
1623  5B                        POP     BX              ; RECOVER POINTER

1624  C3                        RET                     ; ALL DONE

1625                    GRAPH_POSITION  ENDP
                        ;--------------------------------------------------
                        ;
                        ;       GRAPHICS WRITE
                        ;
                        ;       THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
                        ;       POSITION ON THE SCREEN.
                        ;
                        ;       INPUT           AH = CURRENT CRT MODE (MASKED)
                        ;                       AL = CHARACTER TO WRITE
                        ;                       BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
                        ;                            IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BU
                        ;                            (0 IS USED FOR THE BACKGROUND COLOR)
                        ;                       CX = NUMBER OF CHARS TO WRITE
                        ;                       DS = DATA SEGMENT
                        ;                       ES = REGEN SEGMENT
                        ;
                        ;       OUTPUT          NOTHING
                        ;
                        ;       CALL            FONT
                        ;                       GRAPH_POSITION
                        ;                       G_W_1
                        ;                       G_W_2
                        ;                       G_W_4
                        ;
                        ;       WORK            [BP+WC_MODE]= CURRENT CRT MODE (MASKED)
                        ;                       [BP+WFONT - WFONT+7] =  FONT PATTERN
                        ;
                        ;       VOLATILE        AX,BX,CX,DX,SI,DI,DS
                        ;
                        ;       FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN
                        ;       ROM.
                        ;
                        ;--------------------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

1625                    GRAPHICS_WRITE  PROC    NEAR
```

```
1625  55                        PUSH    BP                ; SAVE CURRENT BP
1626  83 EC 12                  SUB     SP,GW_LOCAL       ; ALLOCATE LOCAL WORK AREA
1629  8B EC                     MOV     BP,SP             ; ASSIGN BP AS FRAME POINTER

162B  88 66 09                  MOV     [BP+WC_MODE],AH ; SAVE CURRENT CRT MODE

162E  06                        PUSH    ES                ; SAVE REGEN SEGMENT
162F  50                        PUSH    AX                ; SAVE CODE POINT AND CRT MODE VALUE

1630  53                        PUSH    BX                ; SAVE COLOR ATTRIBUTE
1631  51                        PUSH    CX                ; SAVE NUMBER OF CHAR

1632  16                        PUSH    SS                ; SET DESTINATION SEGMENT FOR FONT
1633  07                        POP     ES                ;   TO ES
                                        ASSUME  ES:STACK

1634  8D 5E 0A                  LEA     BX,[BP+WFONT]     ; SET DESTINATION OFFSET FOR FONT TO BX

1637  32 E4                     XOR     AH,AH             ; CLEAR FOR 1 BYTE CODE
1639  8B C8                     MOV     CX,AX             ; SET CHARACTER CODE TO CX

163B  B0 80                     MOV     AL,80H            ; SET FUNCTION OF REQUEST FONT WITH FULL BOX
163D  B6 00                     MOV     DH,0              ; INDICATES ANK MODE
163F  E8 1A63 R                 CALL    FONT              ; GET FONT PATTERN

1642  8B F3                     MOV     SI,BX             ; SET TOP ADDRESS OF FONT PATTERN

1644  59                        POP     CX                ; RESTORE NUMBER OF CHAR
1645  5B                        POP     BX                ; RESTORE COLOR ATTRIBUTE
                                ;--DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
1646  E8 15E5 R                 CALL    GRAPH_POSITION    ; FIND LOCATION IN REGEN BUFFER
1649  8B F8                     MOV     DI,AX             ; REGEN POINTER IN DI
                                ;--- DETERMINE REGION TO GET CODE POINTS FROM
164B  58                        POP     AX                ; RECOVER CODE POINT AND CRT MODE

164C  06                        PUSH    ES                ; SET FONT PATTERN SEGMENT
164D  1F                        POP     DS                ;    TO DS
                                        ASSUME  DS:STACK

164E  07                        POP     ES                ; RESTORE REGEN SEGMENT
                                        ASSUME  ES:VIDEO_RAM

164F  80 FC 04                  CMP     AH,4              ; 320X200 4 COLOR ?
1652  74 5B                     JE      GRWRT9            ; YES

1654  80 FC 05                  CMP     AH,5              ; 320X200 4 SHADE ?
1657  74 56                     JE      GRWRT9            ; YES

1659  80 FC 0A                  CMP     AH,0AH            ; 640X200 4 COLOR ?
165C  74 42                     JE      GRWRT7            ; YES

165E  80 FC 06                  CMP     AH,6              ; 640X200 2 SHADE ?
1661  75 2B                     JNE     GRWRT5            ; NO, MODE IS 8 OR 9
1663                    GRWRT1:
1663  57                        PUSH    DI                ; SAVE POINTERS
1664  56                        PUSH    SI                ;
                                ;--- 640X200 2 SHADE
1665  B6 04                     MOV     DH,8/2            ; NUMBER OF TIMES THROUGH LOOP
1667                    GRWRT2:
1667  AC                        LODSB                     ; GET WORD FROM CODE POINTS
1668  F6 C3 80                  TEST    BL,XOR_BIT        ; SHOULD WE USE THE FUNCTION
166B  75 04                     JNZ     GRWRT3            ;  TO PUT CHAR IN?

166D  AA                        STOSB                     ; STORE IN REGEN BUFFER
166E  AC                        LODSB                     ; GET NEXT ROW
166F  EB 0A                     JMP SHORT GRWRT4

1671                    GRWRT3:
1671  26: 32 05                  XOR     AL,ES:[DI]        ; EXCLUSIVE OR WITH CURRENT DATA
1674  AA                        STOSB                     ; STORE THE CODE POINT
1675  AC                        LODSB                     ; AGAIN FOR ODD FIELD
1676  26: 32 85 1FFF            XOR     AL,ES:[DI+2000H-1]
167B                    GRWRT4:
167B  26: 88 85 1FFF            MOV     ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
1680  83 C7 4F                  ADD     DI,79             ; MOVE TO NEXT ROW IN REGEN
1683  FE CE                     DEC     DH                ; DONE WITH LOOP ?
1685  75 E0                     JNZ     GRWRT2            ; NO

1687  5E                        POP     SI                ;
1688  5F                        POP     DI                ;

1689  47                        INC     DI                ;
168A  E2 D7                     LOOP    GRWRT1            ;

168C  EB 2E                     JMP     SHORT GRWRT11     ;--- END

168E                    GRWRT5:
168E  D1 E7                     SAL     DI,1              ;--- 160/320X200 16 COLOR
1690  D1 E7                     SAL     DI,1              ;
1692                    GRWRT6:                           ; OFFSET*4 SINCE 4 BYTES/CHAR
1692  56                        PUSH    SI                ;

1693  B6 04                     MOV     DH,8/(4/2)        ; SET LOOP COUNT ( 8 ROW / 4 SCAN / 2 DEC.)
1695  E8 14F3 R                 CALL    G_W_4             ; WRITE ONE CHAR

1698  83 C7 04                  ADD     DI,4              ;
169B  5E                        POP     SI                ;
```

```
169C  E2 F4                     LOOP    GRWRT6          ;

169E  EB 1C                     JMP     SHORT GRWRT11   ;--- END

16A0                    GRWRT7:                         ;--- 640X200 4 COLOR
16A0  D1 E7                     SAL     DI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR

16A2                    GRWRT8:
16A2  56                        PUSH    SI              ;

16A3  B6 04                     MOV     DH,8/(4/2)      ; SET LOOP COUNT ( 8 ROW / 4 SCAN / 2 DEC.)
16A5  E8 1498 R                 CALL    G_W_1           ; WRITE ONE CHAR

16A8  47                        INC     DI              ;
16A9  47                        INC     DI              ;

16AA  5E                        POP     SI              ;
16AB  E2 F5                     LOOP    GRWRT8          ;

16AD  EB 0D                     JMP     SHORT GRWRT11   ;--- END

16AF                    GRWRT9:                         ;--- 320X200 4 COLOR/SHADE
16AF  D1 E7                     SAL     DI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR

16B1                    GRWRT10:
16B1  56                        PUSH    SI              ;

16B2  B6 04                     MOV     DH,8/2          ; SET LOOP COUNT ( 8 ROW / 2 SCAN )
16B4  E8 14C8 R                 CALL    G_W_2           ; WRITE ONE CHAR

16B7  47                        INC     DI              ;
16B8  47                        INC     DI              ;

16B9  5E                        POP     SI              ;
16BA  E2 F5                     LOOP    GRWRT10         ;

16BC                    GRWRT11:                        ;--- END
16BC  83 C4 12                  ADD     SP,GW_LOCAL     ; DEALLOCATE LOCAL WORK
16BF  5D                        POP     BP              ; RESTORE BP

16C0  C3                        RET

16C1                    GRAPHICS_WRITE  ENDP
                        ;-----------------------------------------------
                        ;
                        ;       SET COLOR               INT 10H, AH = 11 (0BH)
                        ;       ---------
                        ;       THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE
                        ;       OVERSCAN COLOR, AND THE FOREGROUND COLOR SET FOR GRAPHICS
                        ;
                        ;       INPUT
                        ;               AH =    CURRENT CRT MODE (MASKED)
                        ;               (BH) HAS COLOR ID
                        ;                       IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                        ;                               FROM THE LOW BITS OF BL (0-31)
                        ;                               IN GRAPHIC MODES, BOTH THE BACKGROUND AND
                        ;                               BORDER ARE SET. IN ALPHA MODES, ONLY THE
                        ;                               BORDER IS SET.
                        ;                       IF BH=1, THE PALETTE SELECTION IS MADE
                        ;                               BASED ON THE LOW BIT OF BL:
                        ;                                  2 COLOR MODE:
                        ;                                       0 = WHITE FOR COLOR 1
                        ;                                       1 = BLACK FOR COLOR 1
                        ;                                  4 COLOR MODES:
                        ;                                       0 = GREEN, RED, YELLOW FOR
                        ;                                           COLORS 1,2,3
                        ;                                       1 = BLUE, CYAN, MAGENTA FOR
                        ;                                           COLORS 1,2,3
                        ;                                  16 COLOR MODES:
                        ;                                       ALWAYS SETS UP PALETTE AS:
                        ;                                       BLUE          FOR COLOR 1
                        ;                                       GREEN         FOR COLOR 2
                        ;                                       CYAN          FOR COLOR 3
                        ;                                       RED           FOR COLOR 4
                        ;                                       MAGENTA       FOR COLOR 5
                        ;                                       YELLOW        FOR COLOR 6
                        ;                                       LIGHT GRAY    FOR COLOR 7
                        ;                                       DARK GRAY     FOR COLOR 8
                        ;                                       LIGHT BLUE    FOR COLOR 9
                        ;                                       LIGHT GREEN   FOR COLOR 10
                        ;                                       LIGHT CYAN    FOR COLOR 11
                        ;                                       LIGHT RED     FOR COLOR 12
                        ;                                       LIGHT MAGENTA FOR COLOR 13
                        ;                                       LIGHT YELLOW  FOR COLOR 14
                        ;                                       WHITE         FOR COLOR 15
                        ;               (BL) HAS THE COLOR VALUE TO BE USED
                        ;
                        ;       OUTPUT
                        ;               THE COLOR SELECTION IS UPDATED
                        ;
                        ;       CALL            ENABLE_VG12
                        ;                       ENABLE_VG2
                        ;                       SUPREG
                        ;
                        ;       VOLATILE        AX,BX,CX,DX
                        ;
                        ;-----------------------------------------------
                                ASSUME  CS:CODE, DS:DATA

16C1                    SET_COLOR       PROC    NEAR

16C1  E8 1DFC R                 CALL    ENABLE_VG12     ; ENABLE VIDEO GENERATER 1 AND 2

16C4  BA 03DA                   MOV     DX,VGA_CTL      ; I/O PORT FOR PALETTE
```

```
16C7                    SETC1:
16C7  EC                        IN      AL,DX           ; SYNC UP VGA FOR REG ADDRESS
16C8  A8 08                     TEST    AL,VERTRET      ; IS VERTICAL RETRACE ON?
16CA  74 FB                     JZ      SETC1           ; NO, WAIT UNTIL IT IS
16CC  B0 06                     MOV     AL,PCSUPER      ;
16CE  EE                        OUT     DX,AL           ; CLEAR SUPERIMPOSE COTROL REGISTER
16CF  32 C0                     XOR     AL,AL           ;     FOR SET PALETTE
16D1  EE                        OUT     DX,AL           ;

16D2  0A FF                     OR      BH,BH           ; IS THIS COLOR 0?
16D4  75 1D                     JNZ     SETC3           ; OUTPUT COLOR 1
                                                        ;--- HANDLE COLOR 0 BY SETTING THE
                                                        ;    BACKGROUND COLOR AND BORDER COLOR
16D6  80 FC 04                  CMP     AH,GRAPHICS     ; IN ALPHA MODE?
16D9  72 0D                     JC      SETC2           ; YES, JUST SET BORDER REG

16DB  B0 10                     MOV     AL,IXPALET      ; SET PALETTE REG 0
16DD  80 FC 0B                  CMP     AH,0BH          ; 640 X 200 X 16 COLOR ?
16E0  75 02                     JNE     SETC11          ; NO

16E2  34 0A                     XOR     AL,1010B        ;
16E4                    SETC11:
16E4  EE                        OUT     DX,AL           ; SELECT VGA REG
16E5  8A C3                     MOV     AL,BL           ; GET COLOR
16E7  EE                        OUT     DX,AL           ; SET IT
16E8                    SETC2:
16E8  B0 02                     MOV     AL,IXBORD       ; SET BORDER REG
16EA  EE                        OUT     DX,AL           ; SELECT VGA BORDER REG
16EB  8A C3                     MOV     AL,BL           ; GET COLOR
16ED  EE                        OUT     DX,AL           ; SET IT

16EE  A2 0066 R                 MOV     CRT_PALLETTE,AL ; SAVE THE COLOR VALUE

16F1  EB 4F                     JMP     SHORT   SETC10

16F3                    SETC3:          ;------ HANDLE COLOR 1 BY CHANGING PALETTE REGISTERS
16F3  B9 02B1 R                 MOV     CX,OFFSET PLTC20; POINT TO 2 COLOR TABLE ENTRY

16F6  80 FC 06                  CMP     AH,6            ; 2 COLOR MODE?
16F9  74 12                     JE      SETC5           ; YES, JUMP

16FB  80 FC 04                  CMP     AH,4            ; 4 COLOR MODE?
16FE  74 0A                     JE      SETC4           ; YES, JUMP
1700  80 FC 05                  CMP     AH,5            ; 4 COLOR MODE?
1703  74 05                     JE      SETC4           ; YES, JUMP

1705  80 FC 0A                  CMP     AH,0AH          ; 4 COLOR MODE?
1708  75 20                     JNE     SETC8           ; NO, GO TO 16 COLOR SET UP
170A                    SETC4:
170A  B9 02B9 R                 MOV     CX,OFFSET PLTC40; POINT TO 4 COLOR TABLE ENTRY
170D                    SETC5:
170D  D0 CB                     ROR     BL,1            ; SELECT ALTERNATE SET?
170F  73 03                     JNC     SETC6           ; NO, JUMP

1711  83 C1 04                  ADD     CX,PLTC20L      ; POINT TO NEXT ENTRY
1714                    SETC6:
1714  8B D9                     MOV     BX,CX           ; TABLE ADDRESS IN BX
1716  43                        INC     BX              ; SKIP OVER BACKGROUND COLOR

1717  B9 0003                  MOV     CX,PLTC20L-1    ; SET NUMBER OF REGS TO FILL
171A  B4 11                     MOV     AH,11H          ; AH IS REGISTER COUNTER
171C                    SETC7:
171C  8A C4                     MOV     AL,AH           ; GET REG NUMBER
171E  EE                        OUT     DX,AL           ; SELECT IT
171F  2E: 8A 07                 MOV     AL,CS:[BX]      ; GET DATA
1722  EE                        OUT     DX,AL           ; SET IT

1723  FE C4                     INC     AH              ; NEXT REG
1725  43                        INC     BX              ; NEXT TABLE VALUE
1726  E2 F4                     LOOP    SETC7

1728  EB 18                     JMP     SHORT SETC10

172A                    SETC8:
172A  8A FC                     MOV     BH,AH           ; SET MODE TO BH
172C  B4 11                     MOV     AH,11H          ; AH IS REGISTER COUNTER
172E  B9 000F                   MOV     CX,15           ; NUMBER OF PALETTES
1731                    SETC9:
1731  8A C4                     MOV     AL,AH           ; GET REG NUMBER
1733  80 FF 0B                  CMP     BH,0BH          ; 640 X 200 X 16 COLOR ?
1736  75 02                     JNE     SETC91          ; NO
1738  34 0A                     XOR     AL,1010B        ;
173A                    SETC91:
173A  EE                        OUT     DX,AL           ; SELECT IT
173B  8A C4                     MOV     AL,AH           ; GET REG NUMBER
173D  EE                        OUT     DX,AL           ; SET PALETTE VALUE

173E  FE C4                     INC     AH              ; NEXT REG
1740  E2 EF                     LOOP    SETC9

1742                    SETC10:
1742  8A 26 0347 R              MOV     AH,SUPIPCR      ;
1746  E8 0929 R                 CALL    SET_SUPREG      ; RESTORE SUPERIMPOSE CONTROL REGISTER

1749  E8 1DD6 R                 CALL    ENABLE_VG2      ; ENABLE VIDEO GENERATER 2
174C  C3                        RET

174D                    SET_COLOR       ENDP
                        ;-----------------------------------------
                        ;
                        ;       WRITE DOT               INT 10H, AH = 12 (0CH)
                        ;       ---------
                        ;       THESE ROUTINES WILL WRITE THE DOT AT THE INDICATED LOCATION
```

```
;
;       INPUT
;               AH = CURRENT CRT MODE (MASKED)
;               DX = ROW (0-199)    (THE ACTUAL VALUE DEPENDS ON THE MODE)
;               CX = COLUMN (0-639) ( THE VALUES ARE NOT RANGE CHECKED )
;               AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
;                    REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
;                    BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
;               DS = DATA SEGMENT
;               ES = REGEN SEGMENT
;
;       OUTPUT          NONE
;
;       CALL            DOT_POSITION
;                       GET_DIRSEG
;
;       VOLATILE        BX,CX,DX,SI
;
;--------------------------------------------

        ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

;
;       REGISTER USAGE          AH      DOT VALUE (TEMPORALY SAVED)
;                               AL      ACCUMULATER
;
;                               BH      CRT MODE
;                               BL      DOT VALUE (NOT CHANGED)
;
;                               CH      MOVED DATA FROM APA
;                               CL      SHIFT COUNT
;
;                               DH      MASK
;                               DL      MASK(COMPLIMENT)
;
174D                    WRITE_DOT       PROC    NEAR

174D  8B D8                     MOV     BX,AX           ; SAVE CRT MODE AND DOT VALUE TO BX

174F  80 FF 0B                  CMP     BH,0BH          ; 640 X 200 X 16 COLOR ?
1752  75 04                     JNE     WDOT1           ; NO

1754  D0 E8                     SHR     AL,1            ; SHIFT PAL ADDR 2-3 TO LSB FOR WRITE IN V-RAM2
1756  D0 E8                     SHR     AL,1            ;
1758                    WDOT1:
1758  E8 17E7 R                 CALL    DOT_POSITION    ; DETERMINE BYTE POSITION OF THE BIT

175B  8A F4                     MOV     DH,AH           ; SET MASK TO DH
175D  F6 D4                     NOT     AH
175F  8A D4                     MOV     DL,AH           ; SET COMPLIMENT OF MASK TO DL

1761  D2 E8                     SHR     AL,CL           ; SHIFT TO ADJUST IN THE DOT POSITION

1763  22 C6                     AND     AL,DH           ; STRIP OFF THE OTHER BITS
1765  26: 8A 2C                 MOV     CH,ES:[SI]      ; GET THE BYTE

1768  F6 C3 80                  TEST    BL,XOR_BIT      ; XOR THE DOT ?
176B  75 6A                     JNZ     WDOT7           ; YES

176D  22 EA                     AND     CH,DL           ; REMOVE THE INDICATED BITS
176F  0A C5                     OR      AL,CH           ; MERGE NEW BIT TO APA
1771                    WDOT2:                          ;                 (ALL POINT ADDRESSABLE BUFFER)
1771  26: 88 04                 MOV     ES:[SI],AL      ; SET IT TO APA

1774  80 FF 0A                  CMP     BH,0AH          ; 640 X 200 X 4/16 COLOR ?
1777  72 5D                     JB      WDOT6           ; NO, END

1779  8A C3                     MOV     AL,BL           ; GET THE DOT VALUE
177B  74 04                     JE      WDOT21          ; 4 COLOR

177D  D0 E8                     SHR     AL,1            ; SHIFT PAL ADDR 2-3 TO LSB FOR WRITE IN V-RAM2
177F  D0 E8                     SHR     AL,1            ;
1781                    WDOT21:
1781  D0 E8                     SHR     AL,1            ; SHIFT 1 BIT TO FIT IN ODD BYTE
1783  D0 C8                     ROR     AL,1            ; LEFT JUSTIFY THE VALUE
1785  D2 E8                     SHR     AL,CL           ; SHIFT TO ADJUST IN THE DOT POSITION

1787  22 C6                     AND     AL,DH           ; STRIP OFF THE OTHER BITS
1789  26: 8A 6C 01              MOV     CH,ES:[SI+1]    ; GET THE BYTE
178D  F6 C3 80                  TEST    BL,XOR_BIT      ; XOR THE DOT ?
1790  75 49                     JNZ     WDOT8           ; YES

1792  22 EA                     AND     CH,DL           ; REMOVE THE INDICATED BITS
1794  0A C5                     OR      AL,CH           ; MERGE NEW BITS TO APA
1796                    WDOT3:
1796  26: 88 44 01              MOV     ES:[SI+1],AL    ; SET IT TO APA

179A  80 FF 0B                  CMP     BH,0BH          ; 640 X 200 X 16 COLOR ?
179D  75 37                     JNE     WDOT6           ; NO

179F  E8 1D95 R                 CALL    GET_DIRSEG      ; GET SEGMENT OF V-RAM 1
17A2  8E C0                     MOV     ES,AX           ; SET IT TO ES

17A4  8A C3                     MOV     AL,BL           ; GET DOT VALUE
17A6  D0 C8                     ROR     AL,1            ; LEFT JUSTIFY THE VALUE
17A8  D2 E8                     SHR     AL,CL           ; SHIFT TO ADJUST IN THE DOT POSITION

17AA  22 C6                     AND     AL,DH           ; STRIP OFF THE OTHER BITS
17AC  26: 8A 2C                 MOV     CH,ES:[SI]      ; GET THE BYTE

17AF  F6 C3 80                  TEST    BL,XOR_BIT      ; XOR THE DOT ?
17B2  75 2B                     JNZ     WDOT9           ; YES

17B4  22 EA                     AND     CH,DL           ; REMOVE OTHER BITS
```

```
17B6  0A C5                     OR      AL,CH           ; MERGE NEW DOT TO APA
17B8                    WDOT4:
17B8  26: 88 04                 MOV     ES:[SI],AL      ; SET IT TO APA

17BB  8A C3                     MOV     AL,BL
17BD  D0 E8                     SHR     AL,1            ; SHIFT NEXT BIT TO FIT IN ODD BYTE
17BF  D0 C8                     ROR     AL,1
17C1  D2 E8                     SHR     AL,CL           ; SHIFT TO SDJUST IN THE DOT POSITION

17C3  22 C6                     AND     AL,DH           ; MASK OTHER BITS OFF
17C5  26: 8A 6C 01              MOV     CH,ES:[SI+1]    ; GET THE BYTE

17C9  F6 C3 80                  TEST    BL,XOR_BIT      ; XOR THE DOT ?
17CC  75 15                     JNZ     WDOT10          ; NO

17CE  22 EA                     AND     CH,DL           ; REMOVE THE OTHER BITS
17D0  0A C5                     OR      AL,CH           ; MERGE NEW BITS TO APA
17D2                    WDOT5:
17D2  26: 88 44 01              MOV     ES:[SI+1],AL    ; SET IT TO APA
17D6                    WDOT6:

17D6  C3                        RET

17D7                    WDOT7:
17D7  32 C5                     XOR     AL,CH
17D9  EB 96                     JMP     SHORT   WDOT2

17DB                    WDOT8:
17DB  32 C5                     XOR     AL,CH
17DD  EB B7                     JMP     SHORT   WDOT3

17DF                    WDOT9:
17DF  32 C5                     XOR     AL,CH
17E1  EB D5                     JMP     SHORT   WDOT4
17E3                    WDOT10:
17E3  32 C5                     XOR     AL,CH
17E5  EB EB                     JMP     SHORT   WDOT5

17E7                    WRITE_DOT       ENDP
```

```
;-----------------------------------------------
;
;       DOT_POSITION
;
;       THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
;        INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
;
;       INPUT
;               AH = CURRENT CRT MODE (MASKED)
;               AL = DOT VALUDE
;               DX = ROW VALUE (0-199)
;               CX = COLUMN VALUE (0-639)
;       OUTPUT
;               SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
;               AH = MASK TO STRIP OFF THE BITS OF INTEREST
;               AL = DOT VALUE (LEFT JUSTIFYED)
;               CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
;               DH = # BITS IN RESULT
;
;       CALL            NONE
;
;       VOLATILE        .CH,DL
;.
;-----------------------------------------------
```

```
17E7    DOT_POSITION    PROC    NEAR

17E7  53                PUSH    BX              ; SAVE BX DURING OPERATION
17E8  50                PUSH    AX              ; WILL SAVE AL DURING OPERATION

                        ;---DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE
                        ;   BY 40( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
17E9  B0 28             MOV     AL,40
17EB  52                PUSH    DX              ; SAVE ROW VALUE
17EC  80 E2 FE          AND     DL,0FEH         ; STRIP OFF ODD/EVEN BIT

17EF  8A FC             MOV     BH,AH           ; SAVE CRT MODE TO BH
17F1  80 FC 09          CMP     AH,09H          ; MODE USING 32K REGEN?
17F4  72 03             JC      DPOS1           ; NO, JUMP

17F6  80 E2 FC          AND     DL,0FCH         ; STRIP OFF LOW 2 BITS
17F9                    DPOS1:
17F9  F6 E2             MUL     DL              ; AX HAS ADDRESS OF 1ST BYTE OF INDICATED ROW

17FB  5A                POP     DX
17FC  F6 C2 01          TEST    DL,1            ; RECOVER IT
17FF  74 03             JZ      DPOS2           ; TEST FOR EVEN/ODD
                                                ; JUMP IF EVEN ROW
1801  05 2000           ADD     AX,2000H        ; OFFSET TO LOCATION OF ODD ROWS
1804                    DPOS2:                  ; EVEN_ROW
1804  80 FF 09          CMP     BH,09H          ; MODE USING 32K REGEN?
1807  72 08             JC      DPOS3           ; NO, JUMP
1809  F6 C2 02          TEST    DL,2            ; TEST FOR ROW 2 OR ROW 3
180C  74 03             JZ      DPOS3           ; JUMP IF ROW 0 OR 1

180E  05 4000           ADD     AX,4000H        ; OFFSET TO LOCATION OF ROW 2 OR 3
1811                    DPOS3:
1811  8B F0             MOV     SI,AX           ; MOVE POINTER TO SI

1813  58                POP     AX              ; RECOVER AL VALUE AND CRT MODE
1814  8B D1             MOV     DX,CX           ; COLUMN VALUE TO DX
```

```
                        ;--DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
                        ;  SET UP THE REGISTERS ACCORDING TO THE MODE
                        ;  CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3/1 FOR HIGH/MED/LOW RES)
                        ;  CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2/1 FOR H/M/L)
                        ;  BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/C0H/F0H FOR H/M/L)
                        ;  BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2/4 FOR H/M/L)

1816  BB 02C0           MOV     BX,2C0H
1819  B9 0302           MOV     CX,302H         ; SET PARMS FOR MED RES

181C  80 FC 04          CMP     AH,4
181F  74 1B             JE      DPOS5           ; HANDLE IF MED RES
1821  80 FC 05          CMP     AH,5
1824  74 16             JE      DPOS5           ; HANDLE IF MED RES

1826  BB 04F0           MOV     BX,4F0H
1829  B9 0101           MOV     CX,101H         ;SET PARMS FOR LOW RES

182C  80 FC 0A          CMP     AH,0AH
182F  73 05             JAE     DPOS4           ; HANDLE MODE A,B AS HIGH RES

1831  80 FC 06          CMP     AH,6
1834  75 06             JNE     DPOS5           ; HANDLE IF LOW RES
1836                  DPOS4:
1836  BB 0180           MOV     BX,180H
1839  B9 0703           MOV     CX,703H         ; SET PARMS FOR HIGH RES
183C                  DPOS5:                  ;--- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
183C  22 EA             AND     CH,DL           ; ADDRESS OF PEL WITHIN BYTE TO CH
                                                ;--- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
183E  D3 EA             SHR     DX,CL           ; SHIFT BY CORRECT AMOUNT
1840  03 F2             ADD     SI,DX           ; INCREMENT THE POINTER

1842  80 FC 0A          CMP     AH,0AH          ; 640X200 4/16 COLOR?
1845  72 02             JB      DPOS6           ; NO, JUMP

1847  03 F2             ADD     SI,DX           ; INCREMENT THE POINTER
1849                  DPOS6:
1849  8A F7             MOV     DH,BH           ; GET THE # OF BITS IN RESULT TO DH
                                                ;--- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
184B  2A C9             SUB     CL,CL           ; ZERO INTO STORAGE LOCATION
184D                  DPOS7:
184D  D0 C8             ROR     AL,1            ; LEFT JUSTIFY THE VALUE IN AL
                                                ; (FOR WRITE)
184F  02 CD             ADD     CL,CH           ; ADD IN THE BIT OFFSET VALUE
1851  FE CF             DEC     BH              ; LOOP CONTROL
1853  75 F8             JNZ     DPOS7           ; ON EXIT, CL HAS SHIFT COUNT TO RESTORE BITS

1855  8A E3             MOV     AH,BL           ;  GET MASK TO AH
1857  D2 EC             SHR     AH,CL           ;  MOVE THE MASK TO CORRECT LOCATION

1859  5B               POP     BX              ;  RECOVER REG
185A  C3               RET                     ;  RETURN WITH EVERYTHING SET UP

185B                  DOT_POSITION    ENDP
                      ;-----------------------------------------------
                      ;
                      ;       READ DOT                INT 10H, AH = 13 (0DH)
                      ;       --------
                      ;       THESE ROUTINES WILL READ THE DOT AT THE INDICATED LOCATION
                      ;
                      ;       INPUT
                      ;               AH = CURRENT CRT MODE (MASKED)
                      ;               DX = ROW (0-199)    (THE ACTUAL VALUE DEPENDS ON THE MODE)
                      ;               CX = COLUMN (0-639) ( THE VALUES ARE NOT RANGE CHECKED )
                      ;               DS = DATA SEGMENT
                      ;               ES = REGEN SEGMENT
                      ;
                      ;       OUTPUT
                      ;               AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
                      ;
                      ;       CALL    DOT_POSITION
                      ;               GET_DIRSEG
                      ;
                      ;       VOLATILE        AH,BX,CX,DX,SI,ES
                      ;
                      ;-----------------------------------------------
                              ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

                      ;
                      ;       REGISTER USAGE          AH      MASK
                      ;                               AL      ACCUMULATER
                      ;
                      ;                               BH      CRT MODE
                      ;                               BL      TEMPORARY SAVE DATA OF DOT VALUE
                      ;                               CH      SHIFT COUNT (SAVED)
                      ;                               CL      SHIFT COUNT
                      ;
                      ;                               DH      # OF BITS IN ONE PIXEL

185B                  READ_DOT        PROC    NEAR

185B  8B D8             MOV     BX,AX           ; SAVE CRT MODE                 TO BH

185D  E8 17E7 R         CALL    DOT_POSITION    ; DETERMINE BYTE POSITION OF THE BIT
1860  8A E9             MOV     CH,CL           ; SAVE SHIFT COUNT TO CH

1862  26: 8A 04         MOV     AL,ES:[SI]      ; GET THE BYTE
1865  22 C4             AND     AL,AH           ; STRIP THE OTHER BITS OFF

1867  D2 E0             SHL     AL,CL           ; SHIFT TO LEFT JUSTIFY
1869  8A CE             MOV     CL,DH           ;
186B  D2 C0             ROL     AL,CL           ; RIGHT JUSTIFY THE RESULT

186D  80 FF 0A          CMP     BH,0AH          ; 640 X 200 X 4/16 COLOR ?
1870  72 3D             JB      RDOT1           ; NO, END
```

```
1872  8A D8                         MOV     BL,AL           ; SAVE EVEN VALUE TO BL

1874  26: 8A 44 01                  MOV     AL,ES:[SI+1]    ; GET THE ODD BYTE
1878  22 C4                         AND     AL,AH           ; STRIP THE OTHER BITS OFF
187A  8A CD                         MOV     CL,CH           ; RESTORE SHIFT COUNT
187C  D2 E0                         SHL     AL,CL           ; LEFT JUSTIFY THE VALUE
187E  D0 C0                         ROL     AL,1            ; RIGHT JUSTIFY THE VALUE
1880  D0 C0                         ROL     AL,1            ; MOVE TO PALETTE ADDRESS 1 POSITION
1882  0A C3                         OR      AL,BL           ; COMBINE EVEN/ODD BIT

1884  80 FF 0B                      CMP     BH,0BH          ; 640 X 200 X 16 COLOR ?
1887  75 26                         JNE     RDOT1           ; NO, END

1889  D0 E0                         SHL     AL,1            ; SHIFT LOW 2 BIT TO PALETTE ADDR 2-3 POSITION
188B  D0 E0                         SHL     AL,1            ;
188D  8A D8                         MOV     BL,AL           ; SAVE EVEN VALUE TO BL

188F  50                            PUSH    AX              ; SAVE AX
1890  E8 1D95 R                     CALL    GET_DIRSEG      ; GET DIRECT SEGMENT OF V-RAM 1
1893  8E C0                         MOV     ES,AX           ; SET IT TO ES
1895  58                            POP     AX              ; RESTORE AX

1896  26: 8A 04                     MOV     AL,ES:[SI]      ; GET THE BYTE
1899  22 C4                         AND     AL,AH           ; STRIP THE OTHER BITS OFF
189B  D2 E0                         SHL     AL,CL           ; LEFT JUSTIFY THE VALUE
189D  D0 C0                         ROL     AL,1            ; RIGHT JUSTIFY THE VALUE
189F  0A D8                         OR      BL,AL           ; COMEBINE EVEN BIT OF V-RAM 1 TO PAL ADDR 0

18A1  26: 8A 44 01                  MOV     AL,ES:[SI+1]    ; GET THE BYTE
18A5  22 C4                         AND     AL,AH           ; STRIP THE OTHER BITS OFF
18A7  D2 E0                         SHL     AL,CL           ; LEFT JUSTIFY THE VALUE
18A9  D0 C0                         ROL     AL,1            ; RIGHT JUSTIFY THE VALUE
18AB  D0 C0                         ROL     AL,1            ; MOVE TO PALETTE ADDRESS 0 POSITION
18AD  0A C3                         OR      AL,BL           ; COMBINE ALL 4 BITS
18AF                        RDOT1:
18AF  C3                            RET

18B0                        READ_DOT        ENDP

                            ;-----------------------------------------------
                            ;
                            ;       WRITE_TTY               INT 10H, AH = 14 (0EH)
                            ;       ---------
                            ;       THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
                            ;        VIDEO CARD.  THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
                            ;        CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
                            ;        IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
                            ;        IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED.  IF THE ROW
                            ;        VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST
                            ;        ROW, FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE
                            ;        LINE. WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING
                            ;        THE NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE
                            ;        PREVIOUS LINE BEFORE THE SCROLL, IN CHARACTER MODE.  IN
                            ;        GRAPHICS MODE, THE 0 COLOR IS USED.
                            ;
                            ;       INPUT
                            ;               AH = CURRENT CRT MODE   (MASKED)
                            ;               AL = CHARACTER TO BE WRITTEN
                            ;                  NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE
                            ;                  HANDLED AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
                            ;               BL = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A
                            ;                    GRAPHICS MODE
                            ;
                            ;       OUTPUT          NONE
                            ;       CALL            BEEP
                            ;                       VIDEO
                            ;
                            ;       VOLATILE        BH, CX, DX
                            ;
                            ;-----------------------------------------------
                                    ASSUME  CS:CODE, DS:DATA
18B0                        WRITE_TTY       PROC    NEAR

18B0  50                            PUSH    AX              ; SAVE REGISTERS
18B1  50                            PUSH    AX              ; SAVE CHAR TO WRITE

18B2  8A 3E 0062 R                  MOV     BH,ACTIVE_PAGE  ; GET CURRENT PAGE SETTING
18B6  53                            PUSH    BX              ; SAVE IT
18B7  8A DF                         MOV     BL,BH           ; IN BL
18B9  32 FF                         XOR     BH,BH
18BB  D1 E3                         SAL     BX,1            ; CONVERT TO WORD OFFSET
18BD  8B 97 035C R                  MOV     DX,[BX+OFFSET CURSOR_POSN] ; GET CURSOR POSITION
18C1  5B                            POP     BX              ; RECOVER CURRENT PAGE

18C2  58                            POP     AX              ; RECOVER CHAR

18C3  8B 0E 0342 R                  MOV     CX,TTY_1ST_CHAR ;--- DX NOW HAS THE CURRENT CURSOR POSITION
18C7  C7 06 0342 R 0000             MOV     TTY_1ST_CHAR,0  ; SAVE 1ST BYTE OF 2 BYTE CODE
                                                            ; CLEAR IT FOR SPECIAL CODE
18CD  3C 08                         CMP     AL,BS
18CF  75 03                         JNE     WTY1            ; IS IT A BACKSPACE?
                                                            ; NO
18D1  E9 19C2 R                     JMP     WTY19           ; BACK_SPACE

18D4                        WTY1:
18D4  3C 0D                         CMP     AL,CR
18D6  75 05                         JNE     WTY2            ; IS IT A CARRIAGE RETURN?
                                                            ; NO

18D8  32 D2                         XOR     DL,DL           ;--- CARRIAGE RETURN
18DA  E9 19BE R                     JMP     WTY18           ; MOVE TO FIRST COLUMN
                                                            ; SET_CURSOR
```

```
18DD                            WTY2:
18DD  3C 0A                             CMP     AL,LF           ; IS IT A LINE FEED
18DF  75 03                             JNE     WTY3            ; NO

18E1  E9 1982 R                         JMP     WTY11           ; LINE_FEED

18E4                            WTY3:
18E4  3C 07                             CMP     AL,BELL         ; IS IT A BELL
18E6  75 08                             JNE     WTY4            ; NO

                                                                ;--- BELL
18E8  B3 02                             MOV     BL,2            ; SET UP COUNT FOR BEEP
18EA  E8 0000 E                         CALL    BEEP            ; SOUND THE POD BELL
18ED  E9 19BA R                         JMP     WTY16           ; TTY_RETURN

18F0                            WTY4:
18F0  80 3E 0049 R 10                   CMP     CRT_MODE,KJ_MODE; KANJI MODE ?
18F5  72 7A                             JB      WTYI0           ; NO, SKIP 2 BYTE CODE HANDLING

18F7  89 0E 0342 R                      MOV     TTY_1ST_CHAR,CX ; RESTORE 1ST BYTE OF 2 BYTE CODE
18FB  0B C9                             OR      CX,CX           ; 1ST BYTE OF 2 BYTE CODE HAS BEEN SET ?
18FD  75 18                             JNZ     WTY6            ; YES

18FF  3C 80                             CMP     AL,080H         ; 1ST BYTE OF 2 BYTE CODE ?
1901  76 6E                             JBE     WTY10           ; NO, GO TO WRITE ONE CHARACTER
1903  3C FD                             CMP     AL,0FDH         ;
1905  73 6A                             JAE     WTY10           ; NO, GO TO WRITE ONE CHARACTER
1907  3C A0                             CMP     AL,0A0H         ;
1909  72 04                             JB      WTY5            ; YES
190B  3C DF                             CMP     AL,0DFH         ;
190D  76 62                             JBE     WTY10           ; NO, GO TO WRITE ONE CHARACTER

190F                            WTY5:
                                                                ;--- CHARACTER IS 1ST BYTE OF 2 BYTE CODE
190F  8A E3                             MOV     AH,BL           ; SET COLOR
1911  A3 0342 R                         MOV     TTY_1ST_CHAR,AX ; SAVE 1ST BYTE OF 2 BYTE CODE AND SET FLAG
1914  E9 19BA R                         JMP     WTYI6           ; RETURN; WRITE PROCESS IS PENDING

1917                            WTY6:
                                                                ;--- CHARACTER MUST BE 2ND BYTE OF 2 BYTE CODE
1917  3C 40                             CMP     AL,040H         ; 2ND BYTE OF 2 BYTE CODE ?
1919  72 08                             JB      WTY7            ; NO
191B  3C FC                             CMP     AL,0FCH         ;
191D  77 04                             JA      WTY7            ; NO
191F  3C 7F                             CMP     AL,07FH         ;
1921  75 08                             JNE     WTY8            ; YES

1923                            WTY7:
                                                                ;--- IGNORE 1ST BYTE
1923  C7 06 0342 R 0000                 MOV     TTY_1ST_CHAR,0  ; RESET FLAG OF 1ST BYTE
1929  EB 46                             JMP     SHORT WTY10     ; GO TO WRITE ONE BYTE

192B                            WTY8:
                                                                ;--- WRITE 2 BYTE CODE
192B  8B 0E 004A R                      MOV     CX,CRT_COLS     ; CHECK COLUMN BOUNDARY
192F  49                                DEC     CX              ; ADJUST FOR COMPARE
1930  3A D1                             CMP     DL,CL           ; IS COLUMN AT END OF LINE ?
1932  72 21                             JB      WTY9            ; NO

1934  50                                PUSH    AX              ; SAVE 2ND BYTE OF 2 BYTE CODE
1935  53                                PUSH    BX              ; SAVE FOREGROUND COLOR

1936  FF 36 0342 R                      PUSH    TTY_1ST_CHAR    ; SAVE 1ST BYTE OF 2 BYTE CODE
193A  C7 06 0342 R 0000                 MOV     TTY_1ST_CHAR,0  ; CLEAR FOR RECURSIVE CALL

1940  B8 0E20                          MOV     AX,0E00H + ' '  ; (AH)=0EH: WRITE SPACE
1943  CD 10                             INT     VIDEO           ; VIDEO I/O

1945  8A DF                             MOV     BL,BH           ; PAGE NUMBER
1947  32 FF                             XOR     BH,BH           ;
1949  D1 E3                             SAL     BX,1            ; CONVERT TO WORD OFFSET
194B  8B 97 035C R                      MOV     DX,[BX+OFFSET CURSOR_POSN]      ; GET CURSOR POSITION

194F  8F 06 0342 R                      POP     TTY_1ST_CHAR    ; RESTORE 2ND BYTE OF 2 BYTE CODE

1953  5B                               POP     BX              ; RESTORE FOREGROUND COLOR
1954  58                                POP     AX              ; RESTORE 2ND BYTE
1955                            WTY9:
                                                                ;--- WRITE 1ST BYTE OF 2 BYTE CODE
1955  50                                PUSH    AX              ; SAVE 2ND BYTE OF 2 BYTE CODE
1956  53                                PUSH    BX              ; SAVE COLOR
1957  A1 0342 R                         MOV     AX,TTY_1ST_CHAR ; GET 1ST BYTE OF 2 BYTE CODE
195A  8A DC                             MOV     BL,AH           ; SET COLOR OF 1ST BYTE
195C  B4 0A                             MOV     AH,0AH          ; (AH)=0AH: WRITE CHARACTER AT CURRENT CURSOR
195E  B9 0001                           MOV     CX,1            ; WRITE ONE CHARACTER
1961  CD 10                             INT     VIDEO           ; VIDEO I/O
1963  5B                                POP     BX              ; RESTORE COLOR OF 2ND BYTE

1964  C7 06 0342 R 0000                 MOV     TTY_1ST_CHAR,0  ; RESET FLAG OF 1ST BYTE
196A  FE C2                             INC     DL              ; INCREMENT ROW

196C  B4 02                             MOV     AH,2            ; FUNCTION OF SET CURSOR POSITION
196E  CD 10                             INT     VIDEO           ; DO IT
1970  58                                POP     AX              ; RESTORE 2ND BYTE
1971                            WTY10:
1971  B4 0A                             MOV     AH,10           ; WRITE CHAR ONLY
1973  B9 0001                           MOV     CX,1            ; ONLY ONE CHAR
1976  CD 10                             INT     VIDEO           ; WRITE THE CHAR
                                                                ;--- POSITION THE CURSOR FOR NEXT CHAR
1978  FE C2                             INC     DL
197A  3A 16 004A R                      CMP     DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
197E  72 3E                             JB      WTY18           ; SET_CURSOR

1980  32 D2                             XOR     DL,DL           ; COLUMN FOR CURSOR
1982                            WTY11:
                                                                ;--- LINE FEED
1982  3A 36 0346 R                      CMP     DH,CRT_ROWS     ; BOTTOM OF SCREEN ?
1986  72 34                             JB      WTY17           ; NO, SET_CURSOR_INC
                                                                ;--- SCROLL REQUIRED
```

```
1988  B4 02                   MOV     AH,2
198A  CD 10                   INT     VIDEO           ; SET THE CURSOR
                                                      ;--- DETERMINE VALUE TO FILL WITH DURING SCROLL
198C  A0 0049 R               MOV     AL,CRT_MODE     ; GET THE CURRENT MODE
198F  24 0F                   AND     AL,KJ_OFF       ; MASK KJ BIT OFF

1991  3C 04                   CMP     AL,GRAPHICS     ; IN ALPHA MODE ?
1993  72 04                   JC      WTY12           ; YES, READ ATTRIBUTE

1995  32 FF                   XOR     BH,BH           ; FILL WITH BACKGROUND
1997  EB 10                   JMP     SHORT WTY14     ; SCROLL-UP

1999                    WTY12:                        ;--- READ ATTRIBUTE
1999  B4 08                   MOV     AH,8
199B  CD 10                   INT     VIDEO           ; READ CHAR/ATTR AT CURRENT CURSOR

199D  80 3E 0049 R 10         CMP     CRT_MODE,KJ_MODE; KJ MODE ?
19A2  72 03                   JB      WTY13           ; NO

19A4  80 E4 77                AND     AH,HAN_MASK     ; MASK KJBIT OFF
19A7                    WTY13:
19A7  8A FC                   MOV     BH,AH           ; STORE IN BH
19A9                    WTY14:
19A9  B8 0601                 MOV     AX,601H         ; SCROLL ONE LINE
19AC  2B C9                   SUB     CX,CX           ; UPPER LEFT CORNER
19AE  8A 36 0046 R            MOV     DH,CRT_ROWS     ; LOWER RIGHT ROW
19B2  8A 16 004A R            MOV     DL,BYTE PTR CRT_COLS ; LOWER RIGHT COLUMN
19B6  FE CA                   DEC     DL
19B8                    WTY15:
19B8  CD 10                   INT     VIDEO           ; SCROLL UP THE SCREEN
19BA                    WTY16:
19BA  58                      POP     AX              ; RESTORE THE CHARACTER

19BB  C3                      RET                     ; RETURN TO CALLER

                        ;---------------------------------- SET NEW CORSOR POSITION
19BC                    WTY17:
19BC  FE C6                   INC     DH              ; NEXT ROW
19BE                    WTY18:
19BE  B4 02                   MOV     AH,2
19C0  EB F6                   JMP     WTY15           ; ESTABLISH THE NEW CURSOR

                        ;---------------------------------- BACK SPACE
19C2                    WTY19:
19C2  0A D2                   OR      DL,DL           ; ALREADY AT END OF LINE
19C4  74 F8                   JE      WTY18           ; SET_CURSOR
19C6  FE CA                   DEC     DL              ; NO -- JUST MOVE IT BACK
19C8  EB F4                   JMP     WTY18           ; SET_CURSOR

19CA                    WRITE_TTY       ENDP
                        ;-----------------------------------------------
                        ;
                        ;       VIDEO STATE             INT 10H, (AH) = 15 (0FH)
                        ;       -----------
                        ;       RETURNS THE CURRENT VIDEO STATE IN AX
                        ;
                        ;       INPUT   NONE
                        ;       OUTPUT
                        ;               AH = NUMBER OF COLUMNS ON THE SCREEN
                        ;               AL = CURRENT VIDEO MODE
                        ;               BH = CURRENT ACTIVE DISPLAY PAGE
                        ;
                        ;       CALL            NONE
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------------------
                                ASSUME  CS:CODE, DS:DATA
19CA
                        VIDEO_STATE     PROC    NEAR
19CA  55                        PUSH    BP              ; SAVE BP

19CB  8A 26 004A R              MOV     AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
19CF  A0 0049 R                 MOV     AL,CRT_MODE     ; CURRENT MODE
19D2  8A 3E 0062 R              MOV     BH,ACTIVE_PAGE  ; GET CURRENT ACTIVE PAGE

19D6  8B EC                     MOV     BP,SP
19D8  89 5E 08                  MOV     [BP+F_BX],BX    ; SET FRAME POINTER
                                                        ; SET RETURN BX
19DB  5D                        POP     BP
19DC  C3                        RET                     ; RESTORE BP
                                                        ; RETURN TO CALLER
19DD                    VIDEO_STATE     ENDP
                        ;-----------------------------------------------------
                        ;
                        ;       SET_PALLETTE                    INT 10H, AH = 16 (10H)
                        ;       ------------
                        ;       THIS ROUTINE WRITES THE PALETTE REGISTERS
                        ;
                        ;       INPUT   AH =    CRT MODE (MASKED)
                        ;               AL = 0  SET PALETTE REGISTERS
                        ;                       (BH) =  VALUE TO SET
                        ;                       (BL) =  PALETTE REGISTER TO SET (00H-0FH)
                        ;               AL = 1  SET BORDER COLOR REGISTER
                        ;                       (BH) =  VALUE TO SET
                        ;               AL = 2  SET ALL PALETTE REGS AND BORDER REG
                        ;                       ES:DX   POINTS TO A 17 BYTE LIST
                        ;                               BYTE 0 - 15 ARE VALUES FOR PALETTE
                        ;                               BYTE 16 IS THE VALUE FOR THE BORDER REG
                        ;       OUTPUT          NONE
```

```
                        ;       CALL            ENABLE_VG12
                        ;                       ENABLE_VG2
                        ;                       WAIT_VERTRET
                        ;                       SET_SUPREG
                        ;
                        ;       VOLATILE        AX,BX,DX,SI,ES
                        ;
                        ;       NOTE: PALETTE REGISTERS ARE WRITE ONLY.
                        ;-------------------------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

19DD                    SET_PALLETTE    PROC    NEAR

19DD  50                        PUSH    AX              ; SAVE AX
19DE  E8 1DFC R                 CALL    ENABLE_VG12     ; ENABLE BOTH VG1 AND VG2

19E1  8B F4                     MOV     SI,SP
19E3  36: 8E 44 10              MOV     ES,SS:[SI+F_ES] ; GET SEG FROM STACK
                                        ASSUME  ES:INDETERMINATE

19E7  8B F2                     MOV     SI,DX           ; OFFSET IN SI

19E9  E8 1A44 R                 CALL    WAIT_VERTRET    ; WAIT UNTIL VERICAL RETRACE

19EC  B0 06                     MOV     AL,PCSUPER      ;
19EE  EE                        OUT     DX,AL           ; CLEAR SUPERIMPOSE CONTROL REGISTER
19EF  32 C0                     XOR     AL,AL           ;       FOR SET PALETTE
19F1  EE                        OUT     DX,AL           ;

19F2  58                        POP     AX              ; RESTORE AX
19F3  0A C0                     OR      AL,AL           ; SET PALETTE REG?
19F5  74 0A                     JZ      SPL1            ; YES, GO DO IT

19F7  3C 02                     CMP     AL,2            ; SET ALL REGS?
19F9  74 1C                     JE      SPL3            ; YES, GO DO IT

                                                        ; SET BORDER COLOR REG?
19FB  77 3C                     JA      SPL5            ; NO, DON'T DO ANYTHING

                                                        ;--- SET BORDER COLOR REGISTER
19FD  B0 02                     MOV     AL,IXBORD       ; SET BORDER COLOR REG NUMBER
19FF  EB 0D                     JMP     SHORT SPL2

1A01                    SPL1:                           ;--- SET PALETTE REGISTER
1A01  8A C3                     MOV     AL,BL           ; GET DESIRED REG NUMBER IN AL
1A03  24 0F                     AND     AL,0FH          ; STRIP UNUSED BITS
1A05  0C 10                     OR      AL,IXPALET      ; MAKE INTO REAL REG NUMBER

1A07  80 FC 0B                  CMP     AH,0BH          ; 640 X 200 X 16 COLOR ?
1A0A  75 02                     JNE     SPL2            ; NO

1A0C  34 0A                     XOR     AL,1010B        ; MAP TO REAL REG NUMBER
1A0E                    SPL2:
1A0E  EE                        OUT     DX,AL           ; SELECT REG
1A0F  8A C7                     MOV     AL,BH           ; GET DATA IN AL
1A11  EE                        OUT     DX,AL           ; SET NEW DATA

1A12  32 C0                     XOR     AL,AL           ; SET REG 0 SO DISPLAY WORKS AGAIN
1A14  EE                        OUT     DX,AL
1A15  EB 22                     JMP     SHORT SPL5

1A17                    SPL3:                           ;--- SET ALL PALETTE REGS AND BORDER REG
1A17  B7 10                     MOV     BH,IXPALET      ; BH IS REG COUNTER
1A19                    SPL4:
1A19  8A C7                     MOV     AL,BH           ; REG ADDRESS IN AL
1A1B  80 FC 0B                  CMP     AH,0BH          ; 640 X 200 X 16 COLOR ?
1A1E  75 02                     JNE     SPL41           ; NO

1A20  34 0A                     XOR     AL,1010B        ; MAP TO REAL REG NUMBER
1A22                    SPL41:
1A22  EE                        OUT     DX,AL           ; SELECT IT

1A23  26: 8A 04                 MOV     AL,BYTE PTR ES:[SI]     ;GET DATA
1A26  EE                        OUT     DX,AL           ; PUT IN VGA REG

1A27  46                        INC     SI              ; NEXT DATA BYTE
1A28  FE C7                     INC     BH              ; NEXT REG
1A2A  80 FF 20                  CMP     BH,20H          ; LAST PALETTE REG?
1A2D  72 EA                     JB      SPL4            ; NO, DO NEXT ONE

1A2F  B0 02                     MOV     AL,IXBORD       ; SET BORDER REG
1A31  EE                        OUT     DX,AL           ; SELECT IT

1A32  26: 8A 04                 MOV     AL,BYTE PTR ES:[SI]     ; GET DATA
1A35  EE                        OUT     DX,AL           ; PUT IN VGA REG
1A36  A2 0066 R                 MOV     CRT_PALLETTE,AL ; SAVE IN RAM
1A39                    SPL5:
1A39  8A 26 0347 R              MOV     AH,SUPIPCR      ; RESTORE SUPERIMPOSE CONTROL REGISTER
1A3D  E8 0929 R                 CALL    SET_SUPREG      ;

1A40  E8 1DD6 R                 CALL    ENABLE_VG2      ; ENABLE VIDEO GENARATER 2
1A43  C3                        RET                     ; ALL DONE

1A44                    SET_PALLETTE    ENDP

                        ;-------------------------------------------------------
                        ;
                        ;       WAIT_VERTRET            WAIT VERTICAL RETRACE
                        ;
                        ;       THIS ROUTINE WAITS UNTIL A VERTICAL RETRACE BEGIN
                        ;
                        ;       INPUT           NONE
```

```
;       OUTPUT          NONE
;       VOLATILE        AL,DX
;
;--------------------------------------------------------

WAIT_VERTRET    PROC    NEAR

1A44                    MOV     DX,VGA_CTL      ; SET VGA CONTROL PORT
1A44  BA 03DA   WAITV1:
1A47                    IN      AL,DX           ; GET VGA STATUS
1A47  EC                AND     AL,VERTRET      ; IN VERTICAL RETRACE?
1A48  24 08             JNZ     WAITV1          ; YES, WAIT FOR IT TO GO AWAY
1A4A  75 FB
1A4C            WAITV2:
1A4C                    IN      AL,DX           ; GET VGA STATUS
1A4C  EC                AND     AL,VERTRET      ; IN VERITCAL RETRACE?
1A4D  24 08             JZ      WAITV2          ; NO, WAIT FOR IT
1A4F  74 FB
                        RET
1A51  C3
WAIT_VERTRET    ENDP
1A52    ;-----------------------------------------
        ;
        ;       NO_OPERATION            INT 10H, AH = 17,18
        ;       ------------
        ;-----------------------------------------
                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

1A52    NO_OPERATION    PROC    NEAR
                        RET
1A52  C3
1A53    NO_OPERATION    ENDP

        ;-----------------------------------------
        ;
        ;       FONT PATTERN            INT 10H, AH = 19
        ;       ------------
        ;       THIS ROUTINE ACCEPTS OR RETURNS FONT PATTERN FROM/IN USER AREA
        ;
        ;       INPUT
        ;               AL =0   REQUEST BASE-FONT
        ;               AL =80H REQUEST BASE-FONT WITH FULL CHARACTER BOX
        ;               AL =40H WRITE FONT PATTERN FROM USER AREA TO GAIJI RAM
        ;
        ;               CX    =  INTERNAL CODE FOR REQUESTED FONT
        ;                         FOR HANKAKU-FONT (CH)=0
        ;               BX    =  OFFSET OF DATA AREA FOR FONT
        ;                            NORMAL-BOX      FULL-BOX
        ;                        -16X16 ; 32 BYTE     36 BYTE
        ;                        -16X8  ; 16 BYTE     18 BYTE
        ;
        ;               DS =    DATA SEGMENT
        ;
        ;       OUTPUT          FONT PATTERN
        ;
        ;       CALL            CHECK_ROSS_CHAR
        ;                       FONT
        ;
        ;       VOLATILE        AX,CX,SI,DI,ES
        ;
        ;-----------------------------------------
                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

1A53    FONT_PATTERN    PROC    NEAR

1A53  55                PUSH    BP              ; SAVE BP
1A54  8B EC             MOV     BP,SP           ; SET SP TO BP
1A56  8E 46 10          MOV     ES,[BP+F_ES]    ; GET SEGMENT VALUE FROM STACK
                        ASSUME  ES:INDETERMINATE

1A59  E8 1B65 R         CALL    CHECK_ROSS_CHAR ; CHECK AND CONVERT ROSSIAN CHARACTER

1A5C  B6 10             MOV     DH,KJ_MODE      ; INDICATES KJ MODE
1A5E  E8 1A63 R         CALL    FONT            ; PERFORM FONT FUNCTION

1A61  5D                POP     BP              ; RESTORE BP
1A62  C3                RET                     ; RETURN TO CALLER

1A63    FONT_PATTERN    ENDP

        ;-----------------------------------------
        ;
        ;       FONT
        ;
        ;       THIS ROUTINE ACCEPTS OR RETURNS FONT PATTERN FROM/IN USER AREA
        ;
        ;       INPUT
        ;               AL =0   REQUEST BASE-FONT
        ;               AL =80H REQUEST BASE-FONT WITH FULL CHARACTER BOX
        ;               AL =40H WRITE FONT PATTERN FROM USER AREA TO APA
        ;
        ;               CX =    INTERNAL CODE FOR REQUESTED FONT
        ;                         FOR HANKAKU-FONT (CH)=0
        ;               DH =    CRT MODE
        ;               ES:BX  = DATA AREA FOR FONT
        ;                            NORMAL-BOX      FULL-BOX
        ;                        -16X16 ; 32 BYTE     36 BYTE
        ;                        -16X8  ; 16 BYTE     18 BYTE
        ;
        ;               DS =    DATA SEGMENT
```

```
                        ;       OUTPUT  (ES:BX) = FONT PATTERN
                        ;
                        ;       CALL            ENABLE_VG1
                        ;                       ENABLE_VG2
                        ;                       ENABLE_KJROM
                        ;                       ENABLE_VRAM
                        ;                       CVTCR
                        ;
                        ;       VOLATILE        AX,CX,SI,DI,ES
                        ;
                        ;-----------------------------------------
                                                ASSUME  CS:CODE ,DS:DATA
1A63                    FONT    PROC    NEAR

1A63  52                        PUSH    DX              ; SAVE CURRENT DX
1A64  1E                        PUSH    DS              ; SAVE CURRENT DS

1A65  50                        PUSH    AX              ; SAVE REGISTERS
1A66  E8 1DB0 R                 CALL    ENABLE_VG1      ; ENABLE VIDEO GENERATER 1

1A69  52                        PUSH    DX              ; SAVE DX
1A6A  BA 03DA                   MOV     DX,VGA_CTL      ; GET GATE ARRAY ADDRESS
1A6D                    FONT1:
1A6D  EC                        IN      AL,DX           ; SEE STATUS REGISTER
1A6E  24 08                     AND     AL,VERTRET      ; IN VERTICAL RETRACE ?
1A70  74 FB                     JZ      FONT1           ; NO, WAIT UNTIL IT COMES
1A72  5A                        POP     DX              ; RESTORE DX

1A73  E8 1DD6 R                 CALL    ENABLE_VG2      ; ENABLE VIDEO GENERATER 2
1A76  58                        POP     AX              ;

1A77  E8 1B88 R                 CALL    ENABLE_KJROM    ; ENABLE KJ-ROM

1A7A  8A E0                     MOV     AH,AL           ; SAVE FUNCTION TO AH
1A7C  24 7F                     AND     AL,7FH          ; MASK FULL BOX FLAG OFF

1A7E  80 FC 40                  CMP     AH,40H          ; WRITE FONT PATTERN ?
1A81  74 50                     JE      FONT9           ; YES

                                ;------------ REQUEST FONT PATTERN
1A83  3C 00                     CMP     AL,0            ; REQUEST FONT ?
1A85  75 74                     JNE     FONT11

1A87  50                        PUSH    AX              ; SAVE REQUESTED FUNCTION
1A88  51                        PUSH    CX              ; SAVE INTERNAL CODE

1A89  8B C1                     MOV     AX,CX           ;
1A8B  E8 1B01 R                 CALL    CVTCR           ; GET KJ ROM SEGMENT ADDRESS

                                                ASSUME  DS:INDETERMINATE
1A8E  8E D8                     MOV     DS,AX           ; SET CHARACTER PATTERN ADDRESS TO DS
1A90  33 F6                     XOR     SI,SI           ; CLEAR FOR ROW 0
1A92  8B FB                     MOV     DI,BX           ; SET PATTERN RETURN ADDRESS

1A94  59                        POP     CX              ; RESTORE INTERNAL CODE

1A95  0A ED                     OR      CH,CH           ; 2 BYTE CODE ?
1A97  75 12                     JNZ     FONT3           ; YES

1A99  80 FE 10                  CMP     DH,KJ_MODE      ; KJ GRAPHICS MODE ?
1A9C  73 23                     JAE     FONT6           ; YES

1A9E  B9 0008                   MOV     CX,8            ; REPEAT COUNT FOR ANK
1AA1  BE 0001                   MOV     SI,1            ; OFFSET FOR ANK
1AA4  58                        POP     AX              ;
1AA5  80 E4 7F                  AND     AH,7FH          ; MASK FULL BOX FLAG OFF
1AA8  50                        PUSH    AX              ;

1AA9  EB 19                     JMP     SHORT FONT7     ; CONTINUE

1AAB                    FONT3:                          ;--- GET LEFT PART FONT OF 2 BYTE CODE
1AAB  B9 0010                   MOV     CX,16           ; SET ROW NUMBER OF CHARACTER BOX
1AAE                    FONT4:
1AAE  AD                        LODSW                   ; GET ONE ROW
1AAF  24 7F                     AND     AL,KJMASKL      ; MASK OFF CONTROL BIT
1AB1  AA                        STOSB                   ; PUT LEFT PART OF ONE ROW
1AB2  E2 FA                     LOOP    FONT4           ; REPEAT UNTIL EXHAUST LEFT PART

1AB4  58                        POP     AX              ; RESTORE FUNCTION
1AB5  50                        PUSH    AX              ; SAVE IT
1AB6  F6 C4 80                  TEST    AH,80H          ; FULL CHARACTER BOX ?
1AB9  74 03                     JZ      FONT5           ; NO

1ABB  33 C0                     XOR     AX,AX           ;
1ABD  AB                        STOSW                   ; SET 0 FOR FULL BOX
1ABE                    FONT5:
1ABE  BE 0001                   MOV     SI,1            ; SET OFFSET FOR RIGHT PART

1AC1                    FONT6:                          ;--- GET FONT OF 1 BYTE CODE OR RIGHT PART
1AC1  B9 0010                   MOV     CX,16           ; SET ROW NUMBER OF CHARACTER BOX
1AC4                    FONT7:
1AC4  AD                        LODSW                   ; GET ONE ROW
1AC5  AA                        STOSB                   ; PUT RIGHT PART OF ONE ROW
1AC6  E2 FC                     LOOP    FONT7           ; REPEAT UNTIL EXHAUST LEFT PART

1AC8  58                        POP     AX              ; RESTORE FUNCTION
1AC9  F6 C4 80                  TEST    AH,80H          ; FULL CHARACTER BOX ?
1ACC  74 03                     JZ      FONT8           ; NO

1ACE  33 C0                     XOR     AX,AX           ;
1AD0  AB                        STOSW                   ; SET 0 FOR FULL BOX
```

```
1AD1
1AD1  EB 28

1AD3
1AD3  81 F9 F040
1AD7  72 22
1AD9  81 F9 F07E
1ADD  77 1C

1ADF  D1 E1
1AE1  81 E1 03FE
1AE5  81 C9 8800

1AE9  06
1AEA  1F

1AEB  8B F3

1AED  8E C1

1AEF  33 FF

1AF1  B9 0010
1AF4
1AF4  AC
1AF5  8A 64 0F

1AF8  AB
1AF9  E2 F9
1AFB
1AFB  E8 1BB4 R

1AFE  1F
1AFF  5A
1B00  C3

1B01
```

```
FONT8:
        JMP     SHORT FONT11    ; END

                        ;---------- WRITE FONT PATTERN
                        ASSUME  DS:DATA
FONT9:
        CMP     CX,EXT_CHAR_L   ; IN EXTERNAL CHARACTER RANGE ?
        JB      FONT11          ; NO
        CMP     CX,EXT_CHAR_H   ;
        JA      FONT11          ; NO

        SAL     CX,1            ; ADJUST FOR CHARACTER PATTERN ADDRESS
        AND     CX,03FEH        ;
        OR      CX,8800H        ; FORCE '10001' TO MSB

        PUSH    ES              ; SET SOURCE SEGMENT
        POP     DS              ;
                        ASSUME  DS:INDETERMINATE
        MOV     SI,BX           ; SET SOURCE OFFSET

        MOV     ES,CX           ; SET DESTINATION SEGMENT
                        ASSUME  ES:INDETERMINATE
        XOR     DI,DI           ; SET DESTINATION OFFSET

        MOV     CX,16           ; SET NUMBER OF ROW
FONT10:
        LODSB                   ; GET LEFT PART OF ONE ROW
        MOV     AH,DS:[SI+15]   ; GET RIGHT PART OF ONE ROW

        STOSW                   ; STORE ONE ROW
        LOOP    FONT10          ; REPEAT ALL ROW
FONT11:
        CALL    ENABLE_VRAM     ; ENABLE VIDEO RAM

                        ASSUME  DS:DATA
        POP     DS              ; RESTORE DS
        POP     DX              ; RESTORE DX
        RET

FONT    ENDP

;-----------------------------------------
;
;       CVTCR
;       -----
;       THIS ROUTINE CONVERT CODE TO KJ-ROM ADDRESS
;
;       INPUT   AX   =  INTERNAL CODE FOR REQUESTED FONT
;
;       OUTPUT  AX   =  SEGMENT ADDRESS OF KJ-ROM
;
;       CALL            NONE
;       VOLATILE        NONE
;
;-----------------------------------------
                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

CVTCR   PROC    NEAR

        OR      AH,AH           ; 2 BYTE CODE ?
        JNZ     CVTCR1          ; YES

                                ;--- 1 BYTE CODE
        SAL     AX,1            ; ADJUST FOR CHARACTER PATTERN ADDRESS
        OR      AX,8000H        ; FORCE '1000000' TO MSB

        JMP     SHORT CVTCR5    ; END


CVTCR1:                         ;--- 2 BYTE CODE
        CMP     AX,RROSS_L_LOW  ; IN REGEN CODE RANGE OF ROSSIAN CHARACTER ?
        JB      CVTCR4          ; NO
        CMP     AX,RROSS_L_HIGH ;
        JBE     CVTCR2          ; YES

        CMP     AX,JIS1_L       ; IN JIS 1 RANGE ?
        JB      CVTCR4          ; NO
        CMP     AX,JIS1_H       ;
        JA      CVTCR3          ; NO
CVTCR2:
        SAL     AX,1            ; ADJUST FOR CHARACTER PATTERN ADDRESS
        AND     AX,3FFEH        ;
        OR      AX,8000H        ; FORCE '10' TO MSB
        JMP     SHORT CVTCR5    ; END
CVTCR3:                         ;--- EXTERNAL CHARACTER
        CMP     AX,EXT_CHAR_L   ; IN EXTERNAL CHARACTER RANGE ?
        JB      CVTCR4          ; NO
        CMP     AX,EXT_CHAR_H   ;
        JA      CVTCR4          ; NO

        SAL     AX,1            ; ADJUST FOR CHARACTER PATTERN ADDRESS
        AND     AX,03FEH        ;
        OR      AX,8800H        ; FORCE '10001' TO MSB
        JMP     SHORT CVTCR5    ; END
CVTCR4:
CVTCR5: MOV     AX,OFFSET WHITE_BOX     ; SET WHITE BOX
        RET
CVTCR   ENDP
```

```
1B01

1B01  0A E4
1B03  75 07

1B05  D1 E0
1B07  0D 8000

1B0A  EB 35

1B0C
1B0C  3D 8100
1B0F  72 2D
1B11  3D 8120
1B14  76 0A

1B16  3D 8140
1B19  72 23
1B1B  3D 9873
1B1E  77 0A
1B20
1B20  D1 E0
1B22  25 3FFE
1B25  0D 8000
1B28  EB 17

1B2A
1B2A  3D F040
1B2D  72 0F
1B2F  3D F07E
1B32  77 0A

1B34  D1 E0
1B36  25 03FE
1B39  0D 8800
1B3C  EB 03

1B3E
1B3E  B8 B0E8
1B41
1B41  C3

1B42
```

```
                        ;-----------------------------------------------
                        ;
                        ;       CHECK_ROSS_CODE
                        ;
                        ;       THIS ROUTINE TRANSLATE REGEN CODE TO ROSSIAN CHARACTER CODE
                        ;
                        ;       INPUT   CX =    2 BYTE REGEN CHARACTER CODE
                        ;
                        ;       OUTPUT  CX =    2 BYTE CHARACTER CODE
                        ;
                        ;       CALL            NONE
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------------------

1B42                    CHECK_ROSS_CODE PROC    NEAR

1B42  81 F9 8100                CMP     CX,RROSS_L_LOW  ; CHECK LOWER CASE OF ROSSIAN CHARACTER
1B46  72 1C                     JB      RCODE2          ; NO
1B48  81 F9 8120                CMP     CX,RROSS_L_HIGH
1B4C  77 06                     JA      RCODE1          ; NO

1B4E  81 C1 0340                ADD     CX,ROSS_L_LOW - RROSS_L_LOW     ; TRANSLATE
1B52  EB 10                     JMP     SHORT RCODE2

1B54                    RCODE1:                         ; CHECK UPPER CASE OF ROSSIAN CHARACTER
1B54  81 F9 8200                CMP     CX,RROSS_U_LOW
1B58  72 0A                     JB      RCODE2          ; NO
1B5A  81 F9 8221                CMP     CX,RROSS_U_HIGH
1B5E  77 04                     JA      RCODE2          ; NO

1B60  81 C1 0270                ADD     CX,ROSS_U_LOW - RROSS_U_LOW     ; TRANSLATE
1B64                    RCODE2:
1B64  C3                        RET

1B65                    CHECK_ROSS_CODE ENDP

                        ;-----------------------------------------------
                        ;
                        ;       CHECK_ROSS_CHAR
                        ;
                        ;       THIS ROUTINE TRANSLATE ROSSIAN CHARACTER CODE TO REGEN CODE
                        ;
                        ;       INPUT   CX =    2 BYTE CHARACTER CODE
                        ;
                        ;       OUTPUT  CX =    2 BYTE REGEN CHARACTER CODE
                        ;
                        ;       CALL            NONE
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------------------

1B65                    CHECK_ROSS_CHAR PROC    NEAR

1B65  81 F9 8440                CMP     CX,ROSS_L_LOW   ; CHECK LOWER CASE OF ROSSIAN CHARACTER
1B69  72 1C                     JB      RCHAR2          ; NO
1B6B  81 F9 8460                CMP     CX,ROSS_L_HIGH
1B6F  77 06                     JA      RCHAR1          ; NO
1B71  81 E9 0340                SUB     CX,ROSS_L_LOW - RROSS_L_LOW     ; TRANSLATE
1B75  EB 10                     JMP     SHORT RCHAR2

1B77                    RCHAR1:                         ; CHECK UPPER CASE OF ROSSIAN CHARACTER
1B77  81 F9 8470                CMP     CX,ROSS_U_LOW
1B7B  72 0A                     JB      RCHAR2          ; NO
1B7D  81 F9 8491                CMP     CX,ROSS_U_HIGH
1B81  77 04                     JA      RCHAR2          ; NO

1B83  81 E9 0270                SUB     CX,ROSS_U_LOW - RROSS_U_LOW     ; TRANSLATE
1B87                    RCHAR2:
1B87  C3                        RET

1B88                    CHECK_ROSS_CHAR ENDP

                        ;-----------------------------------------
                        ;
                        ;       ENABLE_KJROM
                        ;
                        ;       THIS ROUTINE ENABLES KJ-ROM
                        ;
                        ;       INPUT           NONE
                        ;       OUTPUT          NONE
                        ;       CALL            DDS
                        ;                       DISABLE_INT
                        ;                       ENABLE_INT
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------------

1B88                    ENABLE_KJROM    PROC    NEAR

1B88  50                        PUSH    AX              ;
1B89  53                        PUSH    BX              ; SAVE REGISRES
1B8A  52                        PUSH    DX              ;
1B8B  1E                        PUSH    DS              ;

1B8C  E8 0000 E                 CALL    DDS             ; POINT DATA AREA
                                ASSUME  DS:DATA

1B8F  E8 1BE4 R                 CALL    DISABLE_INT     ; DISABLE ALL INTERRUPT
                                                        ; DURING VRAM AND KJ-ROM SWITHING

1B92  BA 01FF                   MOV     DX,SX08BASE     ; ADDRESS OF ADDRESS CONTROLLER
```

```
1B95  BB 0937                   MOV     BX,S8VRAM1*100H + VRAM1_OFF
1B98  E8 1E22 R                 CALL    OUT_GA          ; DISABLE VIDEO RAM 1

1B9B  BB 0A37                   MOV     BX,S8VRAM2*100H + VRAM2_OFF
1B9E  E8 1E22 R                 CALL    OUT_GA          ; DISABLE VIDEO RAM 2

1BA1  BB 0780                   MOV     BX,S8KJROM*100H + KJROM_ON
1BA4  E8 1E22 R                 CALL    OUT_GA          ; ENABLE KANJI ROM

1BA7  C6 06 0353 R FF           MOV     KJROM_STAT,TRUE ; KJ-ROM FLAG ON

1BAC  E8 1BEC R                 CALL    ENABLE_INT      ; ENABLE INTERRUPT

1BAF  1F                        POP     DS              ;
1BB0  5A                        POP     DX              ;
1BB1  5B                        POP     BX              ; RESTORE REGISTERS
1BB2  58                        POP     AX              ;

1BB3  C3                        RET
1BB4                    ENABLE_KJROM    ENDP

                        ;-----------------------------------------
                        ;
                        ;       ENABLE_VRAM
                        ;
                        ;       THIS ROUTINE ENABLES VIDEO RAM
                        ;
                        ;       INPUT           NONE
                        ;       OUTPUT          NONE
                        ;       CALL            DDS
                        ;                       DISABLE_INT
                        ;                       ENABLE_INT
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------------

1BB4                    ENABLE_VRAM     PROC    NEAR

1BB4  50                        PUSH    AX              ;
1BB5  53                        PUSH    BX              ; SAVE REGISTERS
1BB6  52                        PUSH    DX              ;
1BB7  1E                        PUSH    DS              ;

1BB8  E8 0000 E                 CALL    DDS             ; POINT DATA AREA
                                        ASSUME  DS:DATA

1BBB  E8 1BE4 R                 CALL    DISABLE_INT     ; DISABLE ALL INTERRUPT
                                                        ; DURING VRAM AND KJ-ROM SWITHING

1BBE  BA 01FF                   MOV     DX,SX08BASE     ; ADDRESS OF ADDRESS CONTROLLER
1BC1  BB 0730                   MOV     BX,S8KJROM*100H + KJROM_OFF
1BC4  E8 1E22 R                 CALL    OUT_GA          ; DISABLE KANJI ROM

1BC7  BB 09B7                   MOV     BX,S8VRAM1*100H + VRAM1_ON

1BCA  80 3E 034C R 08           CMP     CPU_PAGE,VRAM2_PAGE  ; VIDEO RAM 1 ?
1BCF  72 03                     JB      EVRAM1          ; YES

1BD1  BB 0AB7                   MOV     BX,S8VRAM2*100H + VRAM2_ON
1BD4                    EVRAM1:
1BD4  E8 1E22 R                 CALL    OUT_GA          ; ENABLE VIDEO RAM

1BD7  C6 06 0353 R 00           MOV     KJROM_STAT,FALSE; KJ-ROM FLAG OFF

1BDC  E8 1BEC R                 CALL    ENABLE_INT      ; ENABLE INTERRUPT

1BDF  1F                        POP     DS              ;
1BE0  5A                        POP     DX              ;
1BE1  5B                        POP     BX              ; RESTORE REGISTERS
1BE2  58                        POP     AX              ;

1BE3  C3                        RET
1BE4                    ENABLE_VRAM     ENDP

                        ;-----------------------------------------
                        ;
                        ;       DISABLE_INT
                        ;
                        ;       THIS ROUTINE DISALES ALL INTERRUPT
                        ;
                        ;       INPUT           NONE
                        ;       OUTPUT          NONE
                        ;       CALL            NONE
                        ;       VOLATILE        NONE
                        ;
                        ;-----------------------------------------

1BE4                    DISABLE_INT     PROC    NEAR

1BE4  50                        PUSH    AX

1BE5  FA                        CLI
1BE6  B0 10                     MOV     AL,DISABLE_NMI  ; DISABLE INTERRUPTS
1BE8  E6 A0                     OUT     NMI_PORT,AL     ; DISABLE NMI AND HOLD REQUEST
                                                        ;
1BEA  58                        POP     AX
1BEB  C3                        RET

1BEC                    DISABLE_INT     ENDP
```

```
                        ;----------------------------------------
                        ;
                        ;       ENABLE_INT
                        ;
                        ;       THIS ROUTINE ENABLES ALL INTERRUPT
                        ;
                        ;       INPUT           NONE
                        ;       OUTPUT          NONE
                        ;       CALL            DDS
                        ;       VOLATILE        NONE
                        ;
                        ;----------------------------------------

1BEC                    ENABLE_INT      PROC    NEAR

1BEC  50                        PUSH    AX
1BED  1E                        PUSH    DS

1BEE  E8 0000 E                 CALL    DDS             ; POINT DATA AREA
                                        ASSUME DS:DATA

1BF1  F6 06 0355 R FF           TEST    BYTE PTR IEP_CTRL,TRUE  ; INTERRUPT ENABLEL PROHIBIT ?
1BF6  75 05                     JNZ     ENBLI1                  ; YES, DO NOT ENABLE

1BF8  B0 80                     MOV     AL,ENABLE_NMI   ; ENABLE NMI
1BFA  E6 A0                     OUT     NMI_PORT,AL     ;
1BFC  FB                        STI                     ; ENABLE INTERRUPTS
1BFD                    ENBLI1:
1BFD  1F                        POP     DS
1BFE  58                        POP     AX
1BFF  C3                        RET

1C00                    ENABLE_INT      ENDP
                        ;--------------------------------------------------
                        ;
                        ;       SUPERIMPOSE             INT 10H, AH = 20 (14H)
                        ;       -----------
                        ;       THIS ROUTINE CONTROLS SUPERIMPOSE FUNCTION
                        ;
                        ;       INPUT   AL = 0  SET MODE OF VIDEO GENERATER 1
                        ;                       BH = 4  320X200 4 COLOR  (ANK)
                        ;                       BH = 5  RESERVED
                        ;                       BH = 6  RESERVED
                        ;                       BH = 7  NOT VALID
                        ;                       BH = 8  160X200 16 COLOR (ANK)
                        ;                       BH = 9  320X200 16 COLOR (ANK)
                        ;                       BH = A  640X200 4 COLOR  (ANK)
                        ;                       BH =14  320X200 4 COLOR  (KJ)
                        ;                       BH =15  RESERVED
                        ;                       BH =16  RESERVED
                        ;                       BH =17  NOT VALID
                        ;                       BH =18  160X200 16 COLOR (KJ)
                        ;                       BH =19  320X200 16 COLOR (KJ)
                        ;                       BH =1A  640X200 4 COLOR  (KJ)
                        ;               AL = 1  SET SUPERIMPOSE
                        ;                       BH = 0  OFF
                        ;                       BH = 1  ON
                        ;               AL = 2  SET FORGROUND PAGE
                        ;                       BH = 0  VRAM-1
                        ;               AL = 3  SET TRANSPARENT PALETTE REGISTER
                        ;                       BH =  PALETTE RIGISTER NUMBER
                        ;               AL = 4  SET SUPERIMPOSE MODE
                        ;                       BH = 0: PRIORITY
                        ;                       BH = 1: XOR
                        ;                       BH = 2: AND
                        ;                       BH = 3: OR
                        ;
                        ;       OUTPUT          NONE
                        ;
                        ;       CALL            ENABLE_VG1
                        ;                       ERASE_SCURSOR
                        ;                       SUP_SET_MODE
                        ;                       APPEAR_SCURSOR
                        ;                       OUT_GA
                        ;                       ENABLE_VG2
                        ;
                        ;       VOLATILE        AX,BX,CX,DX,SI,DI,ES
                        ;
                        ;--------------------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

1C00                    SUPERIMPOSE     PROC    NEAR

1C00  E8 1DB0 R                 CALL    ENABLE_VG1      ; SELECT VIDEO GENERATER 1

1C03  0A C0                     OR      AL,AL           ; SUPERIMPOSE MODE SET ?
1C05  75 0F                     JNZ     SIP1            ; NO

1C07  50                        PUSH    AX              ; SAVE CURRENT CRT MODE
1C08  E8 0940 R                 CALL    ERASE_SCURSOR   ; ERASE SOFTWARE CURSOR
1C0B  8A C7                     MOV     AL,BH           ; SET MODE TO AL
1C0D  E8 1C7D R                 CALL    SUP_SET_MODE    ; SET MODE OF VIDEO GENERATER 1

1C10  58                        POP     AX              ; RESTORE CRT MODE
1C11  E8 0961 R                 CALL    APPEAR_SCURSOR  ; APPEAR SOFTWARE CURSOR
1C14  EB 63                     JMP     SHORT SIP7      ; END

1C16                    SIP1:
1C16  3C 02                     CMP     AL,2            ; SUPERIMPOSE ON/OFF FUCTION ?
1C18  73 18                     JAE     SIP2            ; NO
                                                        ;--- AL=1 SUPERIMPOSE OFF/ON
1C1A  8A 26 0347 R              MOV     AH,SUPIPCR      ; GET LAST VALUE OF SUPERIM. CONT. REG.
1C1E  8A C7                     MOV     AL,BH           ; GET VALUE
1C20  D0 E0                     SHL     AL,1            ; SHIFT TO TRANSPARENT BIT POSITION
```

```
1C22  80 E4 FD          AND     AH,NOT TRANSON  ; MASK TRANSPARENT BIT OFF
1C25  24 02             AND     AL,TRANSON      ; GET TRANSPARENT BIT
1C27  0A C4             OR      AL,AH           ; SET TRANSPARENT BIT
1C29  A2 0347 R         MOV     SUPIPCR,AL      ; UPDATE BY NEW VALUE
1C2C  8A D8             MOV     BL,AL           ; SET IT TO BL
1C2E  B7 06             MOV     BH,PCSUPER      ; SET REG. ADDRESS OF SUPERIM. CONT. REG.
1C30  EB 39             JMP     SHORT   SIP5

                SIP2:
1C32                    CMP     AL,3
1C32  3C 03             JAE     SIP3
1C34  73 16                                     ;--- AL=2 SET FORGOUND PAGE
1C36  8A 26 0347 R      MOV     AH,SUPIPCR      ; GET LAST VALUE OF SUPERIM. CONT. REG.
1C3A  8A C7             MOV     AL,BH           ; GET VALUE
1C3C  80 E4 FE          AND     AH,NOT FOREVRAM ; MASK FORGROUND  BIT OFF
1C3F  24 01             AND     AL,FOREVRAM     ; GET FORGROUND BIT
1C41  0A C4             OR      AL,AH           ; SET FORGROUND BIT
1C43  A2 0347 R         MOV     SUPIPCR,AL      ; UPDATE BY NEW VALUE
1C46  8A D8             MOV     BL,AL           ; SET IT TO BL
1C48  B7 06             MOV     BH,PCSUPER      ; SET REG. ADDRESS OF SUPERIM. CONT. REG.
1C4A  EB 1F             JMP     SHORT   SIP5

                SIP3:
1C4C                    JA      SIP4            ; SET SUPERIMPOSE MODE FUNCTION ?, YES
1C4C  77 06
                                                ;--- AL=3 SET TRANSPARENT PALETTE REGISTER
1C4E  8A DF             MOV     BL,BH           ; SET TRANSPARENT PALETTE NUMBER TO BL
1C50  B7 05             MOV     BH,PCTRPALT     ; SET REG. ADDRESS OF TRANSPARENT PALETTE
1C52  EB 1F             JMP     SHORT   SIP6

                SIP4:
1C54                    CMP     AL,4            ; SET SUPERIMPOSE MODE ?
1C54  3C 04             JNE     SIP7            ; NO
1C56  75 21                                     ;--- AL=4 SET SUPERIMPOSE MODE
1C58  D0 E7             SAL     BH,1            ; ADJUST POSITION
1C5A  D0 E7             SAL     BH,1            ; ADJUST POSITION
1C5C  A0 0347 R         MOV     AL,SUPIPCR      ; GET LAST VALUE OF SUPERIM. CONT. REG.
1C5F  24 03             AND     AL,00000011B    ; MASK TRANSPARENT BIT OFF
1C61  0A F8             OR      BH,AL           ; SET SUPERIMPOSE MODE BIT
1C63  88 3E 0347 R      MOV     SUPIPCR,BH      ; UPDATE BY NEW VALUE
1C67  8A DF             MOV     BL,BH           ; SET IT TO BL
1C69  B7 06             MOV     BH,PCSUPER      ; SET REG. ADDRESS OF SUPERIM. CONT. REG.
1C6B                    
                SIP5:
1C6B  F6 C3 02          TEST    BL,TRANSON      ; SET SUPERIMPOSE ON ?
1C6E  75 03             JNE     SIP6            ; YES
1C70  80 E3 01          AND     BL,FOREVRAM     ; MASK MODE CONTROL BITS FOR SUPERIMPOSE OFF
1C73                    
                SIP6:
1C73  BA 03DA           MOV     DX,VGA_CTL      ; GET I/O ADDRESS OF VIDEO GATE ARRAY
1C76  E8 1E22 R         CALL    OUT_GA          ; OUT BX TO GATE ARRAY
1C79                    
                SIP7:
1C79  E8 1DD6 R         CALL    ENABLE_VG2      ; SELECT VIDEO GENERATER 2

1C7C  C3                RET                     ; ALL DONE

1C7D                    SUPERIMPOSE     ENDP
```

```
;-----------------------------------------
;
;       SUP_SET_MODE
;
;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO
;       THE SELECTED MODE FOR SUPERIMPOSE
;       THE SCREEN IS BLANKED.
;
;       INPUT   AL = MODE SELECTED (RANGE 4-A)
;
;       OUTPUT  NONE
;
;       CALL            DISABLE_INT
;                       SET_PALETTE4
;                       SET_PALETTE16
;                       VGA_RESET
;                       ENABLE_VG1
;                       ENABLE_VG2
;                       SET_SUPREG
;                       ENABLE_INT
;
;       VOLATILE        AX,BX,CX,DX,SI,DI
;
;-----------------------------------------
        ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

1C7D    SUP_SET_MODE    PROC    NEAR

1C7D  FF 36 0355 R      PUSH    WORD PTR IEP_CTRL       ; SAVE INTERRUPT ENABLE PROHIBIT FLAG
1C81  80 0E 0355 R FF   OR      BYTE PTR IEP_CTRL,TRUE  ; PROHIBIT INTERRUPT ENABLE
1C86  E8 1BE4 R         CALL    DISABLE_INT             ; DISABLE HARDWARE INTERRUPT

1C89  50                PUSH    AX              ; SAVE INPUT MODE ON STACK
1C8A  24 7F             AND     AL,7FH          ; MASK REGEN NON CLEAR FLAG
1C8C  8A F8             MOV     BH,AL           ; SET INPUT MODE TO BH
1C8E  24 EF             AND     AL,0EFH         ; MASK KJ BIT OFF

1C90  3C 04             CMP     AL,4            ;--- CHECK FOR VALID MODES
1C92  72 24             JB      SSETM2          ; MODE 0-3 IS INVALID
1C94  3C 07             CMP     AL,7            ; CHECK FOR VALID MODES
1C96  74 20             JE      SSETM2          ; MODE 7 IS INVALID
1C98  3C 0A             CMP     AL,0AH          ; UNDER 0BH ?
1C9A  77 1C             JA      SSETM2          ; NO, INVALID

1C9C  8A 26 033B R      MOV     AH,CRT_MODE2    ;--- CHECK VALID SUPERIMPOSE COMBINATION
1CA0  80 E4 0F          AND     AH,KJ_OFF       ; GET CRT MODE OF VIDEO PROCESSOR-2
1CA3  80 FC 02          CMP     AH,2            ; MASK KJ BIT OFF
                                                ; CURRENT MODE IS HIGH BAND ?
```

```
1CA6  72 0A                     JB      SSETM1          ; NO

1CA8  80 FC 04                  CMP     AH,4            ; CURRENT MODE IS HIGH BAND ?
1CAB  72 0F                     JB      SSETM3          ; YES
1CAD  80 FC 09                  CMP     AH,9            ; CURRENT MODE IS HIGH BAND ?
1CB0  73 0A                     JAE     SSETM3          ; YES
1CB2                    SSETM1:                         ;- CURRENT MODE IS LOW BAND
1CB2  3C 09                     CMP     AL,9            ; CHECK LOW BAND
1CB4  73 02                     JAE     SSETM2          ; MODE CONFRICT
1CB6  EB 10                     JMP     SHORT SSETM4    ; OK

1CB8                    SSETM2:
1CB8  58                        POP     AX              ; RESTORE AX
1CB9  E9 1D8D R                 JMP     SSETM11         ; BREAK

1CBC                    SSETM3:                         ;- CURRENT MODE IS HIGH BAND
1CBC  3C 09                     CMP     AL,9            ; CHECK HIGH BAND
1CBE  72 F8                     JB      SSETM2          ; MODE CONFRICT

1CC0  50                        PUSH    AX              ; SAVE AX
1CC1  E4 62                     IN      AL,PORT_C       ; GET DATA FROM PORT-C
1CC3  A8 08                     TEST    AL,EXP64K       ; 64K EXPANTION INSTALLED ?
1CC5  58                        POP     AX              ; RESTORE AX
1CC6  75 F0                     JNZ     SSETM2          ; NO

1CC8                    SSETM4:
1CC8  50                        PUSH    AX              ; SAVE CURRENT CRT MODE AND SETTING MODE(MASKED)
1CC9  8A E7                     MOV     AH,BH           ; SAVE MODE IN AH
1CCB  8B F8                     MOV     DI,AX           ; SAVE MODE IN DI

1CCD  E8 1A44 R                 CALL    WAIT_VERTRET    ; WAIT UNTIL VERTICAL RETRACE
                                                        ;----- TURN OFF VIDEO
1CD0  32 C0                     XOR     AL,AL           ; SET VGA REG 0
1CD2  EE                        OUT     DX,AL           ; SELECT IT

1CD3  A0 033D R                 MOV     AL,CRT_MODE_SET2; GET LAST MODE SET
1CD6  24 F7                     AND     AL,NOT VIDEOENB ; TURN OFF VIDEO
1CD8  EE                        OUT     DX,AL           ; SET IN GATE ARRAY

                                                        ;----- SET DEFAULT PALETTES
1CD9  59                        POP     CX              ; GET CURRENT CRT MODE AND SETTING MODE

1CDA  F6 06 03FC R FF           TEST    SUPRESS_PAL,TRUE; SET PALETTE IS SUPRESSED ?
1CDF  75 2E                     JNZ     SSETM7          ; YES, SKIP

1CE1  B0 06                     MOV     AL,PCSUPER      ;
1CE3  EE                        OUT     DX,AL           ; CLEAR SUPERIMPOSE CONTROL REGISTER
1CE4  32 C0                     XOR     AL,AL           ;       FOR SET PALETTE
1CE6  EE                        OUT     DX,AL           ;

1CE7  B4 10                     MOV     AH,PCPALET      ; SET PALETTE REG 0
1CE9  BB 02BD R                 MOV     BX,OFFSET PLTC41; POINT TO TABLE ENTRY
1CEC  80 F9 04                  CMP     CL,4            ; CHECK FOR 4 COLOR MODE
1CEF  74 11                     JE      SSETM5          ; YES, JUMP
1CF1  80 F9 05                  CMP     CL,5            ; CHECK FOR 4 COLOR MODE
1CF4  74 0C                     JE      SSETM5          ; YES JUMP

1CF6  80 F9 08                  CMP     CL,8            ; CHECK FOR 16 COLOR MODE
1CF9  74 11                     JE      SSETM6          ; YES, JUMP
1CFB  80 F9 09                  CMP     CL,9            ; CHECK FOR 16 COLOR MODE
1CFE  74 0C                     JE      SSETM6          ; YES, JUMP

1D00  EB 0D                     JMP     SHORT SSETM7    ; SKIP SET PALETTES

1D02                    SSETM5:
1D02  80 FD 06                  CMP     CH,06H          ; 2 COLOR MODE ?
1D05  75 08                     JNE     SSETM7          ; NO, SKIP SET PALETTES

1D07  E8 0535 R                 CALL    SET_PALETTE4    ; SET PALETTE 4 COLOR
1D0A  EB 03                     JMP     SHORT SSETM7

1D0C                    SSETM6:
1D0C  E8 0545 R                 CALL    SET_PALETTE16   ; SET PALETTE 16 COLOR

1D0F                    SSETM7:                         ;----- SET UP M0 & M1 IN PAGREG
1D0F  8B C7                     MOV     AX,DI           ; GET CURRENT MODE

1D11  B3 40                     MOV     BL,40H          ; SET UP FOR 16K REGEN
1D13  3C 09                     CMP     AL,09H          ; MODE USE 16K
1D15  72 02                     JC      SSETM8          ; YES, JUMP

1D17  B3 C0                     MOV     BL,0C0H         ; SET UP FOR 32K REGEN
1D19                    SSETM8:
1D19  BA 03DF                   MOV     DX,PAGREG       ; SET PORT ADDRESS OF PAGREG
1D1C  A0 008A R                 MOV     AL,PAGDAT       ; GET LAST DATA OUTPUT
1D1F  24 3F                     AND     AL,3FH          ; CLEAR M0 & M1 BITS
1D21  0A C3                     OR      AL,BL           ; SET NEW BITS
1D23  EE                        OUT     DX,AL           ; STUFF BACK IN PORT
1D24  A2 008A R                 MOV     PAGDAT,AL       ; SAVE COPY IN RAM

                                                        ;--- ENABLE VIDEO AND CORRECT PORT SETTING
1D27  E8 0551 R                 CALL    VGA_RESET       ; RESET VGA

1D2A  B0 01                     MOV     AL,PCPALETM     ; SELECT PALETTE MASK REGISTER
1D2C  EE                        OUT     DX,AL           ; SET IT
1D2D  2E: 8A 47 01              MOV     AL,CS:[BX+PCPALETM]
1D31  24 0F                     AND     AL,0FH          ; MASK EXTRA DATA OFF
1D33  EE                        OUT     DX,AL           ; SET IT

1D34  B0 03                     MOV     AL,PCMODE2      ; SELECT MODE REGISTER 2
1D36  EE                        OUT     DX,AL           ; SET IT
1D37  2E: 8A 47 03              MOV     AL,CS:[BX+PCMODE2]
1D3B  24 0F                     AND     AL,0FH          ; MASK EXTRA DATA OFF
1D3D  EE                        OUT     DX,AL           ; SET IT
```

```
1D3E  8A 26 0347 R
1D42  E8 0929 R

1D45  E8 1DD6 R

1D48  EC
1D49  B0 01
1D4B  EE
1D4C  A0 0352 R
1D4F  0C 10
1D51  EE

1D52  8B C6
1D54  88 26 033D R

1D58  E4 61
1D5A  24 FB
1D5C  E6 61

1D5E  8B F7
1D60  8B C6

1D62  33 FF

1D64  5A
1D65  80 E2 80
1D68  75 15

1D6A  B9 2000

1D6D  3C 09
1D6F  B8 B800
1D72  72 05

1D74  D1 E1
1D76  E8 1D95 R
1D79
1D79  8E C0
1D7B  33 C0
1D7D  F3/ AB

1D7F
1D7F  E8 1DB0 R

1D82  BA 03DA
1D85  EC
1D86  32 C0
1D88  EE
1D89  A0 033D R
1D8C  EE

1D8D
1D8D  8F 06 0355 R
1D91  E8 1BEC R

1D94  C3

1D95
```

```
        MOV     AH,SUPIPCR      ; RESTOER SUPERIMPOSE CONTROL REGISTER
        CALL    SET_SUPREG      ;

        CALL    ENABLE_VG2      ; ENABLE VIDEO GATE ARRAY 2

        IN      AL,DX           ; INSURE ADDRESS STATE
        MOV     AL,IXPALETM     ; SELECT PALETTE MASK REG
        OUT     DX,AL           ; SET IT
        MOV     AL,PALETTE_MASK ; GET LAST VALUE OF PALETTE MASK REGISTER
        OR      AL,VRAM1ENB     ; SET V-RAM1 ENABLE BIT
        OUT     DX,AL           ; SET IT

        MOV     AX,SI           ; PUT MODE SET & PALETTE IN RAM
        MOV     CRT_MODE_SET2,AH
                                ;----- SETUP PORT B
        IN      AL,PORT_B       ; GET CURRENT VALUE OF 8255 PORT B
        AND     AL,NOT PORT_B_ALPHA ; SET UP GRAPHICS MODE
        OUT     PORT_B,AL       ; STUFF BACK IN 8255

        MOV     SI,DI           ; SET MODE TO SI
        MOV     AX,SI           ; GET MODE BACK

                                ;--- FILL REGEN AREA WITH BLANK
        XOR     DI,DI           ; SET UP POINTER FOR REGEN

        POP     DX              ; GET ORIGINAL INPUT BACK
        AND     DL,80H          ; NO CLEAR OF REGEN ?
        JNZ     SSETM10         ; SKIP CLEARING REGEN

        MOV     CX,8192         ; NUMBER OF WORDS TO CLEAR

        CMP     AL,09H          ; REQUIRE 32K BYTE REGEN ?
        MOV     AX,REGEN_START  ; SET SEGMENT OF 16K REGEN BUFFER
        JC      SSETM9          ; NO, JUMP

        SHL     CX,1            ; SET 16K WORDS TO CLEAR
        CALL    GET_DIRSEG      ; GET SEGMENT OF 32K REGEN
SSETM9:
        MOV     ES,AX           ; SET REGEN SEGMENT
        XOR     AX,AX           ; FILL FOR GRAPHICS MODE
        REP     STOSW           ; FILL THE REGEN BUFFER WITH BLANKS

SSETM10:                        ;----- ENABLE VIDEO
        CALL    ENABLE_VG1      ; ENABLE VIDEO GATE ARRAY 1

        MOV     DX,VGA_CTL      ; SET PORT ADDRESS OF VGA
        IN      AL,DX           ; INSURE ADDRESS MODE
        XOR     AL,AL           ;
        OUT     DX,AL           ; SELECT VGA REG 0
        MOV     AL,CRT_MODE_SET2; GET MODE SET VALUE
        OUT     DX,AL           ; SET MODE

SSETM11:
        POP     WORD PTR IEP_CTRL;RESTORE INTERRUPT ENABLE PROHIBIT FLAG
        CALL    ENABLE_INT      ; ENABLE INTERRUPT IF IT IS NOT PROHIBITED

        RET

SUP_SET_MODE    ENDP
```

```
;-----------------------------------------
;
;       GET_DIRSEG
;
;       THIS ROUTINE DETERMINE DIRECT ADDRESS
;               SEGMENT OF V-RAM1 (MAIN RAM)
;
;       INPUT           DS = DATA SEGMENT
;       OUTPUT          AX = SEGMENT VALUE
;       CALL            NONE
;       VOLATILE        NONE
;
;-----------------------------------------
GET_DIRSEG      PROC    NEAR

        MOV     AH,PAGDAT               ; GET COPY OF PAGE REGS
        AND     AX,CPUREG * 100H        ; ISOLATE CPU REG
        SHR     AH,1                    ; SHIFT TO MAKE INTO SEGMENT VALUE

        PUSH    BX                      ; SAVE BX
        MOV     BX,TRUE_MEM             ; GET MEMORY SIZE

        PUSH    CX                      ;
        MOV     CL,6                    ;
        SAL     BX,CL                   ; ADJUST FOR SEGMENT VALUE
        POP     CX                      ;

        SUB     BH,00100000B            ;
        OR      AH,BH                   ; SHIFT VIDEO RAM ADDRESS TO TOP 128K
        POP     BX                      ; RESTORE BX

        RET

GET_DIRSEG      ENDP
```

```
;-----------------------------------------
;
;       ENABLE_VG1
;
;       THIS ROUTINE ENABLES VIDEO GENERATER 1
```

```
                        ;       INPUT           NONE
                        ;       OUTPUT          NONE
                        ;       CALL            DISABLE_INT
                        ;                       ENABLE_INT
                        ;                       OUT_GA
                        ;       VOLATILE        NONE
                        ;
                        ;---------------------------------------

1DB0                    ENABLE_VG1      PROC    NEAR

1DB0  50                        PUSH    AX              ;
1DB1  53                        PUSH    BX              ; SAVE REGISTERS
1DB2  52                        PUSH    DX              ;
1DB3  1E                        PUSH    DS              ;

1DB4  E8 0000 E                 CALL    DDS             ; POINT DATA AREA
                                        ASSUME  DS:DATA

1DB7  E8 1BE4 R                 CALL    DISABLE_INT     ; DISABLE ALL INTERRUPT

1DBA  BA 01FF                   MOV     DX,SX08BASE     ; ADDRESS OF ADDRESS CONTROLLER
1DBD  BB 8D7B                   MOV     BX,S8SX02B*100H + SX02B_OFF
1DC0  E8 1E22 R                 CALL    OUT_GA          ; DISABLE VIDEO GENERATER 2

1DC3  BB 8CFB                   MOV     BX,S8SX02A*100H + SX02A_ON
1DC6  E8 1E22 R                 CALL    OUT_GA          ; ENABLE VIDEO GENERATER 1

1DC9  C6 06 0354 R 01           MOV     VG_STAT,VG1_ON  ; SET VIDEO GENETAETER 1 ON

1DCE  E8 1BEC R                 CALL    ENABLE_INT      ; ENABLE INTERRUPT

1DD1  1F                        POP     DS              ;
1DD2  5A                        POP     DX              ;
1DD3  5B                        POP     BX              ; RESTORE REGISTERS
1DD4  58                        POP     AX              ;

1DD5  C3                        RET

1DD6                    ENABLE_VG1      ENDP

                        ;---------------------------------------
                        ;
                        ;       ENABLE_VG2
                        ;
                        ;       THIS ROUTINE ENABLES VIDEO GENERATER 2
                        ;
                        ;       INPUT           NONE
                        ;       OUTPUT          NONE
                        ;       CALL            DISABLE_INT
                        ;                       ENABLE_INT
                        ;                       OUT_GA
                        ;       VOLATILE        NONE
                        ;
                        ;---------------------------------------

1DD6                    ENABLE_VG2      PROC    NEAR

1DD6  50                        PUSH    AX              ;
1DD7  53                        PUSH    BX              ; SAVE REGISTERS
1DD8  52                        PUSH    DX              ;
1DD9  1E                        PUSH    DS              ;

1DDA  E8 0000 E                 CALL    DDS             ; POINT DATA AREA
                                        ASSUME  DS:DATA

1DDD  E8 1BE4 R                 CALL    DISABLE_INT     ; DISABLE ALL INTERRUPT

1DE0  BA 01FF                   MOV     DX,SX08BASE     ; ADDRESS OF ADDRESS CONTROLLER
1DE3  BB 8C7B                   MOV     BX,S8SX02A*100H + SX02A_OFF
1DE6  E8 1E22 R                 CALL    OUT_GA          ; DISABLE VIDEO GENERATER 1

1DE9  BB 8DFB                   MOV     BX,S8SX02B*100H + SX02B_ON
1DEC  E8 1E22 R                 CALL    OUT_GA          ; ENABLE VIDEO GENERATER 2

1DEF  C6 06 0354 R 00           MOV     VG_STAT,VG2_ON  ; SET VIDEO GENERATER 1 OFF

1DF4  E8 1BEC R                 CALL    ENABLE_INT      ; ENABLE INTERRUPT

1DF7  1F                        POP     DS              ;
1DF8  5A                        POP     DX              ;
1DF9  5B                        POP     BX              ; RESTOER REGISTERS
1DFA  58                        POP     AX              ;

1DFB  C3                        RET

1DFC                    ENABLE_VG2      ENDP

                        ;---------------------------------------
                        ;
                        ;       ENABLE_VG12
                        ;
                        ;       THIS ROUTINE ENABLES VIDEO GENERATER 1 AND 2
                        ;
                        ;       INPUT           NONE
                        ;       OUTPUT          NONE
                        ;       CALL            DISABLE_INT
                        ;                       ENABLE_INT
                        ;                       OUT_GA
                        ;       VOLATILE        NONE
                        ;
                        ;---------------------------------------
```

```
1DFC                            ENABLE_VG12     PROC    NEAR

                                        PUSH    AX              ;
1DFC  50                                PUSH    BX              ; SAVE REGISTERS
1DFD  53                                PUSH    DX              ;
1DFE  52                                PUSH    DS              ;
1DFF  1E

                                        CALL    DDS             ; POINT BIOS DATA AREA
                                                        ASSUME  DS:DATA
1E00  E8 0000 E                         CALL    DISABLE_INT     ; DISABLE ALL INTERRUPT
1E03  E8 1BE4 R

                                        MOV     DX,S8SX08BASE   ; ADDRESS OF ADDRESS CONTROLLER
1E06  BA 01FF                           MOV     BX,S8SX02A*100H + SX02A_ON
1E09  BB 8CFB                           CALL    OUT_GA          ; ENABLE VIDEO GENERATER 1
1E0C  E8 1E22 R

                                        MOV     BX,S8SX02B*100H + SX02B_ON
1E0F  BB 8DFB                           CALL    OUT_GA          ; ENABLE VIDEO GENERATER 2
1E12  E8 1E22 R

                                        MOV     VG_STAT,VG12_ON ; SET FLAG
1E15  C6 06 0354 R 02                   CALL    ENABLE_INT      ; ENABLE INTERRUPT
1E1A  E8 1BEC R

                                        POP     DS              ;
1E1D  1F                                POP     DX              ;
1E1E  5A                                POP     BX              ; RESTORE REGISTERS
1E1F  5B                                POP     AX              ;
1E20  58

                                        RET
1E21  C3

                                ENABLE_VG12     ENDP
1E22

                                ;-----------------------------------------
                                ;
                                ;       OUT_GA
                                ;
                                ;       THIS ROUTINE OUTPUT ADDRESS/DATA TO GATE ARRAY
                                ;
                                ;       INPUT   BH =    INTERNAL ADDRESS
                                ;               BL =    DATA TO WRITE
                                ;               DX =    ADDRESS OF GATE ARRAY
                                ;
                                ;       OUTPUT  NONE
                                ;
                                ;       CALL    NONE
                                ;
                                ;       VOLATILE        AL
                                ;
                                ;-----------------------------------------

1E22                            OUT_GA  PROC    NEAR

1E22  EC                                IN      AL,DX           ; SYNC TO ADDRESS STATE

1E23  8A C7                             MOV     AL,BH
1E25  EE                                OUT     DX,AL           ; WRITE INTERNAL ADDRESS
1E26  8A C3                             MOV     AL,BL
1E28  EE                                OUT     DX,AL           ; WRITE DATA

1E29  C3                                RET

1E2A                            OUT_GA  ENDP
                                ;-----------------------------------------------------
                                ;
                                ;       SET_ALT_CTYPE                   INT 10H, AH = 81H
                                ;       -------------
                                ;
                                ;       THIS ROUTINE SETS THE ALTERNATE CURSOR VALUE
                                ;
                                ;       INPUT   AH =    CURRENT CRT MODE ( MASKED )
                                ;               CX =    CURSOR VALUE CH-START LINE, CL-STOP LINE
                                ;
                                ;       OUTPUT  NONE
                                ;
                                ;       CALL            WRITE_ALT_CURSOR
                                ;                       WRITE_GCURSOR
                                ;
                                ;       VOLATILE        BX,DX,DI
                                ;
                                ;-----------------------------------------------------
                                        ASSUME  CS:CODE, DS:DATA
1E2A                            SET_ALT_CTYPE   PROC    NEAR

1E2A  8B 16 034A R                      MOV     DX,ALT_CURSOR_POSN; GET CURRENT CURSOR POSITION

1E2E  8A 1E 0348 R                      MOV     BL,AC_PRESENT   ; GET ALTERNATE CURSOR PRESENT FLAG
1E32  0A DB                             OR      BL,BL           ; ALTERNATE CURSOR PRESENT ?
1E34  74 15                             JZ      SACT3           ; NO

1E36  51                                PUSH    CX              ; SAVE NEW CURSOR MODE
1E37  8B 0E 0350 R                      MOV     CX,ACURSOR_MODE ; GET OLD CURSOR MODE

1E3B  80 FC 04                          CMP     AH,GRAPHICS     ; GRAPHICS MODE ?
1E3E  73 05                             JAE     SACT1           ; YES

1E40  E8 1E9A R                         CALL    WRITE_ALT_CURSOR; ERASE CURRENT ALTERNATE CURSOR
1E43  EB 03                             JMP     SHORT   SACT2

1E45                            SACT1:
1E45  E8 0618 R                 SACT2:  CALL    WRITE_GCURSOR   ; ERASE CURRENT ALTERNATE GRAPHICS CURSOR
1E48
1E48  59                                POP     CX
1E49  32 DB                             XOR     BL,BL           ; RESTORE NEW CURSOR MODE
1E4B                            SACT3:                          ; CLEAR ALTERNATE CURSOR FLAG
1E4B  89 0E 0350 R                      MOV     ACURSOR_MODE,CX ; SAVE NEW GRAPHICS CURSOR MODE
```

```
1E4F  80 FC 04                        CMP     AH,GRAPHICS     ; GRAPHICS MODE ?
1E52  73 05                           JAE     SACT4           ; YES

1E54  E8 1E9A R                       CALL    WRITE_ALT_CURSOR; WRITE NEW ALTERNATE CURSOR
1E57  EB 04                           JMP     SHORT   SACT5   ;

1E59                            SACT4:
1E59  E8 0618 R                       CALL    WRITE_GCURSOR   ; WRITE NEW GRAPHICS CURSOR
1E5C  4B                              DEC     BX              ; SET ALTERNATE CURSOR FLAG
1E5D                            SACT5:
1E5D  88 1E 0348 R                    MOV     AC_PRESENT,BL   ; SET ALTENATE CURSOR FLAG

1E61  C3                              RET

1E62                            SET_ALT_CTYPE   ENDP
                                ;--------------------------------------------
                                ;
                                ;       SET_ALT_CPOS            INT 10H, AH = 82H
                                ;       ------------
                                ;       THIS ROUTINE SETS THE ALTERNATE CURSOR POSITION TO THE
                                ;       NEW X-Y VALUES PASSED
                                ;
                                ;       INPUT   AH = CRT MODE   (MASKED)
                                ;               DX = ROW,COLUMN OF NEW CURSOR
                                ;
                                ;       OUTPUT          NONE
                                ;
                                ;       CALL            WRITE_ALT_CURSOR
                                ;                       WRITE_GCURSOR
                                ;
                                ;       VOLATILE        BX,CX,DI
                                ;
                                ;--------------------------------------------
                                        ASSUME  CS:CODE, DS:DATA

1E62                            SET_ALT_CPOS    PROC    NEAR

1E62  8B 0E 0350 R                    MOV     CX,ACURSOR_MODE ; GET ALTERNATE GRAPHICS CURSOR MODE

1E66  8A 1E 0348 R                    MOV     BL,AC_PRESENT   ; GET ALTERNATE CURSOR PRESENT FLAG
1E6A  0A DB                           OR      BL,BL           ; ALTERNATE CURSOR PRESENT ?
1E6C  74 15                           JZ      SACPOS3         ; NO, SKIP ERASE OLD CURSOR

1E6E  52                              PUSH    DX                      ; SAVE NEW CURSOR POSITION
1E6F  8B 16 034A R                    MOV     DX,ALT_CURSOR_POSN ; GET OLD ALTERNATE CURSOR POSITION
1E73  80 FC 04                        CMP     AH,GRAPHICS             ; GRAPHICS MODE ?
1E76  73 05                           JAE     SACPOS1                 ; YES

1E78  E8 1E9A R                       CALL    WRITE_ALT_CURSOR; WRITE ALTERNATE CURSOR
1E7B  EB 03                           JMP     SHORT SACPOS2

1E7D                            SACPOS1:
1E7D  E8 0618 R                       CALL    WRITE_GCURSOR   ; WRITE ALTERNATE CURSOR IN GRAPHICS
1E80                            SACPOS2:
1E80  5A                              POP     DX              ; RESTORE NEW CURSOR POSITION
1E81  32 DB                           XOR     BL,BL           ; CLEAR ALTERNATE CURSOR FLAG
1E83                            SACPOS3:
1E83  89 16 034A R                    MOV     ALT_CURSOR_POSN,DX ; SET NEW ALTERNATE CURSOR POSITION

1E87  80 FC 04                        CMP     AH,GRAPHICS     ; GRAPHICS MODE ?
1E8A  73 05                           JAE     SACPOS4         ; YES

1E8C  E8 1E9A R                       CALL    WRITE_ALT_CURSOR; WRITE ALTERNATE CURSOR
1E8F  EB 04                           JMP     SHORT SACPOS5

1E91                            SACPOS4:
1E91  E8 0618 R                       CALL    WRITE_GCURSOR   ; WRITE ALTERNATE CURSOR IN GRAPHICS
1E94  4B                              DEC     BX              ; SET ALTERNATE CURSOR FLAG FOR GRAPHICS
1E95                            SACPOS5:
1E95  88 1E 0348 R                    MOV     AC_PRESENT,BL   ; ALTERNATE CURSOR PRESENT
1E99  C3                              RET

1E9A                            SET_ALT_CPOS    ENDP

                                ;--------------------------------------------
                                ;
                                ;       WRITE_ALT_CURSOR
                                ;
                                ;       THIS ROUTINE WRITES THE ALTERNATE CURSOR
                                ;
                                ;       INPUT   BL = ATTRIBUTE TO BE USED AT ALTERNATE CURSOR POSITION
                                ;                    KANJI BIT IS MASKED
                                ;                       (IF BL=0, USE ATTRIBUTE AT REGEN BUFFER)
                                ;               CX = CURSOR MODE
                                ;               DX = ROW,COLUMN POSITION TO WRITE
                                ;
                                ;       OUTPUT  BL = NEW ATTRIBUTE AT ALTERNATE CURSOR POSITION
                                ;                    KANJI BIT IS MASKED
                                ;                       (IF NO CURSOR HAS WRITTEN, BL=0)
                                ;
                                ;       CALL            FIND_POSN
                                ;
                                ;       VOLATILE        BH,DI
                                ;
                                ;--------------------------------------------

1E9A                            WRITE_ALT_CURSOR        PROC    NEAR

1E9A  F6 C5 20                        TEST    CH,CURSOR_DISABLE       ; CURSOR DISABLED ?
1E9D  75 36                           JNZ     WALTC2          ; YES

1E9F  83 F9 11                        CMP     CX,BLOCK_CURSOR ; BLOCK CURSOR ?
1EA2  75 31                           JNE     WALTC2          ; NO
```

```
1EA4  50                        PUSH    AX              ;
1EA5  51                        PUSH    CX              ; SAVE REGISTERS

1EA6  8B C2                     MOV     AX,DX           ; SET ROW/COLUMN POSITION

1EA8  53                        PUSH    BX              ; SAVE ATTRIBUTE
1EA9  32 ED                     XOR     CH,CH           ; SET ACTIVE PAGE TO CX
1EAB  8A 0E 0062 R              MOV     CL,ACTIVE_PAGE  ;
1EAF  E8 0DEC R                 CALL    FIND_POSN       ; DETERMINE LOCATION IN REGEN BUFFER
1EB2  8B FB                     MOV     DI,BX           ; SET OFFSET TO DI
1EB4  5B                        POP     BX              ; RESTORE ATTRIBUTE

1EB5  26: 8B 05                 MOV     AX,ES:[DI]      ; GET CODE/ATTR FROM REGEN
1EB8  0A DB                     OR      BL,BL           ; ATTRIBUTE SPECIFYED ?
1EBA  74 05                     JZ      WALTC1          ; NO

1EBC  80 E4 88                  AND     AH,ZEN2_MASK    ; GET KJ BIT
1EBF  0A E3                     OR      AH,BL           ; SET ATTRIBUTE
1EC1                    WALTC1:
                                                        ;--- REVERSE ATTRIBUTE OF FORE/BACKGROUND
1EC1  8A FC                     MOV     BH,AH           ; SET ATTRIBUTE TO BH FOR REVERSE
1EC3  B1 04                     MOV     CL,4            ; SET SHIFT COUNTER
1EC5  D2 CF                     ROR     BH,CL           ; SWAP NIBBLE

1EC7  80 E7 77                  AND     BH,HAN_MASK     ; STRIP OFF KJ-BIT
1ECA  80 E4 88                  AND     AH,ZEN2_MASK    ; STRIP FORE/BACKGROUND COLOR BIT OFF
1ECD  0A E7                     OR      AH,BH           ; SET REVERSED COLOR BIT TO AH

1ECF  AB                        STOSW                   ; WRITE CODE/ATTR TO REGEN
1ED0  8A DF                     MOV     BL,BH           ; SET NEW ATTRIBUTE

1ED2  59                        POP     CX              ;
1ED3  58                        POP     AX              ; RESTORE REGISTERS
1ED4  C3                        RET

1ED5                    WALTC2:
1ED5  32 DB                     XOR     BL,BL           ; CLEAR FLAG
1ED7  C3                        RET

1ED8                    WRITE_ALT_CURSOR        ENDP
                        ;-----------------------------------------------
                        ;
                        ;       READ_ALT_CURSOR         INT 10H, AH = 83H
                        ;       ---------------
                        ;       THIS ROUTINE READS THE ALTENATE CURSOR POSITION
                        ;
                        ;       INPUT   NONE
                        ;
                        ;       OUTPUT  DX = ROW, COLUMN OF THE ALTERNATE CURSOR POSITION
                        ;               CX = CURRENT ALTERNATE CURSOR MODE
                        ;
                        ;       CALL    NONE
                        ;
                        ;       VOLATILE        BX
                        ;
                        ;-----------------------------------------
                                ASSUME  CS:CODE, DS:DATA
1ED8                    READ_ALT_CURSOR PROC    NEAR

1ED8  55                        PUSH    BP              ; SAVE BP
1ED9  8B 16 034A R              MOV     DX,ALT_CURSOR_POSN
1EDD  8B 0E 0350 R              MOV     CX,ACURSOR_MODE ; GET ALTERNATE CURSOR MODE

1EE1  8B EC                     MOV     BP,SP           ; SET FRAME POINTER
1EE3  89 56 0C                  MOV     [BP+F_DX],DX    ; SET RETURN DX
1EE6  89 4E 0A                  MOV     [BP+F_CX],CX    ; SET RETURN CX

1EE9  5D                        POP     BP              ; RESTORE BP
1EEA  C3                        RET

1EEB                    READ_ALT_CURSOR ENDP
                        ;-----------------------------------------
                        ;
                        ;       READ_AC_CURRENT FOR KANA-KANJI CONVERSION
                        ;       ---------------         INT 10H, AH = 88H
                        ;
                        ;       THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE
                        ;       CURRENT CURSOR POSITION AND RETURNS THEM TO THE CALLER
                        ;
                        ;       INPUT   AH =    CURRENT CRT MODE (MASKED)
                        ;               DS =    DATA SEGMENT
                        ;               ES =    REGEN SEGMENT
                        ;
                        ;       OUTPUT  AL =    CHAR READ
                        ;               AH =    ATTRIBUTE READ
                        ;
                        ;       CALL    READ_AC_CURRENT
                        ;
                        ;-----------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM
                        ;K_READ_AC_CURRENT      PROC    NEAR
                        ;
                        ;       MOV     BH,ACTIVE_PAGE  ; SET DISPLAY PAGE
                        ;
                        ;       JMP     READ_AC_CURRENT ; HANDLE BY NATIVE ROUTINE
                        ;
                        ;K_READ_AC_CURRENT      ENDP
                        ;-----------------------------------------
                        ;
                        ;       WRITE_AC_CURRENT FOR KANA-KANJI CONVERSION
                        ;       ----------------        INT 10H, AH = 89H
```

```
                        ;       THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER AT
                        ;       THE CURRENT CURSOR POSITION
                        ;
                        ;       INPUT
                        ;               AH = CURRENT CRT MODE (MASKED)
                        ;               AL = CHAR TO WRITE
                        ;               BL = ATTRIBUTE OF CHAR TO WRITE
                        ;               DX = ROW/COLUMN FOR WRITE CHARACTER
                        ;               DS = DATA SEGMENT
                        ;               ES = REGEN SEGMENT
                        ;
                        ;       OUTPUT
                        ;               NONE
                        ;
                        ;       CALL    WRITE_AC_CURRENT
                        ;
                        ;-----------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

1EEB                    K_WRITE_AC_CURRENT      PROC    NEAR

1EEB  80 FC 04                  CMP     AH,GRAPHICS     ; TEXT MODE ?
1EEE  72 0A                     JB      KWAC1           ; YES

1EF0  F6 C3 80                  TEST    BL,XOR_BIT      ; XOR WRITE BIT ON ?
1EF3  75 05                     JNZ     KWAC1           ; YES

1EF5  C6 06 0348 R 00           MOV     AC_PRESENT,FALSE; CLEAR ALTERNATE CURSOR PRESENT FLAG
1EFA                    KWAC1:

1EFA  E9 0F32 R                 JMP     WRITE_AC_CURRENT; HANDLE BY NATIVE ROUTINE
1EFD                    K_WRITE_AC_CURRENT      ENDP
                        ;-----------------------------------------
                        ;
                        ;       WRITE_C_CURRENT FOR KANA-KANJI CONVERSION
                        ;       ---------------         INT 10H, AH = 8AH
                        ;
                        ;       THIS ROUTINE WRITES THE CHARACTER AT
                        ;       THE CURRENT CURSOR POSITION
                        ;
                        ;       INPUT
                        ;               AH = CURRENT CRT MODE (MASKED)
                        ;               AL = CHAR TO WRITE
                        ;               BL = COLOR OF CHAR (GRAPHICS)
                        ;               DX = ROW/COLUMN FOR WRITE CHARACTER
                        ;               DS = DATA SEGMENT
                        ;               ES = REGEN SEGMENT
                        ;
                        ;       OUTPUT
                        ;               NONE
                        ;
                        ;       CALL    WRITE_C_CURRENT
                        ;
                        ;-----------------------------------------
                                ASSUME  CS:CODE, DS:DATA, ES:VIDEO_RAM

1EFD                    K_WRITE_C_CURRENT       PROC    NEAR

1EFD  80 FC 04                  CMP     AH,GRAPHICS     ; TEXT MODE ?
1F00  72 0A                     JB      KWC1            ; YES

1F02  F6 C3 80                  TEST    BL,XOR_BIT      ; XOR WRITE BIT ON ?
1F05  75 05                     JNZ     KWC1            ; YES

1F07  C6 06 0348 R 00           MOV     AC_PRESENT,FALSE; CLEAR ALTERNATE CURSOR PRESENT FLAG
1F0C                    KWC1:

1F0C  E9 0F56 R                 JMP     WRITE_C_CURRENT ; HANDLE BY NATIVE ROUTINE

1F0F                    K_WRITE_C_CURRENT       ENDP
                        ;-----------------------------------------
                        ;
                        ;       WRITE_TTY FOR KANA-KANJI CONVERSION
                        ;       ---------               INT 10H, AH =    (8EH)
                        ;
                        ;       THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
                        ;        VIDEO CARD.  THE INPUT CHARACTER IS WRITTEN TO THE CURRENT ALTERNATE
                        ;        CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
                        ;
                        ;       INPUT
                        ;               AH = CURRENT CRT MODE   (MASKED)
                        ;               AL = CHARACTER TO BE WRITTEN
                        ;                  NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE
                        ;                  HANDLED AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
                        ;               BH = DISPLAY PAGE
                        ;               BL = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A
                        ;                       GRAPHICS MODE
                        ;
                        ;       OUTPUT          NONE
                        ;
                        ;       CALL            VIDEO
                        ;
                        ;       VOLATILE        BH, CX, DX
                        ;
                        ;-----------------------------------------
                                ASSUME  CS:CODE, DS:DATA

1F0F                    K_WRITE_TTY     PROC    NEAR

1F0F  50                        PUSH    AX              ; SAVE REGISTER
1F10  8B 16 034A R              MOV     DX,ALT_CURSOR_POSN      ; GET CURSOR POSITION
                                                        ;--- DX NOW HAS THE CURRENT CURSOR POSITION
```

```
1F14  F7 06 0344 R FFFF          TEST    K_TTY_1ST_CHAR,TRUE; 1ST BYTE OF 2 BYTE CODE HAS BEEN SET ?
1F1A  75 17                      JNZ     KWTY2               ; YES

1F1C  3C 80                      CMP     AL,080H             ; 1ST BYTE OF 2 BYTE CODE ?
1F1E  76 54                      JBE     KWTY6               ; NO, GO TO WRITE ONE CHARACTER
1F20  3C FD                      CMP     AL,0FDH             ;
1F22  73 50                      JAE     KWTY6               ; NO, GO TO WRITE ONE CHARACTER
1F24  3C A0                      CMP     AL,0A0H             ;
1F26  72 04                      JB      KWTY1               ; YES
1F28  3C DF                      CMP     AL,0DFH             ;
1F2A  76 48                      JBE     KWTY6               ; NO, GO TO WRITE ONE CHARACTER

1F2C                        KWTY1:                           ;--- CHARACTER IS 1ST BYTE OF 2 BYTE CODE
1F2C  8A E3                      MOV     AH,BL               ; SET COLOR
1F2E  A3 0344 R                  MOV     K_TTY_1ST_CHAR,AX; SAVE 1ST BYTE OF 2 BYTE CODE AND SET FLAG
1F31  EB 54                      JMP     SHORT KWTY7         ; RETURN; WRITE PROCESS IS PENDING

1F33                        KWTY2:                           ;--- CHARACTER MUST BE 2ND BYTE OF 2 BYTE CODE
1F33  3C 40                      CMP     AL,040H             ; 2ND BYTE OF 2 BYTE CODE ?
1F35  72 08                      JB      KWTY3               ; NO
1F37  3C FC                      CMP     AL,0FCH             ;
1F39  77 04                      JA      KWTY3               ; NO
1F3B  3C 7F                      CMP     AL,07FH             ;
1F3D  75 08                      JNE     KWTY4               ; YES

1F3F                        KWTY3:                           ;--- IGNORE 1ST BYTE
1F3F  C7 06 0344 R 0000          MOV     K_TTY_1ST_CHAR,0; RESET FLAG OF 1ST BYTE
1F45  EB 2D                      JMP     SHORT KWTY6         ; GO TO WRITE ONE BYTE

1F47                        KWTY4:                           ;--- WRITE 2 BYTE CODE
1F47  8B 0E 004A R               MOV     CX,CRT_COLS         ; CHECK COLUMN BOUNDARY
1F4B  49                         DEC     CX                  ; ADJUST FOR COMPARE
1F4C  3A D1                      CMP     DL,CL               ; IS COLUMN AT END OF LINE ?
1F4E  72 08                      JB      KWTY5               ; NO

                                                             ;--- IN CASE OF WRITE 2 BYTE CHAR ON THE END
                                                             ; LINE, IT IS IGNORED
1F50  C7 06 0344 R 0000          MOV     K_TTY_1ST_CHAR,0; CLEAR FIRST BYTE
1F56  EB 2F                      JMP     SHORT   KWTY7       ; TERMINATE

1F58                        KWTY5:                           ;--- WRITE 1ST BYTE OF 2 BYTE CODE
1F58  50                         PUSH    AX                  ; SAVE 2ND BYTE OF 2 BYTE CODE
1F59  53                         PUSH    BX                  ; SAVE COLOR
1F5A  A1 0344 R                  MOV     AX,K_TTY_1ST_CHAR ; GET 1ST BYTE OF 2 BYTE CODE
1F5D  8A DC                      MOV     BL,AH               ; SET COLOR
1F5F  B4 8A                      MOV     AH,8AH              ; (AH)=8AH: WRITE CHARACTER AT CURRENT CURSOR
1F61  B9 0001                    MOV     CX,1                ; WRITE ONE CHARACTER
1F64  CD 10                      INT     VIDEO               ; VIDEO I/O
1F66  5B                         POP     BX                  ; RESTORE COLOR

1F67  C7 06 0344 R 0000          MOV     K_TTY_1ST_CHAR,0; RESET FLAG OF 1ST BYTE
1F6D  FE C2                      INC     DL                  ; INCREMENT ROW

1F6F  B4 82                      MOV     AH,82H              ; FUNCTION OF SET CURSOR POSITION
1F71  CD 10                      INT     VIDEO               ; DO IT
1F73  58                         POP     AX                  ; RESTORE 2ND BYTE
1F74                        KWTY6:                           ;--- WRITE THE CHAR TO THE SCREEN
1F74  B4 8A                      MOV     AH,8AH              ; WRITE CHAR ONLY
1F76  B9 0001                    MOV     CX,1                ; ONLY ONE CHAR
1F79  CD 10                      INT     VIDEO               ; WRITE THE CHAR
                                                             ;--- POSITION THE CURSOR FOR NEXT CHAR
1F7B  FE C2                      INC     DL
1F7D  3A 16 004A R               CMP     DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
1F81  73 04                      JAE     KWTY7               ; SKIP CURSOR MOVE

1F83  B4 82                      MOV     AH,82H
1F85  CD 10                      INT     VIDEO               ; ADVANCE CURSOR POSITION
1F87                        KWTY7:
1F87  58                         POP     AX                  ; RESTORE THE CHARACTER

1F88  C3                         RET                         ; RETURN TO CALLER

1F89                        K_WRITE_TTY     ENDP

2000                             ORG     2000H                       ; ADJUST SIZE TO 8K

2000                        CODE    ENDS
                                    END
```

```
                        ;--- INT 11 --------------------------------------
                        ; EQUIPMENT DETERMINATION
                        ;       THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
                        ;       DEVICES ARE ATTACHED TO THE SYSTEM.
                        ; INPUT
                        ;       NO REGISTERS
                        ;       THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
                        ;        DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
                        ;       PORT 62 (0->3) = LOW ORDER BYTE OF EQUIPMENT
                        ;       PORT 3FA = INTERRUPT ID REGISTER OF 8250
                        ;               BITS 7-3 ARE ALWAYS 0
                        ;       PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
                        ;               CAN BE READ AS WELL AS WRITTEN
                        ;  OUTPUT
                        ;       (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
                        ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED
                        ;       BIT 13 = RESERVED
                        ;       BIT 12 = GAME I/O ATTACHED
                        ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
                        ;       BIT 8  0 = DMA CHIP PRESENT ON SYSTEM, 1 = NO DMA ON SYSTEM
                        ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES
                        ;               00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
                        ;
                        ;       BIT 5,4 = APPLICATION MODE FLAG ( 1: EXTENSION , 0: NATIVE )
                        ;
                        ;                       01 - NATIVE MODE
                        ;                       10 - EXTENSION MODE
                        ;                       11 - (RESERVED)
                        ;
                        ;       BIT 3,2 = PLANAR RAM SIZE
                        ;       BIT 1 RESERVED
                        ;       BIT 0 = 1 (IPL DISKETTE INSTALLED)
                        ;       NO OTHER REGISTERS AFFECTED
                        ;-------------------------------------------------
0000                            ASSUME  CS:CODE,DS:DATA
0000  FB                EQUIPMENT      PROC    FAR
0001  1E                      STI                     ; INTERRUPTS BACK ON
0002  B8 ---- R                PUSH    DS              ; SAVE SEGMENT REGISTER
0005  8E D8                    MOV     AX,DATA         ; ESTABLISH ADDRESSING
0007  A1 0010 R                MOV     DS,AX
000A  1F                       MOV     AX,EQUIP_FLAG   ; MOVE EQUIPMENT FLAG
000B  CF                       POP     DS              ; RECOVER SEGMENT
000C                           IRET                    ; RETURN TO CALLER
                        EQUIPMENT      ENDP
                        ;--- INT 12 --------------------------------
                        ; MEMORY_SIZE_DETERMINE
                        ; INPUT
                        ;       NO REGISTERS
                        ;       THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
                        ; OUTPUT
                        ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
                        ;-----------------------------------------------
000C                            ASSUME  CS:CODE,DS:DATA
000C  FB                MEMORY_SIZE_DETERMINE   PROC    FAR
000D  1E                      STI                     ; INTERRUPTS BACK ON
000E  B8 ---- R                PUSH    DS              ; SAVE SEGMENT
0011  8E D8                    MOV     AX,DATA         ; ESTABLISH ADDRESSING
0013  A1 0013 R                MOV     DS,AX
0016  1F                       MOV     AX,MEMORY_SIZE  ; GET VALUE
0017  CF                       POP     DS              ; RECOVER SEGMENT
0018                           IRET                    ; RETURN TO CALLER
                        MEMORY_SIZE_DETERMINE   ENDP
                        ;-- INT 13 ------------------------------------------------------------
                        ; DISKETTE I/O
                        ;       THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
                        ; INPUT
                        ;       (AH)=0  RESET DISKETTE SYSTEM
                        ;               HARD RESET TO NEC, PREPARE COMMAND, RECAL REQD ON
                        ;               ALL DRIVES
                        ;       (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)
                        ;               DISKETTE_STATUS FROM LAST OP'N IS USED
                        ;       REGISTERS FOR READ/WRITE/VERIFY/FORMAT
                        ;       (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
                        ;                           (BIT 6  :1 80 TRACK DISKETTE ACCESS)
                        ;                           (       :0 40 TRACK ACCESS)
                        ;       (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
                        ;       (CH) - TRACK NUMBER (0-79, NOT VALUE CHECKED)
                        ;       (CL) - SECTOR NUMBER (1-9, NOT VALUE CHECKED, NOT USED FOR
                        ;              FORMAT)
                        ;       (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT
                        ;              USED FOR FORMAT, HOWEVER, CANNOT BE ZERO!!!)
                        ;       (ES:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY)
                        ;
                        ;       (AH)=2  READ THE DESIRED SECTORS INTO MEMORY
                        ;       (AH)=3  WRITE THE DESIRED SECTORS FROM MEMORY
                        ;       (AH)=4  VERIFY THE DESIRED SECTORS
                        ;       (AH)=5  FORMAT THE DESIRED TRACK
                        ;               FOR THE FORMAT OPERATION, THE BUFFER POINTER
                        ;               (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED
                        ;               ADDRESS FIELDS FOR THE TRACK.  EACH FIELD IS
                        ;               COMPOSED OF 4 BYTES, (C,H,R,N), WHERE
                        ;               C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
                        ;               N= NUMBER OF BYTES PER SECTOR (00=128, 01=256,
                        ;               02=512, 03=1024,). THERE MUST BE ONE ENTRY FOR
                        ;               EVERY SECTOR ON THE TRACK. THIS INFORMATION IS USED
                        ;               TO FIND THE REQUESTED SECTOR DURING READ/WRITE
                        ;               ACCESS.
                        ; DATA VARIABLE -- DISK_POINTER
                        ;     DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
                        ; OUTPUT
                        ;       AH = STATUS OF OPERATION
                        ;               STATUS BITS ARE DEFINED IN THE EQUATES FOR
                        ;               DISKETTE_STATUS VARIABLE IN THE DATA SEGMENT OF
                        ;               THIS MODULE
```

```
                        ;       CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)
                        ;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
                        ;       FOR READ/WRITE/VERIFY
                        ;               DS,BX,DX,CH,CL PRESERVED
                        ;               AL = NUMBER OF SECTORS ACTUALLY READ
                        ;               **** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
                        ;       NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE
                        ;               APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN
                        ;               RETRY THE OPERATION. ON READ ACCESSES, NO MOTOR
                        ;               START DELAY IS TAKEN, SO THAT THREE RETRIES ARE
                        ;               REQUIRED ON READS TO ENSURE THAT THE PROBLEM IS NOT
                        ;               DUE TO MOTOR START-UP.
                        ;
                        ;--------------------------------------------------------
0018                            ASSUME  CS:CODE,DS:DATA,ES:DATA
0018                    DISKETTE_IO     PROC    FAR
0018  FB                       STI                     ; INTERRUPTS BACK ON
0019  06                       PUSH    ES              ; SAVE ES
001A  50                       PUSH    AX              ; ALLOCATE ONE WORD OF STORAGE FOR
                                                       ; TIMER1 INITIAL VALUE
001B  50                       PUSH    AX              ; ALLOCATE ONE WORD ON STACK FOR
                                                       ; USE IN PROCS ENABLE AND DISABLE.
                                                       ;  WILL HOLD 8259 MASK.
001C  50                       PUSH    AX              ; SAVE COMMAND AND N_SECTORS
001D  53                       PUSH    BX              ; SAVE ADDRESS
001E  51                       PUSH    CX
001F  1E                       PUSH    DS              ; SAVE SEGMENT REGISTER VALUE
0020  56                       PUSH    SI              ; SAVE ALL REGISTERS DURING
                                                       ; OPERATION
0021  57                       PUSH    DI
0022  55                       PUSH    BP
0023  52                       PUSH    DX
0024  8B EC                    MOV     BP,SP           ; SET UP POINTER TO HEAD PARM
0026  E8 0000 E                CALL    DDS             ; SET DS=DATA
0029  E8 004F R                CALL    J1              ; CALL THE REST TO ENSURE DS
                                                       ; RESTORED
002C  B3 04                    MOV     BL,4            ; GET THE MOTOR WAIT PARAMETER
002E  E8 0331 R                CALL    GET_PARM
0031  88 26 0040 R             MOV     MOTOR_COUNT,AH  ; SET THE TIMER COUNT FOR THE MOTOR
0035  8A 26 0041 R             MOV     AH,DISKETTE_STATUS      ; GET STATUS OF OPERATION
0039  88 66 0F                 MOV     [BP+15],AH      ; RETURN STATUS IN AL
003C  5A                       POP     DX              ; RESTORE ALL REGISTERS
003D  5D                       POP     BP
003E  5F                       POP     DI
003F  5E                       POP     SI
0040  1F                       POP     DS
0041  59                       POP     CX
0042  5B                       POP     BX              ; RECOVER OFFSET
0043  58                       POP     AX
0044  83 C4 04                 ADD     SP,4            ; DISCARD DUMMY SPACE FOR 8259 MASK
0047  07                       POP     ES              ; RECOVER SEGMENT
0048  80 FC 01                 CMP     AH,1            ; SET THE CARRY FLAG TO INDICATE
004B  F5                       CMC                     ;  SUCCESS OR FAILURE
004C  CA 0002                  RET     2               ; THROW AWAY SAVED FLAGS
004F                    DISKETTE_IO     ENDP
004F                    J1      PROC    NEAR
004F  8A F0                    MOV     DH,AL           ; SAVE # SECTORS IN DH
0051  80 26 003F R 7F          AND     MOTOR_STATUS,07FH       ; INDICATE A READ OPERATION
0056  80 E2 3F                 AND     DL,OFF_DBL_TRK  ; RESET FOR 80 TRACK PARM FLAG  JX

0059  0A E4                    OR      AH,AH           ; AH=0
005B  74 27                    JZ      DISK_RESET
005D  FE CC                    DEC     AH              ; AH=1
005F  74 74                    JZ      DISK_STATUS
0061  C6 06 0041 R 00          MOV     DISKETTE_STATUS,0 ; RESET THE STATUS INDICATOR
0066  80 FA 03                 CMP     DL,DRV_RANGE    ; TEST FOR DRIVE IN 0-3 RANGE   JX
0069  77 13                    JA      J3              ; ERROR IF ABOVE
006B  FE CC                    DEC     AH              ; AH=2
006D  74 6D                    JZ      DISK_READ
006F  FE CC                    DEC     AH              ; AH=3
0071  75 03                    JNZ     J2              ; TEST_DISK_VERF
0073  E9 00FF R                JMP     DISK_WRITE
0076                    J2:                             ; TEST_DISK_VERF
0076  FE CC                    DEC     AH              ; AH=4
0078  74 62                    JZ      DISK_VERF
007A  FE CC                    DEC     AH              ; AH=5
007C  74 62                    JZ      DISK_FORMAT
007E                    J3:                             ; BAD_COMMAND
007E  C6 06 0041 R 01          MOV     DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS
                                                       ; TRANSFERRED
0083  C3                       RET                     ; UNDEFINED OPERATION
0084                    J1      ENDP
                        ;------ RESET THE DISKETTE SYSTEM
0084                    DISK_RESET      PROC    NEAR
0084  BA 00F2                 MOV     DX,HEC_CTL      ; ADAPTER CONTROL PORT
0087  FA                       CLI                     ; NO INTERRUPTS
0088  A0 003F R               MOV     AL,MOTOR_STATUS ; FIND OUT IF MOTOR IS RUNNING
008B  24 0F                    AND     AL,DRV_SUPPORT  ; DRIVE BITS
008D  EE                       OUT     DX,AL           ; RESET THE ADAPTER
008E  C6 06 003E R 00          MOV     SEEK_STATUS,0   ; SET RECAL REQUIRED ON ALL DRIVES
0093  C6 06 0041 R 00          MOV     DISKETTE_STATUS,0 ; SET OK STATUS FOR DISKETTE
0098  0C 80                    OR      AL,FDC_RESET    ; TURN OFF RESET
009A  EE                       OUT     DX,AL
009B  FB                       STI                     ; TURN OFF THE RESET
009C  BE 00BC R                MOV     SI,OFFSET J4_2  ; REENABLE THE INTERRUPTS
009F  56                       PUSH    SI              ; DUMMY RETURN FOR
                                                       ; PUSH RETURN IF ERROR
00A0  B9 0010                  MOV     CX,10H          ; IN HEC_OUTPUT
                                                       ; NUMBER OF SENSE INTERRUPTS TO
00A3  B4 08                    J4_0:   MOV     AH,08H  ; ISSUE
                                                       ; COMMAND FOR SENSE INTERRUPT
                                                       ; STATUS
```

```
00A5  E8 0307 R                 CALL    NEC_OUTPUT      ; OUTPUT  THE SENSE INTERRUPT
                                                        ; STATUS
00A8  E8 03A9 R                 CALL    RESULTS         ; GET STATUS FOLLOWING COMPLETION
                                                        ; OF RESET
00AB  A0 0042 R                 MOV     AL,NEC_STATUS   ; IGNORE ERROR RETURN AND DO OWN
                                                        ; TEST
00AE  3C C0                     CMP     AL,0C0H         ; TEST FOR DRIVE READY TRANSITION
00B0  74 12                     JZ      J7              ; EVERYTHING OK
00B2  E2 EF                     LOOP    J4_0            ; RETRY THE COMMAND
00B4  80 0E 0041 R 20   J4_1:   OR      DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
00B9  5E                        POP     SI
00BA  EB 18                     JMP     SHORT J8

00BC  BE 00BC R         J4_2:   MOV     SI,OFFSET J4_2  ; NEC_OUTPUT FAILED, RETRY THE
                                                        ; SENSE INTERRUPT
00BF  56                        PUSH    SI              ; OFFSET OF BAD RETURN IN
                                                        ; NEC_OUTPUT
00C0  E2 E1                     LOOP    J4_0            ; RETRY
00C2  EB F0                     JMP     SHORT J4_1
                        ;------ SEND SPECIFY COMMAND TO NEC
00C4  5E                J7:     POP     SI              ; GET RID OF DUMMY ARGUMENT
00C5  B4 03                     MOV     AH,03H          ; SPECIFY COMMAND
00C7  E8 0307 R                 CALL    NEC_OUTPUT      ; OUTPUT THE COMMAND
00CA  B3 01                     MOV     BL,1            ; STEP RATE TIME AND HEAD UNLOAD
00CC  E8 0331 R                 CALL    GET_PARM        ; OUTPUT TO THE NEC CONTROLLER
00CF  B3 03                     MOV     BL,3            ; PARM1 HEAD LOAD AND NO DMA
00D1  E8 0331 R                 CALL    GET_PARM        ;  TO THE NEC CONTROLLER
00D4                    J8:                             ; RESET_RET
00D4  C3                        RET                     ; RETURN TO CALLER
00D5                    DISK_RESET      ENDP
                        ;------ DISKETTE STATUS ROUTINE
00D5                    DISK_STATUS     PROC    NEAR

00D5  A0 0041 R                 MOV     AL,DISKETTE_STATUS
00D8  88 46 0E                  MOV     BYTE PTR[BP+14],AL ; PUT STATUS ON STACK, IT WILL
                                                        ; POP IN AL
00DB  C3                        RET
00DC                    DISK_STATUS     ENDP
                        ;------ DISKETTE VERIFY
00DC                    DISK_VERF       LABEL   NEAR
                        ;------ DISKETTE READ
00DC                    DISK_READ       PROC    NEAR
00DC  B4 46             J9:                             ; DISK_READ_CONT
                                MOV     AH,046H         ; SET UP READ COMMAND FOR NEC
00DE  EB 26                             ;               ; CONTROLLER
00E0                            JMP     SHORT RW_OPN    ; GO DO THE OPERATION
                        DISK_READ       ENDP
                        ;------ DISKETTE FORMAT
00E0                    DISK_FORMAT     PROC    NEAR
00E0  80 0E 003F R 80           OR      MOTOR_STATUS,80H ; INDICATE A WRITE OPERATION
00E5  B4 4D                     MOV     AH,04DH         ; ESTABLISH THE FORMAT COMMAND
00E7  EB 1D                     JMP     SHORT RW_OPN    ; DO THE OPERATION
00E9                    J10:                            ; CONTINUATION OF RW_OPN FOR FMT
00E9  B3 07                     MOV     BL,7            ; GET THE
00EB  E8 0331 R                 CALL    GET_PARM        ;  BYTES/SECTOR VALUE TO NEC
00EE  B3 09                     MOV     BL,9            ; GET THE
00F0  E8 0331 R                 CALL    GET_PARM        ;  SECTORS/TRACK VALUE TO NEC
00F3  B3 0F                     MOV     BL,15           ; GET THE
00F5  E8 0331 R                 CALL    GET_PARM        ;  GAP LENGTH VALUE TO NEC
00F8  BB 0011                   MOV     BX,17           ; GET THE FILLER BYTE
00FB  53                        PUSH    BX              ; SAVE PARAMETER INDEX ON STACK
00FC  E9 018F R                 JMP     J16             ;  TO THE CONTROLLER
00FF                    DISK_FORMAT     ENDP
                        ;------ DISKETTE WRITE ROUTINE
00FF                    DISK_WRITE      PROC    NEAR
00FF  80 0E 003F R 80           OR      MOTOR_STATUS,80H ; INDICATE A WRITE OPERATION
0104  B4 45                     MOV     AH,045H         ; NEC COMMAND TO WRITE TO DISKETTE
                        DISKIO1.INC

0106                    DISK_WRITE      ENDP
                        ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
                        ;--------------------------------------
                        ; RW_OPN
                        ;       THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
                        ;--------------------------------------
0106                    RW_OPN  PROC    NEAR
0106  50                       PUSH    AX              ; SAVE THE COMMAND
                        ;------ TURN ON THE MOTOR AND SELECT THE DRIVE
0107  51                      PUSH    CX              ; SAVE THE T/S PARMS
0108  FA                      CLI                     ; NO INTERRUPTS WHILE DETERMINING
                                                        ; MOTOR STATUS
0109  C6 06 0040 R FF         MOV     MOTOR_COUNT,0FFH ; SET LARGE COUNT DURING OPERATION
010E  E8 044E R                 CALL    GET_DRIVE       ; GET THE DRIVE PARAMETER FROM THE
                                                        ; STACK
0111  84 06 003F R              TEST    MOTOR_STATUS,AL ; TEST MOTOR FOR OPERATING
0115  75 1F                     JNZ     J14             ; IF RUNNING, SKIP THE WAIT
0117  80 26 003F R F0           AND     MOTOR_STATUS,0F0H ; TURN OFF RUNNING DRIVE
011C  08 06 003F R              OR      MOTOR_STATUS,AL         ; TURN ON THE CURRENT MOTOR
0120  FB                        STI                     ; INTERRUPTS BACK ON
0121  0C 80                    OR      AL,FDC_RESET    ; NO RESET.  TURN ON MOTOR
0123  E6 F2                    OUT     NEC_CTL,AL
                        ;------ WAIT FOR MOTOR BOTH READ AND WRITE
0125  B3 14                     MOV     BL,20           ; GET MOTOR START TIME
0127  E8 0331 R                 CALL    GET_PARM
012A  0A E4                     OR      AH,AH           ; TEST FOR NO WAIT
012C                    J12:                            ; TEST_WAIT_TIME
012C  74 08                     JZ      J14             ; EXIT WITH TIME EXPIRED
012E  2B C9                     SUB     CX,CX           ; SET UP 1/8 SECOND LOOP TIME
0130  E2 FE             J13:    LOOP    J13             ; WAIT FOR THE REQUIRED TIME
0132  FE CC                     DEC     AH              ; DECREMENT TIME VALUE
0134  EB F6                     JMP     J12             ; ARE WE DONE YET
0136                    J14:                            ; MOTOR_RUNNING
0136  FB                       STI                     ; INTERRUPTS BACK ON FOR BYPASS
                                                        ; WAIT
```

```
0137  59                        POP     CX
                        ;------ DO THE SEEK OPERATION
0138  E8 045D R                  CALL    SEEK            ; MOVE TO CORRECT TRACK
013B  58                        POP     AX              ; RECOVER COMMAND
013C  8A FC                     MOV     BH,AH           ; SAVE COMMAND IN BH
013E  B6 00                     MOV     DH,0            ; SET NO SECTORS READ IN CASE OF
                                                        ; ERROR
0140  73 03                     JNC     J14_1           ; IF NO ERROR CONTINUE, JUMP AROUND
                                                        ; JMP
0142  E9 0299 R                 JMP     J17             ; CARRY SET JUMP TO MOTOR WAIT
0145  BE 0299 R         J14_1: MOV     SI,OFFSET J17   ; DUMMY RETURN ON STACK FOR
                                                        ; NEC_OUTPUT
0148  56                        PUSH    SI              ;  SO THAT IT WILL RETURN TO MOTOR
                                                        ; OFF LOCATION
                        ;------ SEND OUT THE PARAMETERS TO THE CONTROLLER
0149  E8 0307 R                 CALL    NEC_OUTPUT      ; OUTPUT THE OPERATION COMMAND
014C  8A 66 01                  MOV     AH,[BP+1]       ; GET THE CURRENT HEAD NUMBER
014F  D0 E4                     SAL     AH,1            ; MOVE IT TO BIT 2

0151  D0 E4                     SAL     AH,1
0153  80 E4 04                  AND     AH,4            ; ISOLATE THAT BIT
0156  0A E2                     OR      AH,DL           ; OR IN THE DRIVE NUMBER
0158  E8 0307 R                 CALL    NEC_OUTPUT
                        ;------ TEST FOR FORMAT COMMAND
015B  80 FF 4D                  CMP     BH,04DH         ; IS THIS A FORMAT OPERATION?
015E  75 02                     JNE     J15             ; NO. CONTINUE WITH R/W/V
0160  EB 87                     JMP     J10             ; IF SO, HANDLE SPECIAL
0162  8A E5              J15:  MOV     AH,CH           ; CYLINDER NUMBER
0164  E8 0307 R                 CALL    NEC_OUTPUT
0167  8A 66 01                  MOV     AH,[BP+1]       ; HEAD NUMBER FROM STACK
016A  E8 0307 R                 CALL    NEC_OUTPUT
016D  8A E1                     MOV     AH,CL           ; SECTOR NUMBER
016F  E8 0307 R                 CALL    NEC_OUTPUT
0172  B3 07                     MOV     BL,7            ; BYTES/SECTOR PARM FROM BLOCK
0174  E8 0331 R                 CALL    GET_PARM        ;  TO THE NEC
0177  B3 08                     MOV     BL,8            ; EOT PARM FROM BLOCK
0179  E8 0331 R                 CALL    GET_PARM        ;  RETURNED IN AH
017C  02 4E 0E                  ADD     CL,[BP+14]      ; ADD CURRENT SECTOR TO NUMBER IN
                                                        ; TRANSFER
017F  FE C9                     DEC     CL              ; CURRENT_SECTOR + N_SECTORS - 1
0181  8A E1                     MOV     AH,CL           ; EOT PARAMETER IS THE CALCULATED
                                                        ; ONE
0183  E8 0307 R                 CALL    NEC_OUTPUT
0186  B3 0B                     MOV     BL,11           ; GAP LENGTH PARM FROM BLOCK
0188  E8 0331 R                 CALL    GET_PARM        ;  TO THE NEC
018B  BB 000D                   MOV     BX,13           ; DTL PARM FROM BLOCK
018E  53                        PUSH    BX              ; SAVE INDEX TO DISK PARAMETER ON
                                                        ; STACK
018F  FC                 J16:   CLD                     ; FORWARD DIRECTION
                        ;------ START TIMER1 WITH INITIAL VALUE OF FFFF
0190  B0 70                     MOV     AL,01110000B    ; SELECT TIMER1,LSB-MSB, MODE 0,
                                                        ; BINARY COUNTER
0192  E6 43                     OUT     TIM_CTL,AL      ; INITIALIZE THE COUNTER
0194  50                        PUSH    AX
0195  58                        POP     AX              ; ALLOW ENOUGH TIME FOR THE 8253 TO
                                                        ; INITIALIZE ITSELF
0196  B0 FF                     MOV     AL,0FFH         ; INITIAL COUNT VALUE FOR THE 8253
0198  E6 41                     OUT     TIMER+1,AL      ; OUTPUT LEAST SIGNIFICANT BYTE
019A  50                        PUSH    AX
019B  58                        POP     AX              ; WAIT
019C  E6 41                     OUT     TIMER+1,AL      ; OUTPUT MOST SIGNIFACNT BYTE
                        ;------INITIALIZE CX FOR JUMP AFTER LAST PARAMETER IS PASSED TO NEC
019E  8A 46 0F                  MOV     AL,[BP+15]      ; RETRIEVE COMMAND PARAMETER
01A1  A8 01                     TEST    AL,01H          ; IS THIS AN ODD NUMBERED FUNCTION?
01A3  74 05                     JZ      J16_1           ; JUMP IF NOT ODD NUMBERED
01A5  B9 0210 R                 MOV     CX,OFFSET WRITE_LOOP
01A8  EB 0C                     JMP     SHORT J16_3
01AA  3C 02              J16_1: CMP     AL,2            ; IS THIS A READ?
01AC  75 05                     JNZ     J16_2           ; JUMP IF VERIFY
01AE  B9 01FC R                 MOV     CX,OFFSET READ_LOOP
01B1  EB 03                     JMP     SHORT J16_3
01B3  B9 01E2 R          J16_2: MOV     CX,OFFSET VERIFY_LOOP

                        ;------FINISH INITIALIZATION
01B6                     J16_3:
                        ;--------------------------------------------------------------
                        ;***NOTE***
                        ;ALL INTERRUPTS ARE ABOUT TO BE DISABLED.  THERE IS A POTENTIAL
                        ;       THAT THIS TIME PERIOD WILL BE LONG ENOUGH TO MISS TIME OF
                        ;       DAY INTERRUPTS.  FOR THIS REASON, TIMER1 WILL BE USED TO
                        ;       KEEP TRACK OF THE NUMBER OF TIME OF DAY INTERRUPTS WHICH
                        ;       WILL BE MISSED. THIS INFORMATION IS USED AFTER THE DISKETTE
                        ;       OPERATION TO UPDATE THE TIME OF DAY.
                        ;--------------------------------------------------------------
01B6  B0 10                     MOV     AL,10H          ; DISABLE NMI
01B8  E6 A0                     OUT     NMI_PORT,AL     ; NO KEYBOARD INTERRUPT
01BA  E8 043A R                 CALL    CLOCK_WAIT      ; WAIT IF TIMER0 IS ABOUT TO
                                                        ; INTERRUPT
                        ;------ ENABLE WATCHDOG TIMER
                        ;--------------------------------------------------------------
                        ;***NOTE***
                        ;       GIVEN THE CURRENT SYSTEM CONFIGURATION A  METHOD IS NEEDED
                        ;       TO PULL THE NEC OUT OF "FATAL ERROR" SITUATIONS.  A TIMER
                        ;       ON THE ADAPTER CARD IS PROVIDED WHICH WILL PERFORM THIS
                        ;       FUNCTION. THE WATCHDOG TIMER ON THE ADAPTER CARD IS ENABLED
                        ;       AND STROBED BEFORE THE 8259 INTERRUPT 6 LINE IS ENABLED.
                        ;       THIS IS BECAUSE OF A GLITCH ON THE LINE LARGE ENOUGH TO
                        ;       TRIGGER AN INTERRUPT.
                        ;--------------------------------------------------------------
01BD  E8 044E R                 CALL    GET_DRIVE
01C0  BA 00F2                   MOV     DX,NEC_CTL      ; GET BIT MASK FOR DRIVE
01C3  0C E0                     OR      AL,FDC_RESET+WD_ENABLE+WD_STROBE
01C5  EE                        OUT     DX,AL           ; CONTROL PORT TO NEC
                                                        ; OUTPUT CONTROL INFO FOR
                                                        ; WATCHDOG(WD) ENABLE
```

```
01C6  24 AF                   AND     AL,FDC_RESET+WD_ENABLE+DRV_SUPPORT     ;
01C8  EE                      OUT     DX,AL           ; OUTPUT CONTROL INFO TO STROBE
                                                      ; WATCHDOG
01C9  BA 00F4                 MOV     DX,NEC_STAT     ; PORT TO NEC STATUS
01CC  B0 20                   MOV     AL,20H          ; SELECT TIMER1 INPUT FROM TIMER0
                                                      ; OUTPUT
01CE  E6 A0                   OUT     NMI_PORT,AL
                      ;------ READ TIMER1 NOW AND SAVE THE INITIAL VALUE
01D0  E8 0000 E               CALL    READ_TIME       ; GET TIMER1 VALUE
01D3  89 46 12                MOV     [BP+I8],AX      ; SAVE INITIAL VALUE FOR CLOCK
                                                      ; UPDATE IN TEMPORAY STORAGE
01D6  E8 0405 R               CALL    DISABLE         ; DISABLE ALL INTERRUPTS
01D9  5B                      ;------ NEC BEGINS OPERATION WHEN NEC RECEIVES LAST PARAMETER
01DA  E8 0331 R               POP     BX              ; GET PARAMTER FROM STACK
01DD  58                      CALL    GET_PARM        ; OUTPUT LAST PARAMETER TO THE NEC
                              POP     AX              ; CAN NOW DISCARD THAT DUMMY RETURN
                                                      ; ADDRESS
01DE  06                      PUSH    ES
01DF  1F                      POP     DS              ; INITIALIZE DS FOR WRITE
01E0  FF E1                   JMP     CX              ; JUMP TO APPROPRIATE R/W/V LOOP
                      ;-----------------------------------------------------------------------
                      ;***NOTE***

                      ;       DATA IS TRANSFERRED USING POLLING ALGORITHMS.  THESE LOOPS
                      ;       TRANSFER A DATA BYTE AT A TIME WHILE POLLING THE NEC FOR
                      ;       NEXT DATA BYTE AND COMPLETION STATUS.
                      ;-----------------------------------------------------------------------
01E2                  ;-----VERIFY OPERATION
01E2  EC              VERIFY_LOOP:
01E3  A8 20                   IN      AL,DX           ; READ STATUS
                              TEST    AL,BUSY_BIT     ; HAS NEC ENTERED EXECUTION PHASE
                                                      ; YET?
01E5  74 FB                   JZ      VERIFY_LOOP     ; NO, CONTINUE SAMPLING
01E7                  J22_2:
01E7  A8 80                   TEST    AL,RQM          ; IS DATA READY?
01E9  75 07                   JNZ     J22_4           ; JUMP IF DATA TRANSFER IS READY
01EB  EC                      IN      AL,DX           ; READ STATUS PORT
01EC  A8 20                   TEST    AL,BUSY_BIT     ; ARE WE DONE?
01EE  75 F7                   JNZ     J22_2           ; JUMP IF MORE TRANSFERS
01F0  EB 35                   JMP     SHORT OP_END    ; TRANSFER DONE
01F2  42              J22_4:  INC     DX              ; POINT AT NEC DATA REGISTER
01F3  EC                      IN      AL,DX           ; READ DATA
01F4  4A                      DEC     DX              ; POINT AT NEC STATUS REGISTER
01F5  EC                      IN      AL,DX           ; READ STATUS PORT
01F6  A8 20                   TEST    AL,BUSY_BIT     ; ARE WE DONE?
01F8  75 ED                   JNZ     J22_2           ; CONTINUE
01FA  EB 2B                   JMP     SHORT OP_END    ; WE ARE DONE
01FC                  ;-----READ OPERATION
01FC  EC              READ_LOOP:
01FD  A8 20                   IN      AL,DX           ; READ STATUS REGISTER
                              TEST    AL,BUSY_BIT     ; HAS NEC STARTED THE EXECUTION
                                                      ; PHASE?
01FF  74 FB                   JZ      READ_LOOP       ; HAS NOT STATRED YET
0201  EC              J22_5:  IN      AL,DX           ; READ STATUS PORT
0202  A8 20                   TEST    AL,BUSY_BIT     ; HAS NEC COMPLETED EXECUTION
                                                      ; PHASE?
0204  74 21                   JZ      OP_END          ; JUMP IF EXECUTION PHASE IS OVER
0206  A8 80                   TEST    AL,RQM          ; IS DATA READY?
0208  74 F7                   JZ      J22_5           ; READ THE DATA
020A  42                      INC     DX              ; POINT AT NEC_DATA
020B  EC                      IN      AL,DX           ; READ DATA
020C  AA                      STOSB                   ; TRANSFER DATA
020D  4A                      DEC     DX              ; POINT AT NEC_STATUS
020E  EB F1                   JMP     J22_5           ; CONTINUE WITH READ OPERATION
                      ;-----WRITE AND FORMAT OPERATION
0210
0210  EC              WRITE_LOOP:
0211  A8 20                   IN      AL,DX           ; READ NEC STATUS PORT
                              TEST    AL,BUSY_BIT     ; HAS THE NEC ENTERED EXECUTION
                                                      ; PHASE YET?
0213  74 FB                   JZ      WRITE_LOOP      ; NO, CONTINUE LOOPING
0215  B9 2080                 MOV     CX,BUSY_BIT*256+RQM
0218                  J22_7:
0218  EC                      IN      AL,DX           ; READ STATUS PORT
0219  84 C5                   TEST    AL,CH           ; IS THE FEC STILL IN THE EXECUTION
                                                      ; PHASE?
021B  74 0A                   JZ      OP_END          ; JUMP IF EXECUTION PHASE IS DONE.
021D  84 C1                   TEST    AL,CL           ; IS THE DATA PORT READY FOR THE
                                                      ; TRANSFER?
021F  74 F7                   JZ      J22_7           ; JUMP TO WRITE DATA
0221  42                      INC     DX              ; POINT AT DATA REGISTER
0222  AC                      LODSB                   ; TRANSFER BYTE
0223  EE                      OUT     DX,AL           ; WRITE THE BYTE ON THE DISKETTE
0224  4A                      DEC     DX              ; POINT AT THE STATUS REGISTER
0225  EB F1                   JMP     J22_7           ; CONTINUE WITH WRITE OR FORMAT
0227  9C                    ;------TRANSFER PROCESS IS OVER
0228  E8 044E R     OP_END: PUSHF                   ; SAVE THE CARRY BIT SET IN
                                                      ; DISK_INT
022B  0C 80                   CALL    GET_DRIVE       ; GET BIT MASK FOR DRIVE SELECTION
022D  BA 00F2                 OR      AL,FDC_RESET    ; NO RESET, KEEP DRIVE SPINNING
0230  EE                      MOV     DX,NEC_CTL      ;
                              OUT     DX,AL           ; DISABLE WATCHDOG
0231  E8 0000 E               ;------ UPDATE TIME OF DAY
0234  E8 043A R               CALL    DDS             ; POINT DS AT BIOS DATA SEGMENT
                              CALL    CLOCK_WAIT      ; WAIT IF TIMER0 IS CLOSE TO
                                                      ; WRAPPING
0237  E8 0000 E               CALL    READ_TIME
023A  8B 5E 12                MOV     BX,[BP+18]      ; GET THE INITIAL VALUE OF TIMER1
023D  2B C3                   SUB     AX,BX           ; UPDATE NUMBER OF INTERRUPTS
                                                      ; MISSED
023F  F7 D8                   NEG     AX              ; PUT IT IN AX
0241  50                      PUSH    AX              ; SAVE IT FOR REUSE IN ISSUING USER
                                                      ; TIMER INTERRUPTS
```

```
0242  01 06 006C R                ADD     TIMER_LOW,AX    ; ADD NUMBER OF TIMER INTERRUPTS TO
                                                          ; TIME
0246  73 04                       JNC     J16_4           ; JUMP IF TIMER_LOW DID NOT SPILL
                                                          ; OVER TO TIMER_HI
0248  FF 06 006E R                INC     TIMER_HIGH
024C  83 3E 006E R 18     J16_4:  CMP     TIMER_HIGH,018H ; TEST FOR COUNT TOTALING 24 HOURS
0251  75 19                       JNZ     J16_5           ; JUMP IF NOT 24 HOURS
0253  81 3E 006C R 00B0           CMP     TIMER_LOW,0B0H  ; LOW VALUE = 24 HOUR VALUE?
0259  7C 11                       JL      J16_5           ; NOT 24 HOUR VALUE?
                          ;------ TIMER HAS GONE 24 HOURS
025B  C7 06 006E R 0000           MOV     TIMER_HIGH,0    ; ZERO OUT TIMER_HIGH VALUE
0261  81 2E 006C R 00B0           SUB     TIMER_LOW,0B0H  ; VALUE REFLECTS CORRECT TICKS PAST
                                                          ; 00B0H
0267  C6 06 0070 R 01             MOV     TIMER_OFL,1     ; INDICATES 24 HOUR THRESHOLD
026C  E8 0414 R           J16_5:  CALL    ENABLE          ; ENABLE ALL INTERRUPTS
026F  59                          POP     CX              ; CX:=AX, COUNT FOR NUMBER OF USER
                                                          ; TIME INTERRUPTS
0270  E3 26                       JCXZ    J16_7           ; IF ZERO DO NOT ISSUE ANY
                                                          ; INTERRUPTS
0272  1E                          PUSH    DS              ; SAVE ALL REGISTERS SAVED PRIOR TO
0273  50                          PUSH    AX              ; INT 1C CALL FROM TIMERINT
                                                          ; THIS PROVIDES A COMPATIBLE
0274  52                          PUSH    DX              ; INTERFACE TO 1C
0275                      J16_6:                          ;
0275  CD 1C                       INT     1CH             ; TRANSFER CONTROL TO USER
                                                          ; INTERRUPT
0277  E2 FC                       LOOP    J16_6           ; DO ALL USER TIMER INTERRUPTS
0279  5A                          POP     DX
027A  58                          POP     AX
027B  1F                          POP     DS              ; RESTORE REGISTERS
                          ;------ CLOCK IS UPDATED AND USER INTERRUPTS 1C HAVE BEEN ISSUED.
                          ;       CHECK IF KEYSTROKE OCCURED
027C  0A C0                       OR      AL,AL           ; AL WAS SET DURING CALL TO ENABLE
027E  74 18                       JZ      J16_7           ; NO KEY WAS PRESSED WHILE SYSTEM
                                                          ; WAS MASKED
0280  BB 0080                     MOV     BX,080H         ; DURATION OF TONE
0283  B9 0048                     MOV     CX,048H         ; FREQUNCY OF TONE
0286  E8 0000 E                  CALL    KB_NOISE        ; NOTIFY USER OF MISSED KEYBORAD
                                                          ; INPUT
                          ;-------CLEAR SHIFT STATES DONT LEAVE POSSIBILTY OF DANGLING STATES
                          ;       OF MISSED BREAKS
0289  80 26 0017 R F0             AND     KB_FLAG,0F0H    ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                          ; SHIFTS
028E  80 26 0018 R 0F             AND     KB_FLAG_1,0FH   ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                          ; NUM AND SCROLL SHIFT
0293  80 26 0088 R 1F             AND     KB_FLAG_2,1FH   ; CLEAR FUNCTION STATES
0298  9D                  J16_7:  POPF                    ; GET THE FLAGS
0299                      J17:
0299  72 40                       JC      J20
029B  E8 03A9 R                   CALL    RESULTS         ; GET THE NEC STATUS
029E  72 3B                       JC      J20             ; LOOK FOR ERROR
                          ;------ CHECK THE RESULTS RETURNED BY THE CONTROLLER
02A0  FC                          CLD                     ; SET THE CORRECT DIRECTION
02A1  BE 0042 R                   MOV     SI,OFFSET NEC_STATUS ; POINT TO STATUS FIELD
02A4  AC                          LODS    NEC_STATUS      ; GET ST0
02A5  24 C0                       AND     AL,0C0H         ; TEST FOR NORMAL TERMINATION
02A7  74 58                       JZ      J22             ; OPN_OK
02A9  3C 40                       CMP     AL,040H         ; TEST FOR ABNORMAL TERMINATION
02AB  75 25                       JNZ     J18             ; NOT ABNORMAL, BAD NEC
                          ;-----------------------------------------------------------------
                          ;***NOTE***
                          ;       THE CURRENT SYSTEM CONFIGURATION HAS NO DMA.  IN ORDER TO
                          ;       STOP THE NEC AN EOT MUST BE PASSED TO FORCE THE NEC TO HALT
                          ;       THEREFORE, THE STATUS RETURNED BY THE NEC WILL ALWAYS SHOW
                          ;       AN EOT ERROR. IF THIS IS THE ONLY ERROR RETURNED AND THE
                          ;       NUMBER OF SECTORS TRANSFERRED EQUALS THE NUMBER SECTORS
                          ;       REQUESTED IN THIS INTERRUPT CALL THEN THE OPERATION HAS
                          ;       COMPLETED SUCCESSFULLY. IF AN EOT ERROR IS RETURNED AND THE
                          ;       REQUESTED NUMBER OF SECTORS IS NOT THE NUMBER OF SECTORS
                          ;       TRANSFERRED THEN THE ERROR IS LEGITIMATE. WHEN THE EOT
                          ;       ERROR IS INVALID THE STATUS BYTES RETURNED ARE UPDATED TO
                          ;       REFLECT THE STATUS OF THE OPERATION IF DMA HAD BEEN PRESENT
                          ;-----------------------------------------------------------------
02AD  AC                          LODS    NEC_STATUS      ; GET ST1
02AE  3C 80                       CMP     AL,80H          ; IS THIS THE ONLY ERROR?
02B0  74 2A                       JE      J21_1           ; NORMAL TERMINATION, NO ERROR
02B2  D0 E0                       SAL     AL,1            ; NOT EOT ERROR, BYPASS ERROR BITS
02B4  D0 E0                       SAL     AL,1
02B6  D0 E0                       SAL     AL,1            ; TEST FOR CRC ERROR
02B8  B4 10                       MOV     AH,BAD_CRC
02BA  72 18                       JC      J19             ; RW_FAIL
02BC  D0 E0                       SAL     AL,1            ; TEST FOR DMA OVERRUN
02BE  B4 08                       MOV     AH,BAD_DMA
02C0  72 12                       JC      J19             ; RW_FAIL
02C2  D0 E0                       SAL     AL,1
02C4  D0 E0                       SAL     AL,1            ; TEST FOR RECORD NOT FOUND
02C6  B4 04                       MOV     AH,RECORD_NOT_FND
02C8  72 0A                       JC      J19             ; RW_FAIL
02CA  D0 E0                       SAL     AL,1
02CC  D0 E0                       SAL     AL,1            ; TEST MISSING ADDRESS MARK
02CE  B4 02                       MOV     AH,BAD_ADDR_MARK
02D0  72 02                       JC      J19             ; RW_FAIL
                          ;------ NEC MUST HAVE FAILED
02D2                      J18:
02D2  B4 20                       MOV     AH,BAD_NEC      ; RW-NEC-FAIL
02D4
02D4  08 26 0041 R        J19:    OR      DISKETTE_STATUS,AH ; RW-FAIL
02D8  E8 03EA R                   CALL    NUM_TRANS       ; HOW MANY WERE REALLY TRANSFERRED
02DB                              RET                     ; RW_ERR
02DB  C3                  J20:    ;------ OPERATION WAS SUCCESSFUL ; RETURN TO CALLER
```

```
02DC                                    J21_1:
02DC  8A 5E 0E                                  MOV     BL,[BP+14]      ; GET NUMBER OF SECTORS PASSED
                                                                        ; FROM STACK
02DF  E8 03EA R                                 CALL    NUM_TRANS       ; HOW MANY GOT MOVED, AL CONTAINS
                                                                        ; NUM OF SECTORS
02E2  3A D8                                     CMP     BL,AL           ; NUMBER REQUESTED=NUMBER ACTUALLY
                                                                        ; TRANSFERRED?
02E4  74 0C                                     JE      J21_2           ; TRANSFER SUCCESSFUL
                                        ;------ OPERATION ATTEMPTED TO ACCESS DATA PAST REAL EOT.  THIS IS
                                        ;       A REAL ERROR
02E6  80 0E 0041 R 04                           OR      DISKETTE_STATUS ,RECORD_NOT_FND
02EB  C6 06 0043 R 80                           MOV     NEC_STATUS+1,80H ; ST1 GETS CORRECT VALUE
02F0  F9                                        STC
02F1  C3                                        RET
02F2  33 C0                             J21_2:  XOR     AX,AX           ; CLEAR AX FOR NEC_STATUS UPDATE
02F4  33 F6                                     XOR     SI,SI           ; INDEX TO NEC_STATUS ARRAY
02F6  88 84 0042 R                              MOV     NEC_STATUS[SI],AL ; ZERO OUT BYTE, ST0
02FA  46                                        INC     SI              ; POINT INDEX AT SECOND BYTE
02FB  88 84 0042 R                              MOV     NEC_STATUS[SI],AL ; ZERO OUT BUYE, ST1
02FF  EB 03                                     JMP     SHORT J21_3     ; OPN_OK
0301  E8 03EA R                         J22:   CALL    NUM_TRANS
0304  32 E4                             J21_3: XOR     AH,AH                   ; NO ERRORS
0306  C3                                        RET
0307                                    RW_OPN  ENDP

                                        ;--------------------------------------------------
                                        ; NEC_OUTPUT
                                        ;       THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER
                                        ;       AFTER TESTING FOR CORRECT DIRECTION AND CONTROLLER READY
                                        ;       THIS ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED
                                        ;       WITHIN A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE
                                        ;       STATUS ON COMPLETION
                                        ; INPUT
                                        ;       (AH)    BYTE TO BE OUTPUT
                                        ; OUTPUT
                                        ;       CY = 0  SUCCESS
                                        ;       CY = 1  FAILURE -- DISKETTE STATUS UPDATED
                                        ;               IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE
                                        ;               LEVEL HIGHER THAN THE CALLER OF NEC_OUTPUT
                                        ;               THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY
                                        ;               CALL OF NEC_OUTPUT
                                        ;       (AL) DESTROYED
                                        ;--------------------------------------------------
0307                                    NEC_OUTPUT      PROC    NEAR
0307  52                                        PUSH    DX              ; SAVE REGISTERS
0308  51                                        PUSH    CX
0309  BA 00F4                                   MOV     DX,NEC_STAT     ; STATUS PORT
030C  33 C9                                     XOR     CX,CX           ; COUNT FOR TIME OUT
030E  EC                                J23:    IN      AL,DX           ; GET STATUS
030F  A8 40                                     TEST    AL,DIO          ; TEST DIRECTION BIT
0311  74 0C                                     JZ      J25             ; DIRECTION OK
0313  E2 F9                                     LOOP    J23
0315                                    J24:                            ; TIME_ERROR
0315  80 0E 0041 R 80                           OR      DISKETTE_STATUS,TIME_OUT
031A  59                                        POP     CX
031B  5A                                        POP     DX              ; SET ERROR CODE AND RESTORE REGS
031C  58                                        POP     AX              ; DISCARD THE RETURN ADDRESS
031D  F9                                        STC                     ; INDICATE ERROR TO CALLER
031E  C3                                        RET
031F  33 C9                             J25:    XOR     CX,CX           ; RESET THE COUNT
0321  EC                                J26:    IN      AL,DX           ; GET THE STATUS
0322  A8 80                                     TEST    AL,RQM          ; IS IT READY?
0324  75 04                                     JNZ     J27             ; YES, GO OUTPUT
0326  E2 F9                                     LOOP    J26             ; COUNT DOWN AND TRY AGAIN
0328  EB EB                                     JMP     J24             ; ERROR CONDITION
032A                                    J27:                            ; OUTPUT
032A  8A C4                                     MOV     AL,AH           ; GET BYTE TO OUTPUT
032C  42                                        INC     DX              ; DATA PORT IS 1 GREATER THAN
                                                                        ; STATUS PORT
032D  EE                                        OUT     DX,AL           ; OUTPUT THE BYTE
032E  59                                        POP     CX              ; RECOVER REGISTERS
032F  5A                                        POP     DX
0330  C3                                        RET                     ; CY = 0 FROM TEST INSTRUCTION
0331                                    NEC_OUTPUT      ENDP
                                        ;--------------------------------------------
                                        ; GET_PARM
                                        ;  THIS ROUTINE FETCHES THE INDEXED POINTER FROM

                                        ;  THE DISK_BASE BLOCK POINTED AT BY THE DATA
                                        ;  VARIABLE DISK_POINTER
                                        ; A BYTE FROM THAT TABLE IS THEN MOVED INTO AH,
                                        ;  THE INDEX OF THAT BYTE BEING THE PARM IN BX
                                        ; ENTRY --
                                        ;   BL = INDEX OF BYTE TO BE FETCHED * 2
                                        ;        IF THE LOW BIT OF BL IS ON, THE BYTE IS IMMEDIATELY
                                        ;        OUTPUT TO THE NEC CONTROLLER
                                        ; EXIT --
                                        ;    AH = THAT BYTE FROM BLOCK
                                        ;    BX = DESTROYED
                                        ;--------------------------------------------
0331                                    GET_PARM        PROC    NEAR
0331  1E                                        PUSH    DS              ; SAVE SEGMENT
0332  56                                        PUSH    SI              ; SAVE REGISTER
0333  2B C0                                     SUB     AX,AX           ; ZERO TO AX
0335  32 FF                                     XOR     BH,BH           ; ZERO BH
0337  8E D8                                     MOV     DS,AX
                                                ASSUME  DS:ABS0
0339  C5 36 0078 R                             LDS     SI,DISK_POINTER ; POINT TO BLOCK
033D  D1 EB                                     SHR     BX,1            ; DIVIDE BX BY 2, AND SET FLAG FOR
                                                                        ; EXIT
033F  9C                                        PUSHF                   ; SAVE OUTPUT BIT
0340  8A 20                                     MOV     AH,[SI+BX]      ; GET THE BYTE
0342  83 FB 01                                  CMP     BX,1            ; IS THIS THE PARM WITH DMA
                                                                        ; INDICATOR
```

```
0345  75 05                        JNZ     J27_1
0347  80 CC 01                     OR      AH,1            ; TURN ON NO DMA BIT
034A  EB 0C                        JMP     SHORT J27_2
034C  83 FB 0A              J27_1: CMP     BX,10           ; MOTOR STARTUP DELAY?
034F  75 07                        JNE     J27_2
0351  80 FC 04                     CMP     AH,4            ; GREATER THAN OR EQUAL TO 1/2 SEC?
0354  7D 02                        JGE     J27_2           ; YES, OKAY
0356  B4 04                        MOV     AH,4            ; NO, FORCE 1/2 SECOND DELAY
0358  9D                    J27_2: POPF                    ; GET OUTPUT BIT
0359  5E                           POP     SI              ; RESTORE REGISTER
035A  1F                           POP     DS              ; RESTORE SEGMENT
                                   ASSUME  DS:DATA
035B  72 AA                        JC      NEC_OUTPUT      ; IF FLAG SET, OUTPUT TO CONTROLLER
035D  C3                           RET                     ; RETURN TO CALLER
035E                       GET_PARM        ENDP
                           ;--------------------------------------------------
                           ; BOUND_SETUP
                           ;       THIS ROUTINE SETS UP BUFFER ADDRESSING FOR READ/WRITE/VERIFY
                           ;       OPERATIONS.
                           ; INPUT
                           ;       ES HAS ORIGINAL BUFFER SEGMENT VALUE
                           ;       BP POINTS AT BASE OF SAVED PARMETERS ON STACK
                           ; OUTPUT
                           ;       ES HAS SEGMENT WHICH WILL ALLOW 64K ACCESS.  THE
                           ;       COMBINATION ES:DI  AND DS:SI POINT TO THE BUFFER. THIS
                           ;       CALCULATED ADDRESS WILL ALWAYS ACCESS 64K OF MEMORY.
                           ;       BX DESTOYED
                           DISKIO2.INC

                           ;--------------------------------------------------
035E                       BOUND_SETUP     PROC    NEAR
035E  51                           PUSH    CX              ; SAVE REGISTERS
035F  8B 5E 0C                     MOV     BX,[BP+12]      ; GET OFFSET OF BUFFER FROM STACK
0362  53                           PUSH    BX              ; SAVE OFFSET TEMPORARILY
0363  B1 04                        MOV     CL,4            ; SHIFT COUNT
0365  D3 EB                        SHR     BX,CL           ; SHIFT OFFSET FOR NEW SEGMENT
                                                           ; VALUE
0367  8C C1                        MOV     CX,ES           ; PUT ES IN REGISTER SUITABLE FOR
                                                           ; ADDING TO
0369  03 CB                        ADD     CX,BX           ; GET NEW VALUE FOR ES
036B  8E C1                        MOV     ES,CX           ; UPDATE THE ES REGISTER
036D  5B                           POP     BX              ; RECOVER ORIGINAL OFFSET
036E  81 E3 000F                   AND     BX,0000FH       ; NEW OFFSET
0372  8B F3                        MOV     SI,BX           ; DS:SI POINT AT BUFFER
0374  8B FB                        MOV     DI,BX           ; ES:DI POINT AT BUFFER
0376  59                           POP     CX
0377  C3                           RET
0378                       BOUND_SETUP     ENDP
                           ;--------------------------------------------------
                           ; CHK_STAT_2
                           ;       THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
                           ;       A RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
                           ;       THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
                           ;       AND THE RESULT RETURNED TO THE CALLER.
                           ; INPUT
                           ;       NONE
                           ; OUTPUT
                           ;       CY = 0 SUCCESS
                           ;       CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
                           ;       (AX) DESTROYED
                           ;--------------------------------------------------
0378                       CHK_STAT_2      PROC    NEAR
0378  53                           PUSH    BX              ; SAVE REGISTERS
0379  56                           PUSH    SI
037A  33 DB                        XOR     BX,BX           ; NUMBER OF SENSE INTERRUPTS TO
                                                           ; ISSUE
037C  BE 0391 R                    MOV     SI,OFFSET J33_3 ; SET UP DUMMY RETURN FROM
                                                           ; NEC_OUTPUT
037F  56                           PUSH    SI              ; PUT ON STACK
0380  B4 08                 J33_2: MOV     AH,08H          ; SENSE INTERUPT STATUS
0382  E8 0307 R                    CALL    NEC_OUTPUT      ; ISSUE SENSE INTERUPT STATUS
0385  E8 03A9 R                    CALL    RESULTS         ;
0388  72 10                        JC      J35             ; NEC TIME OUT, FLAGS SET IN
                                                           ; RESULTS
038A  A0 0042 R                    MOV     AL,NEC_STATUS   ; GET STATUS
038D  A8 20                        TEST    AL,SEEK_END     ; IS SEEK OR RECAL OPERATION DONE?
038F  75 0D                        JNZ     J35_1           ; JUMP IF EXECUTION OF SEEK OR
                                                           ; RECAL DONE
0391  4B                    J33_3: DEC     BX              ; DEC LOOP COUNTER
0392  75 EC                        JNZ     J33_2           ; DO ANOTHER LOOP
0394  80 0E 0041 R 80              OR      DISKETTE_STATUS,TIME_OUT
0399  F9                    J34:   STC                     ; RETURN ERROR INDICATION FOR
                                                           ; CALLER
039A  5E                    J35:   POP     SI              ; RESTORE REGISTERS
039B  5E                           POP     SI
039C  5B                           POP     BX
039D  C3                           RET
                           ;-----SEEK END HAS OCCURED, CHECK FOR NORMAL TERMINATION
039E  24 C0                 J35_1: AND     AL,0C0H         ; MASK NORMAL TERMINATION BITS
03A0  74 F8                        JZ      J35             ; JUMP IF NORMAL TERMINATION
03A2  80 0E 0041 R 40              OR      DISKETTE_STATUS,BAD_SEEK
03A7  EB F0                        JMP     J34
03A9                       CHK_STAT_2      ENDP
                           ;--------------------------------------------------
                           ; RESULTS
                           ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
                           ;       HAS TO SAY FOLLOWING AN INTERRUPT.
                           ;       IT IS ASSUMED THAT THE NEC DATA PORT = NEC STATUS PORT + 1.
                           ; INPUT
                           ;       NONE
                           ; OUTPUT
                           ;       CY = 0  SUCCESSFUL TRANSFER
                           ;       CY = 1  FAILURE -- TIME OUT IN WAITING FOR STATUS
```

```
                        ;       NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT
                        ;       (AH) DESTROYED
                        ;--------------------------------------------
03A9                    RESULTS PROC    NEAR
03A9  FC                        CLD
03AA  BF 0042 R                 MOV     DI,OFFSET NEC_STATUS ; POINTER TO DATA AREA
03AD  51                        PUSH    CX                  ; SAVE COUNTER
03AE  52                        PUSH    DX
03AF  53                        PUSH    BX
03B0  B3 07                     MOV     BL,7                ; MAX STATUS BYTES
03B2                    ;------ WAIT FOR REQUEST FOR MASTER
03B2                    J38:                                ; INPUT_LOOP
03B2  33 C9                     XOR     CX,CX               ; COUNTER
03B4  BA 00F4                   MOV     DX,NEC_STAT         ; STATUS PORT
03B7                    J39:                                ; WAIT FOR MASTER
03B7  EC                        IN      AL,DX               ; GET STATUS
03B8  A8 80                     TEST    AL,080H             ; MASTER READY
03BA  75 0C                     JNZ     J40A                ; TEST_DIR
03BC  E2 F9                     LOOP    J39                 ; WAIT_MASTER
03BE  80 0E 0041 R 80           OR      DISKETTE_STATUS,TIME_OUT
03C3                    J40:                                ; RESULTS_ERROR
03C3  F9                        STC                         ; SET ERROR RETURN
03C4                    ;------ RESULT OPERATION IS DONE
03C4  5B                J44:   POP     BX
03C5  5A                       POP     DX
03C6  59                       POP     CX
03C7  C3                       RET
03C8                    ;------ TEST THE DIRECTION BIT
03C8  EC                J40A:  IN      AL,DX               ; GET STATUS REG AGAIN
03C9  A8 40                    TEST    AL,040H             ; TEST DIRECTION BIT
03CB  75 07                    JNZ     J42                 ; OK TO READ STATUS
03CD                    J41:                                ; NEC_FAIL
03CD  80 0E 0041 R 20           OR      DISKETTE_STATUS,BAD_NEC
03D2  EB EF                     JMP     J40                 ; RESULTS_ERROR
03D4                    ;------ READ IN THE STATUS
03D4                    J42:                                ; INPUT_STAT
03D4  42                        INC     DX                  ; POINT AT DATA PORT
03D5  EC                        IN      AL,DX               ; GET THE DATA
03D6  88 05                     MOV     [DI],AL             ; STORE THE BYTE
03D8  47                        INC     DI                  ; INCREMENT THE POINTER
03D9  B9 000A                   MOV     CX,10               ; LOOP TO KILL TIME FOR NEC
03DC  E2 FE                J43: LOOP    J43
03DE  4A                        DEC     DX                  ; POINT AT STATUS PORT
03DF  EC                        IN      AL,DX               ; GET STATUS
03E0  A8 10                     TEST    AL,010H             ; TEST FOR NEC STILL BUSY
03E2  74 E0                     JZ      J44                 ; RESULTS DONE
03E4  FE CB                     DEC     BL                  ; DECREMENT THE STATUS COUNTER
03E6  75 CA                     JNZ     J38                 ; GO BACK FOR MORE
03E8  EB E3                     JMP     J41                 ; CHIP HAS FAILED
                        ;--------------------------------------------
                        ; NUM_TRANS
                        ;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
                        ;       WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE
                        ; INPUT
                        ;       (CH) = CYLINDER OF OPERATION
                        ;       (CL) = START SECTOR OF OPERATION
                        ; OUTPUT
                        ;       (AL) = NUMBER ACTUALLY TRANSFERRED
                        ;       NO OTHER REGISTERS MODIFIED
                        ;--------------------------------------------
03EA                    NUM_TRANS       PROC    NEAR
03EA  A0 0045 R                 MOV     AL,NEC_STATUS+3 ; GET CYLINDER ENDED UP ON
03ED  3A 46 0B                  CMP     AL,[BP+11]      ; SAME AS WE STARTED
03F0  A0 0047 R                 MOV     AL,NEC_STATUS+5 ; GET ENDING SECTOR
03F3  74 07                     JZ      J45             ; IF ON SAME CYL, THEN NO ADJUST
03F5  B3 08                     MOV     BL,8
03F7  E8 0331 R                 CALL    GET_PARM        ; GET EOT VALUE
03FA  8A C4                     MOV     AL,AH           ;  INTO AL
03FC  FE C0                J45: INC     AL              ; USE EOT+1 FOR CALCULATION
03FE  2A 46 0A                  SUB     AL,[BP]+10      ; SUBTRACT START FROM END
0401  88 46 0E                  MOV     [BP+14],AL
0404  C3                        RET
0405                    NUM_TRANS       ENDP
0405                    RESULTS ENDP
                        ;--------------------------------------------
                        ; DISABLE
                        ;       THIS ROUTINE WILL DISABLE ALL INTERRUPTS EXCEPT FOR
                        ;       INTERRUPT 6 SO WATCH DOG TIME OUT CAN OCCUR IN ERROR
                        ;       CONDITIONS.
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       NONE
                        ;       ALL REGISTERS REMAIN INTACT
                        ;--------------------------------------------

0405                    DISABLE         PROC    NEAR
0405  50                       PUSH    AX
0406                    ;------ DISABLE ALL INTERRUPTS AT THE 8259 LEVEL EXCEPT DISKETTE
0406  E4 21                    IN      AL,INTA01       ; READ CURRENT MASK
0408  89 46 10                 MOV     [BP+16],AX      ; SAVE MASK ON THE SPACE ALLOCATED
                                                       ; ON THE STACK
040B  B0 BF                     MOV     AL,0BFH         ; MASK OFF ALL INTERRUPTS EXCEPT
                                                       ; DISKETTE
040D  E6 21                     OUT     INTA01,AL       ; OUTPUT MASK TO THE 8259
040F  E8 035E R                 CALL    BOUND_SETUP     ; SETUP REGISTERS TO ACCESS BUFFER
0412  58                        POP     AX
0413  C3                       RET
0414                    DISABLE         ENDP
                        ;--------------------------------------------
                        ; ENABLE
                        ;       THIS PROC ENABLES ALL INTERRUPTS.  IT ALSO SETS THE 8253 TO
```

```
                        ;       THE MODE REQUIRED FOR KEYBOARD DATA DESERIALIZATION.
                        ;       BEFORE THE LATCH FOR KEYBOARD DATA IS RESET, BIT 0 OF THE
                        ;       8255 IS READ TO DETERMINE WHETHER ANY KEYSTROKES OCCURED
                        ;       WHILE THE SYSTEM WAS MASKED OFF.
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       AL=1 MEANS A KEY WAS STRUCK DURING DISKETTE I/O.  (OR NOISE
                        ;            ON THE LINE)
                        ;       AL=0 MEANS THAT NO KEY WAS PRESSED.
                        ;       AX IS DESTROYED.  ALL OTHER REGISTERS REMAIN INTACT.
                        ;--------------------------------------------------
0414                    ENABLE          PROC    NEAR
0414  52                        PUSH    DX                      ; SAVE DX
                        ;------ RETURN TIMER1 TO STATE NEEDED FOR KEYBOARD I/O
0415  B0 76                     MOV     AL,01110110B            ;
0417  E6 43                     OUT     TIM_CTL,AL
0419  50                        PUSH    AX
041A  58                        POP     AX                      ; WAIT FOR 8253 TO INITIALIZE
                                                                ; ITSELF
041B  B0 FF                     MOV     AL,0FFH                 ; INITIAL VALUE FOR 8253
041D  E6 41                     OUT     TIMER+1,AL              ; LSB
041F  50                        PUSH    AX
0420  58                        POP     AX                      ; WAIT
0421  E6 41                     OUT     TIMER+1,AL              ; MSB
                        ;------ CHECK IF ANY KEYSTROKES OCCURED DURING DISKETTE TRANSFER
0423  8E 46 10                  MOV     ES,[BP+16]              ; GET ORIGINAL ES VALUE FROM THE
                                                                ; STACK
0426  E4 62                     IN      AL,62H                  ; READ PORT C OF 8255
0428  24 01                     AND     AL,01H                  ; BIT=1 MEANS KESTROKE HAS OCCURED
042A  50                        PUSH    AX                      ; SAVE IT ON THE STACK
                        ;------ ENABLE NMI INTERRUPTS
042B  E4 A0                     IN      AL,NMI_PORT             ; RESET LATCH
042D  B0 80                     MOV     AL,80H                  ; MASK TO ENABLE NMI
042F  E6 A0                     OUT     NMI_PORT,AL             ; ENABLE NMI
                        ;-----  ENABLE ALL INTERRUPTS WHICH WERE ENABLED BEFORE TRANSFER
0431  8B 46 10                  MOV     AX,[BP+16]              ; GET MASK FROM THE STACK

0434  E6 21                     OUT     INTA01,AL
0436  58                        POP     AX                      ; PASS BACK KEY STROKE FLAG
0437  5A                        POP     DX
0438  FB                        STI
0439  C3                        RET
043A                    ENABLE          ENDP
                        ;--------------------------------------------------
                        ;CLOCK_WAIT
                        ;       THIS PROCEDURE IS CALLED WHEN THE TIME OF DAY
                        ;       IS BEING UPDATED.  IT WAITS IF TIMER0 IS ALMOST
                        ;       READY TO WRAP UNTIL IT IS SAFE TO READ AN ACCURATE
                        ;       TIMER1.
                        ;INPUT
                        ;       NONE.
                        ;OUTPUT
                        ;       NONE. AX IS DESTROYED.
                        ;--------------------------------------------------
043A                    CLOCK_WAIT      PROC    NEAR
043A  32 C0                     XOR     AL,AL                   ; READ MODE TIMER0 FOR 8253
043C  E6 43                     OUT     TIM_CTL,AL              ; OUTPUT TO THE 8253
043E  50                        PUSH    AX
043F  58                        POP     AX                      ; WAIT FOR 8253 TO INITIALIZE
                                                                ; ITSELF
0440  E4 40                     IN      AL,TIMER0               ; READ LEAST SIGNIFICANT BYTE
0442  86 C4                     XCHG    AL,AH                   ; SAVE IT
0444  E4 40                     IN      AL,TIMER0               ; READ MOST SIGNIFICANT BYTE
0446  86 C4                     XCHG    AL,AH                   ; REARRANGE FOR PROPER ORDER
0448  3D 012C                   CMP     AX,THRESHOLD            ; IS TIMER0 CLOSE TO WRAPPING?
044B  72 ED                     JC      CLOCK_WAIT              ; JUMP IF CLOCK IS WITHIN THRESHOLD
044D  C3                        RET                             ; OK TO READ TIMER1
044E                    CLOCK_WAIT      ENDP
                        ;--------------------------------------------------------------------
                        ;GET_DRIVE
                        ;       THIS ROUTINE WILL CALCULATE A BIT MASK FOR THE DRIVE WHICH
                        ;       IS SELECTED BY THE CURRENT INT 13 CALL.  THE DRIVE SELECTED
                        ;       CORRESPONDS TO THE BIT IN THE MASK,  I.E. DRIVE ZERO
                        ;       CORRESPONDS TO BIT ZERO AND A 01H IS RETURNED. THE BIT IS
                        ;       CALCULATED BY ACCESSING THE PARAMETERS PASSED TO INT 13
                        ;       WHICH WERE SAVED ON THE STACK.
                        ;INPUT
                        ;       BYTE PTR[BP] MUST POINT TO DRIVE FOR SELECTION.
                        ;OUPTUT
                        ;       AL CONTAINS THE BIT MASK.  ALL OTHER REGISTERS ARE INTACT
                        ;--------------------------------------------------------------------
044E                    GET_DRIVE       PROC    NEAR
044E  51                        PUSH    CX                      ; SAVE REGISTER.
044F  8A 4E 00                  MOV     CL,BYTE PTR[BP]         ; GET DRIVE NUMBER
0452  80 E1 3F                  AND     CL,OFF_DBL_TRK          ; IGNORE J SUPPORT PARAMETER
0455  B0 01                     MOV     AL,1                    ; INITIALIZE AL WITH VALUE FOR

0457  D2 E0                     SHL     AL,CL                   ; SHIFTING
                                                                ; SHIFT BIT POSITION BY DRIVE
0459  24 0F                     AND     AL,DRV_SUPPORT          ; NUMBER (DRIVE IN RANGE 0-3)
                                                                ; ONLY FOUR DRIVES ARE SUPPORTED.

045B  59                        POP     CX                      ;  RANGE CHECK
045C  C3                        RET                             ; RESTORE REGISTERS
045D                    GET_DRIVE       ENDP
                        ;--------------------------------------------------
                        ;  SEEK
                        ;       THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE
                        ;       TO THE NAMED TRACK.  IF THE DRIVE HAS NOT BEEN ACCESSED
                        ;       SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE WILL BE
                        ; INPUT RECALIBRATED.
                        ;       (DL) = DRIVE TO SEEK ON
```

```
                        ;       (CH) = TRACK TO SEEK TO
                        ;       (BP) = BIT 6 :1 80 TRACK MEDIA
                        ;              BIT 6 :0 40 TRACK
                        ; OUTPUT
                        ;       CY = 0 SUCCESS
                        ;       CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
                        ;       (AX) DESTROYED
                        ;--------------------------------------------------------
045D                    SEEK    PROC    NEAR
045D  56                        PUSH    SI              ; SAVE REGISTER
045E  53                        PUSH    BX              ; SAVE REGISTER
045F  51                        PUSH    CX
0460  BE 0074 R                 MOV     SI,OFFSET TRACK0 ; BASE OF CURRENT HEAD POSITIONS
0463  B0 01                     MOV     AL,1            ; ESTABLISH MASK FOR RECAL
0465  8A CA                     MOV     CL,DL           ; USE DRIVE AS A SHIFT COUNT
0467  81 E1 00FF                AND     CX,0FFH         ; MASK OFF HIGH BYTE
046B  03 F1                     ADD     SI,CX           ; POINT SI AT CORRECT DRIVE
046D  D2 C0                     ROL     AL,CL           ; GET MASK FOR DRIVE
                        ;------ SI CONTAINS OFFSET FOR CORRECT DRIVE, AL CONTAINS BIT MASK
                        ;       IN POSITION 0,1 OR 2
046F  59                        POP     CX              ; RESTORE PARAMETER REGISTER
0470  BB 04F0 R                 MOV     BX,OFFSET PJ32  ; SET UP ERROR RECOVERY ADDRESS
0473  53                        PUSH    BX              ; NEEDED FOR ROUTINE NEC_OUTPUT
0474  84 06 003E R              TEST    SEEK_STATUS,AL  ; TEST DRIVE FOR RECAL
0478  75 39                     JNZ     PJ28             ; NO_RECAL
047A  08 06 003E R              OR      SEEK_STATUS,AL  ; TURN ON THE NO RECAL BIT IN FLAG
047E  80 3C 00                  CMP     BYTE PTR[SI],0  ; LAST REFERENCED TRACK=0?
0481  74 30                     JZ      PJ28            ; YES IGNORE RECAL
0483  B4 07                     MOV     AH,07H          ; RECALIBRATE COMMAND
0485  E8 0307 R                 CALL    NEC_OUTPUT
0488  8A E2                     MOV     AH,DL           ; RECAL REQUIRED ON DRIVE IN DL
048A  E8 0307 R                 CALL    NEC_OUTPUT      ; OUTPUT THE DRIVE NUMBER
                        ;------ HEAD IS MOVING TO CORRECT TRACK
                        ;               HOOK FOR 40 TRACK TO 80 TRACK SUPPORT
048D  E8 0378 R                 CALL    CHK_STAT_2      ; CHECK STATUS
0490  73 19                     JNC     RDYI            ; LEAVE ONE MORE CHANCE FOR RETRY
0492  C6 06 0041 R 00           MOV     DISKETTE_STATUS,0 ;CLEAR STATUS
0497  B4 05                     MOV     AH,5            ;SET  PARM AS  5 MILL
0499  E8 04F9 R                 CALL    DELAY_N_MS      ;WAIT SUB
049C  B4 07                     MOV     AH,07H          ;RECABRATE COMMAND
049E  E8 0307 R                 CALL    NEC_OUTPUT      ;OU T

04A1  8A E2                     MOV     AH,DL           ;RECAB DRIVE IN DL
04A3  E8 0307 R                 CALL    NEC_OUTPUT      ;OUT
04A6  E8 0378 R                 CALL    CHK_STAT_2      ;CHECK STATUS
04A9  72 48                     JC      PJ32_2          ;IF ERROR, JUMP TO SET CARRRY
04AB                    RDY1:
04AB  B4 12                     MOV     AH,18           ;RETRY OK WAIT UNTILL SETTLE
04AD  E8 04F9 R                 CALL    DELAY_N_MS      ;IX
                        ;       DRIVE IS SYNC WITH CONTROLLER, SEEK TO TRACK
                        ;       SEEK READY               ;
04B0  C6 04 00                  MOV     BYTE PTR[SI],0
                        ;------ DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK
04B3  8A 04             PJ28:    MOV     AL,BYTE PTR[SI] ; GET THE PCN
04B5  2A C5                     SUB     AL,CH           ; GET SEEK_WAIT VALUE
04B7  74 36                     JZ      PJ31_1           ; ALREADY ON CORRECT TRACK
04B9  B4 0F                     MOV     AH,0FH          ; SEEK COMMAND TO NEC
04BB  E8 0307 R                 CALL    NEC_OUTPUT
04BE  8A E2                     MOV     AH,DL           ; DRIVE NUMBER
04C0  E8 0307 R                 CALL    NEC_OUTPUT
04C3  51                        PUSH    CX              ; SAVE FOR DOS
04C4  F6 46 00 40               TEST    BYTE PTR [BP],040H  ; IF DOUBLE TRACK PARAMETER
04C8  75 02                     JNE     GO_SEEK         ; LEAVE PARAMETER AS IS
04CA  D0 E5             PCJR:   SHL     CH,1            ; MULTIPLY BY TWO WHEN 40 TRACK MEDIA
04CC  8A E5             GO_SEEK: MOV    AH,CH           ; TRACK NUMBER
04CE  E8 0307 R                 CALL    NEC_OUTPUT
04D1  59                        POP     CX              ; RESTORE FOR DOS
04D2  E8 0378 R                 CALL    CHK_STAT_2      ; GET ENDING INTERRUPT AND SENSE
                                                        ; STATUS
                        ;------ WAIT FOR HEAD SETTLE
04D5  9C                        PUSHF                   ; SAVE STATUS FLAGS
04D6  51                        PUSH    CX              ; SAVE REGISTER
04D7  B3 12                     MOV     BL,18           ; HEAD SETTLE PARAMETER
04D9  E8 0331 R                 CALL    GET_PARM
04DC                    PJ29:                            ; HEAD_SETTLE
04DC  B9 0226                   MOV     CX,550          ; 1 MS LOOP
04DF  0A E4                     OR      AH,AH           ; TEST FOR TIME EXPIRED
04E1  74 06                     JZ      PJ31
04E3  E2 FE             PJ30:   LOOP    PJ30            ; DELAY FOR 1 MS
04E5  FE CC                     DEC     AH              ; DECREMENT THE COUNT
04E7  EB F3                     JMP     PJ29             ; DO IT SOME MORE
04E9  59                PJ31:   POP     CX              ; RESTORE REGISTER
04EA  9D                        POPF
04EB  72 06                     JC      PJ32_2
04ED  88 2C                     MOV     BYTE PTR[SI],CH
04EF  5B                PJ31_1:  POP     BX              ; GET RID OF DUMMY RETURN
04F0                    PJ32:                            ; SEEK_ERROR
04F0  5B                        POP     BX              ; RESTORE REGISTER
04F1  5E                        POP     SI              ; UPDATE CORRECT
04F2  C3                        RET                     ; RETURN TO CALLER
04F3  C6 04 FF          PJ32_2:  MOV     BYTE PTR[SI],0FFH ; UNKNOWN STATUS ABOUT SEEK
                                                        ; OPERATION
04F6  5B                        POP     BX              ; GET RID OF DUMMY RETURN
04F7  EB F7                     JMP     SHORT PJ32
04F9                    SEEK    ENDP

                        ;       INPUT   AH: N MILL SEC WAIT
04F9                    DELAY_N_MS PROC NEAR
04F9  51                        PUSH    CX              ; SOFTWARE TIMER
04FA  B9 0226           DE00:   MOV     CX,550          ; JX
04FD  0A E4                     OR      AH,AH           ; JX
04FF  74 06                     JZ      DE40            ; JX
0501  E2 FE             DE20:   LOOP    DE20            ; JX
```

```
0503  FE CC                        DEC     AH                      ; JX
0505  EB F3                        JMP     DE00                    ; JX
0507  59                    DE40:  POP     CX                      ; JX
0508  C3                          RET
0509                        DELAY_N_MS ENDP

                            ;--------------------------------------------
                            ; DISK_BASE
                            ;  THIS IS THE SET OF PARAMETERS REQUIRED FOR
                            ;  DISKETTE OPERATION.  THEY ARE POINTED AT BY THE
                            ;  DATA VARIABLE DISK_POINTER.  TO MODIFY THE PARAMETERS,
                            ;  BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
                            ;--------------------------------------------
0509                        DISK_BASE       LABEL   BYTE
0509  EF                            DB      11101111B               ; SRT=E, HD UNLOAD=0F - 1ST SPECIFY
                                                                    ; BYTE
050A  03                            DB      3                       ; HD LOAD=1, MODE=NO DMA - 2ND
                                                                    ; SPECIFY BYTE
050B  25                            DB      MOTOR_WAIT              ; WAIT AFTER OPN TIL MOTOR OFF
050C  02                            DB      2                       ; 512 BYTES/SECTOR
050D  09                            DB      9                       ; EOT ( LAST SECTOR ON TRACK)
050E  2A                            DB      02AH                    ; GAP LENGTH
050F  FF                            DB      0FFH                    ; DTL
0510  50                            DB      050H                    ; GAP LENGTH FOR FORMAT
0511  F6                            DB      0F6H                    ; FILL BYTE FOR FORMAT
0512  0F                            DB      15                      ; HEAD SETTLE TIME (MILLISECONDS)
0513  04                            DB      4                       ; MOTOR START TIME (1/8 SECONDS)

                            ;--------------------------------------------
                            ; DISK_INT
                            ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT. AN INTERRUPT
                            ;       WILL OCCUR ONLY WHEN THE ONE-SHOT TIMER IS FIRED.  THIS
                            ;       OCCURS IN AN ERROR SITUATION.  THIS ROUTINE SETS ERRORS IN
                            ;       THE DISKETTE STATUS BYTE AND DISABLES THE ONE-SHOT TIMER.
                            ;       THEN THE RETURN ADDRESS ON THE STACK IS CHANGED TO RETURN
                            ;       TO THE OP_END LABEL.
                            ; INPUT
                            ;       NONE.
                            ; OUTPUT
                            ;       NONE.  DS POINTS AT BIOS DATA AREA.  CARRY FLAG IS SET SO
                            ;       THAT ERROR WILL BE CAUGHT IN THE ENVIRONMEMT RETURNED TO.
                            ;--------------------------------------------
0514                        DISK_INT        PROC    FAR
0514  1E                            PUSH    DS
0515  50                            PUSH    AX
0516  52                            PUSH    DX
0517  55                            PUSH    BP                      ; SAVE REGISTER
0518  E8 0000 E                     CALL    DDS                     ; SAVE THE BP REGISTER
                                                                    ; SETUP DS TO POINT AT BIOS DATA
                            ;------ CHECK IF INTERRUPT OCCURED IN INT13 OR WHETHER IT IS A
                            ;       SPURIOUS INTERRUPT
051B  8B EC                         MOV     BP,SP                   ; POINT BP AT STACK
051D  0E                            PUSH    CS                      ; WAS IT IN THE BIOS AREA
051E  58                            POP     AX
051F  3B 46 0A                      CMP     AX,WORD PTR[BP+10] ; GET INTERRUPTED SEGMENT
0522  75 48                         JNE     DI3                     ; NOT IN BIOS, ERROR CONDITION
0524  8B 46 08                      MOV     AX,WORD PTR[BP+8] ; GET IP ON THE STACK
0527  3D 01E2 R                     CMP     AX,OFFSET VERIFY_LOOP ; RANGE CHECK IP FOR DISK
                                                                    ; TRANSFER
052A  7C 40                         JL      DI3                     ; BELOW TRANSFER CODE
052C  3D 0228 R                     CMP     AX,OFFSET OP_END+1 ; UPPER RANGE OF TRANSFER CODE
052F  7D 3B                         JGE     DI3                     ; ABOVE RANGE OF WATCHDOG TERRAIN
                            ;-------VALID DISKETTE INTERRUPT CHANGE RETURN ADDRESS ON STACK TO
                            ;       PULL OUT OF LOOP
0531  C7 46 08 0227 R               MOV     WORD PTR[BP+8],OFFSET OP_END
0536  81 4E 0C 0001                 OR      WORD PTR[BP+12],1 ; TURN ON CARRY FLAG IN FLAGS ON
                                                                    ; STACK
                            ;--------------------------------------------------------------------
                            ;***NOTE***
                            ; A WRITE PROTECTED DISKETTE WILL ALWAYS GET STUCK IN WRITE LOOP
                            ; WAITING FOR BEGINNING OF EXECUTION PHASE.  WHEN THE WATCHDOG
                            ; FIRES AND THE STATUS IN PORT NEC_STAT = DXH (X MEANS DON'T CARE)
                            ; AND WRITE PROTECT IS  CHECKED FOR.
                            ;--------------------------------------------------------------------
053B  BA 00F4                       MOV     DX,NEC_STAT
053E  EC                            IN      AL,DX
053F  24 F0                         AND     AL,0F0H                 ; GET NEC STATUS BYTE
0541  3C D0                         CMP     AL,0D0H                 ; MASK HIGH NIBBLE
0543  75 14                         JNE     DI1                     ; IS EXECUTION PHASE DONE
0545  E8 03A9 R                     CALL    RESULTS                 ; STUCK IN LOOP
                                                                    ; GET STATUS OF OPERATION
0548  BE 0042 R                     MOV     SI,OFFSET NEC_STATUS ; ADDRESS OF BYTES RETURNED BY
                                                                    ; NEC
054B  8A 44 01                      MOV     AL,[SI+1]               ; GET ST1
054E  A8 02                         TEST    AL,02H                  ; WRITE PROTECT SIGNAL ACTIVE?
0550  74 07                         JZ      DI1                     ; TIME OUT ERROR
0552  80 0E 0041 R 03               OR      DISKETTE_STATUS,WRITE_PROTECT
0557  EB 13                         JMP     SHORT DI3
                            ;-------TIME OUT ERROR
0559  80 0E 0041 R 80       DI1:    OR      DISKETTE_STATUS,TIME_OUT
055E  C6 06 003E R 00               MOV     SEEK_STATUS,0   ; SET RECAL ON DRIVES
                            ;------ RESET THE NEC AND DISABLE WATCHDOG
0563  BA 00F2               DI2:    MOV     DX,NEC_CTL
0566  5D                           POP     BP                      ; ADDRESS TO NEC CONTROL PORT
                                                                    ; POINT BP AT BASE OF STACKED
0567  E8 044E R                     CALL    GET_DRIVE               ; PARAMETERS
056A  55                            PUSH    BP                      ; RESET ADAPTER AND DISABLE WD
056B  EE                    DI3:    OUT     DX,AL                   ; RESTORE FOR RETURNED CALL
056C  B0 20                         MOV     AL,EOI
056E  E6 20                         OUT     INTA00,AL               ; GIVE EOI TO 8259
0570  5D                            POP     BP
0571  5A                            POP     DX
```

```
0572  58                        POP     AX
0573  1F                        POP     DS
0574  CF                        IRET
0575                    DISK_INT        ENDP                    ; RETURN FROM INTERRUPT

                        ;-----INT 14-----------------------------------------
                        ;RS232_IO
                        ;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
                        ;       PORT ACCORDING TO THE PARAMETERS:

                        ;       (AH)=0  INITIALIZE THE COMMUNICATIONS PORT
                        ;               (AL) HAS PARMS FOR INITIALIZATION
                        ;
                        ;---7-------6-------5-------4-------3-------2---------1-------0----
                        ;-------- BAUD RATE ---:::----PARITY----:::-STOPBIT-:::--WORD LENGTH--
                        ;
                        ;  000 - 110                X0 - NONE       0 - 1     10 - 7 BITS
                        ;  001 - 150                01 - ODD        1 - 2     11 - 8 BITS
                        ;  010 - 300                11 - EVEN
                        ;  011 - 600
                        ;  100 - 1200
                        ;  101 - 2400
                        ;  110 - 4800
                        ;  111 - 4800
                        ;               ON RETURN, THE RS232 INTERRUPTS ARE DISABLED AND
                        ;                       CONDITIONS ARE SET AS IN CALL TO COMMO
                        ;                       STATUS (AH=3)
                        ;       (AH)=1  SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
                        ;               (AL) REGISTER IS PRESERVED
                        ;               ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS
                        ;                       UNABLE TO TRANSMIT THE BYTE OF DATA OVER
                        ;                       THE LINE. IF BIT 7 OF AH IS NOT SET, THE
                        ;                       REMAINDER OF AH IS SET AS IN A STATUS
                        ;                       REQUEST, REFELECTING THE CURRENT STATUS OF
                        ;                       THE LINE.
                        ;       (AH)=2  RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
                        ;                       RETURNING TO CALLER
                        ;               ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY
                        ;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY
                        ;                       BITS LEFT ON, ARE THE ERROR BITS
                        ;                       (7,4,3,2,1). IN THIS CASE, THE TIME OUT BIT
                        ;                       INDICATES DATA SET READY WAS NOT RECEIVED.
                        ;                       THUS, AH IS NON ZERO ONLY WHEN AN ERROR
                        ;                       OCCURRED.(NOTE: IF THE TIME-OUT BIT IS SET,
                        ;                       OTHER BITS IN AH MAY NOT BE RELIABLE.)
                        ;       (AH)=3  RETURN THE COMMO PORT STATUS IN (AX)
                        ;               AH CONTAINS THE LINE CONTROL STATUS
                        ;               BIT 7 = TIME OUT
                        ;               BIT 6 = TRANS SHIFT REGISTER EMPTY
                        ;               BIT 5 = TRAN HOLDING REGISTER EMPTY
                        ;               BIT 4 = BREAK DETECT
                        ;               BIT 3 = FRAMING ERROR
                        ;               BIT 2 = PARITY ERROR
                        ;               BIT 1 = OVERRUN ERROR
                        ;               BIT 0 = DATA READY
                        ;               AL CONTAINS THE MODEM STATUS
                        ;               BIT 7 = RECIEVED LINE SIGNAL DETECT

                        ;               BIT 6 = RING INDICATOR
                        ;               BIT 5 = DATA SET READY
                        ;               BIT 4 = CLEAR TO SEND
                        ;               BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
                        ;               BIT 2 = TRAILING EDGE RING DETECTOR
                        ;               BIT 1 = DELTA DATA SET READY
                        ;               BIT 0 = DELTA CLEAR TO SEND
                        ;       (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
                        ; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE
                        ;    CARD.  LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
                        ; DATA AREA RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
                        ;       VALUE FOR TIMEOUT (DEFAULT=1)
                        ;OUTPUT
                        ;               AX      MODIFIED ACCORDING TO PARMS OF CALL
                        ;               ALL OTHERS UNCHANGED
                        ;-----------------------------------------------------

                                ASSUME  CS:CODE,DS:DATA
0575                    A1      LABEL   WORD
0575  03F9                      DW      1017    ; 110 BAUD      ; TABLE OF INIT VALUE
0577  02EA                      DW      746     ; 150
0579  0175                      DW      373     ; 300
057B  00BA                      DW      186     ; 600
057D  005D                      DW      93      ; 1200
057F  002F                      DW      47      ; 2400
0581  0017                      DW      23      ; 4800
0583  0017                      DW      23      ; 4800
0585                    RS232_IO        PROC    FAR
                        ;------ VECTOR TO APPROPRIATE ROUTINE
0585  FB                        STI                     ; INTERRUPTS BACK ON
0586  1E                        PUSH    DS              ; SAVE SEGMENT
0587  52                        PUSH    DX
0588  56                        PUSH    SI
0589  57                        PUSH    DI
058A  51                        PUSH    CX
058B  53                        PUSH    BX
058C  8B F2                     MOV     SI,DX           ; RS232 VALUE TO SI
058E  8B FA                     MOV     DI,DX           ; AND TO DI (FOR TIMEOUTS)
0590  D1 E6                     SHL     SI,1            ; WORD OFFSET
0592  E8 0000 E                 CALL    DDS             ; POINT TO BIOS DATA SEGMENT
0595  8B 94 0000 R              MOV     DX,RS232_BASE[SI] ; GET BASE ADDRESS
0599  0B D2                     OR      DX,DX           ; TEST FOR 0 BASE ADDRESS
059B  74 13                     JZ      A3              ; RETURN
059D  0A E4                     OR      AH,AH           ; TEST FOR (AH)=0
059F  74 16                     JZ      A4              ; COMMUN INIT
05A1  FE CC                     DEC     AH              ; TEST FOR (AH)=1
```

```
05A3  74 47                     JZ      A5              ; SEND AL
05A5  FE CC                     DEC     AH              ; TEST FOR (AH)=2
05A7  74 6C                     JZ      A12             ; RECEIVE INTO AL
05A9  FE CC                     DEC     AH              ; TEST FOR (AH)=3
05AB  75 03                     JNZ     A3
05AD  E9 063F R                 JMP     A18             ; COMMUNICATION STATUS
05B0                    A3:                             ; RETURN FROM RS232
05B0  5B                        POP     BX

05B1  59                        POP     CX
05B2  5F                        POP     DI
05B3  5E                        POP     SI
05B4  5A                        POP     DX
05B5  1F                        POP     DS
05B6  CF                        IRET                    ; RETURN TO CALLER, NO ACTION
                        ;------ INITIALIZE THE COMMUNICATIONS PORT
05B7  8A E0             A4:     MOV     AH,AL           ; SAVE INIT PARMS IN AH
05B9  83 C2 03                  ADD     DX,3            ; POINT TO 8250 CONTROL REGISTER
05BC  B0 80                     MOV     AL,80H
05BE  EE                        OUT     DX,AL           ; SET DLAB=1
                        ;------ DETERMINE BAUD RATE DIVISOR
05BF  8A D4                     MOV     DL,AH           ; GET PARMS TO DL
05C1  B1 04                     MOV     CL,4
05C3  D2 C2                     ROL     DL,CL
05C5  81 E2 000E                AND     DX,0EH          ; ISOLATE THEM
05C9  BF 0575 R                 MOV     DI,OFFSET A1    ; BASE OF TABLE
05CC  03 FA                     ADD     DI,DX           ; PUT INTO INDEX REGISTER
05CE  8B 94 0000 R              MOV     DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
05D2  42                        INC     DX
05D3  2E: 8A 45 01              MOV     AL,CS:[DI]+1    ; GET HIGH ORDER OF DIVISOR
05D7  EE                        OUT     DX,AL           ; SET MS OF DIV TO 0
05D8  4A                        DEC     DX
05D9  2E: 8A 05                 MOV     AL,CS:[DI]      ; GET LOW ORDER OF DIVISOR
05DC  EE                        OUT     DX,AL           ; SET LOW OF DIVISOR
05DD  83 C2 03                  ADD     DX,3
05E0  8A C4                     MOV     AL,AH           ; GET PARMS BACK
05E2  24 1F                     AND     AL,01FH         ; STRIP OFF THE BAUD BITS
05E4  EE                        OUT     DX,AL           ; LINE CONTROL TO 8 BITS
05E5  4A                        DEC     DX
05E6  4A                        DEC     DX
05E7  B0 00                     MOV     AL,0
05E9  EE                        OUT     DX,AL           ; INTERRUPT ENABLES ALL OFF
05EA  EB 53                     JMP     SHORT A18       ; COM_STATUS
                        ;------ SEND CHARACTER IN (AL) OVER COMMO LINE
05EC                    A5:
05EC  50                        PUSH    AX              ; SAVE CHAR TO SEND
05ED  83 C2 04                  ADD     DX,4            ; MODEM CONTROL REGISTER
05F0  B0 03                     MOV     AL,3            ; DTR AND RTS
05F2  EE                        OUT     DX,AL           ; DATA TERMINAL READY, REQUEST TO
                                                        ; SEND
05F3  42                        INC     DX              ; MODEM STATUS REGISTER
05F4  42                        INC     DX
05F5  B7 30                     MOV     BH,30H          ; DATA SET READY & CLEAR TO SEND
05F7  E8 064E R                 CALL    WAIT_FOR_STATUS ; ARE BOTH TRUE?
05FA  74 08                     JE      A9              ; YES, READY TO TRANSMIT CHAR
05FC  59                A7:     POP     CX
05FD  8A C1                     MOV     AL,CL           ; RELOAD DATA BYTE
05FF  80 CC 80          A8:     OR      AH,80H          ; INDICATE TIME OUT
0602  EB AC                     JMP     A3              ; RETURN
0604                    A9:                             ; CLEAR_TO_SEND
0604  4A                        DEC     DX              ; LINE STATUS REGISTER
0605  B7 20                     MOV     BH,20H          ; IS TRANSMITTER READY

0607  E8 064E R                 CALL    WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
060A  75 F0                     JNZ     A7              ; RETURN WITH TIME OUT SET
060C  83 EA 05                  SUB     DX,5            ; DATA PORT
060F  59                        POP     CX              ; RECOVER IN CX TEMPORARILY
0610  8A C1                     MOV     AL,CL           ; MOVE CHAR TO AL FOR OUT, STATUS
                                                        ; IN AH
0612  EE                        OUT     DX,AL           ; OUTPUT CHARACTER
0613  EB 9B                     JMP     A3              ; RETURN
                        ;------ RECEIVE CHARACTER FROM COMMO LINE
0615  83 C2 04           A12:    ADD     DX,4            ; MODEM CONTROL REGISTER
0618  B0 01                     MOV     AL,1            ; DATA TERMINAL READY
061A  EE                        OUT     DX,AL
061B  42                        INC     DX
061C  42                        INC     DX              ; MODEM STATUS REGISTER
061D  B7 20                     MOV     BH,20H
061F  E8 064E R                 CALL    WAIT_FOR_STATUS ; DATA SET READY
0622  75 DB                     JNZ     A8              ; TEST FOR DSR
0624  4A                        DEC     DX              ; RETURN WITH ERROR
0625  EC                A16:    IN      AL,DX           ; LINE STATUS REGISTER
0626  A8 01                     TEST    AL,1
0628  75 09                     JNZ     A17             ; RECEIVE BUFFER FULL
062A  F6 06 0071 R 80           TEST    BIOS_BREAK,80H  ; TEST FOR REC. BUFF. FULL
062F  74 F4                     JZ      A16             ; TEST FOR BREAK KEY
0631  EB CC                     JMP     A8              ; LOOP IF NO BREAK KEY
0633  24 1E              A17:    AND     AL,00011110B    ; SET TIME OUT ERROR
                                                        ; TEST FOR ERROR CONDITIONS ON RECV
0635  8A E0                     MOV     AH,AL           ; CHAR
0637  8B 94 0000 R              MOV     DX,RS232_BASE[SI] ; DATA PORT
063B  EC                        IN      AL,DX
063C  E9 05B0 R                 JMP     A3              ; GET CHARACTER FROM LINE
                        ;------ COMMO PORT STATUS ROUTINE ; RETURN
063F  8B 94 0000 R      A18:    MOV     DX,RS232_BASE[SI]
0643  83 C2 05                  ADD     DX,5
0646  EC                        IN      AL,DX           ; CONTROL PORT
0647  8A E0                     MOV     AH,AL           ; GET LINE CONTROL STATUS
0649  42                        INC     DX              ; PUT IN AH FOR RETURN
064A  EC                        IN      AL,DX           ; POINT TO MODEM STATUS REGISTER
064B  E9 05B0 R                 JMP     A3              ; GET MODEM CONTROL STATUS
                        ;-------------------------------------- ; RETURN
                        ;       WAIT FOR STATUS ROUTINE
```

```
                        ;ENTRY: BH=STATUS BIT(S) TO LOOK FOR,
                        ;       DX=ADDR. OF STATUS REG
                        ;EXIT:  ZERO FLAG ON = STATUS FOUND
                        ;       ZERO FLAG OFF = TIMEOUT.
                        ;       AH=LAST STATUS READ
                        ;---------------------------------------
064E                    WAIT_FOR_STATUS PROC    NEAR
064E  8A 9D 007C R              MOV     BL,RS232_TIM_OUT[DI] ;LOAD OUTER LOOP COUNT
0652  2B C9             WFS0:   SUB     CX,CX
0654  EC                WFS1:  IN      AL,DX           ;GET STATUS
0655  8A E0                     MOV     AH,AL           ; MOVE TO AH
0657  22 C7                    AND     AL,BH           ;ISOLATE BITS TO TEST
0659  3A C7                    CMP     AL,BH           ;EXACTLY = TO MASK
                        RS232.INC

065B  74 08                     JE      WFS_END         ;RETURN WITH ZERO FLAG ON
065D  E2 F5                     LOOP    WFS1            ;TRY AGAIN
065F  FE CB                     DEC     BL
0661  75 EF                     JNZ     WFS0
0663  0A FF                     OR      BH,BH           ;SET ZERO FLAG OFF
0665                    WFS_END:
0665  C3                        RET
0666                    WAIT_FOR_STATUS ENDP
0666                    RS232_IO        ENDP

                        ;--- INT 15 ------------------------------------------------------
                        ; CASSETTE I/O
                        ;       (AH) = 0  TURN CASSETTE MOTOR ON
                        ;       (AH) = 1  TURN CASSETTE MOTOR OFF
                        ;       (AH) = 2  READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                        ;                 (ES,BX) = POINTER TO DATA BUFFER
                        ;                 (CX) = COUNT OF BYTES TO READ
                        ; ON EXIT
                        ;       (ES,BX) = POINTER TO LAST BYTE READ + 1
                        ;       (DX) = COUNT OF BYTES ACTUALLY READ
                        ;       (CY) = 0 IF NO ERROR OCCURRED
                        ;            = 1 IF ERROR OCCURRED
                        ;       (AH) = ERROR RETURN IF (CY)= 1
                        ;                 = 01 IF CRC ERROR WAS DETECTED
                        ;                 = 02 IF DATA TRANSITIONS ARE LOST
                        ;                 = 04 IF NO DATA WAS FOUND
                        ;       (AH) = 3 WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                        ;                 (ES,BX) = POINTER TO DATA BUFFER
                        ;                 (CX) = COUNT OF BYTES TO WRITE
                        ; ON EXIT
                        ;       (EX,BX) = POINTER TO LAST BYTE WRITTEN + 1
                        ;       (CX) = 0
                        ;       (AH) = ANY OTHER THAN ABOVE VALUES CAUSES (CY)= 1
                        ;                 AND (AH)= 80 TO BE RETURNED (INVALID COMMAND).
                        ;-----------------------------------------------------------------
0666                            ASSUME  DS:DATA, ES:NOTHING,SS:NOTHING,CS:CODE
0666                    CASSETTE_IO     PROC    FAR
0666  FB                        STI                     ; INTERRUPTS BACK ON
0667  1E                        PUSH    DS              ; ESTABLISH ADDRESSING TO DATA
0668  E8 0000 E                 CALL    DDS
066B  80 26 0071 R 7F           AND     BIOS_BREAK, 7FH ; MAKE SURE BREAK FLAG IS OFF
0670  E8 0677 R                 CALL    W1              ; CASSETTE_IO_CONT
0673  1F                        POP     DS
0674  CA 0002                  RET     2               ; INTERRUPT RETURN
0677                    CASSETTE_IO     ENDP
0677                    W1      PROC    NEAR
                        ;-----------------------------------------------------
                        ; PURPOSE:
                        ;  TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH
                        ;  AH           ROUTINE
                        ;-----------------------------------------------------
                        ;  0            MOTOR ON
                        ;  1            MOTOR OFF
                        ;  2            READ CASSETTE BLOCK
                        ;  3            WRITE CASSETTE BLOCK
                        ;-----------------------------------------------------
0677  0A E4                     OR      AH,AH           ; TURN ON MOTOR?
0679  74 13                     JZ      MOTOR_ON        ; YES, DO IT
067B  FE CC                     DEC     AH              ; TURN OFF MOTOR?
067D  74 18                     JZ      MOTOR_OFF       ; YES, DO IT
067F  FE CC                     DEC     AH              ; READ CASSETTE BLOCK?
0681  74 1A                     JZ      READ_BLOCK      ; YES, DO IT

0683  FE CC                     DEC     AH              ; WRITE CASSETTE BLOCK?
0685  75 03                     JNZ     W2              ;  NOT_DEFINED
0687  E9 07A4 R                 JMP     WRITE_BLOCK     ; YES, DO IT
068A                    W2:                             ; COMMAND NOT DEFINED
068A  B4 80                     MOV     AH,080H         ; ERROR, UNDEFINED OPERATION
068C  F9                        STC                     ; ERROR FLAG
068D  C3                        RET
068E                    W1      ENDP
068E                    MOTOR_ON        PROC    NEAR
                        ;---------------------------------------
                        ; PURPOSE:
                        ;       TO TURN ON CASSETTE MOTOR
                        ;---------------------------------------
068E  E4 61                     IN      AL,PORT_B       ; READ CASSETTE OUTPUT
0690  24 F7                     AND     AL,NOT 08H      ; CLEAR BIT TO TURN ON MOTOR
0692  E6 61             W3:     OUT     PORT_B,AL       ; WRITE IT OUT
0694  2A E4                     SUB     AH,AH           ; CLEAR AH
0696  C3                        RET
0697                    MOTOR_ON        ENDP
0697                    MOTOR_OFF       PROC    NEAR
                        ;---------------------------------------
                        ; PURPOSE:                             :
                        ;       TO TURN CASSETTE MOTOR OFF      :
                        ;---------------------------------------
0697  E4 61                     IN      AL,PORT_B       ; READ CASSETTE OUTPUT
```

```
0699  0C 08                    OR      AL,08H          ; SET BIT TO TURN OFF
069B  EB F5                    JMP     W3              ; WRITE IT, CLEAR ERROR, RETURN
069D                   MOTOR_OFF       ENDP
069D                   READ_BLOCK      PROC    NEAR
                       ;--------------------------------------------------------
                       ; PURPOSE:
                       ;       TO READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                       ;
                       ; ON ENTRY:
                       ;       ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
                       ;       BX POINTS TO START OF MEMORY BUFFER
                       ;       CX CONTAINS NUMBER OF BYTES TO READ
                       ; ON EXIT:
                       ;       BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
                       ;       CX CONTAINS DECREMENTED BYTE COUNT
                       ;       DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                       ;
                       ;       CARRY FLAG IS CLEAR IF NO ERROR DETECTED
                       ;       CARRY FLAG IS SET IF CRC ERROR DETECTED
                       ;--------------------------------------------------------
069D  53                       PUSH    BX              ; SAVE BX
069E  51                       PUSH    CX              ; SAVE CX
069F  56                       PUSH    SI              ; SAVE SI
06A0  BE 0007                  MOV     SI, 7           ; SET UP RETRY COUNT FOR LEADER
06A3  E8 085D R                CALL    BEGIN_OP        ; BEGIN BY STARTING MOTOR
06A6                   W4:                             ; SEARCH FOR LEADER
06A6  E4 62                    IN      AL,PORT_C       ; GET INITIAL VALUE
06A8  24 10                    AND     AL,010H         ; MASK OFF EXTRANEOUS BITS

06AA  A2 006B R                MOV     LAST_VAL,AL     ; SAVE IN LOC LAST_VAL
06AD  BA 3F7A                  MOV     DX,16250        ; # OF TRANSITIONS TO LOOK FOR
06B0                   W5:                             ; WAIT_FOR_EDGE
06B0  F6 06 0071 R 80          TEST    BIOS_BREAK, 80H ; CHECK FOR BREAK KEY
06B5  75 03                    JNZ     W6A             ; JUMP IF NO BREAK KEY
                                                       ; JUMP IF BREAK KEY HIT
06B7  4A                       DEC     DX
06B8  75 03                    JNZ     W7
06BA  E9 073C R                JMP     W17             ; JUMP IF BEGINNING OF LEADER
06BD  E8 077C R        W6A:    CALL    READ_HALF_BIT   ; JUMP IF NO LEADER FOUND
06C0  E3 EE            W7:     JCXZ    W5              ; IGNORE FIRST EDGE
06C2  BA 0378                  MOV     DX,0378H        ; JUMP IF NO EDGE DETECTED
06C5  B9 0200                  MOV     CX,200H         ; CHECK FOR HALF BITS
                                                       ; MUST HAVE AT LEAST THIS MANY ONE
                                                       ; SIZE PULSES BEFORE CHCKNG FOR
06C8  FA                       CLI                     ; SYNC BIT (0)
06C9                   W8:                             ; DISABLE INTERRUPTS
06C9  F6 06 0071 R 80          TEST    BIOS_BREAK, 80H ; SEARCH-LDR
06CE  75 6C                    JNZ     W17             ; CHECK FOR BREAK KEY
06D0  51                       PUSH    CX              ; JUMP IF BREAK KEY HIT
06D1  E8 077C R                CALL    READ_HALF_BIT   ; SAVE REG CX
06D4  0B C9                    OR      CX, CX          ; GET PULSE WIDTH
06D6  59                       POP     CX              ; CHECK FOR TRANSITION
06D7  74 CD                    JZ      W4              ; RESTORE ONE BIT COUNTER
06D9  3B D3                    CMP     DX,BX           ; JUMP IF NO TRANSITION
06DB  E3 04                    JCXZ    W9              ; CHECK PULSE WIDTH
                                                       ; IF CX=0 THEN WE CAN LOOK
06DD  73 C7                    JNC     W4              ; FOR SYNC BIT (0)
                                                       ; JUMP IF ZERO BIT (NOT GOOD
06DF  E2 E8                    LOOP    W8              ; LEADER)
                                                       ; DEC CX AND READ ANOTHER HALF ONE
06E1                   W9:                             ; BIT
06E1  72 E6                    JC      W8              ; FIND-SYNC
                                                       ; JUMP IF ONE BIT (STILL LEADER)
                       ;----- A SYNCH BIT HAS BEEN FOUND.  READ SYN CHARACTER:
06E3  E8 077C R                CALL    READ_HALF_BIT   ; SKIP OTHER HALF OF SYNC BIT (0)
06E6  E8 074E R                CALL    READ_BYTE       ; READ SYNC BYTE
06E9  3C 16                    CMP     AL, 16H         ; SYNCHRONIZATION CHARACTER
06EB  75 49                    JNE     W16             ; JUMP IF BAD LEADER FOUND.
                       ;----- GOOD CRC SO READ DATA BLOCK(S)
06ED  5E                       POP     SI              ; RESTORE REGS
06EE  59                       POP     CX
06EF  5B                       POP     BX
                       ;--------------------------------------------------------
                       ; READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                       ; ON ENTRY:
                       ;       ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
                       ;       BX POINTS TO START OF MEMORY BUFFER
                       ;       CX CONTAINS NUMBER OF BYTES TO READ
                       ; ON EXIT:
                       ;       BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
                       ;       CX CONTAINS DECREMENTED BYTE COUNT
                       ;       DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                       ;--------------------------------------------------------
06F0  51               W10:    PUSH    CX              ; SAVE BYTE COUNT
06F1                                                   ; COME HERE BEFORE EACH
06F1  C7 06 0069 R FFFF        MOV     CRC_REG,0FFFFH  ; 256 BYTE BLOCK IS READ
06F7  BA 0100                  MOV     DX,256          ; INIT CRC REG
06FA                   W11:                            ; SET DX TO DATA BLOCK SIZE
06FA  F6 06 0071 R 80          TEST    BIOS_BREAK, 80H ; RD_BLK
06FF  75 23                    JNZ     W13             ; CHECK FOR BREAK KEY
0701  E8 074E R                CALL    READ_BYTE       ; JUMP IF BREAK KEY HIT
0704  72 1E                    JC      W13             ; READ BYTE FROM CASSETTE
                                                       ; CY SET INDICATES NO DATA
0706  E3 05                    JCXZ    W12             ; TRANSITIONS
                                                       ; IF WE'VE ALREADY REACHED
0708  26: 88 07                MOV     ES:[BX],AL      ; END OF MEMORY BUFFER
070B  43                       INC     BX              ; SKIP REST OF BLOCK
070C  49                       DEC     CX              ; STORE DATA BYTE AT BYTE PTR
070D                   W12:                            ; INC BUFFER PTR
070D  4A                       DEC     DX              ; DEC BYTE COUNTER
070E  7F EA                    JG      W11             ; LOOP UNTIL DATA BLOCK HAS BEEN READ FROM CASSETTE
                                                       ; DEC BLOCK CNT
                                                       ; RD_BLK
```

```
0710  E8 074E R                 CALL    READ_BYTE           ; NOW READ TWO CRC BYTES
0713  E8 074E R                 CALL    READ_BYTE
0716  2A E4                     SUB     AH,AH               ; CLEAR AH
0718  81 3E 0069 R 1D0F        CMP     CRC_REG,1D0FH       ; IS THE CRC CORRECT?
071E  75 06                     JNE     W14                 ; IF NOT EQUAL CRC IS BAD
0720  E3 06                     JCXZ    W15                 ; IF BYTE COUNT IS ZERO
                                                            ; THEN WE HAVE READ ENOUGH
                                                            ; SO WE WILL EXIT
0722  EB CD                     JMP     W10                 ; STILL MORE, SO READ ANOTHER BLOCK
0724                        W13:                            ; MISSING-DATA
                                                            ; NO DATA TRANSITIONS SO
0724  B4 01                     MOV     AH,01H              ; SET AH=02 TO INDICATE
                                                            ; DATA TIMEOUT
0726                        W14:                            ; BAD-CRC
0726  FE C4                     INC     AH                  ; EXIT EARLY ON ERROR
                                                            ; SET AH=01 TO INDICATE CRC ERROR
0728                        W15:                            ; RD-BLK-EX
0728  5A                       POP     DX                  ; CALCULATE COUNT OF
0729  2B D1                     SUB     DX,CX               ; DATA BYTES ACTUALLY READ
                                                            ; RETURN COUNT IN REG DX
072B  50                       PUSH    AX                  ; SAVE AX (RET CODE)
072C  F6 C4 90                 TEST    AH, 90H             ; CHECK FOR ERRORS
072F  75 13                     JNZ     W18                 ; JUMP IF ERROR DETECTED
0731  E8 074E R                 CALL    READ_BYTE           ; READ TRAILER
0734  EB 0E                     JMP     SHORT W18           ; SKIP TO TURN OFF MOTOR
0736                        W16:                            ; BAD-LEADER
0736  4E                       DEC     SI                  ; CHECK RETRIES
0737  74 03                     JZ      W17                 ; JUMP IF TOO MANY RETRIES
0739  E9 06A6 R                 JMP     W4                  ; JUMP IF NOT TOO MANY RETRIES
073C                        W17:                            ; NO VALID DATA FOUND
                            ;----- NO DATA FROM CASSETTE ERROR, I.E. TIMEOUT
073C  5E                       POP     SI                  ; RESTORE REGS
073D  59                       POP     CX                  ; RESTORE REGS

073E  5B                       POP     BX
073F  2B D2                     SUB     DX,DX               ; ZERO NUMBER OF BYTES READ
0741  B4 04                     MOV     AH,04H              ; TIME OUT ERROR (NO LEADER)
0743  50                       PUSH    AX
0744                        W18:                            ; MOT-OFF
0744  FB                       STI                         ; REENABLE INTERRUPTS
0745  E8 0697 R                 CALL    MOTOR_OFF           ; TURN OFF MOTOR
0748  58                       POP     AX                  ; RESTORE RETURN CODE
0749  80 FC 01                  CMP     AH,01H              ; SET CARRY IF ERROR (AH>0)
074C  F5                       CMC
074D  C3                       RET                         ; FINISHED
074E                        READ_BLOCK  ENDP
                            ;--------------------------------------
                            ; PURPOSE:
                            ;       TO READ A BYTE FROM CASSETTE
                            ; ON EXIT
                            ;       REG AL CONTAINS READ DATA BYTE
                            ;--------------------------------------
074E                        READ_BYTE   PROC    NEAR
074E  53                       PUSH    BX                  ; SAVE REGS BX,CX
074F  51                       PUSH    CX
0750  B1 08                     MOV     CL,8H               ; SET BIT COUNTER FOR 8 BITS
0752                        W19:                            ; BYTE-ASM
0752  51                       PUSH    CX                  ; SAVE CX
                            ;-------------------------------
                            ; READ DATA BIT FROM CASSETTE
                            ;-------------------------------
0753  E8 077C R                 CALL    READ_HALF_BIT       ; READ ONE PULSE
0756  E3 20                     JCXZ    W21                 ; IF CX=0 THEN TIMEOUT
                                                            ; BECAUSE OF NO DATA TRANSITIONS
0758  53                       PUSH    BX                  ; SAVE 1ST HALF BIT'S
                                                            ; PULSE WIDTH (IN BX)
0759  E8 077C R                 CALL    READ_HALF_BIT       ; READ COMPLEMENTARY PULSE
075C  58                       POP     AX                  ; COMPUTE DATA BIT
075D  E3 19                     JCXZ    W21                 ; IF CX=0 THEN TIMEOUT DUE TO
                                                            ; NO DATA TRANSITIONS
075F  03 D8                     ADD     BX,AX               ; PERIOD
0761  81 FB 06F0                CMP     BX, 06F0H           ; CHECK FOR ZERO BIT
0765  F5                       CMC                         ; CARRY IS SET IF ONE BIT
0766  9F                       LAHF                        ; SAVE CARRY IN AH
0767  59                       POP     CX                  ; RESTORE CX
                                                            ; NOTE:
                                                            ; MS BIT OF BYTE IS READ FIRST.
                                                            ; REG CH IS SHIFTED LEFT WITH
                                                            ; CARRY BEING INSERTED INTO LS
                                                            ; BIT OF CH.
                                                            ; AFTER ALL 8 BITS HAVE BEEN
                                                            ; READ, THE MS BIT OF THE DATA
                                                            ; BYTE WILL BE IN THE MS BIT OF
                                                            ; REG CH
0768  D0 D5                     RCL     CH,1                ; ROTATE REG CH LEFT WITH CARRY TO
                                                            ;   LS BIT OF REG CH
076A  9E                       SAHF                        ; RESTORE CARRY FOR CRC ROUTINE
076B  E8 0849 R                 CALL    CRC_GEN             ; GENERATE CRC FOR BIT
076E  FE C9                     DEC     CL                  ; LOOP TILL ALL 8 BITS OF DATA
                                                            ; ASSEMBLED IN REG CH
0770  75 E0                     JNZ     W19                 ; BYTE_ASM
0772  8A C5                     MOV     AL,CH               ; RETURN DATA BYTE IN REG AL
0774  F8                       CLC
0775                        W20:                            ; RD-BYT-EX
0775  59                       POP     CX                  ; RESTORE REGS CX,BX
0776  5B                       POP     BX
0777  C3                       RET                         ; FINISHED
0778                                                        ; NO-DATA
0778  59                       POP     CX                  ; RESTORE CX
0779  F9                       STC                         ; INDICATE ERROR
077A  EB F9                     JMP     W20                 ; RD_BYT_EX
077C                        READ_BYTE   ENDP
```

```
                        ;-----------------------------------------------
                        ; PURPOSE:
                        ;       TO COMPUTE TIME TILL NEXT DATA
                        ;       TRANSITION (EDGE)
                        ; ON ENTRY:
                        ;       EDGE_CNT CONTAINS LAST EDGE COUNT
                        ; ON EXIT:
                        ;       AX CONTAINS OLD LAST EDGE COUNT
                        ;       BX CONTAINS PULSE WIDTH (HALF BIT)
                        ;-----------------------------------------------
077C                    READ_HALF_BIT   PROC    NEAR
077C  B9 0064                   MOV     CX, 100         ; SET TIME TO WAIT FOR BIT
077F  8A 26 006B R              MOV     AH,LAST_VAL     ; GET PRESENT INPUT VALUE
0783                    W22:                            ; RD-H-BIT
0783  E4 62                     IN      AL,PORT_C       ; INPUT DATA BIT
0785  24 10                     AND     AL,010H         ; MASK OFF EXTRANEOUS BITS
0787  3A C4                     CMP     AL,AH           ; SAME AS BEFORE?
0789  E1 F8                     LOOPE   W22             ; LOOP TILL IT CHANGES
078B  A2 006B R                 MOV     LAST_VAL,AL     ; UPDATE LAST_VAL WITH NEW VALUE
078E  B0 40                     MOV     AL,40H          ; READ TIMER'S COUNTER COMMAND
0790  E6 43                     OUT     TIM_CTL,AL      ; LATCH COUNTER
0792  8B 1E 0067 R              MOV     BX,EDGE_CNT     ; BX GETS LAST EDGE COUNT
0796  E4 41                     IN      AL,TIMER+1      ; GET LS BYTE
0798  8A E0                     MOV     AH,AL           ; SAVE IN AH
079A  E4 41                     IN      AL,TIMER+1      ; GET MS BYTE
079C  86 C4                     XCHG    AL,AH           ; XCHG AL,AH
079E  2B D8                     SUB     BX,AX           ; SET BX EQUAL TO HALF BIT PERIOD
07A0  A3 0067 R                 MOV     EDGE_CNT,AX     ; UPDATE EDGE COUNT;
07A3  C3                        RET
07A4                    READ_HALF_BIT   ENDP
                        ;-------------------------------------------------------------
                        ; PURPOSE
                        ;       WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE.
                        ;       THE DATA IS PADDED TO FILL OUT THE LAST 256 BYTE BLOCK.
                        ; ON ENTRY:
                        ;       BX POINTS TO MEMORY BUFFER ADDRESS
                        ;       CX CONTAINS NUMBER OF BYTES TO WRITE
                        ; ON EXIT:

                        ;       BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                        ;       CX IS ZERO
                        ;-------------------------------------------------------------
07A4                    WRITE_BLOCK     PROC    NEAR
07A4  53                        PUSH    BX
07A5  51                        PUSH    CX
07A6  E4 61                     IN      AL,PORT_B       ; DISABLE SPEAKER
07A8  24 FD                     AND     AL,NOT 02H
07AA  0C 01                     OR      AL, 01H         ; ENABLE TIMER
07AC  E6 61                     OUT     PORT_B,AL
07AE  B0 B6                     MOV     AL,0B6H         ; SET UP TIMER - MODE 3 SQUARE WAVE
07B0  E6 43                     OUT     TIM_CTL,AL
07B2  E8 085D R                 CALL    BEGIN_OP        ; START MOTOR AND DELAY
07B5  B8 04A0                   MOV     AX,1184         ; SET NORMAL BIT SIZE
07B8  E8 0842 R                 CALL    W31             ; SET_TIMER
07BB  B9 0800                   MOV     CX,0800H        ; SET CX FOR LEADER BYTE COUNT
07BE                    W23:                            ; WRITE LEADER
07BE  F9                        STC                     ; WRITE ONE BITS
07BF  E8 082C R                 CALL    WRITE_BIT
07C2  E2 FA                     LOOP    W23             ; LOOP 'TIL LEADER IS WRITTEN
07C4  FA                        CLI                     ; DISABLE INTS.
07C5  F8                        CLC                     ; WRITE SYNC BIT (0)
07C6  E8 082C R                 CALL    WRITE_BIT
07C9  59                        POP     CX              ; RESTORE REGS CX,BX
07CA  5B                        POP     BX
07CB  B0 16                     MOV     AL, 16H         ; WRITE SYNC CHARACTER
07CD  E8 0815 R                 CALL    WRITE_BYTE
                        ;-------------------------------------------------------------
                        ; PURPOSE
                        ;       WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                        ; ON ENTRY:
                        ;       BX POINTS TO MEMORY BUFFER ADDRESS
                        ;       CONTAINS NUMBER OF BYTES TO WRITE
                        ; ON EXIT:
                        ;       BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                        ;       CX IS ZERO
                        ;-------------------------------------------------------------
07D0                    WR_BLOCK:
07D0  C7 06 0069 R FFFF         MOV     CRC_REG,0FFFFH  ; INIT CRC
07D6  BA 0100                   MOV     DX,256          ; FOR 256 BYTES
07D9                    W24:                            ; WR-BLK
07D9  26: 8A 07                 MOV     AL,ES:[BX]      ; READ BYTE FROM MEM
07DC  E8 0815 R                 CALL    WRITE_BYTE      ; WRITE IT TO CASSETTE
07DF  E3 02                     JCXZ    W25             ; UNLESS CX=0, ADVANCE PTRS & DEC
                                                        ; COUNT
07E1  43                        INC     BX              ; INC BUFFER POINTER
07E2  49                        DEC     CX              ; DEC BYTE COUNTER
07E3                    W25:                            ; SKIP-ADV
07E3  4A                        DEC     DX              ; DEC BLOCK CNT
07E4  7F F3                     JG      W24             ; LOOP TILL 256 BYTE BLOCK
                                                        ;  IS WRITTEN TO TAPE
                        ;-------------------------------------------------------------
                        ; WRITE CRC

                        ;       WRITE 1'S COMPLEMENT OF CRC REG TO CASSETTE
                        ;       WHICH IS CHECKED FOR CORRECTNESS WHEN THE BLOCK IS READ
                        ; REG AX IS MODIFIED
                        ;-------------------------------------------------------------
07E6  A1 0069 R                 MOV     AX,CRC_REG      ; WRITE THE ONE'S COMPLEMENT OF THE
                                                        ;  TWO BYTE CRC TO TAPE
07E9  F7 D0                     NOT     AX              ; FOR 1'S COMPLEMENT
07EB  50                        PUSH    AX              ; SAVE IT
07EC  86 E0                     XCHG    AH,AL           ; WRITE MS BYTE FIRST
07EE  E8 0815 R                 CALL    WRITE_BYTE      ; WRITE IT
```

```
07F1  58                        POP     AX              ; GET IT BACK
07F2  E8 0815 R                 CALL    WRITE_BYTE      ; NOW WRITE LS BYTE
07F5  0B C9                     OR      CX,CX           ; IS BYTE COUNT EXHAUSTED?
07F7  75 D7                     JNZ     WR_BLOCK        ; JUMP IF NOT DONE YET
07F9  51                        PUSH    CX              ; SAVE REG CX
07FA  FB                        STI                     ; RE-ENABLE INTERUPTS
07FB  B9 0020                   MOV     CX, 32          ; WRITE OUT TRAILER BITS
07FE                    W26:                            ; TRAIL-LOOP
07FE  F9                        STC
07FF  E8 082C R                 CALL    WRITE_BIT
0802  E2 FA                     LOOP    W26             ; WRITE UNTIL TRAILER WRITTEN
0804  59                        POP     CX              ; RESTORE REG CX
0805  B0 B0                     MOV     AL, 0B0H        ; TURN TIMER2 OFF
0807  E6 43                     OUT     TIM_CTL, AL
0809  B8 0001                   MOV     AX, 1
080C  E8 0842 R                 CALL    W31             ; SET_TIMER
080F  E8 0697 R                 CALL    MOTOR_OFF       ; TURN MOTOR OFF
0812  2B C0                     SUB     AX,AX           ; NO ERRORS REPORTED ON WRITE
0814  C3                        RET                     ; FINISHED
0815                    WRITE_BLOCK     ENDP
                        ;--------------------------------
                        ; WRITE A BYTE TO CASSETTE.
                        ; BYTE TO WRITE IS IN REG AL.
                        ;--------------------------------
0815                    WRITE_BYTE      PROC    NEAR
0815  51                        PUSH    CX              ; SAVE REGS CX,AX
0816  50                        PUSH    AX
0817  8A E8                    MOV     CH,AL           ; AL=BYTE TO WRITE.
                                                        ;   (MS BIT WRITTEN FIRST)
0819  B1 08                     MOV     CL,8            ; FOR 8 DATA BITS IN BYTE.
                                                        ;   NOTE: TWO EDGES PER BIT
081B                    W27:                            ; DISASSEMBLE THE DATA BIT
081B  D0 D5                     RCL     CH,1            ; ROTATE MS BIT INTO CARRY
081D  9C                        PUSHF                   ; SAVE FLAGS.
                                                        ;   NOTE: DATA BIT IS IN CARRY
081E  E8 082C R                 CALL    WRITE_BIT       ; WRITE DATA BIT
0821  9D                        POPF                    ; RESTORE CARRY FOR CRC CALC
0822  E8 0849 R                 CALL    CRC_GEN         ; COMPUTE CRC ON DATA BIT
0825  FE C9                     DEC     CL              ; LOOP TILL ALL 8 BITS DONE
0827  75 F2                     JNZ     W27             ; JUMP IF NOT DONE YET
0829  58                        POP     AX              ; RESTORE REGS AX,CX
082A  59                        POP     CX
082B  C3                        RET                     ; WE ARE FINISHED

082C                    WRITE_BYTE      ENDP
                        ;----------------------------------------
082C                    WRITE_BIT       PROC    NEAR
                        ; PURPOSE:
                        ;
                        ;  TO WRITE A DATA BIT TO CASSETTE
                        ;  CARRY FLAG CONTAINS DATA BIT
                        ;  I.E. IF SET   DATA BIT IS A ONE
                        ;       IF CLEAR DATA BIT IS A ZERO
                        ;
                        ;  NOTE: TWO EDGES ARE WRITTEN PER BIT
                        ;        ONE BIT  HAS 500 USEC BETWEEN EDGES
                        ;            FOR A 1000 USEC PERIOD (1 MILLISEC)
                        ;
                        ;        ZERO BIT HAS 250 USEC BETWEEN EDGES
                        ;            FOR A  500 USEC PERIOD (.5 MILLISEC)
                        ; CARRY FLAG IS DATA BIT
                        ;----------------------------------------
082C  B8 04A0                   MOV     AX,1184         ;ASSUME IT'S A '1'
082F  72 03                     JC      W28             ; SET AX TO NOMINAL ONE SIZE
0831  B8 0250                   MOV     AX,592          ; JUMP IF ONE BIT
0834                            JMP     W28             ; NO, SET TO NOMINAL ZERO SIZE
0834  50                W28:                            ; WRITE-BIT-AX
                                PUSH    AX              ;WRITE BIT WITH PERIOD EQ TO VALUE
                                                        ; AX
0835  E4 62             W29:   IN      AL,PORT_C       ;INPUT TIMER_0 OUTPUT
0837  24 20                    AND     AL,020H
0839  74 FA                    JZ      W29             ;LOOP TILL HIGH
083B  E4 62             W30:   IN      AL,PORT_C       ;NOW WAIT TILL TIMER'S OUTPUT IS
                                                        ; LOW
083D  24 20                    AND     AL,020H
083F  75 FA                    JNZ     W30

                                                        ;RELOAD TIMER WITH PERIOD
                                                        ;FOR NEXT DATA BIT
0841  58                        POP     AX              ;RESTORE PERIOD COUNT
0842                    W31:                            ; SET TIMER
0842  E6 42                     OUT     042H, AL        ; SET LOW BYTE OF TIMER 2
0844  8A C4                     MOV     AL, AH
0846  E6 42                     OUT     042H, AL        ; SET HIGH BYTE OF TIMER 2
0848  C3                        RET
0849                    WRITE_BIT       ENDP
                        ;----------------------------------------
0849                    CRC_GEN         PROC    NEAR
                        ;  UPDATE CRC REGISTER WITH NEXT DATA BIT
                        ;  CRC IS USED TO DETECT READ ERRORS
                        ;  ASSUMES DATA BIT IS IN CARRY
                        ;  REG AX IS MODIFIED
                        ;  FLAGS ARE MODIFIED
                        ;----------------------------------------
0849  A1 0069 R                 MOV     AX,CRC_REG
                                                        ;THE FOLLOWING INSTUCTIONS
                                                        ;WILL SET THE OVERFLOW FLAG
                                                        ;IF CARRY AND MS BIT OF CRC

                                                        ;ARE UNEQUAL
084C  D1 D8                     RCR     AX,1
084E  D1 D0                     RCL     AX,1
0850  F8                        CLC                     ;CLEAR CARRY
0851  71 04                     JNO     W32             ;SKIP IF NO OVERFLOW
```

```
0853  35 0810                    XOR     AX,0810H        ;IF DATA BIT XORED WITH
                                                         ; CRC REG BIT 15 IS ONE
                                                         ;THEN XOR CRC REG WITH
                                                         ; 0810H
0856  F9                         STC                     ;SET CARRY
0857  D1 D0                W32:  RCL     AX,1            ;ROTATE CARRY (DATA BIT)
                                                         ;INTO CRC REG
0859  A3 0069 R                  MOV     CRC_REG,AX      ;UPDATE CRC_REG
085C  C3                         RET                     ;FINISHED
085D                     CRC_GEN ENDP
                         ;-----------------------------------------------------------------------
085D                     BEGIN_OP PROC    NEAR           ; START TAPE AND DELAY
085D  E8 068E R                  CALL    MOTOR_ON        ;TURN ON MOTOR
0860  B3 42                      MOV     BL,42H          ;DELAY FOR TAPE DRIVE
                                                         ;TO GET UP TO SPEED  (1/2 SEC)
0862  B9 0700            W33:    MOV     CX,700H         ;INNER LOOP= APPROX. 10 MILLISEC
0865  E2 FE              W34:    LOOP    W34
0867  FE CB                      DEC     BL
0869  75 F7                      JNZ     W33
086B  C3                         RET
086C                     BEGIN_OP ENDP
08D0                             ORG     BEGIN+08D0H
08D0                     CODE    ENDS
                                 END
```

```
            ;-----------------------------------------------------
            ;
            ;
            ; KBDNMI - KEYBOARD NMI INTERRUPT ROUTINE
            ;
            ;     THIS ROUTINE OBTAINS CONTROL UPON AN NMI INTERRUPT, WHICH
            ;     OCCURS UPON A KEYSTROKE FROM THE KEYBOARD.
            ;
            ;     THIS ROUTINE WILL DE-SERIALIZE THE BIT STREAM IN ORDER TO
            ;     GET THE KEYBOARD SCAN CODE ENTERED.  IT THEN ISSUES INT 41
            ;     PASSING THE SCAN CODE IN AL TO THE KEY PROCESSOR. UPON RETURN
            ;     IT RE-ENABLES NMI AND RETURNS TO SYSTEM (IRET).
            ;
            ;-----------------------------------------------------
0000                        ASSUME CS:CODE,DS:DATA
0000        KBDNMI  PROC    FAR
            ;-----------DISABLE INTERRUPTS
0000  FA                   CLI
            ;-----------SAVE REGS & DISABLE NMI
0001  56                   PUSH    SI
0002  57                   PUSH    DI
0003  50                   PUSH    AX              ; SAVE REGS
0004  53                   PUSH    BX
0005  51                   PUSH    CX
0006  52                   PUSH    DX
0007  1E                   PUSH    DS
0008  06                   PUSH    ES
            ;-----------INIT COUNTERS
0009  BE 0008              MOV     SI,8            ; SET UP 8 OF DATA BITS
000C  32 DB                 XOR     BL,BL           ; INIT. PARITY COUNTER
            ;-----------SAMPLE 5 TIMES TO  VALIDATE START BIT
000E  32 E4                 XOR     AH,AH
0010  B9 0005               MOV     CX,5            ; SET COUNTER
0013  E4 62         I1:     IN      AL,PORT_C       ; GET SAMPLE
0015  A8 40                 TEST    AL,40H          ; TEST IF 1
0017  74 02                 JZ      I2              ; JMP IF 0
0019  FE C4                 INC     AH              ; KEEP COUNT OF 1'S
001B  E2 F6         I2:     LOOP    I1              ; KEEP SAMPLING
001D  80 FC 03              CMP     AH,3            ; VALID START BIT ?
0020  73 03                 JNB     I25             ; JUMP IF OK
0022  E9 00B0 R             JMP     I8              ; INVALID (SYNC ERROR) NO AUDIO
                                                    ; OUTPUT
            ;-----------VALID START BIT, LOOK FOR TRAILING EDGE
0025  B9 0032       I25:    MOV     CX,50           ; SET UP WATCHDOG TIMEOUT
0028  E4 62         I3:     IN      AL,PORT_C       ; GET SAMPLE
002A  A8 40                 TEST    AL,40H          ; TEST IF 0
002C  74 05                 JZ      I5              ; JMP IF TRAILING EDGE FOUND
002E  E2 F8                 LOOP    I3              ; KEEP LOOKING FOR TRAILING EDGE
0030  EB 7E 90              JMP     I8              ; SYNC ERROR (STUCK ON 1'S)
            ;-----------READ CLOCK TO SET START OF BIT TIME
0033  B0 40         I5:     MOV     AL,40H          ; READ CLOCK
0035  E6 43                 OUT     TIM_CTL,AL      ; M
0037  90                    NOP                     ; M
0038  90                    NOP                     ; M
0039  E4 41                 IN      AL,TIMER+1      ; M
003B  8A E0                 MOV     AH,AL           ; M
003D  E4 41                 IN      AL,TIMER+1      ; M
003F  86 E0                 XCHG    AH,AL           ; M
0041  8B F8                 MOV     DI,AX           ; SAVE CLOCK TIME IN DI
            ;-----------VERIFY VALID TRANSITION
0043  B9 0004               MOV     CX,4            ; SET COUNTER
0046  E4 62         I6:     IN      AL,PORT_C       ; GET SAMPLE
0048  A8 40                 TEST    AL,40H          ; TEST IF 0
004A  75 64                 JNZ     I8              ; JMP IF INVALID TRANSITION (SYNC)
004C  E2 F8                 LOOP    I6              ; KEEP LOOKING FOR VALID TRANSITION
            ;-----------SET UP DISTANCE TO MIDDLE OF 1ST DATA BIT
004E  BA 0220               MOV     DX,544          ; 310 USEC AWAY (.838 US / CT)
            ;-------START LOOKING FOR TIME TO READ DATA BITS AND ASSEMBLE BYTE
0051  E8 00D3 R     I7:     CALL    I30
0054  BA 020E               MOV     DX,526          ; SET NEW DISTANCE TO NEXT HALF BIT
0057  50                    PUSH    AX              ; SAVE 1ST HALF BIT
0058  E8 00D3 R             CALL    I30
005B  8A C8                 MOV     CL,AL           ; PUT 2ND HALF BIT IN CL
005D  58                    POP     AX              ; RESTORE 1ST HALF BIT
005E  3A C8                 CMP     CL,AL           ; ARE THEY OPPOSITES ?
0060  74 59                 JE      I9              ; NO, PHASE ERROR
            ;-----------VALID DATA BIT, PLACE IN SCAN BYTE
0062  D0 EF                 SHR     BH,1            ; SHIFT PREVIOUS BITS
0064  0A F8                 OR      BH,AL           ; OR IN NEW DATA BIT
0066  4E                    DEC     SI              ; DECREMENT DATA BIT COUNTER
0067  75 E8                 JNZ     I7              ; CONTINUE FOR MORE DATA BITS
            ;-----------WAIT FOR TIME TO SAMPLE PARITY BIT
0069  E8 00D3 R             CALL    I30
006C  50                    PUSH    AX              ; SAVE 1ST HALF BIT
006D  E8 00D3 R             CALL    I30
0070  8A C8                 MOV     CL,AL           ; PUT 2ND HALF BIT IN CL
0072  58                    POP     AX              ; RESTORE 1ST HALF BIT
0073  3A C8                 CMP     CL,AL           ; ARE THEY OPPOSITES ?
0075  74 44                 JE      I9              ; NO, PHASE ERROR
            ;-----------VALID PARITY BIT, CHECK PARITY
0077  80 E3 01              AND     BL,1            ; CHECK IF ODD PARITY
007A  74 3F                 JZ      I9              ; JMP IF PARITY ERROR
            ;-----------VALID CHARACTER, SEND TO CHARACTER PROCESSING
007C  8A C7                 MOV     AL,BH           ; PLACE SCAN CODE IN AL
007E  E8 0000 E             CALL    DDS
0081  F6 06 0338 R 04       TEST    JKB_FLAG_2,NMI_FLG
0086  74 15                 JZ      I7_2
0088  A8 80                 TEST    AL,80H
008A  74 05                 JZ      I7_1
008C  FB                    STI                     ; ENABLE INTERRUPTS
008D  CD 48                 INT     48H
008F  EB 1F                 JMP     SHORT I8
0091
0091  BB 0040       I7_1:   MOV     BX,40H          ; DURATION OF ERROR BEEP
```

```
0094  B9 0048                    MOV     CX,48H          ; FREQUENCY OF TONE
0097  E8 0000 E                  CALL    KB_NOISE        ; BUFFER FULL BEEP
009A  FB                        STI                     ; ENABLE INTERRUPTS
009B  EB 13                     JMP     SHORT I8
009D                    I7_2:
009D  80 0E 0338 R 04           OR      JKB_FLAG_2,NMI_FLG
00A2  8A D8                     MOV     BL,AL           ; STORE AL
00A4  E4 A0                     IN      AL,0A0H         ; ENABLE NMI
00A6  8A C3                     MOV     AL,BL           ; RESTORE AL
00A8  FB                        STI                     ; ENABLE INTERRUPTS
00A9  CD 48                     INT     48H             ; CHARACTER PROCESSING
00AB  80 26 0338 R FB           AND     JKB_FLAG_2,NOT NMI_FLG
                        ;-----------RESTORE REGS AND RE-ENABEL NMI
00B0  07                        POP     ES              ; RESTORE REGS
00B1  1F                        POP     DS
00B2  5A                        POP     DX
00B3  59                        POP     CX
00B4  5B                        POP     BX
00B5  E4 A0                     IN      AL,0A0H         ; ENABLE NMI
00B7  58                        POP     AX
00B8  5F                        POP     DI
00B9  5E                        POP     SI
00BA  CF                        IRET                    ; RETURN TO SYSTEM
                        ;-----------PARITY, SYNCH OR PHASE ERROR. OUTPUT MISSED KEY BEEP
00BB  E8 0000 E                 CALL    DDS             ; SETUP ADDRESSING
00BE  83 FE 08                  CMP     SI,8            ; ARE WE ON THE FIRST DATA BIT?
00C1  74 ED                     JE      I8              ; NO AUDIO FEEDBACK (MIGHT BE A
                                                        ; ..GLITCH)
00C3  F6 06 0018 R 01           TEST    KB_FLAG_1,01H   ; CHECK IF TRANSMISSION ERRORS
                                                        ; ..ARE TO BE REPORTED
00C8  75 03                     JNZ     I10             ; 1=DO NOT BEEP, 0=BEEP
00CA  E8 0AF8 R                 CALL    ERROR_BEEP      ; CALL ERROR BEEP ROUTINE
00CD  FE 06 0012 R              INC     KBD_ERR         ; KEEP TRACK OF KEYBOARD ERRORS
00D1  EB DD                     JMP     SHORT I8        ; RETURN FROM INTERRUPT
00D3                    KBDNMI  ENDP
00D3                    I30     PROC    NEAR
00D3  B0 40             I31:    MOV     AL,40H          ; READ CLOCK
00D5  E6 43                     OUT     TIM_CTL,AL      ; *
00D7  90                        NOP                     ; *
00D8  90                        NOP                     ; *
00D9  E4 41                     IN      AL,TIMER+1      ; *
00DB  8A E0                     MOV     AH,AL           ; *
00DD  E4 41                     IN      AL,TIMER+1      ; *
00DF  86 E0                     XCHG    AH,AL           ; *
00E1  8B CF                     MOV     CX,DI           ; GET LAST CLOCK TIME
00E3  2B C8                     SUB     CX,AX           ; SUB CURRENT TIME
00E5  3B CA                     CMP     CX,DX           ; IS IT TIME TO SAMPLE ?
00E7  72 EA                     JC      I31             ; NO, KEEP LOOKING AT TIME
00E9  2B CA                     SUB     CX,DX           ; UPDATE 0 OF COUNTS OFF
00EB  8B F8                     MOV     DI,AX           ; SAVE CURRENT TIME AS LAST TIME
00ED  03 F9                     ADD     DI,CX           ; ADD DIFFERENCE FOR NEXT TIME
                        ;-----------START SAMPLING DATA BIT (5 SAMPLES)
00EF  B9 0005                   MOV     CX,5            ; SET COUNTER
                        ;--------------------------------------------------------
                        ;
                        ; SAMPLE LINE
                        ;
                        ;     PORT_C IS SAMPLED CX TIMES AND IF THER ARE 3 OR MORE 1"S
                        ;     THEN 80H IS RETURNED IN AL, ELSE 00H IS RETURNED IN AL.
                        ;     PARITY COUNTER IS MAINTAINED IN ES.
                        ;
                        ;--------------------------------------------------------
00F2  32 E4                     XOR     AH,AH           ; CLEAR COUNTER
00F4  E4 62             I32:    IN      AL,PORT_C       ; GET SAMPLE
00F6  A8 40                     TEST    AL,40H          ; TEST IF 1
00F8  74 02                     JZ      I33             ; JMP IF 0
00FA  FE C4                     INC     AH              ; KEEP COUNT OF 1'S
00FC  E2 F6             I33:    LOOP    I32             ; KEEP SAMPLING
00FE  80 FC 03                  CMP     AH,3            ; VALID 1 ?
0101  72 05                     JB      I34             ; JMP IF NOT VALID 1
0103  B0 80                     MOV     AL,080H         ; RETURN 80H IN AL (1)
0105  FE C3                     INC     BL              ; INCREMENT PARITY COUNTER
0107  C3                        RET                     ; RETURN TO CALLER
0108  32 C0             I34:    XOR     AL,AL           ; RETURN 0 IN AL (0)
010A  C3                        RET                     ; RETURN TO CALLER
010B                    I38     ENDP
                        ;--------------------------------------------------------
                        ;KEY62_INT
                        ;       THE PURPOSE OF THIS ROUTINE IS TO TRANSLATE SCAN CODES AND
                        ;       SCAN CODE COMBINATIONS FROM THE 62 KEY KEYBOARD TO THEIR
                        ;       EQUIVILENTS ON THE 83 KEY KEYBOARD.  THE SCAN CODE IS
                        ;       PASSED IN AL. EACH SCAN CODE PASSED EITHER TRIGGERS ONE OR
                        ;       MORE CALLS TO INTERRUPT 9 OR SETS FLAGS TO RETAIN KEYBOARD
                        ;       STATUS.  WHEN INTERRUPT 9 IS CALLED THE TRANSLATED SCAN
                        ;       CODES ARE PASSED TO IT IN AL.  THE INTENT OF THIS CODE WAS
                        ;       TO KEEP INTERRUPT 9 INTACT FROM ITS ORIGIN IN THE PC FAMILY
                        ;       THIS ROUTINE IS IN THE FRONT END OF INTERRUPT 9 AND
                        ;       TRANSFORMS A 62 KEY KEYBOARD TO LOOK AS IF IT WERE AN 83
                        ;       KEY VERSION.
                        ;       IT IS ASSUMED THAT THIS ROUTINE IS CALLED FROM THE NMI
                        ;       DESERIALIZATION ROUTINE AND THAT ALL REGISTERS WERE SAVED
                        ;       IN THE CALLING ROUTINE.  AS A CONSEQUENCE ALL REGISTERS ARE
                        ;       DESTROYED.
                        ;--------------------------------------------------------
                        ;EQUATES
= 0080                  BREAK_BIT       EQU     80H
= 0054                  FN_KEY          EQU     54H
= 0055                  PHK             EQU     FN_KEY+1
= 0056                  EXT_SCAN        EQU     PHK+1   ; BASE CODE FOR SCAN CODES
                                                        ; EXTENDING BEYOND 85
= 006A                  EXT_SCAN_END    EQU     06AH    ; END OF EXTENDET SCAN CODE (=106)
= 00FF                  AND_MASK        EQU     0FFH    ; USED TO SELECTIVELY REMOVE BITS
= 001F                  CLEAR_FLAGS     EQU     AND_MASK - (FN_FLAG+FN_BREAK+FN_PENDING)
                        ;SCAN CODES
```

```
= 0010                  Q_KEY           EQU     16
= 0031                  N_KEY           EQU     49
= 001C                  ENTER_KEY       EQU     28
= 0023                  H_KEY           EQU     35
= 0014                  T_KEY           EQU     20
= 0048                  UP_ARROW        EQU     72
= 0050                  DOWN_ARROW      EQU     80
= 004B                  LEFT_ARROW      EQU     75
= 004D                  RIGHT_ARROW     EQU     77
= 000C                  MINUS           EQU     12
= 000D                  EQUALS          EQU     13
= 000B                  NUM_0           EQU     11
                        ;-----------------------------------------------------
                        ;NEW TRANSLATED SCAN CODES
                        ;-----------------------------------------------------
= 0010                  PAUSE           EQU     16      ; HOLD FUNCTION IS SPECIAL CASE.
= 0045                  NUM_LOCK        EQU     69
= 0047                  HOME            EQU     71
= 004F                  END_KEY         EQU     79
= 0049                  PAGE_UP         EQU     73
= 0051                  PAGE_DOWN       EQU     81
= 004A                  KEYPAD_MINUS    EQU     74
= 004E                  KEYPAD_PLUS     EQU     78
= 0093                  JITTKOU_KEY     EQU    147
= 0099                  WARIKOMI_KEY    EQU    153
= 009A                  SHURIYOU_KEY    EQU    154
                                ASSUME CS:CODE,DS:DATA
                        ;----TABLE OF VALID SCAN CODES
010B                    KB0     LABEL BYTE
                                DB Q_KEY, N_KEY
                                DB UP_ARROW, DOWN_ARROW, LEFT_ARROW, RIGHT_ARROW
                                DB ENTER_KEY,H_KEY,T_KEY
= 0009                  KB0LEN          EQU $ - KB0
                        ;----TABLE OF NEW SCAN CODES
0114                    KB1     LABEL BYTE
                                DB PAUSE, NUM_LOCK
                                DB HOME,END_KEY,PAGE_UP,PAGE_DOWN
                                DB JITTKOU_KEY,WARIKOMI_KEY,SHURIYOU_KEY
                        ;-----------------------------------------------
                        ;NOTE:  THERE IS A ONE TO ONE CORRESPONDENCE BETWEEN
                        ;       THE SIZE OF KB0 AND KB1.
                        ;-----------------------------------------------
                        ;TABLE OF NUMERIC KEYPAD SCAN CODES
                        ;       THESE SCAN CODES WERE NUMERIC KEYPAD CODES ON
                        ;       THE 102 KEY KEYBOARD.
                        ;-----------------------------------------------
011D                    NUM_CODES LABEL   BYTE
                                  DB      72H,73H,74H,75H,76H
                                  DB      77H,78H,79H,7AH,70H   ; 10 NUMBERS ON KEYPAD
                        ;-----------------------------------------------
                        ;EXTAB
                        ;       TABLE OF SCAN CODES FOR MAPPING EXTENDED SET
                        ;       OF SCAN CODES (SCAN CODES > 85).  THIS TABLE
                        ;       ALLOWS OTHER DEVICES TO USE THE KEYBOARD INTERFACE.
                        ;       IF THE DEVICE GENERATES A SCAN CODE > 85 THIS TABLE
                        ;       CAN BE USED TO MAP THE DEVICE TO THE KEYBOARD.  THE
                        ;       DEVICE ALSO HAS THE OPTION OF HAVING A UNIQUE SCAN
                        ;       CODE PUT IN THE KEYBOARD BUFFER (INSTEAD OF MAPPING
                        ;       TO THE KEYBOARD). THE EXTENDED SCAN CODE PUT IN THE
                        ;       BUFFER WILL BE CONTINUOUS BEGINNING AT 150.  A ZERO
                        ;       WILL BE USED IN PLACE OF AN ASCII CODE. (E.G. A
                        ;       DEVICE GENERATING SCAN CODE 86 AND NOT MAPPING 86
                        ;       TO THE KEYBOARD WILL HAVE A [150,0] PUT IN THE
                        ;       KEYBOARD BUFFER)
                        ;       TABLE FORMAT:
                        ;       THE FIRST BYTE IS A LENGTH INDICATING THE NUMBER
                        ;       OF SCAN CODES MAPPED TO THE KEYBOARD.  THE REMAINING
                        ;       ENTRIES ARE WORDS. THE FIRST BYTE (LOW BYTE) IS A
                        ;       SCAN CODE AND THE SECOND BYTE (HIGH BYTE) IS ZERO.
                        ;       A DEVICE GENERATING N SCAN CODES IS ASSUMED TO GENERATE THE
                        ;       FOLLOWING STREAM 86,87,88,....86+(N-1). THE SCAN CODE BYTES
                        ;       IN THE TABLE CORRESPOND TO THIS SET WITH THE FIRST DATA
                        ;       BYTE MATCHING 86, THE SECOND MATCHING 87 ETC.
                        ;   NOTES:
                        ;       (1) IF A DEVICE GENERATES A BREAK CODE, NOTHING IS
                        ;           PUT IN THE BUFFER.
                        ;       (2) A LENGTH OF 0 INDICATES THAT ZERO SCAN CODES HAVE BEEN
                        ;           MAPPED TO THE KEYBOARD AND ALL EXTENDED SCAN CODES WILL
                        ;           BE USED.
                        ;       (3) A DEVICE CAN MAP SOME OF ITS SCAN CODES TO THE KEYBOARD
                        ;           AND HAVE SOME ITS SCAN CODES IN THE EXTENDED SET.
                        ;-----------------------------------------------
0127                    EXTAB   LABEL BYTE
0127  14                       DB      20                      ; LENGTH OF TABLE
0128  0048 0049 004D 0051       DW      72,73,77,81,80,79,75,71,57,28
      0050 004F 004B 0047
      0039 001C
013C  0011 0012 001F 002D      ;DW      17,18,31,45,44,43,30,16,15,1
      002C 002B 001E 0010
      000F 0001
0150                    KEY62_INT PROC FAR
0150  FB                       STI                             ; FORWARD DIRECTION
0151  FC                       CLD                             ; SET UP ADDRESSING
0152  E8 0000 E                CALL    DDS                     ; SAVE SCAN CODE
0155  8A E0                    MOV     AH,AL                   ; ADJUST OUTPUT FOR USER
0157  E8 0331 R                CALL    TPM                     ; MODIFICATION
                               JNC     KBX0                    ; JUMP IF OK TO CONTINUE
015A  73 01                    IRET                            ; RETURN FROM INTERRUPT.
015C  CF
                        ;----EXTENDED SCAN CODE CHECK
015D  3C FF             KBX0:   CMP     AL,0FFH                 ; IS THIS AN OVERRUN CHAR?
015F  74 48                     JE      KB0_1                   ; PASS IT TO INTERRUPT 9
0161  24 7F                     AND     AL,AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
```

```
0163  3C 56                CMP     AL,EXT_SCAN     ; IS THIS A SCAN CODE > 85
0165  7C 3B                JL      KBX4            ; REPLACE BREAK BIT
0167  3C 6A                CMP     AL,EXT_SCAN_END ; IS THIS A SCAN CODE < 106
0169  7D 37                JGE     KBX4            ; REPLACE BREAK BIT
                      ;----SCAN CODE IS IN EXTENDED SET
016B  1E                   PUSH    DS
016C  33 F6                XOR     SI,SI
016E  8E DE                MOV     DS,SI
                           ASSUME  DS:ABS0
0170  C4 3E 0124 R         LES     DI,DWORD PTR EXST ; GET THE POINTER TO THE EXTENDED
                                                     ; SET
0174  26: 8A 0D            MOV     CL,BYTE PTR ES:[DI] ; GET LENGTH BYTE
0177  1F                   POP     DS
                           ASSUME  DS:DATA
                      ;----DOES SCAN CODE GET MAPPED TO KEYBOARD OR TO NEW EXTENDED SCAN
                      ;    CODES?
0178  2C 56                SUB     AL,EXT_SCAN     ; CONVERT TO BASE OF NEW SET
017A  FE C9                DEC     CL              ; LENGTH - 1
017C  3A C1                CMP     AL,CL           ; IS CODE IN TABLE?
017E  7F 14                JG      KBX1            ; JUMP IF SCAN CODE IS NOT IN TABLE
                      ;----GET SCAN CODE FROM TABLE
0180  47                   INC     DI              ; POINT DI PAST LENGTH BYTE
0181  8B D8                MOV     BX,AX
0183  32 FF                XOR     BH,BH           ; PREPARE FOR ADDING TO 16 BIT
                                                   ; REGISTER
0185  D1 E3                SHL     BX,1
0187  03 FB                ADD     DI,BX           ; OFFSET TO CORRECT TABLE ENTRY
0189  26: 8A 05            MOV     AL,BYTE PTR ES:[DI] ; TRANSLATED SCAN CODE IN AL
018C  3C 56                CMP     AL,EXT_SCAN     ; IS THIS A SCAN CODE > 85
018E  7C 12                JL      KBX4            ; REPLACE BREAK BIT
0190  3C 6A                CMP     AL,EXT_SCAN_END ; IS THIS A SCAN CODE < 106
0192  7F 0E                JG      KBX4            ; REPLACE BREAK BIT
                      ;----SCAN CODE GETS MAPPED TO EXTENDED SCAN CODES
0194  F6 C4 80      KBX1:  TEST    AH,BREAK_BIT    ; IS THIS A BREAK CODE?
0197  74 01                JZ      KBX2            ; MAKE CODE, PUT IN BUFFER
0199  CF                   IRET
019A  80 C4 40      KBX2:  ADD     AH,64           ; BREAK CODE, RETURN FROM INTERRUPT
019D  32 C0                XOR     AL,AL           ; EXTENDED SET CODES BEGIN AT 150
019F  CD 78                INT     78H             ; ZERO OUT ASCII VALUE (NUL)
01A1  CF                   IRET                    ; KANAKAN ROUTINE
01A2  80 E4 80      KBX4:  AND     AH,BREAK_BIT    ; MASK BREAK BIT ON ORIGINAL SCAN
01A5  0A C4                OR      AL,AH           ; UPDATE NEW SCAN CODE
01A7  8A E0                MOV     AH,AL           ; SAVE AL IN AH AGAIN
                      ;----83 KEY KEYBOARD FUNCTIONS SHIFT+PRTSC AND CTRL+NUMLOCK
01A9  3C 45         KB0_1: CMP     AL,NUM_KEY      ; IS THIS A NUMLOCK?
01AB  75 14                JNE     KB0_3           ; CHECK FOR PRTSC
01AD  F6 06 0017 R 04      TEST    KB_FLAG,CTL_SHIFT ; IS CTRL KEY BEING HELD DOWN?
01B2  74 0A                JZ      KB0_2           ; NUMLOCK WITHOUT CTRL, CONTINUE
01B4  F6 06 0017 R 08      TEST    KB_FLAG,ALT_SHIFT ; IS ALT KEY HELD CONCURRENTLY?
01B9  75 03                JNZ     KB0_2           ; PASS IT ON
01BB  E9 02F0 R            JMP     KB16_1          ; PUT KEYBOARD IN HOLD STATE
01BE  E9 027A R     KB0_2: JMP     CONT_INT        ; CONTINUE WITH INTERRUPT 48H
                      ;----CHECK FOR PRTSC
01C1  3C 37         KB0_3: CMP     AL,55
01C3  75 11                JNZ     KB2             ; IS THIS A PRTSC KEY?
01C5  F6 06 0017 R 03      TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; NOT A PRTSC KEY
                                                   ; EITHER SHIFT
01CA  74 F2                JZ      KB0_2           ; ACTIVE?
01CC  F6 06 0017 R 04      TEST    KB_FLAG,CTL_SHIFT ; PROCESS SCAN IN INT9
01D1  75 EB                JNZ     KB0_2           ; IS THE CTRL KEY PRESSED?
01D3  E9 030F R            JMP     PRTSC           ; NOT A VALID PRTSC (PC COMPATIBLE)
                      ;----FUNCTION KEY HANDLER    ; HANDLE THE PRINT SCREEN FUNCTION
01D6  8A E0         KB2:   MOV     AH, AL          ; SAVE CHARACTER
01D8  24 7F                AND     AL, AND_MASK - BREAK_BIT ; MASK BREAK BIT
01DA  3C 54                CMP     AL, FN_KEY      ; CHECK FOR FUNCTION KEY
01DC  75 23                JNZ     KB4             ; JUMP IF NOT FUNCTION KEY
01DE  F6 C4 80             TEST    AH, BREAK_BIT   ; IS THIS A FUNCTION BREAK
01E1  75 0B                JNZ     KB3             ; JUMP IF FUNCTION BREAK
01E3  80 26 0088 R 1F      AND     KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL PREVIOUS
                                                   ; FUNCTIONS
01E8  80 0E 0088 R A0      OR      KB_FLAG_2, FN_FLAG + FN_PENDING
01ED  CF                   IRET                    ; RETURN FROM INTERRUPT
                      ;----FUNCTION BREAK
01EE  F6 06 0088 R 20 KB3: TEST    KB_FLAG_2,FN_PENDING
01F3  75 06                JNZ     KB3_1           ; JUMP IF FUNCTION IS PENDING
01F5  80 26 0088 R 1F      AND     KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL FLAGS
01FA  CF                   IRET
01FB  80 0E 0088 R 40 KB3_1: OR    KB_FLAG_2,FN_BREAK ; SET BREAK FLAG
0200  CF              KB3_2: IRET                   ; RETURN FROM INTERRUPT
                      ;----CHECK IF FUNCTION FLAG ALREADY SET
0201  3C 55          KB4:   CMP     AL,PHK
0203  74 FB                JZ      KB3_2           ; IS THIS A PHANTOM KEY?
0205  F6 06 0088 R 98 KB4_0: TEST  KB_FLAG_2,FN_FLAG+FN_LOCK ; JUMP IF PHANTOM SEQUENCE
                                                   ; ARE WE IN FUNCTION
020A  75 21                JNZ     KB5             ; STATE?
                      ;----CHECK IF NUM_STATE IS ACTIVE
020C  F6 06 0017 R 20      TEST    KB_FLAG,NUM_STATE
0211  74 16                JZ      KB4_1
0213  3C 0B                CMP     AL,NUM_0        ; JUMP IF NOT IN NUM_STATE
0215  77 12                JA      KB4_1           ; ARE WE IN NUMERIC KEYPAD REGION?
0217  FE C8                DEC     AL              ; JUMP IF NOT IN KEYPAD
0219  74 0E                JZ      KB4_1           ; CHECK LOWER BOUND OF RANGE
                      ;----TRANSLATE SCAN CODE TO NUMERIC KEYPAD
021B  FE C8                DEC     AL              ; JUMP IF NOT IN RANGE (ESC KEY)
021D  BB 011D R            MOV     BX,OFFSET NUM_CODES
0220  2E: D7                XLAT    CS:NUM_CODES    ; AL IS OFFSET INTO TABLE
0222  80 E4 80             AND     AH,BREAK_BIT    ; NEW SCAN CODE IS IN AL
                                                   ; ISOLATE BREAK BIT ON ORIGINAL
0225  0A C4                OR      AL,AH           ; SCAN CODE
0227  EB 51                JMP     SHORT CONT_INT  ; UPDATE KEYPAD SCAN CODE
0229  8A C4         KB4_1: MOV     AL,AH           ; CONTINUE WITH INTERRUPT
022B  EB 4D                JMP     SHORT CONT_INT  ; GET BACK BREAK BIT IF SET
                      ;----CHECK FOR VALID FUNCTION KEY
022D                  KB5:
```

```
022D  3C 01                       CMP     AL,1            ; CHECK FOR ESC KEY (=1)
022F  75 25                       JNE     KB7             ; NOT ESCAPE KEY
                            ;----ESCAPE KEY, LOCK KEYBOARD IN FUNCTION LOCK
0231  F6 C4 80                    TEST    AH,BREAK_BIT    ; IS THIS A BREAK CODE?
0234  75 2C                       JNZ     KB8             ; NO PROCESSING FOR ESCAPE BREAK
0236  F6 06 0088 R 80             TEST    KB_FLAG_2,FN_FLAG ; TOGGLES ONLY WHEN FN HELD
                                                          ; CONCURRENTLY
023B  74 25                       JZ      KB8             ; NOT HELD CONCURRENTLY
023D  F6 06 0088 R 40             TEST    KB_FLAG_2,FN_BREAK ; HAS THE FUNCTION KEY BEEN
                                                          ; RELEASED?
0242  75 1E                       JNZ     KB8             ; CONTINUE IF RELEASED.  PROCESS AS
                                                          ; ESC
0244  F6 06 0017 R 03             TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT?
0249  74 17                       JZ      KB8             ; NOT HELD DOWN
024B  80 36 0088 R 10             XOR     KB_FLAG_2,FN_LOCK ; TOGGLE STATE
0250  80 26 0088 R 1F             AND     KB_FLAG_2,CLEAR_FLAGS ; TURN OFF OTHER STATES
0255  CF                          IRET                    ; RETURN FROM INTERUPT
0256                        ;----CHECK TABLE FOR OTHER VALID SCAN CODES
0256  0E                   KB7:
0257  07                          PUSH    CS
0258  BF 010B R                   POP     ES              ; ESTABLISH ADDRESS OF TABLE
025B  B9 0009                      MOV     DI, OFFSET KB0  ; BASE OF TABLE
025E  F2/ AE                       MOV     CX, KB0LEN      ; LENGTH OF TABLE
0260  74 1D                       REPNE   SCASB           ; SEARCH TABLE FOR A MATCH
                                   JE      KB10            ; JUMP IF MATCH
0262  F6 06 0088 R 40       ;----ILLEGAL CHARACTER
0267  74 0F                 KB8:  TEST    KB_FLAG_2,FN_BREAK ; HAS BREAK OCCURED?
                                   JZ      KB9             ; FUNCTION KEY HAS NOT BEEN
                                                           ; RELEASED
0269  F6 C4 80                     TEST    AH,BREAK_BIT    ; IS THIS A BREAK OF AN ILLEGAL
026C  75 0A                        JNZ     KB9             ; DON'T RESET FLAGS ON ILLEGAL
                                                           ; BREAK
026E  80 26 0088 R 1F       KB85:  AND     KB_FLAG_2,CLEAR_FLAGS ; NORMAL STATE
0273  C6 06 0087 R 00              MOV     CUR_FUNC,0      ; RETRIEVE ORIGINAL SCAN CODE
                            ;----FUNCTION BREAK IS NOT SET
0278  8A C4                 KB9:   MOV     AL,AH           ; RETRIEVE ORIGINAL SCAN CODE
027A                        CONT_INT:
027A  E6 60                        OUT     KBPORT,AL
027C  CD 09                        INT     9H              ; ISSUE KEYBOARD INTERRUPT
027E                        RET_INT:
027E  CF                           IRET
                            ;----BEFORE TRANSLATION CHECK FOR ALT+FN+N_KEY AS NUM LOCK
027F  3C 31                 KB10:  CMP     AL,N_KEY        ; IS THIS A POTENTIAL NUMLOCK?
0281  75 07                        JNE     KB10_1          ; NOT A NUMKEY, TRANSLATE IT
0283  F6 06 0017 R 08              TEST    KB_FLAG,ALT_SHIFT ; ALT HELD DOWN ALSO?
0288  74 D8                        JZ      KB8             ; TREAT AS ILLEGAL COMBINATION
028A  B9 010C R             KB10_1: MOV    CX, OFFSET KB0 + 1 ; GET OFFSET TO TABLE
028D  2B F9                        SUB     DI, CX          ; UPDATE INDEX TO NEW SCAN CODE
                                                           ; TABLE
028F  2E: 8A 85 0114 R             MOV     AL, CS:KB1[DI]  ; MOV NEW SCAN CODE INTO REGISTER
                            ;----TRANSLATED CODE IN AL OR AH OFFSET TO THE TABLE "SCAN"
0294  F6 C4 80              KB12:  TEST    AH,BREAK_BIT    ; IS THIS A BREAK CHAR?
0297  74 31                        JZ      KB13            ; JUMP IF MAKE CODE
                            ;----CHECK FOR TOGGLE KEY
0299  3C 45                        CMP     AL,NUM_LOCK     ; IS THIS A NUM LOCK?
029B  75 08                        JNZ     KB12_2          ; JUMP IF NOT A TOGGLE KEY
029D  0C 80                        OR      AL,80H          ; TURN ON BREAK BIT
029F  E6 60                        OUT     KBPORT,AL
02A1  CD 09                        INT     9H              ; TOGGLE STATE
02A3  24 7F                        AND     AL,AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
02A5  F6 06 0088 R 40       KB12_2: TEST   KB_FLAG_2,FN_BREAK ; HAS FUNCTION BREAK OCCURED?
02AA  74 11                        JZ      KB12_3          ; JUMP IF BREAK HAS NOT OCCURED
02AC  3A 06 0087 R                 CMP     AL,CUR_FUNC     ; IS THIS A BREAK OF OLD VALID
                                                           ; FUNCTION
02B0  75 CC                        JNE     RET_INT         ; ALLOW FURTHER CURRENT FUNCTIONS
02B2  80 26 0088 R 1F              AND     KB_FLAG_2,CLEAR_FLAGS
02B7                        KB12_20:
02B7  C6 06 0087 R 00              MOV     CUR_FUNC,0      ; CLEAR CURRENT FUNCTION
02BC  CF                           IRET                    ; RETURN FROM INTERRUPT
02BD  3A 06 0087 R          KB12_3: CMP    AL,CUR_FUNC     ; IS THIS BREAK OF FIRST FUNCTION?
02C1  75 BB                        JNE     RET_INT         ; IGNORE
02C3  80 26 0088 R DF              AND     KB_FLAG_2,AND_MASK-FN_PENDING ; TURN OFF PENDING
                                                           ; FUNCTION
02C8  EB ED                        JMP     KB12_20         ; CLEAR CURRENT FUNCTION AND RETURN
                            ;----VALID MAKE KEY HAS BEEN PRESSED
02CA  F6 06 0088 R 40       KB13:  TEST    KB_FLAG_2,FN_BREAK ; CHECK IF FUNCTION KEY HAS BEEN
                                                           ; PRESSED
02CF  74 0D                        JZ      KB14_1          ; JUMP IF NOT SET
                            ;----FUNCTION BREAK HAS ALREADY OCCURED
02D1  80 3E 0087 R 00              CMP     CUR_FUNC,0      ; IS THIS A NEW FUNCTION?
02D6  74 06                        JZ      KB14_1          ; INITIALIZE NEW FUNCTION
02D8  38 06 0087 R                 CMP     CUR_FUNC,AL     ; IS THIS NON-CURRENT FUNCTION
02DC  75 90                        JNZ     KB85            ;JUMP IF NO FUNCTION IS PENDING
                                                           ; ..TO RETRIEVE ORIGINAL SCAN CODE
02DE  A2 0087 R             ;----CHECK FOR SCAN CODE GENERATION SEQUENCE
02E1  3C 93                 KB14_1: MOV    CUR_FUNC,AL     ; INITIALIZE CURRENT FN
02E3  72 07                        CMP     AL,JITTKOU_KEY  ; IS THIS A ENTER,H OR T KEY ?
02E5  8A E0                        JB      KB16            ; NO. JMP KB16
02E7  32 C0                        MOV     AH,AL           ; EXTENDED SET CODE
02E9  CD 78                        XOR     AL,AL           ; CLEAR AL
02EB  CF                           INT     78H             ; KANAKAN ROUTINE
02EC                               IRET                    ;
02EC  3C 10                 KB16:                          ; IS THIS THE HOLD FUNCTION
02EE  75 8A                        CMP     AL,PAUSE        ; NO. JMP CONT_INT
                                   JNE     CONT_INT
02F0  F6 06 0018 R 08       ;----PUT KEYBOARD IN HOLD STATE ; CANNOT GO IN HOLD STATE IF
02F5  75 17                 KB16_1: TEST   KB_FLAG_1,HOLD_STATE ; ITS ACTIVE
02F7                                                       ; DONE WITH INTERRUPT
02FC  80 0E 0018 R 08              JNZ     KB16_2
0301  E4 A0                        OR      KB_FLAG_1,HOLD_STATE ; TURN ON HOLD FLAG
0303  F6 06 0018 R 08              AND     JKB_FLAG_2,0FBH ; RESET KEYBOARD LATCH
0308  80 80                        IN      AL,HMI_PORT
                            HOLD:  TEST    KB_FLAG_1,HOLD_STATE ; STILL IN HOLD STATE?
                                   MOV     AL,80H
```

```
030A  E6 A0                          OUT     NMI_PORT,AL     ; ENABLE NMI
030C  75 F5                          JNZ     HOLD            ; CONTINUE LOOPING UNTIL KEY IS
                                                             ; PRESSED
030E  CF                     KB16_2: IRET                    ; RETURN FROM INTERRUPT 48H
                             ;----PRINT SCREEN FUNCTION
030F  F6 06 0018 R 08        PRTSC:  TEST    KB_FLAG_1,HOLD_STATE ; IS HOLD STATE IN PROGRESS?
0314  74 06                          JZ      KB16_3          ; OK TO CONTINUE WITH PRTSC
0316  80 26 0018 R F7                AND     KB_FLAG_1,0FFH-HOLD_STATE ; TURN OFF FLAG
031B  CF                              IRET
031C                         KB16_3:
031C  80 26 0338 R FB                AND     JKB_FLAG_2,0FBH
0321  83 C4 06                       ADD     SP,3H2          ; GET RID OF CALL TO INTERRUPT 48H
0324  07                             POP     ES              ; POP REGISTERS THAT AREN'T
                                                             ; MODIFIED IN INT5
0325  1F                            POP     DS
0326  5A                            POP     DX
0327  59                            POP     CX
0328  5B                            POP     BX
0329  E4 A0                         IN      AL,NMI_PORT     ; RESET KEYBOARD LATCH
032B  CD 05                         INT     5H              ; ISSUE INTERRUPT
032D  58                            POP     AX
032E  5F                            POP     DI
032F  5E                            POP     SI              ; POP THE REST
0330  CF                            IRET
0331                         KEY62_INT ENDP
                             ;--------------------------------------------------------------
                             ;TYPAMATIC
                             ;       THIS ROUTINE WILL CHECK KEYBOARD STATUS BITS IN KB_FLAG_2
                             ;       AND DETERMINE WHAT STATE THE KEYBOARD IS IN.  APPROPRIATE
                             ;       ACTION WILL BE TAKEN.
                             ;INPUT
                             ;       AL= SCAN CODE OF KEY WHICH TRIGGERED NON-MASKABLE INTERRUPT
                             ;OUTPUT
                             ;       CARRY BIT = 1 IF NO ACTION IS TO BE TAKEN.
                             ;       CARRY BIT = 0 MEANS SCAN CODE IN AL SHOULD BE PROCESSED
                             ;                     FURTHER.
                             ;       MODIFICATIONS TO THE VARIABLES CUR_CHAR AND VAR_DELAY ARE
                             ;       MADE.  ALSO THE PUTCHAR BIT IN KB_FLAG_2 IS TOGGLED WHEN
                             ;       THE KEYBOARD IS IN HALF RATE MODE.
                             ;--------------------------------------------------------------
0331                         TPM     PROC    NEAR
0331  53                            PUSH    BX
0332  38 06 0085 R                  CMP     CUR_CHAR,AL     ; IS THIS A NEW CHARACTER?
0336  74 31                         JZ      TP2             ; JUMP IF SAME CHARACTER
                             ;----NEW CHARACTER CHECK FOR BREAK SEQUENCES
0338  A8 80                          TEST    AL,BREAK_BIT    ; IS THE NEW KEY A BREAK KEY?
033A  74 12                          JZ      TP0             ; JUMP IF NOT A BREAK
033C  24 7F                          AND     AL,07FH         ; CLEAR BREAK BIT
033E  38 06 0085 R                   CMP     CUR_CHAR,AL     ; IS NEW CHARACTER THE BREAK OF
                                                             ; LAST MAKE?
0342  8A C4                          MOV     AL,AH           ; RETRIEVE ORIGINAL CHARACTER
0344  75 05                          JNZ     TP              ; JUMP IF NOT THE SAME CHARACTER
0346  C6 06 0085 R 00                MOV     CUR_CHAR,00     ; CLEAR CURRENT CHARACTER
034B  F8                     TP:     CLC                     ; CLEAR CARRY BIT
034C  5B                             POP     BX
034D  C3                             RET                     ; RETURN
                             ;----INITIALIZE A NEW CHARACTER
034E  A2 0085 R              TP0:    MOV     CUR_CHAR,AL     ; SAVE NEW CHARACTER
0351  80 26 0086 R F0                AND     VAR_DELAY,0F0H  ; CLEAR VARIABLE DELAY
0356  80 26 0088 R FE                AND     KB_FLAG_2,0FEH  ; INITIAL PUTCHAR BIT AS ZERO
035B  F6 06 0088 R 02                TEST    KB_FLAG_2,INIT_DELAY ; ARE WE INCREASING THE
                                                             ; INITIAL DELAY?
0360  74 E9                          JZ      TP              ; DEFAULT DELAY
0362  80 0E 0086 R 0F                OR      VAR_DELAY,DELAY_RATE ; INCREASE DELAY BY 2X
0367  EB E2                          JMP     SHORT TP
                             ;----CHECK IF WE ARE IN TYPAMATIC MODE AND IF DELAY IS OVER
0369  F6 06 0088 R 08        TP2:    TEST    KB_FLAG_2,TYPE_OFF ; IS TYPAMATIC TURNED OFF?
036E  75 2B                          JNZ     TP4             ; JUMP IF TYPAMATIC RATE IS OFF
0370  8A 1E 0086 R                   MOV     BL,VAR_DELAY    ; GET VAR_DEALY
0374  80 E3 0F                       AND     BL,0FH          ; MASK OFF HIGH ORDER(SCREEN RANGE)
0377  0A DB                          OR      BL,BL           ; IS INITIAL DELAY OVER?
0379  74 0D                          JZ      TP3             ; JUMP IF DELAY IS OVER
037B  FE CB                          DEC     BL              ; DECREASE DELAY WAIT BY ANOTHER
                                                             ; CHARACTER
037D  80 26 0086 R F0                AND     VAR_DELAY,0F0H
0382  08 1E 0086 R                   OR      VAR_DELAY,BL
0386  EB 13                          JMP     SHORT TP4
                             ;----CHECK IF TIME TO OUTPUT CHAR
0388  F6 06 0088 R 04        TP3:    TEST    KB_FLAG_2,HALF_RATE ; ARE WE IN HALF RATE MODE
038D  74 BC                          JZ      TP              ; JUMP IF WE ARE IN NORMAL MODE
038F  80 36 0088 R 01                XOR     KB_FLAG_2,PUTCHAR ; TOGGLE BIT
0394  F6 06 0088 R 01                TEST    KB_FLAG_2,PUTCHAR ; IS IT TIME TO PUT OUT A CHAR
0399  75 B0                          JNZ     TP
039B                         TP4:                            ; NOT TIME TO OUTPUT CHARACTER
039B  F9                             STC                     ; SKIP THIS CHARACTER
039C  5B                             POP     BX              ; SET CARRY FLAG
039D  C3                             RET
039E                         TPM     ENDP
                             PAGE
                             ;---- INT 16 ------------------------------------------------------
                             ; KEYBOARD I/O
                             ;       THESE ROUTINES PROVIDE KEYBOARD SUPPORT
                             ; INPUT
                             ;       (AH)=0  READ THE NEXT JIS8BIT CHARACTER STRUCK FROM THE
                             ;               KEYBOARD, RETURN THE RESULT IN (AL), SCAN CODE IN
                             ;               (AH)
                             ;
                             ;       (AH)=1  SET THE Z FLAG TO INDICATE IF AN JIS8BIT CHARACTER IS
                             ;               AVAILABLE TO BE READ.
                             ;               (ZF)=1 -- NO CODE AVAILABLE
                             ;               (ZF)=0 -- CODE IS AVAILABLE
                             ;               IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE
                             ;               READ IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER
```

```
;       (AH)=2  RETURN THE CURRENT SHIFT STATUS IN AL AND AH REGISTERS
;               AL REGISTER
;                  7  6  5  4  3  2  1  0
;                  !  !  !  !  !  !  !  !
;                  !  !  !  !  !  !  !--+->01=RIGHT SHIFT KEY IS DEPRESSED
;                  !  !  !  !  !  !  !     10=LEFT SHIFT KEY IS DEPRESSED
;                  !  !  !  !  !  !  !     11=LOCK SHIFT KEY IS DEPRESSED
;                  !  !  !  !  !  !  !-------->CONTROL SHIFT KEY IS DEPRESSED
;                  !  !  !  !  !  !----------->ALTERNATE SHIFT KEY IS DEPRESSED
;                  !  !  !  !  !-------------->SCROLL LOCK STATE HAS BEEN TOGGLED
;                  !  !  !  !----------------->RESERVED
;                  !  !  !-------------------->CAPS LOCK STATE HAS BEEN TOGGLED
;                  !  !----------------------->NUM LOCK STATE HAS BEEN TOGGLED
;                  !-------------------------->INSERT STATE IS ACTIVE
;
;               AH REGISTER
;                  7  6  5  4  3  2  1  0
;                  !  !  !  !  !  !  !  !
;                  !  !  !  !  !  !  !  !->1=ZENKAKU MODE, 0=HANKAKU MODE
;                  !  !  !  !  !  !  !--+->00=ALPHA/NUMERIC SHIFT
;                  !  !  !  !  !  !        01=KATAKANA SHIFT
;                  !  !  !  !  !  !        10=HIRAGANA SHIFT
;                  !  !  !  !  !  !----------->RESERVED TO 1
;                  !--+--+--+--+-------------->RESERVED
;
;       (AH)=3  SET TYPAMATIC RATES.  THE TYPAMATIC RATE CAN BE
;               CHANGED USING THE FOLLOWING FUNCTIONS:
;               (AL)=0      RETURN TO DEFAULT.  RESTORES ORIGINAL
;                           STATE. I.E. TYPAMATIC ON, NORMAL INITIAL
;                           DELAY, AND NORMAL TYPAMATIC RATE.
;               (AL)=1      INCREASE INITIAL DELAY.  THIS IS THE
;                           DELAY BETWEEN THE FIRST CHARACTER AND
;                           THE BURST OF TYPAMATIC CHARS.
;               (AL)=2      HALF_RATE.  SLOWS TYPAMATIC CHARACTERS
;                           BY ONE HALF.
;               (AL)=3      COMBINES AL=1 AND AL=2.  INCREASES
;                           INITIAL DELAY AND SLOWS TYPAMATIC
;                           CHARACTERS BY ONE HALF.
;               (AL)=4      TURN OFF TYPAMATIC CHARACTERS.  ONLY THE
;                           FIRST CHARACTER IS HONORED.  ALL OTHERS
;                           ARE IGNORED.
;               AL IS RANGE CHECKED.  IF AL<0 OR AL>4 THE STATE
;               REMAINS THE SAME.
;
;               ****NOTE****  EACH TIME THE TYPAMATIC RATES ARE
;               CHANGED ALL PREVIOUS STATES ARE REMOVED.  I.E. IF
;               THE KEYBOARD IS IN THE HALF RATE MODE AND YOU WANT
;               TO ADD AN INCREASE IN TYPAMATIC DELAY, YOU MUST
;               CALL THIS ROUTINE WITH AH=3 AND AL=3.
;
;       (AH)=4  ADJUST KEYBOARD BY THE VALUE IN AL AS FOLLOWS:
;               (AL)=0      TURN OFF KEYBOARD CLICK.
;               (AL)=1      TURN ON KEYBOARD CLICK.
;               AL IS RANGE CHECKED.  THE STATE IS UNALTERED IF
;               AL <> 1,0.
;
;       (AH=5)  CHANGE THE KBD SHIFT STATUS
;               AL REGSTER
;               7  6  5  4  3  2  1  0
;               !  !  !  !  !  !  !  !
;               !  !  !  !  !  !  !--+---> 00=HANKAKU , 01=ZENKAKU
;               !  !  !  !  !  !           10=TOGGLE  , 11=NOT CHANGE
;               !  !  !  !  !--+--------> 00=ALPHA/NUMERIC , 01=KATAKANA
;               !  !  !  !                 10=HIRAGANA      , 11= NOT CHANGE
;               !  !  !--+--------------> 00=CAPS OFF , 01=CAPS ON
;               !  !                       10=TOGGLE   , 11=NOT CHANGE
;               !--+--------------------> 00=EXIT KANAKAN , 01=ENTER KANAKAN
;                                          10=TOGGLE   , 11=NOT CHANGE
;
;       (AH=85) SAME AS AH=5  BUT THE INDICATER ISN'T CHANGED.
;
;                  ****NOTE**** WHEN YOU USE THIS FUNCTION,
;                  YOU MUST HAVE THE STACK WHOSE LENGTH IS MORE THAN 130 WORD.
;                  THIS FUNCTION MAY USE THAT.
;
;       (AH=7)  INDICATOR AND KANAKAN ON/OFF
;               AL REGSTER
;               7  6  5  4  3  2  1  0
;               !  !  !  !  !  !  !  !
;               !  !  !  !  !  !  !  !---> KANAKAN    0=ON , 1=OFF
;               !  !  !  !  !  !  !--+---> INDICATOR  0=ON , 1=OFF
;               !  !  !  !  !  !  !--+---> RESERVED
;               !--+--+--+--+--+--+--+---> RESERVED
;
; OUTPUT
;       AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
;       ALL REGISTERS RETAINED
;--------------------------------------------------------------------------
;
ASSUME  CS:CODE,DS:DATA
KEYBOARD_IO     PROC    FAR
        STI                     ; INTERRUPTS BACK ON
        PUSH    DS              ; SAVE CURRENT DS
        PUSH    BX              ; SAVE BX TEMPORARILY
        CALL    DDS             ; POINT DS AT BIOS DATA SEGMENT
        OR      AH,AH           ; AH=0
        JZ      K1              ; ASCII_READ
        DEC     AH              ; AH=1
        JZ      K2              ; ASCII_STATUS
        DEC     AH              ; AH=2
        JZ      K3              ; SHIFT_STATUS
        DEC     AH              ; AH=3
        JZ      TRATE           ; SET TYPAMATIC RATES
        DEC     AH              ; AH=4
        JZ      KCLICK          ; ON/OFF KEYBOARD CLICK
        DEC     AH
```

```
039E                    
039E  FB                
039F  1E                
03A0  53                
03A1  E8 0000 E         
03A4  0A E4             
03A6  74 22             
03A8  FE CC             
03AA  74 36             
03AC  FE CC             
03AE  74 43             
03B0  FE CC             
03B2  74 48             
03B4  FE CC             
03B6  74 55             
03B8  FE CC             
```

```
03BA  F6 C4 7F                  TEST    AH,7FH          ; AH=5 OR AH=85H
03BD  74 65                     JZ      CHG_SHIFT       ; CHANGE THE KBD SHIFT STATUS
03BF  FE CC                     DEC     AH              ; AH=6
03C1  74 5E                     JZ      RET_INT16       ; ILLEGAL FUNCTION CALL
03C3  FE CC                     DEC     AH              ; AH=7
03C5  75 5A                     JNZ     RET_INT16       ; ILLEGAL FUNCTION CALL
03C7  E9 049B R                 JMP     CHG_SW          ; CHANGE THE KANAKAN AND INDICATOR SWITCH
                        ;
                        ;------ READ THE KEY TO FIGURE OUT WHAT TO DO
                        ;
03CA                    K1:                             ; ASCII READ
03CA  FB                        STI                     ; INTERRUPTS BACK ON DURING LOOP
03CB  90                        NOP                     ; ALLOW AN INTERRUPT TO OCCUR
03CC  FA                        CLI                     ; INTERRUPTS BACK OFF
03CD  8B 1E 001A R              MOV     BX,BUFFER_HEAD  ; GET POINTER TO HEAD OF BUFFER
03D1  3B 1E 001C R              CMP     BX,BUFFER_TAIL  ; TEST END OF BUFFER
03D5  74 F3                     JZ      K1              ; LOOP UNTIL SOMETHING IN BUFFER
03D7  8B 07                     MOV     AX,[BX]         ; GET SCAN CODE AND ASCII CODE
03D9  E8 04B9 R                 CALL    K4              ; MOVE POINTER TO NEXT POSITION
03DC  89 1E 001A R              MOV     BUFFER_HEAD,BX  ; STORE VALUE IN VARIABLE
03E0  EB 3F                     JMP     SHORT   RET_INT16
                        ;
                        ;------ ASCII STATUS
                        ;
03E2                    K2:
03E2  FA                        CLI                     ; INTERRUPTS OFF
03E3  8B 1E 001A R              MOV     BX,BUFFER_HEAD  ; GET HEAD POINTER
03E7  3B 1E 001C R              CMP     BX,BUFFER_TAIL  ; IF EQUAL (Z=1) THEN NOTHING THERE
03EB  8B 07                     MOV     AX,[BX]
03ED  FB                        STI                     ; INTERRUPTS BACK ON
03EE  5B                        POP     BX              ; RECOVER REGISTER
03EF  1F                        POP     DS              ; RECOVER SEGMENT
03F0  CA 0002                   RET     2               ; THROW AWAY FLAGS
                        ;
                        ;------ SHIFT STATUS
                        ;
03F3                    K3:
03F3  A0 0017 R                 MOV     AL,KB_FLAG      ; GET THE SHIFT STATUS FLAGS
03F6  8A 26 0336 R              MOV     AH,JKB_FLAG     ; GET THE JKB_FLAG
03FA  EB 25                     JMP     SHORT   RET_INT16
                        ;
                        ;------ SET TYPAMATIC
                        ;
03FC                    TRATE:
03FC  3C 04                     CMP     AL,4            ; CHECK FOR CORRECT RANGE
03FE  7F 21                     JG      RET_INT16       ; IF ILLEGAL VALUE IN AL IGNORE
0400  80 26 0088 R F1           AND     KB_FLAG_2,0F1H  ; MASK OFF ANY OLD TYPAMATIC STATES
0405  D0 E0                     SHL     AL,1            ; SHIFT TO PROPER POSITION
0407  08 06 0088 R              OR      KB_FLAG_2,AL
040B  EB 14                     JMP     SHORT   RET_INT16
                        ;
                        ;------- ADJUST KEY CLICK
                        ;
040D                    KCLICK:
040D  8A C8                     OR      AL,AL           ; TURN OFF KEYBOARD CLICK?
040F  75 07                     JNZ     KCLICK1         ; JUMP FOR RANGE CHECK
0411  80 26 0018 R FB           AND     KB_FLAG_1,AND_MASK-CLICK_ON ; TURN OFF CLICK
0416  EB 09                     JMP     SHORT RET_INT16
0418                    KCLICK1:
0418  3C 01                     CMP     AL,1            ; RANGE CHECK
041A  75 05                     JNE     RET_INT16       ; NOT IN RANGE, RETURN
041C  80 0E 0018 R 04           OR      KB_FLAG_1,CLICK_ON ; TURN ON KEYBOARD CLICK
                        ;
                        ;------ INTERRUPT RETURN
                        ;
0421                    RET_INT16:
0421  5B                        POP     BX              ; RECOVER REGISTER
0422  1F                        POP     DS              ; RECOVER REGISTER
0423  CF                        IRET                    ; RETURN TO CALLER
                        ;
                        ;-----  CHANGE KEYBOADR STATUS BY THE AL
                        ;
0424                    CHG_SHIFT:
0424  50                        PUSH    AX
0425  51                        PUSH    CX
                        ;
0426  8B C8                     MOV     CX,AX           ; SAVE AX INTO CX
0428  24 C0                     AND     AL,0C0H         ;
042A  3C C0                     CMP     AL,0C0H         ; CHNGE KANAKAN ?
042C  74 04                     JE      CH_CAP          ; IF AL=0C0H THEN CH_CAP
042E  B4 FF                     MOV     AH,0FFH         ;
0430  CD 78                     INT     78H             ; KANAKAN ROUTINE
0432                    CH_CAP:
0432  8A C1                     MOV     AL,CL           ;
0434  24 30                     AND     AL,30H          ;
0436  3C 30                     CMP     AL,30H          ;
0438  74 1D                     JE      CH_JAP          ;
043A  3C 20                     CMP     AL,20H          ; TOGGLE ?
043C  74 14                     JE      CAP_TOG         ;
043E  3C 10                     CMP     AL,10H          ; CAPS BIT ON ?
0440  74 08                     JE      CAP_ENT
0442  80 26 0017 R BF           AND     KB_FLAG,NOT CAPS_STATE
0447  EB 0E 90                  JMP     CH_JAP
044A                    CAP_ENT:
044A  80 0E 0017 R 40           OR      KB_FLAG,CAPS_STATE
044F  EB 06 90                  JMP     CH_JAP
0452                    CAP_TOG:
0452  80 36 0017 R 40           XOR     KB_FLAG,CAPS_STATE
0457                    CH_JAP:
0457  8A C1                     MOV     AL,CL
0459  24 0C                     AND     AL,0CH
045B  3C 0C                     CMP     AL,0CH
045D  74 0B                     JE      CH_H_Z
045F  D0 E8                     SHR     AL,1
```

```
0461  80 26 0336 R F9          AND     JKB_FLAG,NOT HOT_ALPHA_STATE
0466  08 06 0336 R             OR      JKB_FLAG,AL
046A                     CH_H_Z:
046A  8A C1                    MOV     AL,CL
046C  24 03                    AND     AL,03H
046E  3C 03                    CMP     AL,03H
0470  74 1D                    JE      CHG_SHIFT_R
0472  3C 02                    CMP     AL,02H          ; TOGGLE ?
0474  74 14                    JE      H_Z_TOG
0476  3C 01                    CMP     AL,01H
0478  74 08                    JE      Z_ENT
047A  80 26 0336 R FE          AND     JKB_FLAG,NOT ZENKAKU_STATE
047F  EB 0E 90                 JMP     CHG_SHIFT_R
0482                     Z_ENT:
0482  80 0E 0336 R 01          OR      JKB_FLAG,ZENKAKU_STATE
0487  EB 06 90                 JMP     CHG_SHIFT_R
048A                     H_Z_TOG:
048A  80 36 0336 R 01          XOR     JKB_FLAG,ZENKAKU_STATE
048F                     CHG_SHIFT_R:
048F  F6 C5 80                 TEST    CH,80H
0492  75 03                    JNZ     CHG_S_R
0494  E8 0EA9 R                CALL    IND
0497                     CHG_S_R:
0497  59                      POP     CX
0498  58                      POP     AX
0499  EB 86                    JMP     RET_INT16

                         ;
                         ;------ CHANGE KANAKN AND INDICATOR SWITCH
                         ;
049B                     CHG_SW:
049B  3C 03                    CMP     AL,3
049D  7F 17                    JG      CHG_SW_3
049F  80 26 0338 R FC          AND     JKB_FLAG_2,0FCH
04A4  08 06 0338 R             OR      JKB_FLAG_2,AL
04A8  A8 01                    TEST    AL,1
04AA  74 05                    JZ      CHG_SW_1
04AC  B8 FF02                  MOV     AX,0FF02H
04AF  EB 03                    JMP     SHORT CHG_SW_2
04B1                     CHG_SW_1:
04B1  B8 FF01                  MOV     AX,0FF01H
04B4                     CHG_SW_2:
04B4  CD 78                    INT     78H
04B6                     CHG_SW_3:
04B6  E9 0421 R                JMP     RET_INT16
04B9                     KEYBOARD_IO    ENDP
                         ;
                         ;------ INCREMENT A BUFFER POINTER
                         ;
04B9                     K4     PROC    NEAR
04B9  43                       INC     BX              ; MOVE TO NEXT WORD IN LIST
04BA  43                       INC     BX
04BB  3B 1E 0082 R             CMP     BX,BUFFER_END   ; AT END OF BUFFER?
04BF  75 04                    JNE     K5              ; NO, CONTINUE
04C1  8B 1E 0080 R             MOV     BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
04C5  C3                 K5:   RET
04C6                     K4     ENDP
                         ;------ TABLE OF SHIFT KEYS AND MASK VALUES
04C6                     K6     LABEL   BYTE
04C6  6B 6C 6D                 DB      KANJI_KEY,MUHEN_KEY,HENKAN_KEY
04C9  3A 6E 6F                 DB      ALPHA_KEY,KATAKANA_KEY,HIRAGANA_KEY
04CC  52                       DB      INS_KEY         ; INSERT KEY
04CD  45 46 38 1D              DB      NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
04D1  2A 36                    DB      LEFT_KEY,RIGHT_KEY
= 000D                   K6L    EQU     $-K6
                         ;------ SHIFT_MASK_TABLE
04D3                     K7     LABEL   BYTE
04D3  80 20 10                 DB      KANJI_SHIFT,MUHEN_SHIFT,HENKAN_SHIFT
04D6  40 02 04                 DB      CAPS_SHIFT,KATAKANA_SHIFT,HIRAGANA_SHIFT
04D9  80                       DB      INS_SHIFT       ; INSERT MODE SHIFT
04DA  20 10 08 04              DB      NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
04DE  02 01                    DB      LEFT_SHIFT,RIGHT_SHIFT
                         ;------ SCAN CODE TABLES
04E0  1B FF 00 FF FF FF  K8            DB      27,-1,0,-1,-1,-1,30,-1
      1E FF
04E8  FF FF FF 1F FF 7F                DB      -1,-1,-1,31,-1,127,-1,17
      FF 11
04F0  17 05 12 14 19 15                DB      23,5,18,20,25,21,9,15
      09 0F
04F8  10 1B 1D 0A FF 01                DB      16,27,29,10,-1,1,19
      13
04FF  04 06 07 08 0A 0B                DB      4,6,7,8,10,11,12,-1,-1
      0C FF FF
0508  FF FF 1C 1A 18 03                DB      -1,-1,28,26,24,3,22,2
      16 02
0510  0E 0D FF FF FF FF                DB      14,13,-1,-1,-1,-1,-1,-1
      FF FF
0518  20 FF                            DB      ' ',-1
                         ;-------- CTL TABLE SCAN
051A                     K9     LABEL   BYTE
051A  5E 5F 60 61 62 63                DB      94,95,96,97,98,99,100,101
      64 65
0522  66 67 FF FF 77 FF                DB      102,103,-1,-1,119,-1,132,-1
      84 FF
052A  73 FF 74 FF 75 FF                DB      115,-1,116,-1,117,-1,118,-1
      76 FF
0532  FF                               DB      -1
                         ;------- LC TABLE
0533                     K10    LABEL   BYTE
0533  1B 31 32 33 34 35                DB      01BH,'1234567890-=',08H,09H
      36 37 38 39 30 2D
0542  3D 08 09
      71 77 65 72 74 79                DB      'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
      75 69 6F 70 5B 5D
      0D FF 61 73 64 66
```

```
      67 68 6A 6B 6C 3B
      27
055B  60 FF 5C 7A 78 63          DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
      76 62 6E 6D 2C 2E
      2F FF 2A FF 20
056C  FF                         DB      -1
056D                      ;------- UC TABLE
056D  1B 21 40 23 24 25   K11     LABEL   BYTE
      5E 26 2A 28 29 5F          DB      27,'!@#$',37,05EH,'&*()_+',08H,0
      2B 08 00
057C  51 57 45 52 54 59          DB      'QWERTYUIOP  ',0DH,-1,'ASDFGHJKL:"'
      55 49 4F 50 7B 7D
      0D FF 41 53 44 46
      47 48 4A 4B 4C 3A
      22
0595  7E FF 7C 5A 58 43          DB      7EH,-1,'|ZXCVBNM<>?',-1,0,-1,' ',-1
      56 42 4E 4D 3C 3E
      3F FF 00 FF 20 FF
05A7                      ;------- UC TABLE SCAN
05A7  54 55 56 57 58 59   K12     LABEL   BYTE
      5A                         DB      84,85,86,87,88,89,90
05AE  5B 5C 5D                   DB      91,92,93
05B1                      ;------- ALT TABLE SCAN
05B1  68 69 6A 6B 6C      K13     LABEL   BYTE
05B6  6D 6E 6F 70 71             DB      104,105,106,107,108
                                 DB      109,110,111,112,113
05BB                      ;------- NUM STATE TABLE
05BB  37 38 39 2D 34 35  K14     LABEL   BYTE
      36 2B 31 32 33 30          DB      '789-456+1230.'
      2E
05C8                      ;------- BASE CASE TABLE
05C8  47 48 49 FF 4B FF  K15     LABEL   BYTE
      4D                         DB      71,72,73,-1,75,-1,77
05CF  FF 4F 50 51 52 53          DB      -1,79,80,81,82,83
                          ;*****************************************************************
                          ;------- KEYBOARD INTERRUPT ROUTINE
                          ;*****************************************************************
05D5                      KB_INT  PROC    FAR
05D5  FB                          STI                     ; ALLOW FURTHER INTERRUPTS
05D6  50                          PUSH    AX
05D7  53                          PUSH    BX
05D8  51                          PUSH    CX
05D9  52                          PUSH    DX
05DA  56                          PUSH    SI
05DB  57                          PUSH    DI
05DC  1E                          PUSH    DS
05DD  06                          PUSH    ES
05DE  FC                          CLD                     ; FORWARD DIRECTION
05DF  E8 0000 E                   CALL    DDS             ;
05E2  8A 3E 0017 R                MOV     BH,KB_FLAG      ;
05E6  8A 1E 0336 R                MOV     BL,JKB_FLAG     ;
05EA  81 E3 4007                  AND     BX,4007H        ; MASK KBD STATUS FLAG
05EE  53                          PUSH    BX              ; MEMORIZE KBD STATUS
05EF  8A E0                       MOV     AH,AL           ; SAVE SCAN CODE IN AH
                          ;------- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
05F1  3C FF                       CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR?
05F3  75 06                       JNZ     K16             ; NO, TEST FOR SHIFT KEY
05F5  E8 0AF0 R                   CALL    ERROR_BEEP      ; CALL ERROR BEEP ROUTINE
05F8  E9 07C9 R                   JMP     K26             ; END OF INTERRUPT
                          ;------- TEST FOR SHIFT KEYS
05FB                      K16:                            ; TEST_SHIFT
05FB  24 7F                       AND     AL,07FH         ; TURN OFF THE BREAK BIT
05FD  0E                          PUSH    CS
05FE  07                          POP     ES              ; ESTABLISH ADDRESS OF SHIFT TABLE
05FF  BF 04C6 R                   MOV     DI,OFFSET K6    ; SHIFT KEY TABLE
0602  B9 000D                     MOV     CX,K6L          ; LENGTH
0605  F2/ AE                      REPNE   SCASB           ; LOOK THROUGH THE TABLE FOR A
                                                          ; MATCH
0607  8A C4                       MOV     AL,AH           ; RECOVER SCAN CODE
0609  74 03                       JE      K17             ; JUMP IF MATCH FOUND
060B  E9 07B9 R                   JMP     K25             ; IF NO MATCH, THEN SHIFT NOT FOUND
                          ;------- SHIFT KEY FOUND
060E  81 EF 04C7 R        K17:    SUB     DI,OFFSET K6+1  ; ADJUST PTR TO SCAN CODE MATCH
0612  2E: 8A A5 04D3 R            MOV     AH,CS:K7[DI]    ; GET MASK INTO AH
0617  A8 80                       TEST    AL,80H          ; TEST FOR BREAK KEY
0619  74 03                       JZ      K17_1
061B  E9 0727 R                   JMP     K23
                          ;------- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
061E                      K17_1:
061E  83 FF 09                    CMP     DI,9
0621  72 07                       JB      K18             ; IS THIS A TOGGLE KEY
                          ;                               ; YES, HANDLE TOGGLE KEY
                          ;------- PLAIN SHIFT KEY, SET SHIFT ON
0623  08 26 0017 R                OR      KB_FLAG,AH      ; TURN ON SHIFT BIT
0627  E9 07C9 R                   JMP     K26             ; INTERRUPT_RETURN
                          ;------- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
062A                      K18:                            ; SHIFT-TOGGLE
062A  F6 06 0017 R 04             TEST    KB_FLAG, CTL_SHIFT ; CHECK CTL SHIFT STATE
062F  74 03                       JZ      K18_1
0631  E9 07B9 R                   JMP     K25             ; JUMP IF CTL STATE
0634                      K18_1:
0634  F6 06 0017 R 08             TEST    KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
0639  74 68                       JZ      K22             ; JUMP IF NOT ALTERNATE SHIFT
                          ;
                          ;------- ALTERNATE SHIFT
                          ;
063B  3C 3A                       CMP     AL,ALPHA_KEY
063D  75 11                       JNZ     K19             ; ALTERNATE-ALPHA_KEY = CAPS
063F  84 26 0018 R                TEST    AH,KB_FLAG_1    ; IS KEY ALREADY DEPRESSED
```

```
0643  75 5B                     JNZ     K21             ; JUMP IF KEY ALREADY DEPRESSED
0645  08 26 0018 R              OR      KB_FLAG_1,AH    ; INDICATE THAT THE KEY IS
                                                       ; DEPRESSED
0649  30 26 0017 R              XOR     KB_FLAG,AH      ; TOGGLE THE SHIFT STATE
064D  E9 07C9 R                 JMP     K26             ; INTERRUPT RETURN
0650                    K19:
0650  3C 6E                     CMP     AL,KATAKANA_KEY         ; IS THIS THE KATAKANA_KEY
0652  75 14                     JNZ     K20                     ; NOT KATAKANA_KEY
0654  F6 06 0337 R 10           TEST    JKB_FLAG_1,HANKAKU_SHIFT  ; IS KEY ALREADY DEPRESSED
0659  75 45                     JNZ     K21                     ; JUMP IF KEY ALREADY DEPRESSED
065B  80 0E 0337 R 10           OR      JKB_FLAG_1,HANKAKU_SHIFT  ; INDICATE THAT THE KEY IS DEPRESSED
0660  80 26 0336 R FE           AND     JKB_FLAG,NOT ZENKAKU_STATE      ; TOGGLE THE SHIFT STATE
0665  EB 39 90                  JMP     K21                     ; INTERRUPT RETURN
0668                    K20:
0668  3C 6F                     CMP     AL,HIRAGANA_KEY         ; IS THIS THE HIRAGANA_KEY
066A  74 03                     JZ      K20_1
066C  EB 15 90                  JMP     K20_2                   ; NOT HIRAGANA_KEY
066F                    K20_1:
066F  F6 06 0337 R 08           TEST    JKB_FLAG_1,ZENKAKU_SHIFT  ; IS KEY ALREADY DEPRESSED
0674  75 2A                     JNZ     K21                     ; JUMP IF KEY ALREADY DEPRESSED
0676  80 0E 0337 R 08           OR      JKB_FLAG_1,ZENKAKU_SHIFT  ; INDICATE THAT THE KEY IS DEPRESSED
067B  80 0E 0336 R 01           OR      JKB_FLAG,ZENKAKU_STATE    ; TOGGLE THE SHIFT STATE
0680  EB 1E 90                  JMP     K21
0683                    K20_2:
0683  3C 6B                     CMP     AL,KANJI_KEY            ; IS KNUM_KEY
0685  74 07                     JZ      K20_3
0687  3C 52                     CMP     AL,INS_KEY              ; IS INS_KEY
0689  74 15                     JE      K21
068B  EB 7F 90                  JMP     K22_6
068E                    K20_3:
068E  F6 06 0338 R 40           TEST    JKB_FLAG_2,KNUM_SHIFT   ; IS KEY ALREADY DEPRESSED
0693  75 0B                     JNZ     K21                     ; JUMP IF KEY ALREADY DEPRESSED
0695  80 0E 0338 R 40           OR      JKB_FLAG_2,KNUM_SHIFT   ; INDICATE THAT THE KEY IS DEPRESSED
069A  8A E0                     MOV     AH,AL                   ;
069C  B0 38                     MOV     AL,38H
069E  CD 78                     INT     78H                     ; KANAKAN ROUTINE
06A0                    K21:
06A0  E9 07C9 R                 JMP     K26                     ; INTERRUPT RETURN
                        ;
                        ;------ NOT ALTERNATE SHIFT
                        ;
06A3                    K22:                            ; SHIFT TOGGLE KEY HIT; PROCESS IT
06A3  3C 3A                     CMP     AL,ALPHA_KEY            ; IS THIS THE ALPHA_KEY
06A5  75 14                     JNZ     K22_1                   ; NOT ALPHA_KEY
06A7  F6 06 0337 R 01           TEST    JKB_FLAG_1,ALPHA_SHIFT  ; IS KEY ALREADY DEPRESSED
06AC  75 70                     JNZ     K22_7                   ; JUMP IF KEY ALREADY DEPRESSED
06AE  80 0E 0337 R 01           OR      JKB_FLAG_1,ALPHA_SHIFT  ; INDICATE THAT THE KEY IS DEPRESSED
06B3  80 26 0336 R F9           AND     JKB_FLAG,ALPHA_STATE    ; TOGGLE THE SHIFT STATE
06B8  E9 07C9 R                 JMP     K26                     ; INTERRUPT RETURN
06BB                    K22_1:
06BB  3C 6E                     CMP     AL,KATAKANA_KEY         ; IS THIS THE KATAKANA_KEY
06BD  74 04                     JZ      K22_2                   ; YES, KATAKANA_KEY
06BF  3C 6F                     CMP     AL,HIRAGANA_KEY         ; IS THIS THE HIRAGANA_KEY
06C1  75 16                     JNZ     K22_3                   ; NOT HIRAGANA_KEY
06C3                    K22_2:
06C3  84 26 0337 R              TEST    AH,JKB_FLAG_1   ; IS KEY ALREADY DEPRESSED
06C7  75 55                     JNZ     K22_7           ; JUMP IF KEY ALREADY DEPRESSED
06C9  08 26 0337 R              OR      JKB_FLAG_1,AH   ; INDICATE THAT THE KEY IS DEPRESSED
06CD  80 26 0336 R F9           AND     JKB_FLAG,ALPHA_STATE
06D2  08 26 0336 R              OR      JKB_FLAG,AH     ; TOGGLE THE SHIFT STATE
06D6  E9 07C9 R                 JMP     K26             ; INTERRUPT RETURN
06D9                    K22_3:
06D9  3C 6B                     CMP     AL,KANJI_KEY    ; IS THIS THE KANJI_KEY
06DB  74 0A                     JZ      K22_4           ; KANJI_KEY
06DD  3C 6C                     CMP     AL,MUHEN_KEY    ; IS THIS THE MUHEN_KEY
06DF  74 06                     JZ      K22_4           ; MUHEN_KEY
06E1  3C 6D                     CMP     AL,HENKAN_KEY   ; IS THIS THE HENKAN_KEY
06E3  74 02                     JZ      K22_4           ; HENKAN_KEY
06E5  EB 25                     JMP     SHORT K22_6     ;
06E7                    K22_4:
06E7  84 26 0338 R              TEST    AH,JKB_FLAG_2   ; IS KEY ALREADY DEPRESSED
06EB  75 31                     JNZ     K22_7           ; JUMP IF KEY ALREADY DEPRESSED
06ED  08 26 0338 R              OR      JKB_FLAG_2,AH   ; INDICATE THAT THE KEY IS DEPRESSED
06F1  F6 06 0336 R 05           TEST    JKB_FLAG,ZENKAKU_CHAR
06F6  75 08                     JNZ     K22_5           ; IF ZENKAKU CHARACTER
06F8  8A E0                     MOV     AH,AL           ; SET AH SCANCODE
06FA  B0 20                     MOV     AL,20H          ; SPACE CHARACTER
06FC  CD 78                     INT     78H             ; KANAKAN ROUTINE
06FE  EB 1E                     JMP     SHORT K22_7     ; INTERRUPT RETURN
0700
0700  8A E0                     MOV     AH,AL           ; SET AH SCANCODE
0702  B0 81                     MOV     AL,81H          ; ZENKAKU SPACE 1ST BYTE
0704  CD 78                     INT     78H             ; KANAKAN ROUTINE
0706  B0 40                     MOV     AL,40H          ; ZENKAKU SPACE 2ND BYTE
0708  CD 78                     INT     78H             ; KANAKAN ROUTINE
070A  EB 12                     JMP     SHORT K22_7     ; INTERRUPT RETURN
070C
070C  84 26 0018 R              TEST    AH,KB_FLAG_1    ; IS KEY ALREADY DEPRESSED
0710  75 0C                     JNZ     K22_7           ; JUMP IF KEY ALREADY DEPRESSED
0712  08 26 0018 R              OR      KB_FLAG_1,AH    ; INDICATE THAT THE KEY IS
                                                        ; DEPRESSED
0716  30 26 0017 R              XOR     KB_FLAG,AH      ; TOGGLE THE SHIFT STATE
071A  3C 52                     CMP     AL,INS_KEY      ; TEST FOR 1ST MAKE OF INSERT KEY
071C  74 03                     JE      K22_8           ;
071E                    K22_7:                          ; JUMP IF NOT INSERT KEY
0721  E9 07C9 R                 JMP     K26             ;
0724                    K22_8:                          ; SET SCAN CODE INTO AH, 0 INTO AL
      B8 5200                   MOV     AX,INS_KEY*256  ; PUT INTO OUTPUT BUFFER
      E9 0A53 R                 JMP     K57             ;
0727                    ;------ BREAK SHIFT FOUND
                        ;                               ; BREAK-SHIFT-FOUND
                        K23:
```

```
0727  83 FF 09                CMP     DI,9                    ; IS THIS ATOGGLE KEY
072A  72 1D                   JB      K24                     ; YES, HANDLE TOGGLE KEY
072C  F6 D4                   NOT     AH                      ; INVERT MASK
072E  20 26 0017 R            AND     KB_FLAG,AH              ; TURN OFF SHIFT BIT
0732  3C B8                   CMP     AL,ALT_KEY+80H          ; IS THIS ALTERNATE SHIFT RELEASE
0734  75 10                   JNE     K23_1                   ; INTERRUPT_RETURN
                      ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
0736  A0 0019 R               MOV     AL,ALT_INPUT
0739  32 E4                   XOR     AH,AH                   ; SCAN CODE OF 0
073B  88 26 0019 R            MOV     ALT_INPUT,AH            ; ZERO OUT THE FIELD
073F  0A C0                   OR      AL,AL                   ; WAS THE INPUT=0?
0741  74 03                   JE      K23_1                   ; INTERRUPT_RETURN
0743  E9 0A9A R               JMP     K58                     ; IT WASN'T, SO PUT IN BUFFER
0746                  K23_1:
0746  E9 07C9 R               JMP     K26
0749                  K24:                                    ; BREAK-TOGGLE
0749  3C BA                   CMP     AL,CAPS_KEY+BREAK_BIT           ; SPECIAL CASE OF TOGGLE KEY
074B  75 19                   JNE     K24_1                   ; JUMP AROUND POTENTIAL UPDATE
074D  80 26 0337 R FE         AND     JKB_FLAG_1,NOT ALPHA_SHIFT      ; INDICATE NO LONGER DEPRESSED
0752  80 26 0018 R BF         AND     KB_FLAG_1,NOT CAPS_SHIFT        ; INDICATE NO LONGER DEPRESSED
0757  F6 06 0018 R 02         TEST    KB_FLAG_1,CLICK_SEQUENCE
075C  74 6B                   JZ      K26                     ; INTERRUPT IS OVER
075E  80 26 0018 R FD         AND     KB_FLAG_1,AND_MASK-CLICK_SEQUENCE ; MASK OFF MAKE
                                                              ; OF CLICK
0763  EB 64 90                JMP     K26                     ; INTERRUPT IS OVER
0766                  K24_1:
0766  3C EE                   CMP     AL,KATAKANA_KEY+BREAK_BIT       ; IS THIS THE KATAKANA_KEY
0768  75 0D                   JNZ     K24_2                   ; NOT KATAKANA_KEY
076A  80 26 0337 R FD         AND     JKB_FLAG_1,NOT KATAKANA_SHIFT   ; INDICATE NO LONGER DEPRESSED
076F  80 26 0337 R EF         AND     JKB_FLAG_1,NOT HANKAKU_SHIFT    ; INDICATE NO LONGER DEPRESSED
0774  EB 53 90                JMP     K26                     ; INTERRUPT IS OVER
0777                  K24_2:
0777  3C EF                   CMP     AL,HIRAGANA_KEY+BREAK_BIT       ; IS THIS THE HIRAGANA_KEY
0779  75 0D                   JNZ     K24_3                   ; NOT HIRAGANA_KEY
077B  80 26 0337 R FB         AND     JKB_FLAG_1,NOT HIRAGANA_SHIFT   ; INDICATE NO LONGER DEPRESSED
0780  80 26 0337 R F7         AND     JKB_FLAG_1,NOT ZENKAKU_SHIFT    ; INDICATE NO LONGER DEPRESSED
0785  EB 42 90                JMP     K26                     ; INTERRUPT IS OVER
0788                  K24_3:
0788  3C EB                   CMP     AL,KANJI_KEY+BREAK_BIT          ; IS THIS THE KANJI_KEY
078A  75 0D                   JNZ     K24_4                   ; NOT KANJI_KEY
078C  80 26 0338 R 7F         AND     JKB_FLAG_2,NOT KANJI_SHIFT      ; INDICATE NO LONGER DEPRESSED
0791  80 26 0338 R BF         AND     JKB_FLAG_2,NOT KNUM_SHIFT       ; INDICATE NO LONGER DEPRESSED
0796  EB 31 90                JMP     K26                     ; INTERRUPT IS OVER
0799                  K24_4:
0799  3C EC                   CMP     AL,MUHEN_KEY+BREAK_BIT          ; IS THIS THE MUHEN_KEY
079B  75 08                   JNZ     K24_5                   ; NOT MUHEN_KEY
079D  80 26 0338 R DF         AND     JKB_FLAG_2,NOT MUHEN_SHIFT      ; INDICATE NO LONGER DEPRESSED
07A2  EB 25 90                JMP     K26                     ; INTERRUPT IS OVER
07A5                  K24_5:
07A5  3C ED                   CMP     AL,HENKAN_KEY+BREAK_BIT         ; IS THIS THE HENKAN_KEY
07A7  75 08                   JNZ     K24_6                   ; NOT HIRAGANA_KEY
07A9  80 26 0338 R EF         AND     JKB_FLAG_2,NOT HENKAN_SHIFT     ; INDICATE NO LONGER DEPRESSED
07AE  EB 19 90                JMP     K26                     ; INTERRUPT IS OVER
                      ;------ BREAK OF NORMAL TOGGLE
07B1                  K24_6:
07B1  F6 D4                   NOT     AH                      ; INVERT MASK
07B3  20 26 0018 R            AND     KB_FLAG_1,AH            ; INDICATE NO LONGER DEPRESSED
07B7  EB 10                   JMP     SHORT K26               ; INTERRUPT_RETURN
                      ;------ TEST FOR HOLD STATE
07B9                  K25:                                    ; NO-SHIFT-FOUND
07B9  3C 80                   CMP     AL,80H                  ; TEST FOR BREAK KEY
07BB  73 0C                   JAE     K26                     ; NOTHING FOR BREAK CHARS FROM HERE
                                                              ; ON
07BD  F6 06 0018 R 08         TEST    KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE?
07C2  74 24                   JZ      K28                     ; BRANCH AROUND TEST IF NOT
07C4  80 26 0018 R F7         AND     KB_FLAG_1,NOT HOLD_STATE ; TURN OFF THE HOLD STATE
                                                              ; BIT
07C9                  K26:                                    ; INTERRUPT-RETURN
07C9  8A 36 0017 R            MOV     DH,KB_FLAG
07CD  8A 16 0336 R            MOV     DL,JKB_FLAG
07D1  81 E2 4007              AND     DX,4007H                ; MASK KBD STATUS FLAG
07D5  5B                      POP     BX
07D6  33 DA                   XOR     BX,DX
07D8  74 05                   JZ      K26_1
07DA  B8 05FF                 MOV     AX,05FFH
07DD  CD 16                   INT     16H
07DF                  K26_1:
07DF  07                      POP     ES
07E0  1F                      POP     DS
07E1  5F                      POP     DI
07E2  5E                      POP     SI
07E3  5A                      POP     DX
07E4  59                      POP     CX
07E5  5B                      POP     BX
07E6  58                      POP     AX
07E7  CF                      IRET                            ; RESTORE STATE
                                                              ; RETURN, INTERRUPTS BACK ON WITH
                                                              ; FLAG CHANGE
                      ;------ NOT IN  HOLD STATE, TEST FOR SPECIAL CHARS
07E8                  K28:                                    ; NO-HOLD-STATE
07E8  F6 06 0017 R 08         TEST    KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
07ED  75 03                   JNZ     K29                     ; JUMP IF ALTERNATE SHIFT
07EF  E9 08E9 R               JMP     K38                     ; JUMP IF NOT ALTERNATE
                      ;------ TEST FOR ALT+CTRL KEY SEQUENCES
07F2                  K29:                                    ; TEST-RESET
07F2  F6 06 0017 R 04         TEST    KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
07F7  74 69                   JZ      K31                     ; NO_RESET
07F9  3C 53                   CMP     AL,DEL_KEY              ; SHIFT STATE IS THERE, TEST KEY
07FB  75 09                   JNE     K29_1                   ; NO_RESET
                      ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
07FD  C7 06 0072 R 1234       MOV     RESET_FLAG, 1234H ; SET FLAG FOR RESET FUNCTION
0803  E9 0000 E               JMP     NEAR PTR RESET  ; JUMP TO POWER ON DIAGNOSTICS
0806  3C 52                   CMP     AL,INS_KEY              ; CHECK FOR RESET WITH DIAGNOSTICS
0808  75 09                   JNE     K29_2                   ; CHECK FOR OTHER
                      ;------ CTL-ALT-INS HAS BEEN FOUND
```

```
080A  C7 06 0072 R 3412          MOV     RESET_FLAG, 3412H ; SET FLAG FOR DIAGNOSTICS
0810  E9 0000 E                  JMP     NEAR PTR RESET  ; LEVEL 1 DIAGNOSTICS
0813  3C 3A                     K29_2:  CMP     AL,CAPS_KEY     ; CHECK FOR KEYBORAD CLICK TOGGLE
0815  75 13                             JNE     K29_3           ; CHECK FOR SCREEN ADJUSTMENT
                                 ;------ ALT+CTRL+CAPSLOCK HAS BEEN FOUND
0817  F6 06 0018 R 02                   TEST    KB_FLAG_1,CLICK_SEQUENCE
081C  75 AB                             JNZ     K26             ; JUMP IF SEQUENCE HAS ALREADY
                                                                ; OCCURED
081E  80 36 0018 R 04                   XOR     KB_FLAG_1,CLICK_ON ; TOGGLE BIT FOR AUDIO KEYSTROKE
                                                                ; FEEDBACK
0823  80 0E 0018 R 02                   OR      KB_FLAG_1,CLICK_SEQUENCE ; SET CLICK_SEQUENCE STATE
0828  EB 9F                             JMP     SHORT K26       ; INTERRUPT IS OVER
082A  3C 4D                     K29_3:  CMP     AL,RIGHT_ARROW  ; ADJUST SCREEN TO THE RIGHT?
082C  75 12                             JNE     K29_4           ; LOOK FOR RIGHT ADJUSTMENT
082E  E8 0AD1 R                         CALL    GET_POS         ; GET THE # OF POSITIONS SCREEN IS
                                                                ; SHIFTED
0831  3C FC                             CMP     AL,0-RANGE      ; IS SCREEN SHIFTED AS FAR AS
                                                                ; POSSIBLE?
0833  7C 94                             JL      K26             ; OUT OF RANGE
0835  FE 0E 0089 R                      DEC     HORZ_POS        ; SHIFT VALUE TO THE RIGHT
0839  FE C8                             DEC     AL              ; DECREASE RANGE VALUE
083B  E8 0ADD R                         CALL    PUT_POS         ; RESTORE STORAGE LOCATION
083E  EB 14                             JMP     SHORT K29_5     ; ADJUST
0840  3C 4B                     K29_4:  CMP     AL,LEFT_ARROW   ; ADJUST SREEN TO THE LEFT?
0842  75 1E                             JNE     K31             ; NOT AN ALT_CTRL SEQUENCE
0844  E8 0AD1 R                         CALL    GET_POS         ; GET NUMBER OF POSITIONS SCREEN IS
                                                                ; SHIFTED
0847  3C 04                             CMP     AL,RANGE        ; IS SCREEN SHIFTED AS FAR AS
                                                                ; POSSIBLE?
0849  7F 14                             JG      K29_6
084B  FE 06 0089 R                      INC     HORZ_POS        ; SHIFT SCREEN TO THE LEFT
084F  FE C0                             INC     AL              ; INCREASE NUMBER OF POSITIONS
                                                                ; SCREEN IS SHIFTED
0851  E8 0ADD R                         CALL    PUT_POS         ; PUT POSTION BACK IN STORAGE
0854  B0 02                     K29_5:  MOV     AL,2            ; ADJUST
0856  BA 03D4                           MOV     DX,3D4H         ; ADDRESS TO CRT CONTROLLER
0859  EE                                OUT     DX,AL
085A  A0 0089 R                         MOV     AL,HORZ_POS     ; COLUMN POSITION
085D  42                                INC     DX              ; POINT AT DATA REGISTER
085E  EE                                OUT     DX,AL           ; MOV POSITION
085F  E9 07C9 R                 K29_6:  JMP     K26
                                 ;------ IN ALTERNATE SHIFT, RESET NOT FOUND
0862                             K31:                            ; NO-RESET
0862  3C 39                             CMP     AL,57           ; TEST FOR SPACE KEY
0864  74 0B                             JZ      K31_1           ; GO TO K31_1
0866  3C 6C                             CMP     AL,108          ; TEST FOR MUHENKAN KEY
0868  74 07                             JZ      K31_1           ; GO TO K31_1
086A  3C 6D                             CMP     AL,109          ; TEST FOR HENKAN KEY
086C  74 03                             JZ      K31_1           ; GO TO K31_1
086E  EB 2A 90                          JMP     K32             ; GO TO K32
0871  B0 20                     K31_1:  MOV     AL,' '          ; SET SPACE CHAR
0873  E9 0A53 R                         JMP     K57             ; BUFFER_FILL
                                 ;------ ALT-INPUT-TABLE
0876                             K30     LABEL   BYTE
0876  70 72 73 74 75                    DB      70H,72H,73H,74H,75H
087B  76 77 78 79 7A                    DB      76H,77H,78H,79H,7AH ; 10 NUMBERS ON KEYPAD
                                 ;------ SUPER-SHIFT-TABLE
0880  10 11 12 13 14 15                 DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
      16 17
0888  18 19 1E 1F 20 21                 DB      24,25,30,31,32,33,34,35
      22 23
0890  24 25 26 2C 2D 2E                 DB      36,37,38,44,45,46,47,48
      2F 30
0898  31 32                            DB      49,50
                                 ;------ LOOK FOR KEY PAD ENTRY
089A                             K32:                            ; ALT-KEY-PAD
089A  BF 0876 R                         MOV     DI,OFFSET K30   ; ALT-INPUT-TABLE
089D  B9 000A                           MOV     CX,10           ; LOOK FOR ENTRY USING KEYPAD
08A0  F2/ AE                            REPNE   SCASB           ; LOOK FOR MATCH
08A2  75 13                             JNE     K33             ; NO_ALT_KEYPAD
08A4  81 EF 0877 R                      SUB     DI,OFFSET K30+1 ; DI NOW HAS ENTRY VALUE
08A8  A0 0019 R                         MOV     AL,ALT_INPUT    ; GET THE CURRENT BYTE
08AB  B4 0A                             MOV     AH,10           ; MULTIPLY BY 10
08AD  F6 E4                             MUL     AH
08AF  03 C7                             ADD     AX,DI           ; ADD IN THE LATEST ENTRY
08B1  A2 0019 R                         MOV     ALT_INPUT,AL    ; STORE IT AWAY
08B4  E9 07C9 R                         JMP     K26             ; THROW AWAY THAT KEYSTROKE
                                 ;------ LOOK FOR SUPERSHIFT ENTRY
08B7                             K33:                            ; NO-ALT-KEYPAD
08B7  C6 06 0019 R 00                   MOV     ALT_INPUT,0     ; ZERO ANY PREVIOUS ENTRY INTO
                                                                ; INPUT
08BC  B9 001A                           MOV     CX,26           ; DI,ES ALREADY POINTING
08BF  F2/ AE                            REPNE   SCASB           ; LOOK FOR MATCH IN ALPHABET
08C1  75 05                             JNE     K34             ; NOT FOUND, FUNCTION KEY OR OTHER
08C3  32 C0                             XOR     AL,AL           ; ASCII CODE OF ZERO
08C5  E9 0A53 R                         JMP     K57             ; PUT IT IN THE BUFFER
                                                                ; ALTERNATE SHIFT
08C8                             ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT
08C8                             K34:                            ; ALT-TOP-ROW
08CA  3C 02                             CMP     AL,2            ; KEY WITH '1' ON IT
08CC  72 0C                             JB      K35             ; NOT ONE OF INTERESTING KEYS
08CE  3C 0E                             CMP     AL,14           ;  IS IT IN THE REGION?
08D0  73 08                             JAE     K35             ;  ALT-FUNCTION
08D2  80 C4 76                          ADD     AH,118          ;  CONVERT PSUEDO SCAN CODE TO
                                                                ; RANGE
08D5  32 C0                             XOR     AL,AL           ;  INDICATE AS SUCH
08D7  E9 0A53 R                         JMP     K57             ;  BUFFER_FILL
                                                                ;  PSEUDO SCAN CODES
08DA                             ;------ TRANSLATE ALTERNATE SHIFT
08DA                             K35:                            ; ALT-FUNCTION
08DA  3C 3B                             CMP     AL,59           ; TEST FOR IN TABLE
08DC  73 03                             JAE     K37             ; ALT-CONTINUE
08DE                                                            ; CLOSE-RETURN
08DE  E9 07C9 R                 K36:    JMP     K26             ; IGNORE THE KEY
08E1                                                            ; ALT-CONTINUE
08E1  3C 47                     K37:    CMP     AL,71           ; IN KEYPAD REGION
```

Appendix A.

```
08E1  73 F9
08E3  BB 05B1 R
08E6  E9 0AC7 R

08E9
08E9  F6 06 0017 R 04
08EE  74 66


08F0  3C 46
08F2  75 19
08F4  8B 1E 001A R
08F8  C6 06 0071 R 80
08FD  CD 1B
08FF  2B C0
0901  89 07
0903  E8 04B9 R
0906  89 1E 001C R
090A  E9 07C9 R
090D

090D  3C 37
090F  75 06
0911  B8 7200
0914  E9 0A53 R

0917
0917  BB 04E0 R
091A  3C 3B
091C  7D 03
091E  E9 0A4F R
0921
0921  BB 051A R
0924  3C 7E
0926  75 05
0928  B0 0A
092A  E9 0A91 R
092D
092D  3C 6A
092F  73 47
0931  E9 0AC7 R

0934
0934  4A 4E 70 71 72 73
093A  74 75 76 77 78 79
0940  7A 7B 7C 7D 7E
= 0011

0945
0945  2D 2B 30 2E 31 32
      33 34 35 36 37 38
      39 2A 2F 2C 0D

0956  BF 0934 R
0959  B9 0011
095C  F2/ AE

095E  75 0C
0960  81 EF 0935 R
0964  2E: 8A 85 0945 R
0969  E9 0A53 R
096C
096C  3C 47
096E  73 72
0970  3C 1C
0972  75 07
0974  B0 0D
0976  CD 78
0978
0978  E9 07C9 R
097B
097B  F6 06 0017 R 03
0980  74 7B

0982  3C 0F
0984  75 06
0986  B8 0F00
0989  E9 0A53 R
098C
098C  3C 3B
098E  72 06
0990  BB 05A7 R
0993  E9 0AC7 R
0996
0996  52
0997  8B D0
0999  F6 06 0336 R 04
099E  74 12
09A0  BB 0D4B R
09A3  E8 0B6B R
09A6  8B C2
09A8  BB 0D85 R
09AB  E8 0B6B R
09AE  5A
09AF  E9 07C9 R
09B2
09B2  F6 06 0336 R 02
09B7  74 23
09B9  F6 06 0336 R 01
09BE  74 12
09C0  BB 0C63 R
09C3  E8 0B6B R
09C6  8B C2
09C8  BB 0C9D R
```

```
        JAE     K36             ; IF SO, IGNORE
        MOV     BX,OFFSET K13   ; ALT SHIFT PSEUDO SCAN TABLE
        JMP     K63             ; TRANSLATE THAT
;------ NOT IN ALTERNATE SHIFT
K38:                            ; NOT-ALT-SHIFT
        TEST    KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT?
        JZ      K45             ; NOT-CTL-SHIFT
;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS
;------ TEST FOR BREAK AND PAUSE KEYS
        CMP     AL,SCROLL_KEY   ; TEST FOR BREAK
        JNE     K41             ; NO-BREAK
        MOV     BX,BUFFER_HEAD  ; GET CURRENT BUFFER HEAD
        MOV     BIOS_BREAK,80H  ; TURN ON BIOS_BREAK BIT
        INT     1BH             ; BREAK INTERRUPT VECTOR
        SUB     AX,AX           ; PUT OUT DUMMY CHARACTER
        MOV     [BX],AX         ; PUT DUMMY CHAR AT BUFFER HEAD
        CALL    K4              ; UPDATE BUFFER POINTER
        MOV     BUFFER_TAIL,BX  ; UPDATE TAIL
        JMP     K26             ; DONE WITH INTERUPT
K41:                            ; NO-PAUSE
;------ TEST SPECIAL CASE KEY 55
        CMP     AL,55
        JNE     K42             ; NOT-KEY-55
        MOV     AX,114*256      ; START/STOP PRINTING SWITCH
        JMP     K57             ; BUFFER_FILL
;------ SET UP TO TRANSLATE CONTROL SHIFT
K42:                            ; NOT-KEY-55
        MOV     BX,OFFSET K8    ; SET UP TO TRANSLATE CTL
        CMP     AL,59           ; IS IT IN TABLE?
        JGE     K42_1
        JMP     K56             ; YES, GO TRANSLATE CHAR
K42_1:
        MOV     BX,OFFSET K9    ; CTL TABLE SCAN
        CMP     AL,7EH          ; IS NUMERIC PAD RETURN KEY
        JNE     K42_2           ; NO.
        MOV     AL,0AH          ; SET 'LF' CODE
        JMP     K57_4
K42_2:
        CMP     AL,6AH          ; IS IT IN TABLE?
        JAE     K45_2
        JMP     K63             ; TRANSLATE_SCAN
;------ TEN KEYPAD SCAN CODE TABLE
K43     LABEL   BYTE
        DB      4AH,4EH,70H,71H,72H,73H
        DB      74H,75H,76H,77H,78H,79H
        DB      7AH,7BH,7CH,7DH,7EH
K43L    EQU     $-K43
;------ TEN KEYPAD CHARACTER CODE TABLE
K44     LABEL   BYTE
        DB      '-+0.123456789*/',',0DH


;------ NOT IN CONTROL SHIFT
K45:    MOV     DI,OFFSET K43   ; SCAN CODE TABLE
        MOV     CX,K43L         ; LENGTH
        REPNE   SCASB           ; LOOK THROUGH THE TABLE FOR A
                                ; MATCH
        JNE     K45_1           ; IF NOT MATCH THEN K45_1
        SUB     DI,OFFSET K43+1 ; ADJUST PTR TO SCAN CODE MATCH
        MOV     AL,CS:K44[DI]   ; GET CHARACTER CODE INTO AL
        JMP     K57             ; BUFFER_FILL
K45_1:
        CMP     AL,71           ; TEST FOR KEYPAD REGION
        JAE     K48             ; HANDLE KEYPAD REGION
        CMP     AL,28           ; TEST FOR CR KEY
        JNE     K45_3
        MOV     AL,13           ; CR CODE
        INT     78H             ; KANAKAN ROUTINE
K45_2:
        JMP     K26             ; INTERRUPT RETURN
K45_3:
        TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
        JZ      K54             ; TEST FOR SHIFT STATE
;------ UPPER CASE, HANDLE SPECIAL CASES
        CMP     AL,15           ; BACK TAB KEY
        JNE     K46             ; NOT-BACK-TAB
        MOV     AX,15*256       ; SET PSEUDO SCAN CODE
        JMP     K57             ; BUFFER_FILL
K46:                            ; NOT-PRINT-SCREEN
        CMP     AL,59           ; FUNCTION KEYS
        JB      K47             ; NOT-UPPER-FUNCTION
        MOV     BX,OFFSET K12   ; UPPER CASE PSEUDO SCAN CODES
        JMP     K63             ; TRANSLATE_SCAN
K47:                            ; NOT-UPPER-FUNCTION
        PUSH    DX                      ;
        MOV     DX,AX                   ; STORE AX TO DX
        TEST    JKB_FLAG,HIRAGANA_STATE ; IS HIRAGANA STATE ?
        JZ      K47_1                   ; NO. JMP K47_1
        MOV     BX,OFFSET H_Z_U_1ST     ; ZENKAKU HIRAGANA UPPER SHIFT 1ST BYTE
        CALL    CK                      ; HANDLING
        MOV     AX,DX                   ; RESTORE DX TO AX
        MOV     BX,OFFSET H_Z_U_2ND     ; ZENKAKU HIRAGANA UPPER SHIFT 2ND BYTE
        CALL    CK                      ; HANDLING
        POP     DX                      ;
        JMP     K26                     ; INTERRUPT RETURN
K47_1:
        TEST    JKB_FLAG,KATAKANA_STATE ; IS KATAKANA STATE ?
        JZ      K47_3                   ; NO. JMP K47_3
        TEST    JKB_FLAG,ZENKAKU_STATE  ; IS ZENKAKU STATE ?
        JZ      K47_2                   ; NO. JMP K47_2
        MOV     BX,OFFSET K_Z_U_1ST     ; ZENKAKU KATAKANA UPPER SHIFT 1ST BYTE
        CALL    CK                      ; HANDLING
        MOV     AX,DX                   ; RESTORE DX TO AX
        MOV     BX,OFFSET K_Z_U_2ND     ; ZENKAKU KATAKANA UPPER SHIFT 2ND BYTE
```

```
09CB  E8 0B6B R
09CE  5A
09CF  E9 07C9 R
09D2
09D2  BB 0BB5 R
09D5  E8 0B6B R
09D8  5A
09D9  E9 07C9 R
09DC
09DC  5A
09DD  BB 056D R
09E0  EB 6D

09E2
09E2  3C 53
09E4  77 08
09E6  2C 47
09E8  BB 05C8 R
09EB  E9 0AC9 R
09EE
09EE  3C 6A
09F0  75 06
09F2  E8 0B13 R
09F5  E9 07C9 R
09F8
09F8  B0 20
09FA  EB 57 90

09FD
09FD  3C 3B
09FF  72 04
0A01  32 C0
0A03  EB 4E
0A05
0A05  52
0A06  8B D0
0A08  F6 06 0336 R 04
0A0D  74 12
0A0F  BB 0CD7 R
0A12  E8 0B6B R
0A15  8B C2
0A17  BB 0D11 R
0A1A  E8 0B6B R
0A1D  5A
0A1E  E9 07C9 R
0A21
0A21  F6 06 0336 R 02
0A26  74 23
0A28  F6 06 0336 R 01
0A2D  74 12
0A2F  BB 0BEF R
0A32  E8 0B6B R
0A35  8B C2
0A37  BB 0C29 R
0A3A  E8 0B6B R
0A3D  5A
0A3E  E9 07C9 R
0A41
0A41  BB 0B7B R
0A44  E8 0B6B R
0A47  5A
0A48  E9 07C9 R
0A4B
0A4B  5A
0A4C  BB 0533 R

0A4F
0A4F  FE C8
0A51  2E: D7

0A53
0A53  3D 297E
0A56  75 12
0A58  F6 06 0336 R 01
0A5D  74 22
0A5F  B0 81
0A61  CD 78
0A63  B0 60
0A65  CD 78
0A67  E9 07C9 R
0A6A
0A6A  3D 2B5C
0A6D  75 22
0A6F  F6 06 0336 R 01
0A74  74 0B
0A76  B0 81
0A78  CD 78
0A7A  B0 5F
0A7C  CD 78
0A7E  E9 07C9 R
0A81
0A81  53
0A82  51
0A83  BB 0080
0A86  B9 0048
0A89  E8 0000 E
0A8C  59
0A8D  5B
0A8E  E9 07C9 R
0A91
0A91  3C FF
0A93  74 1F
0A95  80 FC FF
```

```
        CALL    CK                      ; HANDLING
        POP     DX                      ;
        JMP     K26                     ; INTERRUPT RETURN
K47_2:
        MOV     BX,OFFSET K_H_U         ; HANKAKU KATAKANA UPPER SHIFT
        CALL    CK                      ; HANDLING
        POP     DX                      ;
        JMP     K26                     ; INTERRUPT RETURN
K47_3:
        POP     DX
        MOV     BX,OFFSET K11   ; POINT TO UPPER CASE TABLE
        JMP     SHORT K56       ; OK, TRANSLATE THE CHAR
;------
K48:                            ; KEYPAD-REGION
        CMP     AL,83           ; IF SCAN CODE > 83
        JA      K49             ;       THEN K49
        SUB     AL,71           ; CONVERT ORIGIN
        MOV     BX,OFFSET K15   ; BASE CASE TABLE
        JMP     K64             ; CONVERT TO PSEUDO SCAN
K49:
        CMP     AL,6AH          ; IS SPECIAL KEY
        JNE     K50             ; NO. JMP K50
        CALL    SK              ; SPECIAL KEY HANDLING
        JMP     K26             ; INTERRUPT RETURN
K50:
        MOV     AL,' '          ;
        JMP     K57             ;
;------ PLAIN OLD LOWER CASE
K54:                            ; NOT-SHIFT
        CMP     AL,59           ; TEST FOR FUNCTION KEYS
        JB      K55             ; NOT-LOWER-FUNCTION
        XOR     AL,AL           ; SCAN CODE IN AH ALREADY
        JMP     SHORT K57       ; BUFFER_FILL
K55:                            ; NOT-LOWER-FUNCTION
        PUSH    DX                      ;
        MOV     DX,AX                   ; STORE AX TO DX
        TEST    JKB_FLAG,HIRAGANA_STATE ; IS HIRAGANA STATE ?
        JZ      K55_1                   ; NO. JMP K47_1
        MOV     BX,OFFSET H_Z_L_1ST     ; ZENKAKU HIRAGANA LOWER SHIFT 1ST BYTE
        CALL    CK                      ; HANDLING
        MOV     AX,DX                   ; RESTORE DX TO AX
        MOV     BX,OFFSET H_Z_L_2ND     ; ZENKAKU HIRAGANA LOWER SHIFT 2ND BYTE
        CALL    CK                      ; HANDLING
        POP     DX                      ;
        JMP     K26                     ; INTERRUPT RETURN
K55_1:
        TEST    JKB_FLAG,KATAKANA_STATE ; IS KATAKANA STATE ?
        JZ      K55_3                   ; NO. JMP K47_3
        TEST    JKB_FLAG,ZENKAKU_STATE  ; IS ZENKAKU STATE ?
        JZ      K55_2                   ; NO. JMP K47_2
        MOV     BX,OFFSET K_Z_L_1ST     ; ZENKAKU KATAKANA LOWER SHIFT 1ST BYTE
        CALL    CK                      ; HANDLING
        MOV     AX,DX                   ; RESTORE DX TO AX
        MOV     BX,OFFSET K_Z_L_2ND     ; ZENKAKU KATAKANA LOWER SHIFT 2ND BYTE
        CALL    CK                      ; HANDLING
        POP     DX                      ;
        JMP     K26                     ; INTERRUPT RETURN
K55_2:
        MOV     BX,OFFSET K_H_L         ; HANKAKU KATAKANA LOWER SHIFT
        CALL    CK                      ; HANDLING
        POP     DX                      ;
        JMP     K26                     ; INTERRUPT RETURN
K55_3:
        POP     DX
        MOV     BX,OFFSET K10   ; LC TABLE
;------ TRANSLATE THE CHARACTER
K56:                            ; TRANSLATE-CHAR
        DEC     AL              ; CONVERT ORIGIN
        XLAT    CS:K11          ; CONVERT THE SCAN CODE TO ASCII
;------ PUT CHARACTER INTO BUFFER
K57:                            ; BUFFER-FILL
        CMP     AX,0297EH       ; IS THIS A 'TIRDE'
        JNE     K57_1           ; NOT 'TIRDE'
        TEST    JKB_FLAG,ZENKAKU_STATE  ; ZENKAKU_STATE ?
        JZ      K57_3           ; GO TO BEEP
        MOV     AL,81H          ; 1ST BYTE OF 'TIRDE'
        INT     78H             ; KANAKAN
        MOV     AL,60H          ; 2ND BYTE OF 'TIRDE'
        INT     78H             ; KANAKAN
        JMP     K26             ; INTERRUPT RETURN
K57_1:
        CMP     AX,02B5CH       ; IS THIS A 'REVERSE SLASH'
        JNE     K57_4           ; NOT 'REVERSE SLASH'
        TEST    JKB_FLAG,ZENKAKU_STATE  ; ZENKAKU_STATE ?
        JZ      K57_3           ; GO TO BEEP
        MOV     AL,81H          ; 1ST BYTE OF 'REVERSE SLASH'
        INT     78H             ; KANAKAN
        MOV     AL,5FH          ; 2ND BYTE OF 'REVERSE SLASH'
        INT     78H             ; KANAKAN
        JMP     K26             ; INTERRUPT RETURN
K57_3:
        PUSH    BX
        PUSH    CX              ; DURATION OF ERROR BEEP
        MOV     BX,80H          ; FREQUENCY OF TONE
        MOV     CX,48H          ; BUFFER FULL BEEP
        CALL    KB_NOISE
        POP     CX
        POP     BX              ; INTERRUPT RETURN
        JMP     K26
K57_4:                          ; IS THIS AN IGNORE CHAR?
        CMP     AL,-1           ; YES, DO NOTHING WITH IT
        JE      K59             ; LOOK FOR -1 PSEUDO SCAN
        CMP     AH,-1
```

```
0A98  74 1A                     JE      K59             ; NEAR_INTERRUPT_RETURN
                        ;------ HANDLE THE CAPS LOCK PROBLEM
                        K58:                            ; BUFFER-FILL-NOTEST
0A9A  F6 06 0017 R 40           TEST    KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE?
0A9F  74 20                     JZ      K61             ; SKIP IF NOT
                        ;------ IN CAPS LOCK STATE
0AA1  F6 06 0017 R 03           TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT
                                                        ; STATE
0AA6  74 0F                     JZ      K60             ; IF NOT SHIFT, CONVERT LOWER TO
                                                        ; UPPER
                        ;------ CONVERT ANY UPPER CASE TO LOWER CASE
0AA8  3C 41                     CMP     AL,'A'          ; FIND OUT IF ALPHABETIC
0AAA  72 15                     JB      K61             ; NOT_CAPS_STATE
0AAC  3C 5A                     CMP     AL,'Z'
0AAE  77 11                     JA      K61             ; NOT_CAPS_STATE
0AB0  04 20                     ADD     AL,'a'-'A'      ; CONVERT TO LOWER CASE
0AB2  EB 0D                     JMP     SHORT K61       ; NOT_CAPS_STATE
0AB4                    K59:
0AB4  E9 07C9 R                 JMP     K26             ; INTERRUPT_RETURN
                        ;------ CONVERT ANY LOWER CASE TO UPPER CASE
0AB7                    K60:                            ; LOWER-TO-UPPER
0AB7  3C 61                     CMP     AL,'a'          ; FIND OUT IF ALPHABETIC
0AB9  72 06                     JB      K61             ; NOT_CAPS_STATE
0ABB  3C 7A                     CMP     AL,'z'
0ABD  77 02                     JA      K61             ; NOT_CAPS_STATE
0ABF  2C 20                     SUB     AL,'a'-'A'      ; CONVERT TO UPPER CASE
0AC1                    K61:                            ; NOT-CAPS-STATE
0AC1  E8 0DBF R                 CALL    ZEN_AN          ; TRANSLATE A/N TO ZENKAKU
0AC4  E9 07C9 R                 JMP     K26             ; INTERRUPT_RETURN
                        ;------ TRANSLATE SCAN FOR PSEUDO SCAN CODES
0AC7                    K63:                            ; TRANSLATE-SCAN
0AC7  2C 3B                     SUB     AL,59           ; CONVERT ORIGIN TO FUNCTION KEYS
0AC9                    K64:                            ; TRANSLATE-SCAN-ORGD
0AC9  2E: D7                    XLAT    CS:K9           ; CTL TABLE SCAN
0ACB  8A E0                     MOV     AH,AL           ; PUT VALUE INTO AH
0ACD  32 C0                     XOR     AL,AL           ; ZERO ASCII CODE
0ACF  EB 82                     JMP     K57             ; PUT IT INTO THE BUFFER
0AD1                    KB_INT  ENDP
                        ;--------------------------------------------------------------------
                        ;GET_POS
                        ;       THIS ROUTINE WILL SHIFT THE VALUE STORED IN THE HIGH NIBBLE
                        ;       OF THE VARIABLE VAR_DELAY TO THE LOW NIBBLE.
                        ;INPUT
                        ;       NONE.  IT IS ASSUMED THAT DS POINTS AT THE BIOS DATA AREA
                        ;OUTPUT
                        ;       AL CONTAINS THE SHIFTED VALUE.
                        ;--------------------------------------------------------------------
0AD1                    GET_POS PROC    NEAR
0AD1  51                        PUSH    CX              ; SAVE SHIFT REGISTER
0AD2  A0 0086 R                 MOV     AL,BYTE PTR VAR_DELAY ; GET STORAGE LOCATION
0AD5  24 F0                     AND     AL,0F0H         ; MASK OFF LOW NIBBLE
0AD7  B1 04                     MOV     CL,4            ; SHIFT OF FOUR BIT POSITIONS
0AD9  D2 F8                     SAR     AL,CL           ; SHIFT THE VALUE SIGH EXTENDED
0ADB  59                        POP     CX              ; RESTORE THE VALUE
0ADC  C3                        RET
0ADD                    GET_POS ENDP
                        ;--------------------------------------------------------------------
                        ;PUT_POS
                        ;       THIS ROUTINE WILL TAKE THE VALUE IN LOW ORDER NIBBLE IN
                        ;       AL AND STORE IT IN THE HIGH ORDER OF VAR_DELAY
                        ;INPUT
                        ;       AL CONTAINS THE VALUE FOR STORAGE
                        ;OUTPUT
                        ;       NONE.
                        ;--------------------------------------------------------------------
0ADD                    PUT_POS PROC    NEAR
0ADD  51                        PUSH    CX              ; SAVE REGISTER
0ADE  B1 04                     MOV     CL,4            ; SHIFT COUNT
0AE0  D2 E0                     SHL     AL,CL           ; PUT IN HIGH ORDER NIBBLE
0AE2  8A 0E 0086 R              MOV     CL,BYTE PTR VAR_DELAY ; GET DATA BYTE
0AE6  80 E1 0F                  AND     CL,0FH          ; CLEAR OLD VALUE IN HIGH NIBBLE
0AE9  0A C1                     OR      AL,CL           ; COMBINE HIGH AND LOW NIBBLES
0AEB  A2 0086 R                 MOV     BYTE PTR VAR_DELAY,AL ; PUT IN POSITION
0AEE  59                        POP     CX              ; RESTORE REGISTER
0AEF  C3                        RET
0AF0                    PUT_POS ENDP
                        ;--------------------------------------------------------------------
                        ;ERROR_BEEP
                        ;       THIS ROUTINE WILL ERROR BEEP
                        ;INPUT
                        ;       NONE.
                        ;OUTPUT
                        ;       NONE.
                        ;
                        ;  NOTE:
                        ;       THIS ROUTINE DESTROY BX AND CX.
                        ;       THIS ROUTINE CALL KB_NOISE.
                        ;--------------------------------------------------------------------
0AF0                    ERROR_BEEP      PROC    NEAR
0AF0  BB 0080                   MOV     BX,80H
0AF3  B9 0048                   MOV     CX,48H          ; DURATION OF ERROR BEEP
0AF6  E8 0000 E                 CALL    KB_NOISE        ; FREQUENCY OF TONE
0AF9  80 26 0017 R F0           AND     KB_FLAG,0F0H    ; BUFFER FULL BEEP
                                                        ; CLEAR ALT,CLRL,LEFT AND RIGHT
0AFE  80 26 0018 R 0F           AND     KB_FLAG_1,0FH   ; SHIFTS
                                                        ; CLEAR POTENTIAL BREAK OF INS,CAPS
0B03  80 26 0088 R 1F           AND     KB_FLAG_2,1FH   ; ,NUM AND SCROLL SHIFT
0B08  80 26 0337 R 00           AND     JKB_FLAG_1,00H  ; CLEAR FUNCTION STATES
                                                        ; CLEAR HANKAKU,ZENKAKU,HIRAGANA,KATAKANA AND
0B0D  80 26 0338 R 0F           AND     JKB_FLAG_2,0FH  ; ALPHA SHIFT
0B12  C3                        RET                     ; CLEAR KANJI,KNUMBER,MUHEN AND HENKAN SHIFT
0B13                    ERROR_BEEP      ENDP
                        ;--------------------------------------------------------------------
                        ;       SPACIAL CASE KEY (06AH) HANDLING
```

```
0B13
0B13  F6 06 0336 R 05          ;
0B18  75 24                    ;       NNOTEN
0B1A  F6 06 0017 R 03          ;           THIS KEY IS 'JIS' UNIQUE KEY ('ASCII' NOT INCLUDE).
0B1F  74 0D                    ;-----------------------------------------------------------------
0B21  F6 06 0336 R 02          ;
0B26  74 02                    SK      PROC    NEAR
0B28  EB 13                            TEST    JKB_FLAG,ZENKAKU_CHAR           ; ZENKAKU CHARACTER ?
0B2A                                   JNZ     SK6                             ; YES, ZENKAKU STATE
0B2A  B0 7E                            TEST    KB_FLAG,RIGHT_SHIFT+LEFT_SHIFT  ; UPPER CASE?
0B2C  EB 0D                            JZ      SK2                             ; NOT UPPER CASE
0B2E                                   TEST    JKB_FLAG,KATAKANA_STATE         ; KATAKANA STATE ?
0B2E  F6 06 0336 R 02          ;       JZ      SK1                             ; NOT KATAKANA STATE
0B33  74 04                    ;       JMP     SHORT SK5                       ; GO TO RETURN
0B35  B0 B0                    SK1:
0B37  EB 02                            MOV     AL,07EH                         ; 'OVERSCORE'
0B39                           ;       JMP     SHORT SK4                       ;  CALL KANAKAN
0B39  B0 5C                    SK2:
0B3B                                   TEST    JKB_FLAG,KATAKANA_STATE         ; KATAKANA STATE ?
0B3B  CD 78                    ;       JZ      SK3                             ; NOT KATAKANA STATE
0B3D                           ;       MOV     AL,0B0H                         ; 'PROLONGED SOUND'
0B3D  C3                       ;       JMP     SHORT  SK4                      ; CALL KANAKAN
0B3E                           SK3:
0B3E  52                               MOV     AL,05CH                         ; 'YEN SIGN'
0B3F  F6 06 0017 R 03          SK4:
0B44  74 0C                            INT     78H                             ; KANAKAN ROUTINE
0B46  F6 06 0336 R 06          SK5:
0B4B  75 1C                            RET
0B4D  BA 8150                  SK6:
0B50  EB 0F                            PUSH    DX                              ;
0B52                                   TEST    KB_FLAG,RIGHT_SHIFT+LEFT_SHIFT  ; UPPER CASE ?
0B52  F6 06 0336 R 06          ;       JZ      SK7                             ; NOT UPPER CASE
0B57  75 05                    ;       TEST    JKB_FLAG,NOT ALPHA_STATE        ; ALPHA_STATE
0B59  BA 818F                  ;       JNZ     SK10                            ; NOT ALPHA STATE
0B5C  EB 03                    ;       MOV     DX,08150H                       ; 'OVERSCORE'
0B5E                           ;       JMP     SHORT SK9                       ; CALL KANAKAN
0B5E  BA 815B                  SK7:
0B61                                   TEST    JKB_FLAG,NOT ALPHA_STATE        ; ALPHA_STATE
0B61  8A C6                    ;       JNZ     SK8                             ; NOT ALPHA STATE
0B63  CD 78                    ;       MOV     DX,0818FH                       ; 'YEN SIGN'
0B65  8A C2                    ;       JMP     SHORT SK9                       ; CALL KANAKAN
0B67  CD 78                    SK8:
0B69                                   MOV     DX,0815BH                       ; 'PROLONGED SOUND'
0B69  5A                       SK9:
0B6A  C3                               MOV     AL,DH                           ;
0B6B                                   INT     78H                             ; KANAKAN ROUTINE
                                       MOV     AL,DL                           ;
                                       INT     78H                             ; KANAKAN ROUTINE
                               SK10:
                                       POP     DX                              ;
                                       RET                                     ;
                               SK      ENDP
                               PAGE
                               ;-----------------------------------------------------------------
                               ;       JAPANEASE CHARACTER SET HANDLING
                               ;
                               ;               INPUT
                               ;                       AH = SCAN CODE
                               ;                       AL = OFFSET FROM TABLE'S FIRST BYTE + 1
                               ;                       BX = OFFSET OF TABLE'S FIST BYTE
                               ;               OUTPUT
                               ;                       NONE
                               ;-----------------------------------------------------------------
                               ;
0B6B                           CK      PROC    NEAR
0B6B  FE C8                            DEC     AL      ;
0B6D  2E: D7                           XLAT    CS:K11  ;
0B6F  3C FF                            CMP     AL,-1   ;
0B71  74 07                            JE      CK1     ;
0B73  80 FC FF                         CMP     AH,-1   ;
0B76  74 02                            JE      CK1     ;
0B78  CD 78                            INT     78H     ;
0B7A                           CK1:
0B7A  C3                               RET             ;
0B7B                           CK      ENDP
0B7B                           K_H_L   LABEL   BYTE    ; KATAKANA HANKAKU LOWER CASE
0B7B  1B C7 CC B1 B3 B4                DB      1BH,0C7H,0CCH,0B1H,0B3H,0B4H,0B5H,0D4H,0D5H,0D6H,0DCH,0CEH,0CDH,08H
      B5 D4 D5 D6 DC CE
      CD 08
0B89  09 C0 C3 B2 BD B6                DB      09H,0C0H,0C3H,0B2H,0BDH,0B6H,0DDH,0C5H,0C6H,0D7H,0BEH,0DEH,0DFH,-1
      DD C5 C6 D7 BE DE
      DF FF
0B97  FF C1 C4 BC CA B7                DB      -1,0C1H,0C4H,0BCH,0CAH,0B7H,0B8H,0CFH,0C9H,0D8H,0DAH,0B9H,0D1H
      B8 CF C9 D8 DA B9
      D1
0BA4  FF DB C2 BB BF CB                DB      -1,0DBH,0C2H,0BBH,0BFH,0CBH,0BAH,0D0H,0D3H,0C8H,0D9H,0D2H,-1,02AH
      BA D0 D3 C8 D9 D2
      FF 2A
0BB2  FF 20 FF                         DB      -1,20H,-1
0BB5                           ;
0BB5                           K_H_U   LABEL   BYTE    ; KATAKANA HANKAKU UPPER CASE
      1B FF FF A7 A9 AA                DB      1BH,-1,-1,0A7H,0A9H,0AAH,0ABH,0ACH,0ADH,0AEH,0A6H,-1,-1,08H
      AB AC AD AE A6 FF
      FF 08
0BC3  09 FF FF A8 FF FF                DB      09H,-1,-1,0A8H,-1,-1,-1,-1,-1,-1,-1,-1,0A2H,-1
      FF FF FF FF FF FF
      A2 FF
0BD1  FF FF FF FF FF FF                DB      -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,0A3H
      FF FF FF FF FF FF
      A3
0BDE  FF FF AF FF FF FF                DB      -1,-1,0AFH,-1,-1,-1,-1,-1,-1,0A4H,0A1H,0A5H,-1,-1
      FF FF FF A4 A1 A5
      FF 20 FF
0BEC                                   DB      -1,20H,-1
```

```
0BEF
0BEF  1B 83 83 83 83 83
      83 83 83 83 83 83
      83 08
0BFD  09 83 83 83 83 83
      83 83 83 83 83 81
      81 FF
0C0B  FF 83 83 83 83 83
      83 83 83 83 83 83
      83
0C18  FF 83 83 83 83 83
      83 83 83 83 83 83
      FF 81
0C26  FF 81 FF

0C29
0C29  FF 6B 74 41 45 47
      49 84 86 88 8F 7A
      77 FF
0C37  FF 5E 65 43 58 4A
      93 69 6A 89 5A 4A
      4B FF
0C45  FF 60 67 56 6E 4C
      4E 7D 6D 8A 8C 50
      80
0C52  FF 8D 63 54 5C 71
      52 7E 82 6C 8B 81
      FF 96
0C60  FF 40 FF

0C63
0C63  1B FF FF 83 83 83
      83 83 83 83 83 81
      81 08
0C71  09 FF FF 83 FF FF
      FF FF FF FF 81 FF
      81 FF
0C7F  FF FF FF FF FF FF
      FF FF FF FF 81 83
      81
0C8C  FF FF 83 FF FF FF
      FF FF FF 81 81 81
      FF FF
0C9A  FF 81 FF

0C9D
0C9D  FF FF FF 40 44 46
      48 83 85 87 92 92
      58 FF
0CAB  FF FF FF 42 FF FF
      FF FF FF FF 77 FF
      75 FF
0CB9  FF FF FF FF FF FF
      FF FF FF FF 78 96
      76
0CC6  FF FF 62 FF FF FF
      FF FF FF 41 42 45
      FF FF
0CD4  FF 40 FF

0CD7
0CD7  1B 82 82 82 82 82
      82 82 82 82 82 82
      82 08
0CE5  09 82 82 82 82 82
      82 82 82 82 82 81
      81 FF
0CF3  FF 82 82 82 82 82
      82 82 82 82 82 82
      82
0D00  FF 82 82 82 82 82
      82 82 82 82 82 82
      FF 81
0D0E  FF 81 FF

0D11
0D11  FF CA D3 A0 A4 A6
      A8 E2 E4 E6 ED D9
      D6 FF
0D1F  FF BD C4 A2 B7 A9
      F1 C8 C9 E7 B9 4A
      4B FF
0D2D  FF BF C6 B5 CD AB
      AD DC CC E8 EA AF
      DE
0D3A  FF EB C2 B3 BB D0
      B1 DD E0 CB E9 DF
      FF 96
0D48  FF 40 FF

0D4B
0D4B  1B FF FF 82 82 82
      82 82 82 82 82 81
      81 08
0D59  09 FF FF 82 FF FF
      FF FF FF FF 81 FF
      81 FF
0D67  FF FF FF FF FF FF
      FF FF FF FF 81 83
      81
0D74  FF FF 82 FF FF FF
      FF FF FF 81 81 81
      FF FF
```

```
;
K_Z_L_1ST   LABEL   BYTE    ; KATAKANA ZENKAKU LOWER 1ST BYTE
            DB  1BH,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,08H

            DB  09H,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,81H,81H,-1

            DB  -1,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H

            DB  -1,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,-1,81H

            DB  -1,81H,-1
;
K_Z_L_2ND   LABEL   BYTE    ; KATAKANA ZENKAKU LOWER 2ND BYTE
            DB  -1,6BH,74H,41H,45H,47H,49H,84H,86H,88H,8FH,7AH,77H,-1

            DB  -1,5EH,65H,43H,58H,4AH,93H,69H,6AH,89H,5AH,4AH,4BH,-1

            DB  -1,60H,67H,56H,6EH,4CH,4EH,7DH,6DH,8AH,8CH,50H,80H

            DB  -1,8DH,63H,54H,5CH,71H,52H,7EH,82H,6CH,8BH,81H,-1,96H

            DB  -1,40H,-1
;
K_Z_U_1ST   LABEL   BYTE    ; KATAKANA ZENKAKU UPPER 1ST BYTE
            DB  1BH,-1,-1,83H,83H,83H,83H,83H,83H,83H,83H,81H,81H,08H

            DB  09H,-1,-1,83H,-1,-1,-1,-1,-1,-1,81H,-1,81H,-1

            DB  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,81H,83H,81H

            DB  -1,-1,83H,-1,-1,-1,-1,-1,-1,81H,81H,81H,-1,-1

            DB  -1,81H,-1
;
K_Z_U_2ND   LABEL   BYTE    ; KATAKANA ZENKAKU UPPER 2ND BYTE
            DB  -1,-1,-1,40H,44H,46H,48H,83H,85H,87H,92H,92H,58H,-1

            DB  -1,-1,-1,42H,-1,-1,-1,-1,-1,-1,77H,-1,75H,-1

            DB  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,78H,96H,76H

            DB  -1,-1,62H,-1,-1,-1,-1,-1,-1,41H,42H,45H,-1,-1

            DB  -1,40H,-1
;
H_Z_L_1ST   LABEL   BYTE    ; HIRAGANA ZENKAKU LOWER 1ST BYTE
            DB  1BH,82H,82H,82H,82H,82H,82H,82H,82H,82H,82H,82H,82H,08H

            DB  09H,82H,82H,82H,82H,82H,82H,82H,82H,82H,82H,81H,81H,-1

            DB  -1,82H,82H,82H,82H,82H,82H,82H,82H,82H,82H,82H,82H

            DB  -1,82H,82H,82H,82H,82H,82H,82H,82H,82H,82H,82H,-1,81H

            DB  -1,81H,-1
;
H_Z_L_2ND   LABEL   BYTE    ; HIRAGANA ZENKAKU LOWER 2ND BYTE
            DB  -1,0CAH,0D3H,0A0H,0A4H,0A6H,0A8H,0E2H,0E4H,0E6H,0EDH,0D9H,0D6H,-1

            DB  -1,0BDH,0C4H,0A2H,0B7H,0A9H,0F1H,0C8H,0C9H,0E7H,0B9H,4AH,4BH,-1

            DB  -1,0BFH,0C6H,0B5H,0CDH,0ABH,0ADH,0DCH,0CCH,0E8H,0EAH,0AFH,0DEH

            DB  -1,0EBH,0C2H,0B3H,0BBH,0D0H,0B1H,0DDH,0E0H,0CBH,0E9H,0DFH,-1,96H

            DB  -1,40H,-1
;
H_Z_U_1ST   LABEL   BYTE    ; HIRAGANA ZENKAKU UPPER 1ST BYTE
            DB  1BH,-1,-1,82H,82H,82H,82H,82H,82H,82H,82H,81H,81H,08H

            DB  09H,-1,-1,82H,-1,-1,-1,-1,-1,-1,81H,-1,81H,-1

            DB  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,81H,83H,81H

            DB  -1,-1,82H,-1,-1,-1,-1,-1,-1,81H,81H,81H,-1,-1
```

```
0D82  FF 81 FF

0D85
0D85  FF FF FF 9F A3 A5
      A7 E1 E3 E5 F0 92
      58 FF
0D93  FF FF FF A1 FF FF
      FF FF FF FF 77 FF
      75 FF
0DA1  FF FF FF FF FF FF
      FF FF FF FF 78 96
      76
0DAE  FF FF C1 FF FF FF
      FF FF FF 41 42 45
      FF FF
0DBC  FF 40 FF

0DBF

0DBF  52
0DC0  8B D0
0DC2  F6 06 0336 R 81
0DC7  74 1E
0DC9  3C 20
0DCB  7C 1A
0DCD  3C 7E
0DCF  7F 16
0DD1  2C 20
0DD3  BB 0DEB R
0DD6  2E: D7

0DD8  CD 78
0DDA  8B C2
0DDC  2C 20
0DDE  BB 0E4A R
0DE1  2E: D7

0DE3  CD 78
0DE5  5A
0DE6  C3
0DE7
0DE7  CD 78
0DE9  5A
0DEA  C3
0DEB

0DEB
0DEB  81 81 81 81 81 81
      81 81
0DF3  81 81 81 81 81 81
      81 81
0DFB  82 82 82 82 82 82
      82 82
0E03  82 82 81 81 81 81
      81 81
0E0B  81 82 82 82 82 82
      82 82
0E13  82 82 82 82 82 82
      82 82
0E1B  82 82 82 82 82 82
      82 82
0E23  82 82 82 81 81 81
      81 81
0E2B  81 82 82 82 82 82
      82 82
0E33  82 82 82 82 82 82
      82 82
0E3B  82 82 82 82 82 82
      82 82
0E43  82 82 82 81 81 81
      81
0E4A
0E4A  40 49 8D 94 90 93
      95 4C
0E52  69 6A 96 7B 43 7C
      44 5E
0E5A  4F 50 51 52 53 54
      55 56
0E62  57 58 46 47 83 81
      84 48
0E6A  97 60 61 62 63 64
      65 66
0E72  67 68 69 6A 6B 6C
      6D 6E
0E7A  6F 70 71 72 73 74
      75 76
0E82  77 78 79 6D 8F 6E
      4F 51
0E8A  4D 81 82 83 84 85
      86 87
0E92  88 89 8A 8B 8C 8D
      8E 8F
0E9A  90 91 92 93 94 95
      96 97
0EA2  98 99 9A 6F 62 70
      50
```

```
        DB    -1,81H,-1
;
H_Z_U_2ND       LABEL   BYTE    ; HIRAGANA ZENKAKU UPPER 2ND BYTE
        DB    -1,-1,-1,9FH,0A3H,0A5H,0A7H,0E1H,0E3H,0E5H,0F0H,92H,58H,-1

        DB    -1,-1,-1,0A1H,-1,-1,-1,-1,-1,-1,77H,-1,75H,-1

        DB    -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,78H,96H,76H

        DB    -1,-1,0C1H,-1,-1,-1,-1,-1,-1,41H,42H,45H,-1,-1

        DB    -1,40H,-1
;
;-----------------------------------------------------------------------
;    ZEN_AN
;            TRANSLATE A/N TO ZENKAKU A/N
;             INPUT   AX
;                       AH = SCAN CODE
;                       AL = 1BYTE JIS CODE (20H-7EH)
;             OUTPUT  NONE
;-----------------------------------------------------------------------
ZEN_AN  PROC    NEAR
;
        PUSH    DX
        MOV     DX,AX           ; SAVE AX IN DX
        TEST    JKB_FLAG,ZENKAKU_STATE
        JZ      H_KAKU          ; IF NOT ZENKAKU THEN H_KAKU
        CMP     AL,20H          ; IF AL<20H THEN H_KAKU
        JL      H_KAKU          ;
        CMP     AL,7EH          ; IF AL>7EH THEN H_KAKU
        JG      H_KAKU          ;
        SUB     AL,20H          ; CONVERT ORIGIN
        MOV     BX,OFFSET Z_ALPHA1
        XLAT    CS:Z_ALPHA1     ; CONVERT THE SCAN CODE TO 1ST BYTE OF
                                ; CHRACTER CODE
        INT     78H             ; KANAKAN ROUTINE
        MOV     AX,DX           ; RESTORE AX
        SUB     AL,20H          ; CONVERT ORIGIN
        MOV     BX,OFFSET Z_ALPHA2
        XLAT    CS:Z_ALPHA2     ; CONVERT THE SCAN CODE TO 2ND BYTE OF
                                ; CHRACTER CODE
        INT     78H             ; KANAKAN ROUTINE
        POP     DX
        RET
H_KAKU:
        INT     78H
        POP     DX
        RET
ZEN_AN  ENDP
;
Z_ALPHA1        LABEL BYTE      ; ZENKAKU_ALPHA 1ST BYTE
        DB      81H,81H,81H,81H,81H,81H,81H,81H

        DB      81H,81H,81H,81H,81H,81H,81H,81H

        DB      82H,82H,82H,82H,82H,82H,82H,82H

        DB      82H,82H,81H,81H,81H,81H,81H,81H

        DB      81H,82H,82H,82H,82H,82H,82H,82H

        DB      82H,82H,82H,82H,82H,82H,82H,82H

        DB      82H,82H,82H,82H,82H,82H,82H,82H

        DB      82H,82H,82H,81H,81H,81H,81H,81H

        DB      81H,82H,82H,82H,82H,82H,82H,82H

        DB      82H,82H,82H,82H,82H,82H,82H,82H

        DB      82H,82H,82H,82H,82H,82H,82H,82H

        DB      82H,82H,82H,81H,81H,81H,81H

Z_ALPHA2        LABEL BYTE      ; ZENKAKU_ALPHA 2ND BYTE
        DB      40H,49H,8DH,94H,90H,93H,95H,4CH

        DB      69H,6AH,96H,7BH,43H,7CH,44H,5EH

        DB      4FH,50H,51H,52H,53H,54H,55H,56H

        DB      57H,58H,46H,47H,83H,81H,84H,48H

        DB      97H,60H,61H,62H,63H,64H,65H,66H

        DB      67H,68H,69H,6AH,6BH,6CH,6DH,6EH

        DB      6FH,70H,71H,72H,73H,74H,75H,76H

        DB      77H,78H,79H,6DH,8FH,6EH,4FH,51H

        DB      4DH,81H,82H,83H,84H,85H,86H,87H

        DB      88H,89H,8AH,8BH,8CH,8DH,8EH,8FH

        DB      90H,91H,92H,93H,94H,95H,96H,97H

        DB      98H,99H,9AH,6FH,62H,70H,50H
```

```
0EA9
0EA9  F6 06 0338 R 02
0EAE  75 46
0EB0  53
0EB1  52
0EB2  F6 06 0336 R 04
0EB7  74 05
0EB9  BB 0EF7 R
0EBC  EB 1B
0EBE
0EBE  F6 06 0336 R 02
0EC3  74 05
0EC5  BB 0F00 R
0EC8  EB 0F
0ECA
0ECA  F6 06 0017 R 40
0ECF  74 05
0ED1  BB 0F09 R
0ED4  EB 03
0ED6
0ED6  BB 0F12 R
0ED9
0ED9  BA 0A00
0EDC  E8 0F25 R
0EDF  F6 06 0336 R 01
0EE4  74 05
0EE6  BB 0F20 R
0EE9  EB 03
0EEB
0EEB  BB 0F1B R
0EEE
0EEE  BA 0A08
0EF1  E8 0F25 R
0EF4  5A
0EF5  5B
0EF6
0EF6  C3
0EF7

0EF7
0EF7  81 79 82 A9 82 C8
      81 45 24
0F00
0F00  81 79 83 4A 83 69
      81 45 24
0F09
0F09  81 79 43 61 70 73
      81 45 24
0F12
0F12  81 79 89 70 90 94
      81 45 24
0F1B
0F1B  94 BC 81 7A 24
0F20
0F20  91 53 81 7A 24
```

```
;-------------------------------------------------------------
;       DISPLAY SHIFT STATUS ROUTINE
;
;       *** NOTE ***
;          THIS ROUTINE DISPLAY CURRENT SHIFT STATUS.
;-------------------------------------------------------------
;
IND     PROC    NEAR
        TEST    JKB_FLAG_2,INDICATOR_OFF ; INDICATOR SWITCH OFF ?
        JNZ     IND7                     ;
        PUSH    BX                       ;
        PUSH    DX                       ;
        TEST    JKB_FLAG,HIRAGANA_STATE  ; TEST HIRAGANA SHIFT
        JZ      IND1                     ; NOT HIRAGANA SHIFT
        MOV     BX,OFFSET HMESS          ; SET HIRAGANA INDICATOR
        JMP     SHORT IND4               ; DISPLAY INDICATOR
IND1:
        TEST    JKB_FLAG,KATAKANA_STATE  ; TEST KATAKANA SHIFT
        JZ      IND2                     ; NOT KATAKANA SHIFT
        MOV     BX,OFFSET KMESS          ; SET KATAKANA INDICATOR
        JMP     SHORT IND4               ; DISPLAY INDICATOR
IND2:
        TEST    KB_FLAG,CAPS_STATE       ; TEST CAPS SHIFT
        JZ      IND3                     ; NOT CAPS SHIFT
        MOV     BX,OFFSET CMESS          ; SET CAPS INDICATOR
        JMP     SHORT IND4               ; DISPLAY INDICATOR
IND3:
        MOV     BX,OFFSET EMESS          ; SET ALPHA/NUMERIC INDICATOR
IND4:
        MOV     DX,0A00H                 ; SET LOW,COLOUM
        CALL    WRITE                    ; DISPLAY
        TEST    JKB_FLAG,ZENKAKU_STATE   ; TEST ZENKAKU SHIFT
        JZ      IND5                     ; NOT ZENKAKU SHIFT
        MOV     BX,OFFSET MESS2          ; SET ZENKAKU INDICATOR
        JMP     SHORT IND6               ; DISPLAY INDICATOR
IND5:
        MOV     BX,OFFSET MESS1          ; SET HANKAKU INDICATOR
IND6:
        MOV     DX,0A08H                 ; SET LOW,COLOUM
        CALL    WRITE                    ; DISPLAY
        POP     DX                       ;
        POP     BX                       ;
IND7:
        RET
IND     ENDP
;
HMESS   LABEL WORD
        DB      081H,079H,082H,0A9H,082H,0C8H,081H,045H,'$'

KMESS   LABEL WORD
        DB      081H,079H,083H,04AH,083H,069H,081H,045H,'$'

CMESS   LABEL WORD
        DB      081H,079H,043H,061H,070H,073H,081H,045H,'$'

EMESS   LABEL BYTE
        DB      081H,079H,089H,070H,090H,094H,081H,045H,'$'

MESS1   LABEL BYTE
        DB      094H,0BCH,081H,07AH,'$'
MESS2   LABEL BYTE
        DB      091H,053H,081H,07AH,'$'
PAGE
;-------------------------------------------------------------
;       MESSAGE WRITE ROUTINE
;
;         INPUT   BX      OFFSET OF MESSAGE HEAD
;                 DH      ROW
;                 DL      COLUM
;
;         OUTPUT          .NONE.
;
;       *** NOTE ***
;       MESSAGE FORMAT
;
;               |<------------  N+1 BYTE  ------------>|
;               |                                      |
;               (C1),(C2),(C3),-----------------,(CN),'$'
;
;-------------------------------------------------------------
```

```
0F25
0F25  50
0F26  55
0F27  8B EA
0F29  B4 83
0F2B  CD 10
0F2D  51
0F2E  52
0F2F  B9 2000
0F32  B4 81
0F34  CD 10
0F36  8B D5
0F38  B4 82
0F3A  CD 10
0F3C
0F3C  2E: 80 3F 24
0F40  74 0E
0F42  2E: 8A 07
0F45  53
0F46  B3 0F
0F48  B4 8E
0F4A  CD 10
0F4C  5B
0F4D  43
```

```
WRITE   PROC    NEAR
        PUSH    AX
        PUSH    BP
        MOV     BP,DX                    ;
        MOV     AH,83H                   ; READ ALTERNATE CURSOR POSITION
        INT     10H                      ; AND CURSOR TYPE
        PUSH    CX                       ;
        PUSH    DX                       ;
        MOV     CX,2000H                 ; M
        MOV     AH,81H                   ; M
        INT     10H                      ; ERASE ALTERNATE CURSOR
        MOV     DX,BP                    ;
        MOV     AH,82H                   ; M
        INT     10H                      ; SET ALTERNATE CURSOR POSITION
WR1:
        CMP     BYTE PTR CS:[BX],'$'     ; IS END OF MESSAGRE
        JE      WR2                      ; YES,
        MOV     AL,CS:[BX]               ; LOAD CHARACTER
        PUSH    BX
        MOV     BL,0FH                   ; SET COLOR (IN GRAPHIC MODE)
        MOV     AH,8EH
        INT     10H                      ; BIOS CALL 10H
        POP     BX
        INC     BX                       ; INCREMENT POINTER
```

```
0F4E  EB EC                     
0F50                            
0F50  5A                        
0F51  B4 82                     
0F53  CD 10                     
0F55  59                        
0F56  B4 81                     
0F58  CD 10                     
0F5A  5D                        
0F5B  58                        
0F5C  C3                        
0F5D                            
                                
                                
                                
                                
                                
0F5D                            
                                
0F5D                            
0F5D  1E                        
0F5E  53                        
0F5F  51                        
0F60  56                        
0F61  E8 0000 E                 
0F64  8B 1E 001C R              
0F68  8B F3                     
0F6A  E8 04B9 R                 
0F6D  3B 1E 001A R              
0F71  75 33                     
0F73  8B 1E 0339 R              
0F77  8B 37                     
0F79  F7 C6 FF00                
0F7D  74 20                     
0F7F  81 E6 00FF                
0F83  81 FE 0081                
0F87  72 16                     
0F89  81 FE 009F                
0F8D  76 0C                     
0F8F  81 FE 00E0                
0F93  72 0A                     
0F95  81 FE 00EF                
0F99  77 04                     
0F9B                            
0F9B  89 1E 001C R              
0F9F                            
0F9F  53                        
0FA0  E8 0AF0 R                 
0FA3  5B                        
0FA4  EB 1C                     
0FA6                            
0FA6  F6 06 0018 R 04           
0FAB  74 0B                     
0FAD  53                        
0FAE  BB 0001                   
0FB1  B9 0010                   
0FB4  E8 0000 E                 
                                
0FB7  5B                        
0FB8                            
0FB8  89 36 0339 R              
0FBC  89 04                     
0FBE  89 1E 001C R              
0FC2                            
0FC2  5E                        
0FC3  59                        
0FC4  5B                        
0FC5  1F                        
0FC6  CF                        
0FC7                            
                                
                                
0FC7                            
1100                            
1100                            
```

```
                JMP     SHORT WR1               ;
WR2:                                            
                POP     DX                      ;
                MOV     AH,82H                  ; RECOVER ALTERNATE CURSOR POSITION
                INT     10H                     ; M
                POP     CX                      ;
                MOV     AH,81H                  ; RECOVER ALTERNATE CURSOR TYPE
                INT     10H                     ; M
                POP     BP                      
                POP     AX                      
                RET                             
WRITE           ENDP                            
                PAGE                            
;-------------------------------------------------------------------------
; INT 79H  (SAVE DATA INTO BUFFER)
;       INPUT:  AX
;       OUTPUT: NONE
;-------------------------------------------------------------------------
;
BUFFER_QUEING   PROC    FAR
                ASSUME  CS:CODE,DS:DATA
KQ1:                                            
                PUSH    DS                      ;
                PUSH    BX                      ;
                PUSH    CX                      ;
                PUSH    SI                      ;
                CALL    DDS                     ; POINT DS AT BIOS DATA SEGMENT
                MOV     BX,BUFFER_TAIL          ; GET THE END POINTER TO THE BUFFER
                MOV     SI,BX                   ; SAVE THE VALUE
                CALL    K4                      ; ADVANCE THE TAIL
                CMP     BX,BUFFER_HEAD          ; HAS THE BUFFER WRAPPED AROUND?
                JNE     KQ2                     ; BUFFER_FILL_BEEP
                MOV     BX,FIRST_PTR            ;
                MOV     SI,[BX]                 ; --- SI=CONTENT OF PREVIOUS DATA
                TEST    SI,0FF00H               
                JE      ONE_BYT                 
                AND     SI,00FFH                
                CMP     SI,81H                  ; M
                JB      ONE_BYT                 ; M
                CMP     SI,9FH                  ; M
                JBE     TWO_BYT                 ; M
                CMP     SI,0E0H                 ; M
                JB      ONE_BYT                 ; M
                CMP     SI,0EFH                 ; M
                JA      ONE_BYT  ; --- RECOGNIZE PREVIOUS DATA IS FIRST BYTE OF KANJI
TWO_BYT:                                        
                MOV     BUFFER_TAIL,BX          ; BACK THE POINTER
ONE_BYT:                                        
                PUSH    BX                      ; SAVE BUFFER_TAIL
                CALL    ERROR_BEEP              ; CALL ERROR BEEP ROUTINE
                POP     BX                      ; RETRIEVE BUFFER TAIL
                JMP     SHORT KQ4               ; RETURN FROM INTERRUPT
KQ2:                                            
                TEST    KB_FLAG_1,CLICK_ON ; IS AUDIO FEEDBACK ENABLED?
                JZ      KQ3                     ; NO, JUST PUT IN BUFFER
                PUSH    BX                      ; SAVE BUFFER_TAIL VALUE
                MOV     BX,1H                   ; DURATION OF CLICK
                MOV     CX,10H                  ; FREQUENCY OF CLICK
                CALL    KB_NOISE                ; OUTPUT AUDIO FEEDBACK OF KEY
                                                ; STROKE
                POP     BX                      ; RETRIEVE BUFFER_TAIL VALUE
KQ3:                                            
                MOV     FIRST_PTR,SI    ; MEMORIZE POINTER
                MOV     [SI],AX         ; STORE THE VALUE
                MOV     BUFFER_TAIL,BX  ; MOVE THE POINTER UP
KQ4:                                            
                POP     SI                      ;
                POP     CX                      ;
                POP     BX                      ;
                POP     DS                      ;
                IRET                            ; RETURN FROM INTERRUPT
BUFFER_QUEING   ENDP                            
PAGE                                            
;;;
```

This page intentionally left blank.

```
;**************
;**************
;*          *
;* MODULE 5 *
;*          *
;**************
;**************
```

```
;-- INT 5 ---------------------------------------------------------------;
;       THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE        ;
;       SCREEN.                                                         ;
;                                                                       ;
;       50:0    THE STATUS OF THE PRINT SCREEN.                         ;
;                                                                       ;
;               = 0 :  PRINT SCREEN ISN'T CALLED.                       ;
;                                                                       ;
;               = 1 :  PRINT SCREEN IS IN PROGRESS                      ;
;                                                                       ;
;               =-1 :  ERROR ENCOUNTED DURING PRINTING                  ;
;-----------------------------------------------------------------------;

;-----------------------------------------------------------------------;
;       VIDEO MODE TABLE                                                ;
;-----------------------------------------------------------------------;
;       BYTE 1   7 , 6  : 01 CHARACTER / 10 GRAPHIC                     ;
;                5 - 0  : COLOR TABLE OFFSET                            ;
;                                                                       ;
;       BYTE 2   7 - 0  : VERTICAL SCREEN SIZE(SIZE/8 WHEN GRAPHIC)     ;
;                                                                       ;
;       BYTE 3   7 - 0  : HORIZONTAL SCREEN SIZE                        ;
;-----------------------------------------------------------------------;
```

```
0000                    TABLE   PROC    NEAR
0000  40 19 28          MODETBL DB      040H,019H,028H          ; C (40,25)     ;NATIVE
0003  40 19 28                  DB      040H,019H,028H          ; C (40,25)
0006  40 19 50                  DB      040H,019H,050H          ; C (80,25)
0009  40 19 50                  DB      040H,019H,050H          ; C (80,25)
000C  84 19 28                  DB      084H,019H,028H          ; G (320,200)
000F  84 19 28                  DB      084H,019H,028H          ; G (320,200)
0012  80 19 50                  DB      080H,019H,050H          ; G (640,200)
0015  00 00 00                  DB      000H,000H,000H          ; NOT VALID
0018  8C 19 14                  DB      08CH,019H,014H          ; G (160,200)
001B  8C 19 28                  DB      08CH,019H,028H          ; G (320,200)
001E  84 19 50                  DB      084H,019H,050H          ; G (640,200)
0021  00 00 00                  DB      000H,000H,000H          ; NOT VALID
0024  00 00 00                  DB      000H,000H,000H          ; NOT VALID
0027  00 00 00                  DB      000H,000H,000H          ; NOT VALID
002A  00 00 00                  DB      000H,000H,000H          ; NOT VALID
002D  00 00 00                  DB      000H,000H,000H          ; NOT VALID
0030  40 0B 28                  DB      040H,00BH,028H          ; C (40,11)
0033  40 0B 28                  DB      040H,00BH,028H          ; C (40,11)
0036  40 0B 50                  DB      040H,00BH,050H          ; C (80,11)
0039  40 0B 50                  DB      040H,00BH,050H          ; C (80,11)
003C  84 19 28                  DB      084H,019H,028H          ; G (320,200)
003F  84 19 28                  DB      084H,019H,028H          ; G (320,200)
0042  80 19 50                  DB      080H,019H,050H          ; G (640,200)
0045  00 00 00                  DB      000H,000H,000H          ; NOT VALID
0048  8C 19 14                  DB      08CH,019H,014H          ; G (160,200)
004B  8C 19 28                  DB      08CH,019H,028H          ; G (320,200)
004E  84 19 50                  DB      084H,019H,050H          ; G (640,200)
0051  AC 19 50                  DB      0ACH,019H,050H          ; G (640,200)
0054  00                DOTTBL  DB      000H                    ; 2 COLOR
0055  00                        DB      000H
0056  F8                        DB      0F8H
0057  F8                        DB      0F8H
0058  00                        DB      000H                    ; 4 COLOR
0059  00                        DB      000H
005A  80                        DB      080H
005B  80                        DB      080H
005C  70                        DB      070H
005D  70                        DB      070H
005E  F8                        DB      0F8H
005F  F8                        DB      0F8H
0060  00                        DB      000H                    ; 16 COLOR
0061  00                        DB      000H
0062  80                        DB      080H
0063  00                        DB      000H
0064  60                        DB      060H
0065  00                        DB      000H
0066  80                        DB      080H
0067  10                        DB      010H
0068  98                        DB      098H
0069  00                        DB      000H
006A  60                        DB      060H
006B  10                        DB      010H
006C  78                        DB      078H
006D  00                        DB      000H
006E  98                        DB      098H
006F  10                        DB      010H
0070  60                        DB      060H
0071  18                        DB      018H
0072  98                        DB      098H
0073  60                        DB      060H
0074  78                        DB      078H
0075  80                        DB      080H
0076  98                        DB      098H
0077  E0                        DB      0E0H
0078  78                        DB      078H
0079  C0                        DB      0C0H
007A  78                        DB      078H
007B  E0                        DB      0E0H
007C  78                        DB      078H
007D  F0                        DB      0F0H
007E  F8                        DB      0F8H
007F  F8                        DB      0F8H
0080  00                        DB      000H                    ; SUPER 16 COLOR
0081  00                        DB      000H
0082  80                        DB      080H
0083  80                        DB      080H
0084  20                        DB      020H
0085  20                        DB      020H
0086  C0                        DB      0C0H
```

```
0087  C0                        DB      0C0H
0088  28                        DB      028H
0089  28                        DB      028H
008A  E0                        DB      0E0H
008B  E0                        DB      0E0H
008C  58                        DB      058H
008D  58                        DB      058H
008E  F8                        DB      0F8H
008F  F8                        DB      0F8H
0090  00                        DB      000H
0091  00                        DB      000H
0092  80                        DB      080H
0093  80                        DB      080H
0094  20                        DB      020H
0095  20                        DB      020H
0096  C0                        DB      0C0H
0097  C0                        DB      0C0H
0098  28                        DB      028H
0099  28                        DB      028H
009A  E0                        DB      0E0H
009B  E0                        DB      0E0H
009C  58                        DB      058H
009D  58                        DB      058H
009E  F8                        DB      0F8H
009F  F8                        DB      0F8H
00A0                    TABLE   ENDP
                        ;-------------------------------------------------------------------------;
                        ;       MAIN ROUTINE                                                      ;
                        ;-------------------------------------------------------------------------;
00A0                    PRINT_SCREEN    PROC    FAR
00A0  1E                        PUSH    DS                      ;SAVE REGS.
00A1  06                        PUSH    ES
00A2  50                        PUSH    AX
00A3  53                        PUSH    BX
00A4  51                        PUSH    CX
00A5  52                        PUSH    DX
00A6  56                        PUSH    SI
00A7  57                        PUSH    DI
00A8  B8 ---- R                 MOV     AX,DSEGM                ;SET DS
00AB  8E D8                     MOV     DS,AX
00AD  B8 ---- R                 MOV     AX,XXDATA               ;SET ES
00B0  8E C0                     MOV     ES,AX
00B2  80 3E 001E R 01           CMP     STATUS17,1              ;IS INT17 IN PROGRESS ?
00B7  74 64                     JE      RETURN
00B9  26: 80 3E 0000 R 01       CMP     STATUS_BYTE,1           ;IS INT5  IN PROGRESS ?
00BF  74 5C                     JE      RETURN
00C1  26: C6 06 0000 R 01       MOV     STATUS_BYTE,1
00C7  E8 0230 R                 CALL    INIT                    ;INITIAL PROCESS
00CA  89 26 0069 R              MOV     SPSAVE5,SP
00CE  E8 033F R                 CALL    CRLF
00D1  8A 26 0003 R              MOV     AH,CPI                  ;SAVE CPI,LPI
00D5  A0 0004 R                 MOV     AL,LPI
00D8  50                        PUSH    AX
00D9  B4 0F                     MOV     AH,15                   ;GET DISPLAY STATUS AND PAGE
00DB  CD 10                     INT     10H
00DD  88 3E 0068 R              MOV     PAGENO,BH
00E1  32 E4                     XOR     AH,AH                   ;MODE CHECK
00E3  8B F8                     MOV     DI,AX
00E5  03 F8                     ADD     DI,AX
00E7  03 F8                     ADD     DI,AX
00E9  2E: 8A 85 0000 R          MOV     AL,MODETBL[DI]          ;SET VIDEO MODE
00EE  A2 006D R                 MOV     VIDEO_MODE,AL
00F1  80 3E 006D R 00           CMP     VIDEO_MODE,0
00F6  74 13                     JE      ABEND
00F8  7F 38                     JG      CHAR                    ;CHARACTER
00FA  E9 01A8 R                 JMP     GRAPH                   ;GRAPHIC
00FD  BE ---- R         EXIT:   MOV     SI,XXDATA               ;SET NORMAL STATUS
0100  8E C6                     MOV     ES,SI
0102  26: FE 06 0000 R          INC     STATUS_BYTE
0107  5B                        POP     BX                      ;RESTORE CPI,LPI
0108  E8 0308 R                 CALL    CPILPI
010B  8B 26 0069 R      ABEND:  MOV     SP,SPSAVE5              ;RESTORE SP
010F  BE ---- R                 MOV     SI,XXDATA               ;SET ERROR STATUS
0112  8E C6                     MOV     ES,SI
0114  26: 80 2E 0000 R 02       SUB     STATUS_BYTE,2
011A  E8 0242 R                 CALL    POST                    ;POST ROUTINE
011D  5F                RETURN: POP     DI
011E  5E                        POP     SI
011F  5A                        POP     DX
0120  59                        POP     CX
0121  5B                        POP     BX
0122  58                        POP     AX
0123  07                        POP     ES
0124  1F                        POP     DS
0125  CF                        IRET
0126                    PRINT_SCREEN    ENDP
                        ;-------------------------------------------------------------------------;
                        ;       INT 17 CALL ROUTINE                                               ;
                        ;-------------------------------------------------------------------------;
0126                    INT17   PROC    NEAR
0126  52                        PUSH    DX
0127  33 D2                     XOR     DX,DX
0129  CD 17                     INT     17H                     ;INT 17
012B  F6 C4 08                  TEST    AH,08H                  ;ERROR ?
012E  75 DB                     JNZ     ABEND
0130  5A                        POP     DX
0131  C3                        RET
0132                    INT17   ENDP
                        ;-------------------------------------------------------------------------;
                        ;       CHARACTER HARD COPY                                               ;
                        ;-------------------------------------------------------------------------;
0132                    CHAR    PROC    NEAR
0132  BB 7814                   MOV     BX,7814H                ;6 CPI , 6 LPI SET
```

```
0135  E8 0308 R
0138  2E: 8A AD 0001 R
013D  2E: 8A 8D 0002 R
0142  32 F6
0144  32 D2
0146  33 F6
0148  B4 02
014A  8A 3E 0068 R
014E  CD 10
0150  B4 08
0152  8A 3E 0068 R
0156  CD 10
0158  3C 1C
015A  77 22
015C  3C 19
015E  77 1C
0160  3C 18
0162  77 1A
0164  3C 17
0166  77 14
0168  3C 14
016A  77 12
016C  3C 10
016E  77 0C
0170  3C 0F
0172  77 0A
0174  3C 06
0176  77 04
0178  3C 00
017A  77 02
017C  B0 20
017E  88 84 03D4 R
0182  C6 84 0474 R 00
0187  46
0188  FE C2
018A  3A D1
018C  75 BA
018E  51
018F  B4 0B
0191  8D 1E 03D4 R
0195  8D 3E 0474 R
0199  8B CE
019B  E8 0126 R
019E  59
019F  FE C6
01A1  3A F5
01A3  75 9F
01A5  E9 00FD R
01A8

01A8
01A8  BB 7810
01AB  E8 0308 R
01AE  B8 0C05
01B1  B7 00
01B3  E8 0126 R
01B6  B8 001B
01B9  E8 0126 R
01BC  B8 0028
01BF  E8 0126 R
01C2  E8 025A R
01C5  33 C9
01C7  E8 0317 R
01CA  8B 16 0062 R
01CE  4A
01CF  8B 36 006E R
01D3  33 FF
01D5  51
01D6  E8 02BE R
01D9  41
01DA  2B 36 0064 R
01DE  75 F6
01E0  8B 0E 0066 R
01E4  51
01E5  33 FF
01E7  8B 0E 0000 R
01EB  41
01EC  51
01ED  B9 0008
01F0  8A 85 0080 R
01F4  D1 E0
01F6  88 85 0080 R
01FA  47
01FB  E2 F3
01FD  59
01FE  8A C4
0200  32 E4
0202  E8 0126 R
0205  E2 E5
0207  59
0208  E2 DA
020A  59
020B  4A
020C  83 FA 00
020F  7D BE
0211  E8 033F R
0214  A1 006E R
0217  33 D2
0219  F7 36 0064 R
021D  03 C8
021F  3B 0E 0060 R
```

```
        CALL    CPILPI
        MOV     CH,MODETBL+1[DI]        ;SET DISPLAY SIZE TO CX
        MOV     CL,MODETBL+2[DI]
        XOR     DH,DH                   ;0 -> V.POS
CHAR00: XOR     DL,DL                   ;0 -> H.POS
        XOR     SI,SI                   ;0 -> BUF.POS
CHAR10: MOV     AH,2                    ;MOVE CURSOR
        MOV     BH,PAGENO
        INT     10H
        MOV     AH,8                    ;READ CHARACTER & ATTRIBUTE
        MOV     BH,PAGENO
        INT     10H
        CMP     AL,1CH                  ;FORMAT CHARACTER ?
        JA      FORMAT
        CMP     AL,19H
        JA      CNTROL
        CMP     AL,18H
        JA      FORMAT
        CMP     AL,17H
        JA      CNTROL
        CMP     AL,14H
        JA      FORMAT
        CMP     AL,10H
        JA      CNTROL
        CMP     AL,0FH
        JA      FORMAT
        CMP     AL,06H
        JA      CNTROL
        CMP     AL,00H
        JA      FORMAT
CNTROL: MOV     AL,' '
FORMAT: MOV     CODE_INT5[SI],AL        ;SET CHARACTER
        MOV     ATTR_INT5[SI],0         ;SET ATTRIBUTE
        INC     SI                      ;UP BUFFER POSITION
        INC     DL                      ;UP HORIZONTAL POSITION
        CMP     DL,CL                   ;LIMIT ?
        JNE     CHAR10
        PUSH    CX                      ;PRINT CHARACTER
        MOV     AH,11
        LEA     BX,CODE_INT5
        LEA     DI,ATTR_INT5
        MOV     CX,SI
        CALL    INT17
        POP     CX
        INC     DH                      ;UP VERTICAL POSITION
        CMP     DH,CH
        JNE     CHAR00
        JMP     EXIT
CHAR    ENDP
;-------------------------------------------------------------------------;
;       GRAPHIC HARD COPY                                                 ;
;-------------------------------------------------------------------------;
GRAPH   PROC    NEAR
        MOV     BX,7810H                ;6 CPI , 7.5 LPI SET
        CALL    CPILPI
        MOV     AX,0C05H                ;SET UNIDIRECTION
        MOV     BH,0
        CALL    INT17
        MOV     AX,001BH                ;SET 3 BYTE TRANSMISSION
        CALL    INT17
        MOV     AX,0028H
        CALL    INT17
        CALL    SIZESET
        XOR     CX,CX                   ;0     -> H.POS
GR00:   CALL    ESCX1                   ;ESC + X + 1 + N1 + N2
        MOV     DX,VSIZE                ;VSIZE -> V.POS
        DEC     DX
GR10:   MOV     SI,SLICE                ;SET COUNT
        XOR     DI,DI
        PUSH    CX
GR20:   CALL    DOTSET                  ;DOT PATTERN SET
        INC     CX                      ;MOVE POSITION RIGHT
        SUB     SI,HRATIO
        JNZ     GR20
        MOV     CX,VRATIO               ;PRINT DOT PATTERN
GR40:   PUSH    CX
        XOR     DI,DI
        MOV     CX,PRINTER_ID           ;SLICE / 8 -> CX
        INC     CX
GR50:   PUSH    CX                      ;SET DOT PATTERN TO AH
        MOV     CX,8
GR60:   MOV     AL,DOT[DI]
        SHL     AX,1
        MOV     DOT[DI],AL
        INC     DI
        LOOP    GR60
        POP     CX
        MOV     AL,AH                   ;PRINTING
        XOR     AH,AH
        CALL    INT17
        LOOP    GR50
        POP     CX
        LOOP    GR40
        POP     CX
        DEC     DX                      ;UP VERTICAL POSITION
        CMP     DX,0
        JNL     GR10                    ;LIMIT ?
        CALL    CRLF                    ;LINE FEED
        MOV     AX,SLICE                ;SLICE / HRATIO -> AX
        XOR     DX,DX
        DIV     HRATIO
        ADD     CX,AX                   ;UP HORIZONTAL POSITION
        CMP     CX,HSIZE
```

```
0223  7C A2                        JL      GR00
0225  B8 0C05                      MOV     AX,0C05H                ;SET BIDIRECTION
0228  B7 01                        MOV     BH,1
022A  E8 0126 R                    CALL    INT17
022D  E9 00FD R                    JMP     EXIT
0230                         GRAPH ENDP
                             ;-------------------------------------------------------------------------;
                             ;       INITIAL PROCESS                                                   ;
                             ;-------------------------------------------------------------------------;
0230                         INIT  PROC    NEAR
0230  5E                          POP     SI                      ;SAVE RETURN ADDRESS
0231  1E                          PUSH    DS                      ;RESET ES
0232  07                          POP     ES
0233  A0 0026 R                   MOV     AL,LF_CT                ;SAVE LINE FEED COUNT
0236  50                          PUSH    AX
0237  B4 03                       MOV     AH,3                    ;SAVE CURSOR POSITION
0239  8A 3E 0068 R                MOV     BH,PAGENO
023D  CD 10                       INT     10H
023F  52                          PUSH    DX
0240  56                          PUSH    SI                      ;RESTORE RETURN ADDRESS
0241  C3                          RET
0242                         INIT  ENDP
                             ;-------------------------------------------------------------------------;
                             ;       POST PROCESS                                                      ;
                             ;-------------------------------------------------------------------------;
0242                         POST  PROC    NEAR
0242  5E                          POP     SI                      ;SAVE RETURN ADDRESS
0243  5A                          POP     DX                      ;RESTORE CURSOR POSITION
0244  80 3E 006D R 00             CMP     VIDEO_MODE,0
0249  7C 08                       JL      POST10
024B  B4 02                       MOV     AH,2
024D  8A 3E 0068 R                MOV     BH,PAGENO
0251  CD 10                       INT     10H
0253  5B                    POST10: POP    BX                      ;RESTORE LINE FEED COUNT
0254  88 1E 0026 R                MOV     LF_CT,BL
0258  56                          PUSH    SI
0259  C3                          RET
                             ;
025A                         POST  ENDP
                             ;-------------------------------------------------------------------------;
                             ;       DOT SIZE SET                                                      ;
                             ;-------------------------------------------------------------------------;
025A                         SIZESET PROC  NEAR
025A  32 E4                       XOR     AH,AH                   ;SET VERTICAL DOT SIZE
025C  2E: 8A 85 0001 R            MOV     AL,MODETBL+1[DI]
0261  B1 03                       MOV     CL,3
0263  D3 E0                       SHL     AX,CL
0265  A3 0062 R                   MOV     VSIZE,AX
0268  32 E4                       XOR     AH,AH                   ;SET HORIZONTAL DOT SIZE
026A  2E: 8A 85 0002 R            MOV     AL,MODETBL+2[DI]
026F  D3 E0                       SHL     AX,CL
0271  A3 0060 R                   MOV     HSIZE,AX
0274  C7 06 0066 R 0002           MOV     VRATIO,2                ;SET ENLARGE RATIO
027A  81 3E 0062 R 00C8           CMP     VSIZE,200               ;+----------+------+------+
0280  7F 06                       JG      SIZE00                  ;| V. SIZE  | PT-1 | PT-2 |
0282  C7 06 0066 R 0005           MOV     VRATIO,5                ;|----------+------+------|
0288  A1 0000 R              SIZE00: MOV  AX,PRINTER_ID           ;| 200      | X 5  | x 5  |
028B  81 3E 0060 R 0168           CMP     HSIZE,360               ;+----------+------+------+
0291  7F 0C                       JG      SIZE10                  ;| H. SIZE  | PT-1 | PT-2 |
0293  D0 E0                       SHL     AL,1                    ;|----------+------+------|
0295  81 3E 0060 R 00B4           CMP     HSIZE,180               ;| 361 -    | X 1  | x 2  |
029B  7F 02                       JG      SIZE10                  ;| 181 - 360| X 2  | x 4  |
029D  D0 E0                       SHL     AL,1                    ;|   0 - 180| X 4  | x 8  |
029F  A3 0064 R              SIZE10: MOV  HRATIO,AX               ;+----------+------+------+
02A2  2E: 8A 85 0000 R            MOV     AL,MODETBL+0[DI]        ;SET COLOR TABLE OFFSET
02A7  A3 006B R                   MOV     COLORTB,AX
02AA  81 26 006B R 003F           AND     COLORTB,003FH
02B0  8B 0E 0000 R                MOV     CX,PRINTER_ID           ;SET SLICE SIZE (16 OR 24)
02B4  B0 08                       MOV     AL,8
02B6  FE C1                       INC     CL
02B8  F6 E1                       MUL     CL
02BA  A3 006E R                   MOV     SLICE,AX
02BD  C3                          RET
02BE                         SIZESET ENDP
                             ;-------------------------------------------------------------------------;
                             ;       DOT STORE                                                         ;
                             ;-------------------------------------------------------------------------;
02BE                         DOTSET  PROC  NEAR
02BE  51                          PUSH    CX
02BF  52                          PUSH    DX
02C0  B4 0D                       MOV     AH,13                   ;READ DOT
02C2  CD 10                       INT     10H
02C4  3B 0E 0060 R                CMP     CX,HSIZE
02C8  7C 02                       JL      DOT00
02CA  B0 00                       MOV     AL,0
02CC  25 000F                DOT00: AND   AX,000FH                ;SET VERTICAL MASK
02CF  8B D8                       MOV     BX,AX
02D1  D1 E3                       SHL     BX,1
02D3  03 1E 006B R                ADD     BX,COLORTB
02D7  2E: 8A B7 0054 R            MOV     DH,DOTTBL+0[BX]
02DC  2E: 8A 97 0055 R            MOV     DL,DOTTBL+1[BX]
02E1  8B 1E 0064 R                MOV     BX,HRATIO               ;SET HORIZONTAL RATIO
02E5  D1 EB                       SHR     BX,1
02E7  8B CB                       MOV     CX,BX                   ;DOT PATTERN SET
02E9  0B DB                       OR      BX,BX                   ;640 MODE ?
02EB  75 04                       JNZ     DOT20
02ED  8B 0E 0000 R                MOV     CX,PRINTER_ID
02F1  88 B5 0080 R           DOT20: MOV   DOT[DI],DH
02F5  47                          INC     DI
02F6  E2 F9                       LOOP    DOT20
02F8  8B CB                       MOV     CX,BX
02FA  0B DB                       OR      BX,BX                   ;640 MODE ?
02FC  74 07                       JZ      DOT99
```

```
02FE  88 95 0080 R          DOT30:  MOV     DOT[DI],DL
0302  47                            INC     DI
0303  E2 F9                        LOOP    DOT30
0305  5A                    DOT99:  POP     DX
0306  59                            POP     CX
0307  C3                            RET
0308                        DOTSET  ENDP
                            ;-------------------------------------------------------------------------;
                            ;       CPI LPI SET ROUTINE (BH : CPI , BL : LPI)                         ;
                            ;-------------------------------------------------------------------------;
0308                        CPILPI  PROC    NEAR
0308  B8 0C01                        MOV     AX,0C01H                ;CPI SET
030B  E8 0126 R                     CALL    INT17
030E  8A FB                         MOV     BH,BL                   ;LPI SET
0310  B8 0C02                       MOV     AX,0C02H
0313  E8 0126 R                     CALL    INT17
0316  C3                            RET
0317                        CPILPI  ENDP
                            ;-------------------------------------------------------------------------;
                            ;       ESC + X + 1 + N1 + N2                                             ;
                            ;-------------------------------------------------------------------------;
0317                        ESCX1   PROC    NEAR
0317  B8 001B                       MOV     AX,001BH                ;ESC
031A  E8 0126 R                     CALL    INT17
031D  B8 0025                       MOV     AX,0025H                ;X
0320  E8 0126 R                     CALL    INT17
0323  B8 0031                       MOV     AX,0031H                ;1
0326  E8 0126 R                     CALL    INT17
0329  A1 0062 R                     MOV     AX,VSIZE                ;N1N2 = VSIZE X VRATIO
032C  F7 26 0066 R                  MUL     VRATIO
0330  50                            PUSH    AX
0331  8A C4                        MOV     AL,AH
0333  32 E4                        XOR     AH,AH
0335  E8 0126 R                     CALL    INT17
0338  58                            POP     AX
0339  32 E4                        XOR     AH,AH
033B  E8 0126 R                     CALL    INT17
033E  C3                            RET
033F                        ESCX1   ENDP
                            ;-------------------------------------------------------------------------;
                            ;       CR. LF.                                                           ;
                            ;-------------------------------------------------------------------------;
033F                        CRLF    PROC    NEAR
033F  B8 000D                       MOV     AX,000DH                ;CR
0342  E8 0126 R                     CALL    INT17
0345  83 3E 0000 R 01               CMP     PRINTER_ID,1            ;PT-1 ?
034A  75 07                         JNE     CR10                    ;NO
034C  B8 000A                       MOV     AX,000AH                ;LF (FOR PT-1)
034F  E8 0126 R                     CALL    INT17
0352  C3                            RET
0353  B8 041B                CR10:   MOV     AX,041BH                ;VERTICAL FEED (FOR PT-2)
0356  E8 0126 R                     CALL    INT17
0359  B8 0425                       MOV     AX,0425H
035C  E8 0126 R                     CALL    INT17
035F  B8 0435                       MOV     AX,0435H
0362  E8 0126 R                     CALL    INT17
0365  B8 0400                       MOV     AX,0400H
0368  E8 0126 R                     CALL    INT17
036B  B8 0410                       MOV     AX,0410H
036E  E8 0126 R                     CALL    INT17
0371  C3                            RET
0372                        CRLF    ENDP
03C0                                ORG     BEGIN+03C0H
03C0                        CODE    ENDS
                                    END
```

```
**************
**************
*            *
* MODULE 12  *
*            *
**************
**************
```

```
;-- INT 17 ------------------------------------------------------------;
; PRINTER I/O                                                          ;
;   INPUT                                                              ;
;       AH=0 : PRINT A CODE IN AL                                      ;
;              AL : OUTPUT DATA                                        ;
;                                                                      ;
;       AH=1 : INITIALIZE PRINTER                                      ;
;                                                                      ;
;       AH=2 : READ PRINTER STATUS-1 IN AH                             ;
;                7      6      5      4      3      2      1      0    ;
;                |      |      |      |      |      |      |      |    ;
;                |      |      |      |      |      |      |  TIME OUT;
;                |      |      |      |      |      |  RESERVED       ;
;                |      |      |      |      |  RESERVED              ;
;                |      |      |      |  ANY ERROR                     ;
;                |      |      |  SELECT                               ;
;                |      |  OUT OF PAPER                                ;
;                |  RESERVED                                           ;
;            -BUSY                                                     ;
;                                                                      ;
;       AH=3 : READ PRINTER STATUS-2 IN AH                             ;
;                7      6      5      4,3    2,1    0                 ;
;                |      |      |      |      |      |                 ;
;                |      |      |      |      |  PRINTER-ID            ;
;                |      |      |      |  CPI *                        ;
;                |      |      |  LPI *                               ;
;                |      |  RESERVED                                   ;
;                |  RESERVED                                           ;
;             ASF ON                                                   ;
;                                                                      ;
;              * CPI                    * LPI                         ;
;                  00 : 10                  00 : 4                   ;
;                  01 : 12                  01 : 5                   ;
;                  10 : 40/3                10 : 6                   ;
;                  11 : 15                  11 : 7.5                 ;
;                                                                      ;
;       AH=4 : PASS THROUGH A CODE IN AL                               ;
;              AL : OUTPUT DATA                                        ;
;                                                                      ;
;       AH=5 : PRINT A CODE IN AL (DOUBLE SIZE)                        ;
;              AL : OUTPUT DATA                                        ;
;                                                                      ;
;       AH=6 : RESERVED                                                ;
;                                                                      ;
;       AH=7 : RESERVED                                                ;
;                                                                      ;
;       AH=8 : RESERVED                                                ;
;                                                                      ;
;       AH=9 : RESERVED                                                ;
;                                                                      ;
;       AH=A : RESERVED                                                ;
;                                                                      ;
;       AH=B : PRINT CODES WITH ATTRIBUTES                             ;
;              ES:BX : ADDRESS OF CHARACTER BUFFER                     ;
;              ES:DI : ADDRESS OF ATTRIBUTE BUFFER                     ;
;              CX    : LENGTH OF THE BUFFER                            ;
;                                                                      ;
;              ATTRIBUTE                                               ;
;                  7      6      5      4      3,2    1,0             ;
;                  |      |      |      |      |      |               ;
;                  |      |      |      |      |  HORIZONTAL GRID *    ;
;                  |      |      |      |  VERTICAL GRID *             ;
;                  |      |      |  RESERVED                           ;
;                  |      |  HIGH-INTENSITY                            ;
;                  |  UBDERSCORE                                       ;
;               RESERVED                                               ;
;                                                                      ;
;              * GRID LINE                                             ;
;                  00 : NO GRID LINE                                   ;
;                  01 : SINGLE SOLID  LINE                             ;
;                  10 : DOUBLE SOLID  LINE                             ;
;                  11 : SINGLE DASHED LINE                             ;
;                                                                      ;
;       AH=C : SET PRINTER CONTROL VALUE                               ;
;              AL=0 : RESET CONTROL VALUE TO DEFAULT                   ;
;              AL=1 : CHANGE CHARACTER PITCH                           ;
;                  BH=90 : 10    CHAR/INCH                           ;
;                  BH=78 : 12    CHAR/INCH                           ;
;                  BH=6C : 40/3  CHAR/INCH                           ;
;                  BH=60 : 15    CHAR/INCH                           ;
;              AL=2 : CHANGE LINE FEED PITCH                           ;
;                  BH   : PHYSICAL PICH * 120 (INCH)                   ;
;              AL=3 : CHANGE PAGE LENGTH                               ;
;                  BX   : PAGE LENGTH                                  ;
;              AL=5 : CHANGE DIRECTION                                 ;
;                  BH=0 : UNIDIRECTION                                 ;
;                  BH=1 : BIDIRECTION                                  ;
;              AL=6 : CHANGE CHARACTER TYPE                            ;
;                  BH=0 : LARGE TYPE                                   ;
;                  BH=1 : SMALL TYPE                                   ;
;                                                                      ;
;       DX MUST BE ZERO.                                               ;
;----------------------------------------------------------------------;
;                                                                      ;
;----------------------------------------------------------------------;
;----------------------------------------------------------------------;
;  CPI , LPI CONTROL VALUE TABLE                                       ;
;----------------------------------------------------------------------;
0000 TABLE   PROC    NEAR                                              ;
;----------------------------------------------------------------------;
;       VERTICAL GRID CONTROL VALUES                                   ;
;             0   : KEY                                                ;
;             1-3 : SOLID IMAGE                                        ;
```

```
;             4-6 : DASHED IMAGE                                                ;
;             7   : UNDERSCORE IMAGE                                            ;
;             8   : FEED VALUE                                                  ;
;-------------------------------------------------------------------------------;
0000  1E                V_VALUE         DB      1EH                 ; 4 LPI (UPPER)
0001  FF FF F8                          DB      0FFH,0FFH,0F8H
0004  F8 3F F0                          DB      0F8H,03FH,0F0H
0007  80                               DB      080H
0008  07                               DB      7
0009  1E                               DB      1EH                 ; 4 LPI (LOWER)
000A  FF FF FF                          DB      0FFH,0FFH,0FFH
000D  0F FC 1F                          DB      00FH,0FCH,01FH
0010  80                               DB      080H
0011  07                               DB      7
0012  18                               DB      18H                 ; 5 LPI (UPPER)
0013  FF F0 00                          DB      0FFH,0F0H,000H
0016  FC 30 00                          DB      0FCH,030H,000H
0019  10                               DB      010H
001A  04                               DB      4
001B  18                               DB      18H                 ; 5 LPI (LOWER)
001C  FF FF FF                          DB      0FFH,0FFH,0FFH
001F  3F FC 3F                          DB      03FH,0FCH,03FH
0022  10                               DB      010H
0023  04                               DB      4
0024  14                               DB      14H                 ; 6 LPI (UPPER)
0025  FC 00 00                          DB      0FCH,000H,000H
0028  F8 00 00                          DB      0F8H,000H,000H
002B  02                               DB      002H
002C  02                               DB      2
002D  14                               DB      14H                 ; 6 LPI (LOWER)
002E  FF FF FF                          DB      0FFH,0FFH,0FFH
0031  0F FC 1F                          DB      00FH,0FCH,01FH
0034  02                               DB      002H
0035  02                               DB      2
0036  10                               DB      10H                 ; 7.5 LPI (UPPER)
0037  00 00 00                          DB      000H,000H,000H
003A  00 00 00                          DB      000H,000H,000H
003D  01                               DB      001H
003E  00                               DB      0
003F  10                               DB      10H                 ; 7.5 LPI (LOWER)
0040  FF FF FF                          DB      0FFH,0FFH,0FFH
0043  F0 FF 0F                          DB      0F0H,0FFH,00FH
0046  01                               DB      001H
0047  00                               DB      0
;-------------------------------------------------------------------------------;
;       HORIZONTAL GRID CONTROL VALUES                                          ;
;             0   : KEY                                                         ;
;             1-3 : DASHED IMAGE                                                ;
;             4   : HORIZONTAL SIZE                                             ;
;             5   : SPACE BETWEEN CHARACTERS                                    ;
;-------------------------------------------------------------------------------;
0048  90                H_VALUE         DB      90H                 ; 10 CPI
0049  FC 0F C0                          DB      0FCH,00FH,0C0H
004C  12                               DB      18
004D  06                               DB      6
004E  78                               DB      78H                 ; 12 CPI
004F  F8 3E 00                          DB      0F8H,03EH,000H
0052  0F                               DB      15
0053  04                               DB      4
0054  6C                               DB      6CH                 ; 13.4 CPI
0055  F8 7C 00                          DB      0F8H,07CH,000H
0058  0D                               DB      13
0059  03                               DB      3
005A  60                               DB      60H                 ; 15 CPI
005B  F0 F0 00                          DB      0F0H,0F0H,000H
005E  0C                               DB      12
005F  01                               DB      1
0060                    TABLE   ENDP
;-------------------------------------------------------------------------------;
;  MAIN ROUTINE                                                                 ;
;-------------------------------------------------------------------------------;
0060                    PRINTER_IO      PROC    FAR
0060  FB                        STI                             ; INTERRUPTS BACK ON
0061  1E                        PUSH    DS                      ; SAVE REGISTERS
0062  06                        PUSH    ES
0063  56                        PUSH    SI
0064  57                        PUSH    DI
0065  55                        PUSH    BP
0066  52                        PUSH    DX
0067  51                        PUSH    CX
0068  53                        PUSH    BX
0069  50                        PUSH    AX
006A  BE ---- R                   MOV     SI,DSEGM                ; SET DATA SEGMENT ADDRESS
006D  8E DE                       MOV     DS,SI
006F  C6 06 001E R 01             MOV     STATUS17,1              ; SET PROGRESS FLAG ON
0074  89 26 0014 R                MOV     SPSAVE,SP               ; SAVE STACK POINTER
0078  0B D2                       OR      DX,DX                   ; CHECK PRINTER CLASS
007A  75 3D                       JNZ     NO_USE                  ; IMPROPER DEVICE --RETURN--
007C  83 3E 0000 R 00             CMP     PRINTER_ID,0            ; CHECK PRINTER-ID
0081  75 05                       JNE     I00                     ; AVAILABLE
0083  50                          PUSH    AX                      ; INITIALIZE
0084  E8 0832 R                     CALL    INIT1
0087  58                          POP     AX
0088  C6 06 0002 R 10   I00:      MOV     RETURN_CODE,PR_SELECT   ; SET SELECT TO RETURN CODE
008D  80 0E 0002 R 80             OR      RETURN_CODE,PR_BUSY
0092  80 FC 01                    CMP     AH,1                    ; IF INIT IS REQUIRED,
0095  74 35                       JE      PTR1                    ;  JUMP TO PTR1 ROUTINE DIRECTLY.
0097  80 FC 03                    CMP     AH,3                    ; IF STATUS IS REQUIRED,
009A  74 30                       JE      PTR1                    ;  JUMP TO PTR1 ROUTINE DIRECTLY.
009C  F7 06 0000 R 0001   I10:    TEST    PRINTER_ID,1            ; CHECK PRINTER TYPE-1
00A2  75 28                       JNZ     PTR1
00A4  F7 06 0000 R 0002           TEST    PRINTER_ID,2            ; CHECK PRINTER TYPE-2
00AA  74 03                       JZ      NO_DEV
```

```
00AC  E9 0136 R                    JMP     PTR2
00AF  C6 06 0002 R 08      NO_DEV: MOV     RETURN_CODE,PR_ERROR    ; DEVICE IS INVALID
00B4  80 26 0002 R EF              AND     RETURN_CODE,0FFH-PR_SELECT
00B9                       NO_USE:                                 ; FUNCTION IS INVALID
00B9  58                   RETURN: POP     AX                      ; SET RETURN CODE
00BA  8A 26 0002 R                 MOV     AH,RETURN_CODE
00BE  C6 06 001E R 00              MOV     STATUS17,0              ; SET PROGRESS FLAG OFF
00C3  5B                           POP     BX                      ; RECOVER REGISTERS
00C4  59                           POP     CX
00C5  5A                           POP     DX
00C6  5D                           POP     BP
00C7  5F                           POP     DI
00C8  5E                           POP     SI
00C9  07                           POP     ES
00CA  1F                           POP     DS
00CB  CF                           IRET
                           ;-------------------------------------------------------------------;
                           ;  FOLLOWING MAIN BRANCH IS PREPARED FOR PRINTER TYPE-1             ;
                           ;                                                                   ;
                           ;-------------------------------------------------------------------;
00CC  0A E4                PTR1:   OR      AH,AH
00CE  74 32                        JZ      AH0_1                   ; AH=0 PRINT A CODE IN AL
00D0  FE CC                        DEC     AH
00D2  74 38                        JZ      AH1_1                   ; AH=1 PRINTER INITIALIZATION
00D4  FE CC                        DEC     AH
00D6  74 39                        JZ      AH2_1                   ; AH=2 READ PRINTER STATUS
00D8  FE CC                        DEC     AH
00DA  74 3C                        JZ      AH3_1                   ; AH=3 READ PRINTER STATUS-II
00DC  FE CC                        DEC     AH
00DE  74 3D                        JZ      AH4_1                   ; AH=4 PRINT PASS THRU CODE IN AL
00E0  FE CC                        DEC     AH
00E2  74 3E                        JZ      AH5_1                   ; AH=5 DOUBLE SIZE CHARACTER PRINT
00E4  FE CC                        DEC     AH
00E6  74 18                        JZ      AHX_1                   ; AH=6 IS NOT USED
00E8  FE CC                        DEC     AH
00EA  74 14                        JZ      AHX_1                   ; AH=7 IS NOT USED
00EC  FE CC                        DEC     AH
00EE  74 10                        JZ      AHX_1                   ; AH=8 IS NOT USED
00F0  FE CC                        DEC     AH
00F2  74 0C                        JZ      AHX_1                   ; AH=9 IS NOT USED
00F4  FE CC                        DEC     AH
00F6  74 08                        JZ      AHX_1                   ; AH=A IS NOT USED
00F8  FE CC                        DEC     AH
00FA  74 30                        JZ      AHB_1                   ; AH=B PRINT CHARACTER IN SPECIAL BUFFER
00FC  FE CC                        DEC     AH
00FE  74 31                        JZ      AHC_1                   ; AH=C CHANGE PRINTER CONTROL PARAMETER
                           ;----   AH=? ( PTR1 : FUNCTION INVALID ) ----------------------------;
0100  EB B7                AHX_1:  JMP     NO_USE                  ; AH=? INVALID FUNCTION IS SPECIFIED
                           ;----   AH=0 ( PTR1 : PRINT A CODE IN AL ) --------------------------;
0102  C6 06 000D R 00       AH0_1:  MOV     SIZE_AH,NOR             ; SET CHARACTER SIZE NORMAL
0107  E8 019F R                    CALL    CMD_CHK
010A  EB AD                        JMP     RETURN
                           ;----   AH=1 ( PTR1 : PRINTER INITIALIZATION ) ----------------------;
010C  E8 0832 R            AH1_1:  CALL    INIT1
010F  EB A8                        JMP     RETURN
                           ;----   AH=2 ( PTR1 : READ PRINTER STATUS INTO AH ) -----------------;
0111  32 DB                AH2_1:  XOR     BL,BL
0113  E8 0F54 R                    CALL    STATUS
0116  EB A1                        JMP     RETURN
                           ;----   AH=3 ( PTR1 : READ PRINTER STATUS-II INTO AH ) --------------;
0118  E8 0F91 R            AH3_1:  CALL    STATUS2
011B  EB 9C                        JMP     RETURN
                           ;----   AH=4 ( PTR1 : PRINT PASS THROUGH CODE IN AL ) ---------------;
011D  E8 0F0A R            AH4_1:  CALL    FIRE
0120  EB 97                        JMP     RETURN
                           ;----   AH=5 ( PTR1 : DOUBLE SIZE CHARACTER PRINT ) -----------------;
0122  C6 06 000D R 01      AH5_1:  MOV     SIZE_AH,BAI             ; SET CHARACTER SIZE DOUBLE
0127  E8 019F R                    CALL    CMD_CHK
012A  EB 8D                        JMP     RETURN
                           ;----   AH=B ( PTR1 : PRINT ATTRIBUTES AND CHARACTERS IN SPECIAL BUFFER )
012C  E8 0AB7 R            AHB_1:  CALL    ATTR
012F  EB 88                        JMP     RETURN
                           ;----   AH=C ( PTR1 : CHANGE PRINTER CONTROL PARAMETERS ) -----------;
0131  E8 08F8 R            AHC_1:  CALL    CHG_PT1
0134  EB 83                        JMP     RETURN
                           ;-------------------------------------------------------------------;
                           ;  FOLLOWING MAIN BRANCH IS PREPARED FOR PRINTER TYPE-2             ;
                           ;                                                                   ;
                           ;-------------------------------------------------------------------;
0136  0A E4                PTR2:   OR      AH,AH
0138  74 33                        JZ      AH0_2                   ; AH=0 PRINT A CODE IN AL
013A  FE CC                        DEC     AH
013C  74 35                        JZ      AH1_2                   ; AH=1 PRINTER INITIALIZATION
013E  FE CC                        DEC     AH
0140  74 37                        JZ      AH2_2                   ; AH=2 READ PRINTER STATUS
0142  FE CC                        DEC     AH
0144  74 3B                        JZ      AH3_2                   ; AH=3 READ PRINTER STATUS-II
0146  FE CC                        DEC     AH
0148  74 3D                        JZ      AH4_2                   ; AH=4 PRINT PASS THRU CODE IN AL
014A  FE CC                        DEC     AH
014C  74 3F                        JZ      AH5_2                   ; AH=5 DOUBLE SIZE CHARACTER PRINT
014E  FE CC                        DEC     AH
0150  74 18                        JZ      AHX_2                   ; AH=6 IS NOT USED
0152  FE CC                        DEC     AH
0154  74 14                        JZ      AHX_2                   ; AH=7 IS NOT USED
0156  FE CC                        DEC     AH
0158  74 10                        JZ      AHX_2                   ; AH=8 IS NOT USED
015A  FE CC                        DEC     AH
015C  74 0C                        JZ      AHX_2                   ; AH=9 IS NOT USED
015E  FE CC                        DEC     AH
0160  74 08                        JZ      AHX_2                   ; AH=A IS NOT USED
0162  FE CC                        DEC     AH
0164  74 2D                        JZ      AHB_2                   ; AH=B PRINT ATTRIBUTES AND CHARACTERS
```

```
0166  FE CC                          DEC     AH
0168  74 2F                          JZ      AHC_2                   ; AH=C CHANGE PRINTER CONTROL PARAMETERS
                              ;----  AH=? ( PTR2 : FUNCTION INVALID ) ----------------------------;
016A  E9 00B9 R               AHX_2: JMP     NO_USE                  ; AH=? INVALID FUNCTION IS SPECIFIED
                              ;----  AH=0 ( PTR2 : PRINT A CODE IN AL ) ---------------------------;
NT 17


016D  E8 07FA R               AH0_2: CALL    CHAR0
0170  E9 00B9 R                      JMP     RETURN
                              ;----  AH=1 ( PTR2 : PRINTER INITIALIZATION ) -----------------------;
0173  E8 0832 R               AH1_2: CALL    INIT1
0176  E9 00B9 R                      JMP     RETURN
                              ;----  AH=2 ( PTR2 : READ PRINTER STATUS INTO AH ) ------------------;
0179  32 DB                   AH2_2: XOR     BL,BL
017B  E8 0F54 R                      CALL    STATUS
017E  E9 00B9 R                      JMP     RETURN
                              ;----  AH=3 ( PTR2 : READ PRINTER STATUS-II INTO AH ) ---------------;
0181  E8 0F91 R               AH3_2: CALL    STATUS2
0184  E9 00B9 R                      JMP     RETURN
                              ;----  AH=4 ( PTR2 : PRINT PASS THROUGH CODE IN AL ) ----------------;
0187  E8 0F0A R               AH4_2: CALL    FIRE
018A  E9 00B9 R                      JMP     RETURN
                              ;----  AH=5 ( PTR2 : DOUBLE SIZE CHARACTER PRINT ) ------------------;
018D  E8 0816 R               AH5_2: CALL    CHAR5
0190  E9 00B9 R                      JMP     RETURN
                              ;----  AH=B ( PTR2 : PRINT ATTRIBUTE AND CHARACTER IN SPECIAL BUFFER )
0193  E8 0AB7 R               AHB_2: CALL    ATTR
0196  E9 00B9 R                      JMP     RETURN
                              ;----  AH=C ( PTR2 : CHANGE PRINTER CONTROL PARAMETERS ) ------------;
0199  E8 0979 R               AHC_2: CALL    CHG_PT2
019C  E9 00B9 R                      JMP     RETURN
019F                          PRINTER_IO     ENDP
                              ;-- CMD_CHK ------------------------------------------------------------;
                              ; CHECKS A CODE JUST RECEIVED, AND RECOGNIZES IT AS A COMMAND OR      ;
                              ;  A DATA.                                                            ;
                              ;                                                                     ;
                              ;---------------------------------------------------------------------;
019F                          CMD_CHK PROC    NEAR
019F  F6 06 002C R 80                 TEST    FLG1,TWO_BYTE_FLG       ; AL IS SECOND BYTE OF TWO BYTES CODE ?
01A4  75 3D                           JNZ     A51
01A6  F6 06 002C R 01                 TEST    FLG1,ESC_FLG            ; ESC SEQUENCE IS IN PROGRESS ?
01AB  74 06                           JZ      A21
01AD  E8 01F6 R                       CALL    ESC_COD                 ; JUMP TO ESC SEQ  ANALYSE ROUTINE
01B0  EB 43 90                        JMP     ARET
01B3  3C 1B                   A21:    CMP     AL,1BH                  ; "ESC" ?
01B5  75 0E                           JNE     A4
01B7  80 0E 002C R 01                 OR      FLG1,ESC_FLG            ; SET "ESC" RECEIVED INDICATOR
01BC  C7 06 0020 R 0001               MOV     CODEN,1                 ; SET CODE COUNTER = 1
01C2  EB 31 90                        JMP     ARET
01C5  3C 80                   A4:     CMP     AL,80H                  ; ++++++++++++++++++++++++++++++++
01C7  76 0C                           JBE     A41                     ; +                            +
01C9  3C 9F                           CMP     AL,9FH                  ; + CHECK LEGALITY OF 1ST CODE OF +
01CB  76 0E                           JBE     A5                      ; +                            +
01CD  3C DF                           CMP     AL,0DFH                 ; +         TWO BYTES CODE     +
01CF  76 04                           JBE     A41                     ; +                            +
01D1  3C FC                           CMP     AL,0FCH                 ; +                            +
01D3  76 06                           JBE     A5                      ; ++++++++++++++++++++++++++++++++
01D5  E8 03D9 R               A41:    CALL    SINGLE                  ; OTHERS ARE SINGLE CODES
01D8  EB 1B 90                        JMP     ARET
01DB  80 0E 002C R 80         A5:     OR      FLG1,TWO_BYTE_FLG       ; SET TWO BYTES CODE INDICATOR
01E0  EB 06 90                        JMP     A6
01E3  80 26 002C R 7F         A51:    AND     FLG1,0FFH-TWO_BYTE_FLG  ; RESET TWO BYTES CODE INDICATOR
01E8  50                      A6:     PUSH    AX
01E9  E8 04F6 R                       CALL    MOD_C1                  ; FORCE INTO CHARACTER MODE
01EC  80 0E 0009 R 01                 OR      PRINT_MODE,EVEN_PR_FLG  ; SET CHARACTER MODE
01F1  58                              POP     AX
01F2  E8 0453 R                       CALL    CHR_BUF                 ; CODE IS STORED IN BUFFER
01F5  C3                      ARET:   RET
01F6                          CMD_CHK ENDP
                              ;-- ESC_COD -----------------------------------------------------------;
                              ;  CHECKS CODE COUNTER OF ESC-SEQUENCE, AND TRANSFERS CONTROL TO      ;
                              ;    ANALYSE ROUTINE ACCORDING TO COUNTER VALUE.                      ;
                              ;                                                                     ;
                              ;---------------------------------------------------------------------;
01F6                          ESC_COD PROC    NEAR
01F6  83 3E 0020 R 01                 CMP     CODEN,1                 ; +  IN CASE OF CONTINUOUS "ESC"S  +
01FB  75 04                           JNE     BS1                     ; +    ARE RECEIVED, ONLY FIRST    +
01FD  3C 1B                           CMP     AL,1BH                  ; CHECK AL IS DUMMY "ESC"
01FF  74 56                           JE      BRET                    ; IGNORE
0201  FF 06 0020 R            BS1:    INC     CODEN                   ; CODE COUNT INCREMENT
0205  83 3E 0020 R 02                 CMP     CODEN,2                 ; CODE COUNT = 2 ?
020A  75 06                           JNE     BS2
020C  E8 0258 R                       CALL    B100
020F  EB 46 90                        JMP     BRET
0212  83 3E 0020 R 03         BS2:    CMP     CODEN,3                 ; CODE COUNT = 3 ?
0217  75 06                           JNE     BS3
0219  E8 02B7 R                       CALL    B200
021C  EB 39 90                        JMP     BRET
021F  83 3E 0020 R 04         BS3:    CMP     CODEN,4                 ; CODE COUNT = 4 ?
0224  75 06                           JNE     BS4
0226  E8 033B R                       CALL    B300
0229  EB 2C 90                        JMP     BRET
022C  83 3E 0020 R 05         BS4:    CMP     CODEN,5                 ; CODE COUNT = 5 ?
0231  75 06                           JNE     BS5
0233  E8 036B R                       CALL    B400
0236  EB 1F 90                        JMP     BRET
0239  F6 06 002D R 01         BS5:    TEST    FLG2,X1_FLG             ; ESC X 1  IS IN PROGRESS ?
023E  74 06                           JZ      B01
0240  E8 057C R                       CALL    DX1
0243  EB 12 90                        JMP     BRET
0246  F6 06 002D R 02         B01:    TEST    FLG2,X2_FLG             ; ESC X 2  IS IN PROGRESS ?
024B  74 06                           JZ      B02
```

```
024D  E8 05D5 R                       CALL    DX2
0250  EB 05 90                        JMP     BRET
0253  EB 01 90                  B02:  JMP     ER_RTNB
0256  90                        ER_RTNB:NOP
0257  C3                        BRET:   RET
0258                            ESC_COD ENDP
                                ;-- B100 ------------------------------------------------------------;
                                ;  ANALYSES SECOND CODE OF ESC SEQUENCE.                             ;
                                ;-------------------------------------------------------------------;
0258                            B100    PROC    NEAR
0258  3C 25                    B1:     CMP     AL,25H                  ; CHECK AL="PERCENT"
025A  75 08                           JNE     B11
025C  80 0E 002C R 02                 OR      FLG1,X_FLG              ; SET "X" RECEIVED INDICATOR
0261  EB 53 90                        JMP     B1RET
0264  3C 5B                    B11:    CMP     AL,5BH                  ; CHECK AL="POUND"
0266  75 0D                           JNE     B12
0268  C6 06 000C R 01                 MOV     SIZE_ESC,BAI            ; SET CHAR. SIZE TO DOUBLE
026D  80 26 002C R FE                 AND     FLG1,0FFH-ESC_FLG       ; RESET "ESC" INDICATOR
0272  EB 42 90                        JMP     B1RET
0275  3C 5D                    B12:    CMP     AL,5DH                  ; CHECK AL="VERTICAL BAR"
0277  75 0D                           JNE     B13
0279  C6 06 000C R 00                 MOV     SIZE_ESC,NOR            ; SET CHAR. SIZE TO NORMAL
027E  80 26 002C R FE                 AND     FLG1,0FFH-ESC_FLG       ; RESET "ESC" INDICATOR
0283  EB 31 90                        JMP     B1RET
0286  3C 46                    B13:    CMP     AL,46H                  ; CHECK AL="F"
0288  75 08                           JNE     B14
028A  80 0E 002C R 04                 OR      FLG1,F_FLG              ; SET "F" INDICATOR
028F  EB 25 90                        JMP     B1RET
0292  3C 28                    B14:    CMP     AL,28H                  ; CHECK AL="("
0294  74 17                           JE      B15
0296  3C 29                           CMP     AL,29H                  ; CHECK AL=")"
0298  74 13                           JE      B15
029A  3C 4F                           CMP     AL,4FH                  ; CHECK AL="O"
029C  74 0F                           JE      B15
029E  3C 50                           CMP     AL,50H                  ; CHECK AL="P"
02A0  74 0B                           JE      B15
02A2  3C 53                           CMP     AL,53H                  ; CHECK AL="S"
02A4  74 07                           JE      B15
02A6  3C 56                           CMP     AL,56H                  ; CHECK AL="V"
02A8  74 03                           JE      B15
02AA  EB 09 90                        JMP     ER_B1
02AD  80 26 002C R FE           B15:    AND     FLG1,0FFH-ESC_FLG       ; RESET "ESC" RECEIVED INDICATOR
02B2  EB 02 90                        JMP     B1RET
02B5  90                        ER_B1:  NOP
02B6  C3                        B1RET:  RET
02B7                            B100    ENDP
                                ;-- B200 ------------------------------------------------------------;
                                ;  ANALYSES THIRD CODE OF ESC SEQUENCE.                              ;
                                ;-------------------------------------------------------------------;
02B7                            B200    PROC    NEAR
02B7  F6 06 002C R 02           B2:     TEST    FLG1,X_FLG              ; "X" RECEIVED ?
02BC  75 10                           JNZ     B201
02BE  F6 06 002C R 04                 TEST    FLG1,F_FLG              ; "F" RECEIVED ?
02C3  75 03                           JNZ     B21
02C5  EB 72 90                        JMP     ER_B2
02C8  A2 0022 R                 B21:    MOV     N1,AL                   ; AL IS N1 VALUE ( IF "F" RCVD )
02CB  EB 6D 90                        JMP     B2RET
02CE  3C 31                     B201:   CMP     AL,31H                  ; AL IS "1" ?
02D0  74 27                           JE      B202
02D2  3C 32                           CMP     AL,32H                  ; AL IS "2" ?
02D4  74 2B                           JE      B203
02D6  3C 33                           CMP     AL,33H                  ; AL IS "3" ?
02D8  74 2F                           JE      B204
02DA  3C 35                           CMP     AL,35H                  ; AL IS "5" ?
02DC  74 33                           JE      B205
02DE  3C 36                           CMP     AL,36H                  ; AL IS "6" ?
02E0  74 37                           JE      B206
02E2  3C 39                           CMP     AL,39H                  ; AL IS "9" ?
02E4  74 3B                           JE      B207
02E6  3C 34                           CMP     AL,34H                  ; AL IS "4" ?
02E8  74 3F                           JE      B208
02EA  3C 38                           CMP     AL,38H                  ; AL IS "8" ?
02EC  74 3B                           JE      B208
02EE  3C 42                           CMP     AL,42H                  ; AL IS "B" ?
02F0  74 3F                           JE      B209
02F2  3C 55                           CMP     AL,55H                  ; AL IS "U" ?
02F4  74 3B                           JE      B209
02F6  EB 41 90                        JMP     ER_B2                   ; OTHERS ARE ILLEGAL
02F9  80 0E 002D R 01           B202:   OR      FLG2,X1_FLG             ; SET "X1" INDICATOR
02FE  EB 3A 90                        JMP     B2RET
0301  80 0E 002D R 02           B203:   OR      FLG2,X2_FLG             ; SET "X2" INDICATOR
0306  EB 32 90                        JMP     B2RET
0309  80 0E 002D R 04           B204:   OR      FLG2,X3_FLG             ; SET "X3" INDICATOR
030E  EB 2A 90                        JMP     B2RET
0311  80 0E 002D R 08           B205:   OR      FLG2,X5_FLG             ; SET "X5" INDICATOR
0316  EB 22 90                        JMP     B2RET
0319  80 0E 002D R 10           B206:   OR      FLG2,X6_FLG             ; SET "X6" INDICATOR
031E  EB 1A 90                        JMP     B2RET
0321  80 0E 002D R 20           B207:   OR      FLG2,X9_FLG             ; SET "X9" INDICATOR
0326  EB 12 90                        JMP     B2RET
0329  80 0E 002D R 40           B208:   OR      FLG2,IGN_FLG            ; SET COMMAND IGNORE INDICATOR
032E  EB 0A 90                        JMP     B2RET
0331  80 26 002C R FC           B209:   AND     FLG1,0FFH-X_FLG-ESC_FLG ; RESET "X" & "ESC" INDICATOR
0336  EB 02 90                        JMP     B2RET
0339  90                        ER_B2:  NOP
033A  C3                        B2RET:  RET
033B                            B200    ENDP
                                ;-- B300 ------------------------------------------------------------;
                                ;  ANALYSES FOUTH CODE OF ESC SEQUENCE.                              ;
                                ;                                                                    ;
                                ;-------------------------------------------------------------------;
033B                            B300    PROC    NEAR
033B  F6 06 002D R 7F           B3:     TEST    FLG2,7FH                ; X1,2,3,5,6,9 MODE IN PROGRESS ?
```

```
0340  74 06                      JZ      B31
0342  A2 0022 R                  MOV     N1,AL                   ; AL IS N1 VALUE
0345  EB 23 90                   JMP     B3RET
0348  F6 06 002C R 04    B31:    TEST    FLG1,F_FLG              ; "F" MODE IN PROGRESS ?
034D  74 10                      JZ      B32
034F  A2 0023 R                  MOV     N2,AL                   ; AL IS N2 VALUE
0352  8A 26 0022 R               MOV     AH,N1
0356  A3 0024 R                  MOV     N1N2,AX                 ; N1N2 VALUE GEN
0359  E8 07C0 R                  CALL    F_EM                    ; CALL ESC-F EMULATION ROUTINE
035C  EB 0C 90                   JMP     B3RET
035F  F6 06 002D R 40    B32:    TEST    FLG2,IGN_FLG            ; IGNORE INDICATOR ON ?
0364  75 04                      JNZ     B3RET                   ; RETURN
0366  EB 01 90                   JMP     ER_B3
0369  90                 ER_B3:  NOP
036A  C3                 B3RET:  RET
036B                     B300    ENDP
                         ;--B400 ----------------------------------------------------------------;
                         ;  ANALYSES FIFTH CODE OF ESC SEQUENCE.                                 ;
                         ;                                                                       ;
                         ;-----------------------------------------------------------------------;
036B                     B400    PROC    NEAR
036B  F6 06 002D R 40    B4:     TEST    FLG2,IGN_FLG            ; IGNORE INDICATOR ON ?
0370  74 0D                      JZ      B41
0372  80 26 002C R FC            AND     FLG1,0FFH-ESC_FLG-X_FLG ; RESET "ESC" & "X" INDICATOR
0377  80 26 002D R BF            AND     FLG2,0FFH-IGN_FLG       ; RESET IGNORE INDICATOR
037C  EB 5A 90                   JMP     B4RET
037F  A2 0023 R          B41:    MOV     N2,AL                   ; AL IS N2
0382  8A 26 0022 R               MOV     AH,N1                   ;
0386  A3 0024 R                  MOV     N1N2,AX                 ; N1N2 VALUE GEN
0389  F6 06 002D R 01            TEST    FLG2,X1_FLG             ; "X1" IS IN PROGRESS ?
038E  74 06                      JZ      B42
0390  E8 0494 R                  CALL    X1X2_EM                 ; CALL X1 X2 EMULATION ROUTINE
0393  EB 43 90                   JMP     B4RET
0396  F6 06 002D R 02    B42:    TEST    FLG2,X2_FLG             ; "X2" IS IN PROGRESS ?
039B  74 06                      JZ      B43
039D  E8 0494 R                  CALL    X1X2_EM                 ; CALL X1 X2 EMULATION ROUTINE
03A0  EB 36 90                   JMP     B4RET
03A3  F6 06 002D R 04    B43:    TEST    FLG2,X3_FLG             ; "X3" IS IN PROGRESS ?
03A8  74 06                      JZ      B44
03AA  E8 052F R                  CALL    X3_EM                   ; CALL X3 EMULATION ROUTINE
03AD  EB 29 90                   JMP     B4RET
03B0  F6 06 002D R 08    B44:    TEST    FLG2,X5_FLG             ; "X5" IS IN PROGRESS ?
03B5  74 06                      JZ      B45
03B7  E8 0738 R                  CALL    X5_EM                   ; CALL X5 EMULATION ROUTINE
03BA  EB 1C 90                   JMP     B4RET
03BD  F6 06 002D R 10    B45:    TEST    FLG2,X6_FLG             ; "X6" IS IN PROGRESS ?
03C2  74 06                      JZ      B46
03C4  E8 06EF R                  CALL    X6_EM                   ; CALL X6 EMULATION ROUTINE
03C7  EB 0F 90                   JMP     B4RET
03CA  F6 06 002D R 20    B46:    TEST    FLG2,X9_FLG             ; "X9" IS IN PROGRESS ?
03CF  74 06                      JZ      ER_B4
03D1  E8 078E R                  CALL    X9_EM                   ; CALL X9 EMULATION ROUTINE
03D4  EB 02 90                   JMP     B4RET
03D7  90                 ER_B4:  NOP
03D8  C3                 B4RET:  RET
03D9                     B400    ENDP
                         ;-- SINGLE ------------------------------------------------------------;
                         ; ANALYSES ONE BYTE CONTROL CODE WHICH INCLUDES                         ;
                         ;  CANCEL, CARRIAGE RETURN, LINE FEED, FORM FEED, SPACE, DC1 OR DC3.   ;
                         ;                                                                       ;
                         ;-----------------------------------------------------------------------;
03D9                     SINGLE  PROC    NEAR
03D9  3C 18                      CMP     AL,18H                  ; "CAN" ?
03DB  74 1F                      JE      C1
03DD  3C 0A                      CMP     AL,0AH                  ; "LF" ?
03DF  74 21                      JE      C3
03E1  3C 0C                      CMP     AL,0CH                  ; "FF" ?
03E3  74 3B                      JE      C4
03E5  3C 1C                      CMP     AL,1CH                  ; "FS" ?
03E7  74 56                      JE      C5
03E9  3C 0D                      CMP     AL,0DH                  ; "CR" ?
03EB  74 65                      JE      CRET
03ED  3C 08                      CMP     AL,08H                  ; "BS" ?
03EF  74 61                      JE      CRET
03F1  3C 11                      CMP     AL,11H                  ; "DC1" ?
03F3  74 5D                      JE      CRET
03F5  3C 13                      CMP     AL,13H                  ; "DC3" ?
03F7  74 59                      JE      CRET
03F9  EB 4A 90                   JMP     C6                      ; JUMP TO ONE BYTE CHR CODE
                         ;++++++++++++++++++++++++++++++++++++++++ CANCEL ++++++++++++++++++++
03FC  E8 08E1 R          C1:     CALL    RESET
03FF  EB 51 90                   JMP     CRET
                         ;++++++++++++++++++++++++++++++++++++++++ LINE FEED ++++++++++++++++++
0402  E8 047D R          C3:     CALL    PRINT2
0405  51                         PUSH    CX
0406  33 C9                      XOR     CX,CX                   ;CLEAR CX
0408  0A 0E 0026 R               OR      CL,LF_CT                ; CL=0 ?
040C  75 02                      JNZ     C31
040E  B1 01                      MOV     CL,1
0410  B0 0A              C31:    MOV     AL,0AH                  ; "LF"
0412  E8 0F0A R                  CALL    FIRE                    ;
0415  E2 F9                      LOOP    C31
0417  59                 C32:    POP     CX
0418  C6 06 0009 R 00           MOV     PRINT_MODE,0            ; RESET PRINT MODE
041D  EB 33 90                   JMP     CRET
                         ;++++++++++++++++++++++++++++++++++++++++ FORM FEED ++++++++++++++++++
0420  E8 047D R          C4:     CALL    PRINT2
0423  B0 1B                      MOV     AL,1BH                  ; THIS SEQUENCE KEEPS CORRECT
0425  E8 0F0A R                  CALL    FIRE                    ;   PAGE LENGTH FEEDITN BY
0428  B0 32                      MOV     AL,32H                  ;     LINE FEED MOVEMENT OF
042A  E8 0F0A R                  CALL    FIRE                    ;       ONE-SIXTH-INCH
042D  B0 0C                      MOV     AL,0CH                  ; "FF"
042F  E8 0F0A R                  CALL    FIRE                    ;
```

```
0432  B0 1B                        MOV.    AL,1BH                  ; THIS SEQUENCE SETS LINE FEED
0434  E8 0F0A R                     CALL    FIRE                    ;   PITCH FOR ORDINARY USING
0437  B0 30                        MOV     AL,30H                  ;     VALUE OF ONE-NINTH-INCH
0439  E8 0F0A R                     CALL    FIRE                    ;
043C  EB 14 90                     JMP     CRET
                                ;++++++++++++++++++++++++++++++++++++++++ FIXED LENGTH IMAGE TX ++++++
043F  E8 065F R                 C5:     CALL    FS_EM
0442  EB 0E 90                     JMP     CRET
                                ;++++++++++++++++++++++++++++++++++++++++ ONE BYTE CHARACTER +++++++++
0445  50                        C6:     PUSH    AX                      ; SAVE AX
0446  E8 04F6 R                     CALL    MOD_C1                  ; FORCE INTO CHARACTER MODE
0449  80 0E 0009 R 01                OR      PRINT_MODE,EVEN_PR_FLG  ; SET CHARACTER MODE
044E  58                           POP     AX                      ; RECOVER AX
044F  E8 0453 R                    CALL    CHR_BUF                 ; STORE SPACE TO CHR BUFFER
0452  C3                        CRET:   RET
0453                            SINGLE  ENDP
                                ;-- CHR_BUF ---------------------------------------------------------;
                                ; STORES A RECEIVED CHARACTER CODE INTO CHARACTER BUFFER             ;
                                ;  AND INDICATES CHARACTER SIZE INTO ATTRIBUTE BUFFER.               ;
                                ;                                                                    ;
                                ;--------------------------------------------------------------------;
0453                            CHR_BUF PROC    NEAR
0453  56                           PUSH    SI                      ; SAVE SI
0454  53                           PUSH    BX                      ; SAVE BX
0455  83 3E 000E R 7C              CMP     CCP,7CH                 ; IF OVER, IGNORE THE CODE
045A  77 1E                        JA      W1                      ;
045C  8B 36 000E R                 MOV     SI,CCP                  ; CHAR. CODE TO BUFFER
0460  88 84 00A0 R                 MOV     CODE_BUFFER[SI],AL      ;
0464  8A 1E 000C R                 MOV     BL,SIZE_ESC             ; CHECK DOUBLE SIZE CHAR.
0468  0A 1E 000D R                 OR      BL,SIZE_AH              ;
046C  88 9C 0190 R                 MOV     ATTR_BUFFER[SI],BL      ; SIZE ID TO BUFFER
0470  32 FF                        XOR     BH,BH                   ; CLEAR BH
0472  01 1E 000E R                 ADD     CCP,BX                  ; ADJUST CURRENT CHAR. POSITION
0476  FF 06 000E R                 INC     CCP                     ; INDICATOR INCREMENT
047A  5B                        W1:     POP     BX                      ; RECOVER BX
047B  5E                           POP     SI                      ; RECOVER SI
047C  C3                           RET
047D                            CHR_BUF ENDP
                                ;-- PRINT2 ---------------------------------------------------------;
                                ; ACTIVATES SECONDARY PRINTING.                                      ;
                                ;   IN CHARACTER MODE : CALLS CHARACTER PRINT ROUTINE                ;
                                ;   IN GRAPHIC MODE   : CALLS LOW PART PRINT ROUTINE                 ;
                                ;                                                                    ;
                                ;--------------------------------------------------------------------;
047D                            PRINT2  PROC    NEAR
047D  F6 06 0009 R 01              TEST    PRINT_MODE,EVEN_PR_FLG  ; CHARACTER MODE ?
0482  75 08                        JNZ     XX1                     ; NO
0484  F6 06 0009 R 02              TEST    PRINT_MODE,LOW_PR_FLG   ; GRAPHIC MODE ?
0489  75 05                        JNZ     XX2                     ; NO
048B  C3                           RET                             ; FIRST CHARACTER
048C  E8 0AA7 R                 XX1:    CALL    PRINT                   ; CALL CHARACTER PRINT
048F  C3                           RET
0490  E8 0699 R                 XX2:    CALL    LOW_PRT                 ; CALL LOW PART PRINT
0493  C3                           RET
0494                            PRINT2  ENDP
                                ;---- X1X2_EM ------------------------------------------------------;
                                ; CHECKS TRANSMIT DATA COUNT OF ESC-X1 OR ESC-X2 SEQUENCE            ;
                                ;  AND CONVERTS TO ESC-L                                             ;
                                ;                                                                    ;
                                ;--------------------------------------------------------------------;
0494                            X1X2_EM PROC    NEAR
0494  A1 0024 R                 D0:     MOV     AX,N1N2
0497  3D 0000                        CMP     AX,0                    ; TX COUNT = 0 ?
049A  74 4F                        JE      D6                      ; IF ZERO THEN GOTO END
049C  50                        D1:     PUSH    AX                      ; SAVE AX
049D  E8 0515 R                    CALL    MOD_C2                  ; INTO GRAPHIC MODE OK ?
04A0  80 0E 0009 R 02              OR      PRINT_MODE,LOW_PR_FLG   ; INTO GRAPHIC MODE
04A5  80 26 002E R EF              AND     FLG3,0FFH-SL_FUL_FLG    ; RESET SLICE FULL FLAG
04AA  C6 06 0028 R 01              MOV     IM_MOD,1                ; DEFAULT IMAGE MODE ESCX1
04AF  58                           POP     AX                      ; RECOVER AX
04B0  F6 06 002D R 02              TEST    FLG2,X2_FLG             ; ESCX2 ACTIVE ?
04B5  74 0C                        JZ      D3
04B7  C6 06 0028 R 02              MOV     IM_MOD,2                ; IMAGE MODE SET TO ESCX2
04BC  80 26 002E R DF              AND     FLG3,0FFH-BAI_FUL_FLG   ; RESET SECOND SLICE FULL FLAG
04C1  D1 E0                        SHL     AX,1                    ; TX COUNT TO DOUBLE SIZE
04C3  3D 0460                   D3:     CMP     AX,MAX                  ; TX COUNT OVER 1120 ?
04C6  76 03                        JBE     D4                      ; IF NOT, JUMP
04C8  B8 0460                      MOV     AX,MAX                  ; FORCE COUNT SET TO 1120
04CB  A3 0029 R                 D4:     MOV     FS_N,AX                 ; STORE N1N2 VALUE FOR FS
04CE  53                           PUSH    BX                      ; SAVE BX
04CF  51                           PUSH    CX                      ; SAVE CX
04D0  BB 0460                      MOV     BX,MAX                  ; SET MAX. SLICE POSITION
04D3  8B 0E 0010 R                 MOV     CX,CSP                  ; GET CURRENT SLICE POSITION
04D7  03 C8                        ADD     CX,AX                   ; CALCULATE NEXT FINAL SLICE POSITION
04D9  3B D9                        CMP     BX,CX                   ; COMPARE MAX. AND NEXT FINAL
04DB  73 06                        JAE     D5                      ; IF MAX IS NOT LESS, JUMP
04DD  2B 1E 0010 R                 SUB     BX,CSP                  ; CALCULATE CORRECT COUNT
04E1  8B C3                        MOV     AX,BX                   ; CORRECT COUNT SET TO AX
04E3  59                        D5:     POP     CX                      ; RECOVER CX
04E4  5B                           POP     BX                      ; RECOVER BX
04E5  E8 0E88 R                    CALL    ESCL                    ; CALL ESC-L OUT SUB.
04E8  EB 0B 90                     JMP     D7                      ; RETURN
04EB  80 26 002C R FC           D6:     AND     FLG1,0FFH-ESC_FLG-X_FLG ; RESET ESC & X FLAG
04F0  80 26 002D R FC              AND     FLG2,0FFH-X1_FLG-X2_FLG ; RESET X1 & X2 FLAG
04F5  C3                        D7:     RET
04F6                            X1X2_EM ENDP
                                ;-- MOD_C1 ---------------------------------------------------------;
                                ;  CHECKS CURRENT PRINT MODE, AND IF GRAPHIC MODE IS ACTIVE.         ;
                                ;    FORCES LOW PART OF GRAPHIC IMAGE OUT TO PRINTER AND             ;
                                ;    FORCES INTO CHARACTER MODE.                                     ;
                                ;                                                                    ;
                                ;--------------------------------------------------------------------;
```

```
04F6                            MOD_C1  PROC    NEAR
04F6  F6 06 0009 R 02                   TEST    PRINT_MODE,LOW_PR_FLG   ; GRAPHIC MODE ?
04FB  74 17                             JZ      C1END
04FD  E8 0699 R                         CALL    LOW_PRT                 ; CALL LOW PART PRINT
0500  80 26 0009 R FD                   AND     PRINT_MODE,0FFH-LOW_PR_FLG      ; RESET GRAPHIC MODE
0505  B0 0D                             MOV     AL,0DH                  ; FORCE CR + LF OUT TO PRINTER
0507  E8 0F0A R                         CALL    FIRE
050A  B0 0A                             MOV     AL,0AH
050C  E8 0F0A R                         CALL    FIRE
050F  B0 0A                             MOV     AL,0AH
0511  E8 0F0A R                         CALL    FIRE
0514  C3                       C1END:   RET
0515                            MOD_C1  ENDP
                                ;-- MOD_C2 ----------------------------------------------------------------;
                                ;  CHECKS CURRENT PRINT MODE, AND IF CHARCATER MODE IS ACTIVE,              ;
                                ;    FORCES CHARACTER DATA OUT TO PRINTER AND                               ;
                                ;    FORCES INTO GRAPHIC MODE.                                              ;
                                ;                                                                           ;
                                ;---------------------------------------------------------------------------;
0515                            MOD_C2  PROC    NEAR
0515  F6 06 0009 R 01                   TEST    PRINT_MODE,EVEN_PR_FLG  ; CHARACTER MODE ?
051A  74 12                             JZ      C2END
051C  E8 0AA7 R                         CALL    PRINT                   ; CALL CHAR. PRINT ROUTINE
051F  80 26 0009 R FE                   AND     PRINT_MODE,0FFH-EVEN_PR_FLG     ; RESET CHARACTER MODE
0524  B0 0D                             MOV     AL,0DH                  ;FORCE CR + LF OUT TO PRINTER
0526  E8 0F0A R                         CALL    FIRE
0529  B0 0A                             MOV     AL,0AH
052B  E8 0F0A R                         CALL    FIRE
052E  C3                       C2END:   RET
052F                            MOD_C2  ENDP
                                ;-- X3_EM ----------------------------------------------------------------;
                                ; EMULATES ESC-X3 ( HORIZONTAL SKIP ), AND                                 ;
                                ;  CONVERTS ESC-X3 TO ESC-L + NULL DATA .                                   ;
                                ;                                                                           ;
                                ;---------------------------------------------------------------------------;
052F                            X3_EM   PROC    NEAR
052F  8B 0E 0024 R                      MOV     CX,N1N2                 ; GET N1N2 VALUE
0533  83 F9 00                          CMP     CX,0                    ; CHECK ZERO
0536  76 39                             JBE     N33
0538  81 F9 0460                        CMP     CX,MAX                  ; CHECK < 1120
053C  77 33                             JA      N33
053E  51                                PUSH    CX
053F  E8 0515 R                         CALL    MOD_C2                  ; FORCE INTO GRAPHIC MODE
0542  59                                POP     CX
0543  A1 0010 R                         MOV     AX,CSP                  ; GET CURRENT SLICE POSITION
0546  50                                PUSH    AX
0547  5E                              POP     SI
0548  03 C1                           ADD     AX,CX                   ; ADJUST MAX SLICE POSITION
054A  3D 0460                           CMP     AX,MAX                  ;
054D  77 17                             JA      N22
054F  80 0E 0009 R 02                   OR      PRINT_MODE,LOW_PR_FLG   ; INTO GRAPHIC MODE
0554  A3 0010 R                         MOV     CSP,AX                  ;
0557  8B C1                             MOV     AX,CX
0559  E8 0E88 R                         CALL    ESCL                    ; CALL ESC-L OUT
055C  B0 00                    N11:     MOV     AL,0                    ; ZERO OUT TO PRINTER
055E  E8 0F0A R                         CALL    FIRE                    ;   FOR MAKING BLANK
0561  E2 F9                             LOOP    N11
0563  EB 0C 90                         JMP     N33
0566  E8 0699 R                N22:     CALL    LOW_PRT                 ; CALL LOW PART GRAPHIC PRINT
0569  B0 0D                             MOV     AL,0DH                  ; FORCE "CR" OUT
056B  E8 0F0A R                         CALL    FIRE
056E  EB 01 90                          JMP     N33
0571  80 26 002C R FC          N33:     AND     FLG1,0FFH-ESC_FLG-X_FLG ; RESET "ESC" & "X" INDICATOR
0576  80 26 002D R FB                   AND     FLG2,0FFH-X3_FLG        ; RESET "X3" INDICATOR
057B  C3                                RET
057C                            X3_EM   ENDP
                                ;-- DX1 ----------------------------------------------------------------;
                                ; CONTROLS ESC-X1 (IMAGE TRANSMISSSION) DATA STREAM.                      ;
                                ;  IF DATA COUNT IS ODD,  THE DATA IS OUT TO PRINTER,                      ;
                                ;  IF DATA COUNT IS EVEN, THE DATA IS STORED IN IMAGE BUFFER FOR LOW       ;
                                ;   PART PRINTING.                                                         ;
                                ;                                                                          ;
                                ;--------------------------------------------------------------------------;
057C                            DX1     PROC    NEAR
057C  8B 0E 0020 R                      MOV     CX,CODEN                ;GET CODE COUNT
0580  83 E9 05                          SUB     CX,5                    ;CALCULATE CORRECT DATA COUNT
0583  F7 C1 0001                        TEST    CX,01H                  ;CX IS ODD NUMBER ?
0587  74 28                             JZ      F2                      ;IF EVEN, GO TO F1
0589  F6 06 002E R 10                   TEST    FLG3,SL_FUL_FLG         ;CHECK SLICE COUNT EXCEEDS MAX
058E  75 44                             JNZ     F4
0590  FF 06 0010 R                      INC     CSP                     ;UPDATE CSP
0594  81 3E 0010 R 0460                 CMP     CSP,MAX                 ;CHECK MAX POSITION
059A  77 06                             JA      F1
059C  E8 0F0A R                         CALL    FIRE                    ;AL (DATA) IS OUT
059F  EB 33 90                          JMP     F4
05A2  F6 06 002E R 10           F1:     TEST    FLG3,SL_FUL_FLG         ;CHECK SLICE COUNT EXCEEDS MAX
05A7  75 2B                             JNZ     F4
05A9  80 0E 002E R 10                   OR      FLG3,SL_FUL_FLG         ;IF EXCEEDS, SET SLICE FULL IND.
05AE  EB 24 90                          JMP     F4
05B1  F6 06 002E R 10           F2:     TEST    FLG3,SL_FUL_FLG         ;IF SLICE POSITION EXCEEDS MAX.
05B6  75 0A                             JNZ     F3                      ;IGNORE FURTHER DATA
05B8  33 F6                             XOR     SI,SI                   ;CLEAR SI
05BA  03 36 0010 R                      ADD     SI,CSP                  ;GET CSP TO SI
05BE  08 84 00A0 R                      OR      IMAGE_BUFFER[SI],AL     ;DATA IS STORED FOR SECONDARY PRINT
05C2  D1 E9                     F3:     SHR     CX,1                    ;DEVIDE BY 2
05C4  3B 0E 0024 R                      CMP     CX,N1N2                 ;REACHED N1N2 COUNT ?
05C8  75 0A                             JNE     F4
05CA  80 26 002C R FC                   AND     FLG1,0FFH-ESC_FLG-X_FLG ; RESET "ESC" & "X" INDICATOR
```

```
05CF  80 26 002D R FE          AND     FLG2,0FFH-X1_FLG        ; RESET "X1" FLAG
05D4  C3                       RET
05D5                           DX1     ENDP
                               ;-- DX2 ---------------------------------------------------------;
                               ; CONTROLS ESC-X2 (IMAGE TRANSMISSION AND ENLARGE) DATA STREAM.  ;
                               ;  IF DATA COUNT IS ODD,  THE DATA IS OUT TO PRINTER TWICE,      ;
                               ;  IF DATA COUNT IS EVEN, THE DATA IS STORED TWICE IN IMAGE BUFFER;
                               ;  FOR LOW PART PRINTING.                                        ;
                               ;                                                                ;
                               ;----------------------------------------------------------------;
05D5                           DX2     PROC    NEAR
05D5  8B 0E 0020 R             MOV     CX,CODEN                ; GET CODE COUNT TO CX
05D9  83 E9 05                 SUB     CX,5                    ; CALCULATE CORRECT DATA COUNT
05DC  F7 C1 0001               TEST    CX,01H                  ; CX IS ODD NUMBER ?
05E0  74 3A                    JZ      G2
05E2  F6 06 002E R 10          TEST    FLG3,SL_FUL_FLG         ; CHECK SLICE COUNT EXCEEDS MAX
05E7  75 75                    JNZ     G5
05E9  FF 06 0010 R             INC     CSP                     ; UPDATE CSP
05ED  81 3E 0010 R 0460        CMP     CSP,MAX                 ; CHECK MAX POSITION
05F3  77 5D                    JA      G4
05F5  50                       PUSH    AX                      ; SAVE N1N2
05F6  E8 0F0A R                CALL    FIRE
05F9  58                       POP     AX                      ; GET  N1N2
05FA  FF 06 0010 R             INC     CSP                     ; UPDATE CSP
05FE  81 3E 0010 R 0460        CMP     CSP,MAX                 ; CHECK MAX POSOTION
0604  77 0A                    JA      G1
0606  E8 0F0A R                CALL    FIRE
0609  FF 0E 0010 R             DEC     CSP                     ; DECREMENT CSP FOR ADJUSTING
060D  EB 4F 90                 JMP     G5                      ;       LOW PART DATA COUNT
0610  80 0E 002E R 20         G1:     OR      FLG3,BAI_FUL_FLG        ; SET SLICE FULL INDICATOR
0615  FF 0E 0010 R             DEC     CSP                     ; DECREMENT CSP FOR ADJUSTING
0619  EB 43 90                 JMP     G5                      ;       LOW PART DATA COUNT
061C  F6 06 002E R 10        G2:     TEST    FLG3,SL_FUL_FLG         ; IF SLICE POSITION EXCEEDS MAX.
0621  75 1A                    JNZ     G3                      ; IGNORE FURTHER DATA
0623  33 F6                    XOR     SI,SI                   ; CLEAR SI
0625  03 36 0010 R             ADD     SI,CSP                  ; GET CSP TO SI
0629  08 84 00A0 R             OR      IMAGE_BUFFER[SI],AL     ; DATA IS STORE IN IMAGE BUFFER
062D  F6 06 002E R 20          TEST    FLG3,BAI_FUL_FLG        ; CHECK SLICE FULL INDICATOR
0632  75 1E                    JNZ     G4
0634  46                       INC     SI                      ; UPDATE SLICE POSITION
0635  08 84 00A0 R             OR      IMAGE_BUFFER[SI],AL
0639  89 36 0010 R             MOV     CSP,SI                  ; CURRENT SLICE POSITION STORE
063D  D1 E9                  G3:     SHR     CX,1                    ; DEVIDE BY 2
063F  3B 0E 0024 R             CMP     CX,N1N2                 ; REACHED N1N2 COUNT ?
0643  75 19                    JNE     G5
0645  80 26 002C R FC          AND     FLG1,0FFH-ESC_FLG-X_FLG ; RESET "ESC" & "X" INDICATOR
064A  80 26 002D R FD          AND     FLG2,0FFH-X2_FLG        ; RESET "X2" INDICATOR
064F  EB 0D 90                 JMP     G5
0652  F6 06 002E R 10        G4:     TEST    FLG3,SL_FUL_FLG         ; CHECK SLICE FULL INDICATOR
0657  75 05                    JNZ     G5
0659  80 0E 002E R 10          OR      FLG3,SL_FUL_FLG         ; SET SLICE FULL INDICATOR
065E  C3                     G5:     RET
065F                           DX2     ENDP
                               ;-- FS_EM --------------------------------------------------------;
                               ; EMULATES FIXED DATA IMAGE TRANSMISSION DATA STREAM.            ;
                               ;  TRANSMITED DATA COUNT IS STORED IN FS_N.                      ;
                               ;                                                                ;
                               ;  THIS COMMAND USES MOST RECENT ESC-X1 OR ESC-X2 CONTROL VALUES.;
                               ;                                                                ;
                               ;----------------------------------------------------------------;
065F                           FS_EM   PROC    NEAR
065F  83 3E 0029 R 00          CMP     FS_N,0                  ; CHECK TX COUNT
0664  74 32                    JZ      H2                      ; IF ZERO, GO TO END
0666  E8 0515 R                CALL    MOD_C2                  ; ALREADY IMAGE MODE ?
0669  80 0E 0009 R 02          OR      PRINT_MODE,LOW_PR_FLG   ; INTO IMAGE MODE
066E  80 0E 002C R 03          OR      FLG1,ESC_FLG+X_FLG      ; ESC X ON
0673  80 26 002E R EF          AND     FLG3,0FFH-SL_FUL_FLG    ; RESET SLICE FULL INDICATOR
0678  A1 0029 R                MOV     AX,FS_N                 ; TX COUNT INTO AX
067B  C7 06 0020 R 0005        MOV     CODEN,5                 ; ADJUST CODE COUNTER
0681  E8 0E88 R                CALL    ESCL                    ; ESC L AND N1 N2 OUT
0684  F6 06 0028 R 02          TEST    IM_MOD,02H              ; NOW ESC X 2 ?
0689  74 08                    JZ      H1                      ; IF NOT THEN ESC X1
068B  80 0E 002D R 02          OR      FLG2,X2_FLG             ; ESC X2 FLG ON
0690  EB 06 90                 JMP     H2
0693  80 0E 002D R 01        H1:     OR      FLG2,X1_FLG             ; ESC X1 FLG ON
0698  C3                     H2:     RET
0699                           FS_EM   ENDP
                               ;-- LOW_PRT -----------------------------------------------------;
                               ; PRINTS LOW PART OF 8-BIT GRAPHIC IMAGE IN IMAGE BUFFER         ;
                               ;                                                                ;
                               ;----------------------------------------------------------------;
0699                           LOW_PRT PROC    NEAR
0699  B0 1B                    MOV     AL,1BH                  ; THIS SEQUENCE SETS LINE FEED
069B  E8 0F0A R                CALL    FIRE                    ;   PITCH FOR ORDINARY USING
069E  B0 30                    MOV     AL,30H                  ;     VALUE OF ONE-NINTH-INCH.
06A0  E8 0F0A R                CALL    FIRE                    ;
06A3  B0 0D                    MOV     AL,0DH                  ; CR
06A5  E8 0F0A R                CALL    FIRE
06A8  B0 0A                    MOV     AL,0AH                  ; LF
06AA  E8 0F0A R                CALL    FIRE
06AD  A1 0010 R                MOV     AX,CSP                  ; GET CSP TO AX
06B0  3B 06 0012 R             CMP     AX,CSPMAX
06B4  77 03                    JA      E1
06B6  A1 0012 R                MOV     AX,CSPMAX
06B9  3D 0460                E1:     CMP     AX,MAX                  ; CHECK CSP EXCEEDS MAX. ?
06BC  76 03                    JBE     E2                      ;
06BE  B8 0460                  MOV     AX,MAX                  ; IF EXCEEDS, AX = 1120
06C1  50                     E2:     PUSH    AX
06C2  E8 0E88 R                CALL    ESCL                    ; CALL ESC-L OUT ROUTINE
06C5  59                       POP     CX
06C6  33 F6                    XOR     SI,SI                   ; CLEAR SI
06C8  46                       INC     SI                      ; ADJUST SI
06C9  8A 84 00A0 R           E3:     MOV     AL,IMAGE_BUFFER[SI]     ; GET PRINT DATA FROM BUFFER
```

```
06CD  E8 0F0A R                 CALL    FIRE
06D0  C6 84 00A0 R 00           MOV     IMAGE_BUFFER[SI],0      ; CLEAR IMAGE BUFFER
06D5  F6 C4 08                  TEST    AH,PR_ERROR             ; ERROR HAS OCCURED ?
06D8  75 03                     JNZ     E4
06DA  46                        INC     SI
06DB  E2 EC                     LOOP    E3
06DD  80 26 0009 R FD   E4:     AND     PRINT_MODE,0FFH-LOW_PR_FLG      ; RESET GRAPHIC MODE IND.
06E2  C7 06 0010 R 0000         MOV     CSP,0                   ; CLEAR CSP
06E8  C7 06 0012 R 0000         MOV     CSPMAX,0                ; CLEAR CSPMAX
06EE  C3                        RET
06EF                    LOW_PRT ENDP
                        ;-- X6_EM ---------------------------------------------------------------;
                        ; EMULATES ESC-X6 (CR TO SPECIFIED DOT POSITION).                        ;
                        ;                                                                        ;
                        ;------------------------------------------------------------------------;
06EF                    X6_EM   PROC    NEAR
06EF  8B 0E 0024 R              MOV     CX,N1N2                 ; GET N1N2 VALUE TO CX
06F3  83 F9 00                  CMP     CX,0                    ; CHECK ZERO
06F6  76 35                     JBE     J3
06F8  81 F9 0460                CMP     CX,MAX                  ; CHECK EXCEEEDS MAX
06FC  73 2F                     JAE     J3
06FE  51                        PUSH    CX
06FF  8B 0E 0010 R              MOV     CX,CSP
0703  3B 0E 0012 R              CMP     CX,CSPMAX
0707  72 04                     JB      J1
0709  89 0E 0012 R              MOV     CSPMAX,CX
070D  E8 0515 R         J1:     CALL    MOD_C2                  ; CALL CHECK CURRENT MODE
0710  C6 06 0009 R 02           MOV     PRINT_MODE,LOW_PR_FLG   ; FORCE INTO GRAPHIC MODE
0715  B0 0D                     MOV     AL,0DH                  ; "CR"
0717  E8 0F0A R                 CALL    FIRE
071A  59                        POP     CX
071B  8B C1                     MOV     AX,CX                   ; GET N1N2 VALUE
071D  E8 0E88 R                 CALL    ESCL                    ; CALL ESC-L OUT ROUTINE
0720  B0 00             J2:     MOV     AL,0                    ; ZERO OUT TO PRINTER
0722  E8 0F0A R                 CALL    FIRE                    ;   FOR MAKING BLANK
0725  E2 F9                     LOOP    J2
0727  A1 0024 R                 MOV     AX,N1N2                 ; SAVE NEW CURRENT SLICE POSITION
072A  A3 0010 R                 MOV     CSP,AX
072D  80 26 002C R FC   J3:     AND     FLG1,0FFH-ESC_FLG-X_FLG ; RESET "ESC" & "X" INDICATOR
0732  80 26 002D R EF           AND     FLG2,0FFH-X6_FLG        ; RESET "X6" INDICATOR
0737  C3                        RET
0738                    X6_EM   ENDP
                        ;-- X5_EM ---------------------------------------------------------------;
                        ; EMULATES ESC-X5 (FORWARD VERTICAL STEP FEED)                           ;
                        ;                                                                        ;
                        ;------------------------------------------------------------------------;
0738                    X5_EM   PROC    NEAR
0738  83 3E 0024 R 00           CMP     N1N2,0                  ; CHECK N1N2 = ZERO
073D  76 44                     JBE     L4
073F  81 3E 0024 R 0100         CMP     N1N2,100H               ; CHECK EXCEEDS MAX
0745  77 3C                     JA      L4
0747  8B 0E 0010 R              MOV     CX,CSP
074B  51                        PUSH    CX
074C  E8 047D R                 CALL    PRINT2                  ; CALL SECONDARY PRINT
074F  8B 0E 0024 R              MOV     CX,N1N2                 ;
0753  33 D2                     XOR     DX,DX                   ;++++++++++++++++++++++++++++++++
0755  8B C1                     MOV     AX,CX                   ;+  THIS SEQUENCE CALCULATES   +
0757  B9 000D                   MOV     CX,13                   ;+    BY FOLLOWING FORMULA:    +
075A  F7 F1                     DIV     CX                      ;+                             +
075C  8B C8                     MOV     CX,AX                   ;+  CX = 1/13 * CX             +
075E  83 F9 00                  CMP     CX,0                    ;++++++++++++++++++++++++++++++++
0761  75 01                     JNE     L1
0763  41                        INC     CX
0764  B0 1B             L1:     MOV     AL,1BH                  ; "ESC"
0766  E8 0F0A R                 CALL    FIRE
0769  B0 30                     MOV     AL,30H                  ; "0"
076B  E8 0F0A R                 CALL    FIRE
076E  B0 0A             L2:     MOV     AL,0AH                  ; "LF" OUT
0770  83 F9 00                  CMP     CX,0
0773  74 06                     JE      L3
0775  E8 0F0A R                 CALL    FIRE
0778  49                        DEC     CX
0779  EB F3                     JMP     L2
077B  59                L3:     POP     CX
077C  89 0E 0024 R              MOV     N1N2,CX
0780  E8 06EF R                 CALL    X6_EM
0783  80 26 002C R FC   L4:     AND     FLG1,0FFH-ESC_FLG-X_FLG ; RESET "ESC" & "X" INDICATOR
0788  80 26 002D R F7           AND     FLG2,0FFH-X5_FLG        ; RESET "X5" INDICATOR
078D  C3                        RET
078E                    X5_EM   ENDP
                        ;-- X9_EM ---------------------------------------------------------------;
                        ; EMULATES ESC-X9 (LINE FEED PITCH).                                     ;
                        ;       FOLLOWING CONVERSION IS PROCEEDED:                               ;
                        ;          0  =< N1N2 < 13  ..... LINE FEED COUNT = 1  x 1/9 INCH        ;
                        ;         13  =< N1N2 < 26  ..... LINE FEED COUNT = 2  x 1/9 INCH        ;
                        ;         26  =< N1N2 < 39  ..... LINE FEED COUNT = 3  x 1/9 INCH        ;
                        ;         39  =< N1N2 < 52  ..... LINE FEED COUNT = 4  x 1/9 INCH        ;
                        ;         52  =< N1N2 < 65  ..... LINE FEED COUNT = 5  x 1/9 INCH        ;
                        ;         65  =< N1N2 < 78  ..... LINE FEED COUNT = 6  x 1/9 INCH        ;
                        ;         78  =< N1N2 < 91  ..... LINE FEED COUNT = 7  x 1/9 INCH        ;
                        ;         91  =< N1N2 < 104 ..... LINE FEED COUNT = 8  x 1/9 INCH        ;
                        ;        104 =< N1N2        ..... LINE FEED COUNT = 9  x 1/9 INCH        ;
                        ;------------------------------------------------------------------------;
078E                    X9_EM   PROC    NEAR
078E  53                        PUSH    BX
078F  51                        PUSH    CX
0790  A1 0024 R                 MOV     AX,N1N2                 ; GET N1N2 VALUE TO AX
0793  33 D2                     XOR     DX,DX
0795  B9 000D                   MOV     CX,13
0798  F7 F1                     DIV     CX
079A  8B C8                     MOV     CX,AX
079C  41                        INC     CX
079D  83 F9 09                  CMP     CX,9
```

```
07A0  76 03                        JBE     K1
07A2  B9 0009                      MOV     CX,9
07A5  88 0E 0026 R           K1:   MOV     LF_CT,CL
07A9  B0 1B                        MOV     AL,1BH
07AB  E8 0F0A R                    CALL    FIRE
07AE  B0 30                        MOV     AL,30H
07B0  E8 0F0A R                    CALL    FIRE
07B3  80 26 002C R FC              AND     FLG1,0FFH-ESC_FLG-X_FLG ; RESET "ESC" & "X" INDICATOR
07B8  80 26 002D R DF              AND     FLG2,0FFH-X9_FLG        ; RESET "X9" INDICATOR
07BD  59                           POP     CX
07BE  5B                           POP     BX
07BF  C3                           RET
07C0                         X9_EM ENDP
                             ;-- F_EM ----------------------------------------------------------------
                             ; EMULATES ESC-F (SET PAGE LENGTH PER LINE).                             ;
                             ;                                                                       ;
                             ;      ACTUAL LENGTH:    0 < N1N2 < 64                                  ;
                             ;                                                                       ;
                             ;-------------------------------------------------------------------------
07C0                         F_EM  PROC    NEAR
07C0  8B 0E 0024 R                 MOV     CX,N1N2                 ; GET N1N2 VALUE
07C4  83 F9 00                     CMP     CX,0                    ; CHECK ZERO
07C7  76 2B                        JBE     M3
07C9  83 F9 7E               M1:   CMP     CX,7EH                  ; CHECK VALUE >= 126
07CC  76 03                        JBE     M2
07CE  B9 007E                      MOV     CX,7EH                  ; IF EXCEEDS 126, FORCE CX TO 126
07D1  E8 0F0A R                    CALL    FIRE                    ;  ACTIVATE CORRECT COUNT OF LINE
07D4  B0 32                        MOV     AL,32H                  ;   NUMBER IN A PAGE
07D6  E8 0F0A R                    CALL    FIRE
07D9  B0 1B                        MOV     AL,1BH                  ; "ESC"
07DB  E8 0F0A R                    CALL    FIRE
07DE  B0 43                        MOV     AL,43H                  ; "C"
07E0  E8 0F0A R                    CALL    FIRE
07E3  8A C1                        MOV     AL,CL                   ; N1 INTO AL REG
07E5  E8 0F0A R                    CALL    FIRE
07E8  B0 1B                        MOV     AL,1BH                  ; ESC + 0 OUT TO PRINTER FOR
07EA  E8 0F0A R                    CALL    FIRE                    ;  LINE FEED PITCH TO ORDINARY
07ED  B0 30                        MOV     AL,30H                  ;   USING VALUE
07EF  E8 0F0A R                    CALL    FIRE
07F2  80 26 002C R FA        M3:   AND     FLG1,0FFH-ESC_FLG-F_FLG ; RESET "ESC" & "F" INDICATOR
07F7  C3                           RET
07F8                         F_EM  ENDP
                             ;-- CHAR0 ---------------------------------------------------------------
                             ; PROVIDES CODE TRANSMISSION TO PRINTER TYPE-II.                         ;
                             ;  IF CHARACTER PRINT IS REQUIRED, NORMAL SIZE CHARACTER IS PRINTED.     ;
                             ;                                                                       ;
                             ;-------------------------------------------------------------------------
07F8                         CHAR0 PROC    NEAR
07F8  80 3E 000D R 00              CMP     SIZE_AH,NOR             ; CHECK CURRENT CHAR. SIZE
07FD  74 11                        JE      CH0_1                   ; IF ALREADY NORMAL, JUMP
07FF  50                           PUSH    AX                      ; SET NORMAL SIZE MODE
0800  B0 1B                        MOV     AL,1BH
0802  E8 0F0A R                    CALL    FIRE
0805  B0 5D                        MOV     AL,5DH
0807  E8 0F0A R                    CALL    FIRE
080A  C6 06 000D R 00              MOV     SIZE_AH,NOR
080F  58                           POP     AX
0810  E8 0F0A R              CH0_1: CALL    FIRE                    ; DATA TO PRINTER
0813  C3                           RET
0814                         CHAR0 ENDP
                             ;-- CHAR5 ---------------------------------------------------------------
                             ; PROVIDES CODE TRANSMISSION TO PRINTER TYPE-II.                         ;
                             ;  IF CHARACTER PRINT IS REQUIRED, DOUBLE SIZE CHARACTER IS PRINTED     ;
                             ;                                                                      ;
                             ;-------------------------------------------------------------------------
0814                         CHAR5 PROC    NEAR
0814  80 3E 000D R 01              CMP     SIZE_AH,BAI             ; CHECK CURRENT CHAR. SIZE
0819  74 11                        JE      CH5_1                   ; IF ALREADY DOUBLE, JUMP
081B  50                           PUSH    AX                      ; SET DOUBLE SIZE MODE
081C  B0 1B                        MOV     AL,1BH
081E  E8 0F0A R                    CALL    FIRE
0821  B0 5B                        MOV     AL,5BH
0823  E8 0F0A R                    CALL    FIRE
0826  C6 06 000D R 01              MOV     SIZE_AH,BAI
082B  58                           POP     AX
082C  E8 0F0A R              CH5_1: CALL    FIRE                    ; DATA TO PRINTER
082F  C3                           RET
0830                         CHAR5 ENDP
                             ;-- INIT1 ---------------------------------------------------------------
                             ; RECOGNIZES PRINTER TYPE AND INITIALIZE IT.                             ;
                             ;  CONTROL PARAMETERS ARE SET TO DEFAULT VALUES.                        ;
                             ;                                                                       ;
                             ;-------------------------------------------------------------------------
0830                         INIT1 PROC    NEAR
0830  53                           PUSH    BX
0831  51                           PUSH    CX
0832  52                           PUSH    DX
0833  56                           PUSH    SI
0834  FF 36 0014 R                 PUSH    SPSAVE                  ; CLEAR CONTROL AREA
0838  8D 36 0000 R                 LEA     SI,PRINTER_ID
083C  B9 0060                      MOV     CX,96
083F  E8 0E6D R                    CALL    CLEAR
0842  8F 06 0014 R                 POP     SPSAVE
0846  8D 36 00A0 R                 LEA     SI,IMAGE_BUFFER         ; CLEAR IMAGE_BUFFER
084A  B9 0474                      MOV     CX,1140
084D  E8 0E6D R                    CALL    CLEAR
0850  BA 037A                      MOV     DX,PR_CMD_PORT          ; SET INIT LINE LOW
0853  B0 08                        MOV     AL,08H
0855  EE                          OUT     DX,AL
0856  E8 0896 R                    CALL    DELAY
0859  72 31                        JC      R9                      ; IF NOT ATTACHED, JUMP
085B  24 78                        AND     AL,78H
085D  3C 78                        CMP     AL,78H
085F  74 2B                        JE      R9                      ; IF NOT ATTACHED, JUMP
```

```
0863  BA 037A                   MOV     DX,PR_CMD_PORT          ; SET INIT LINE HIGH
0866  B0 0C                     MOV     AL,0CH
0868  EE                        OUT     DX,AL
0869  E8 0896 R                 CALL    DELAY
086C  32 E0                     XOR     AH,AL                   ; SET PRINTER ID
086E  74 1E                     JZ      R9                      ;
0870  D0 E8                     SHR     AL,1
0872  D0 E8                     SHR     AL,1
0874  25 0001                   AND     AX,01H
0877  35 0001                   XOR     AX,01H
087A  40                        INC     AX
087B  A3 0000 R                 MOV     PRINTER_ID,AX           ;
087E  E8 08B9 R           R3:   CALL    DEFAULT                 ; SET DEFAULT VALUE
0881  B3 00                     MOV     BL,0                    ; CALL STATUS READ ROUTINE
0883  E8 0F54 R                 CALL    STATUS                  ; GET LAST STATUS
0886  E8 0A34 R                 CALL    GETVAL                  ; RESET CPI , LPI CONTROL VALUE
0889  5E                        POP     SI
088A  5A                        POP     DX
088B  59                        POP     CX
088C  5B                        POP     BX
088D  C3                        RET
088E  BA 037A             R9:   MOV     DX,PR_CMD_PORT          ; NOT ATTACH
0891  B0 0C                     MOV     AL,0CH
0893  EE                        OUT     DX,AL
0894  EB E8                     JMP     R3
0896                      INIT1 ENDP
                          ;-- DELAY ------------------------------------------------------------;
                          ; WAIT FOR INITIALIZE                                                 ;
                          ;                                                                     ;
                          ;       CARRY : PRINTER STATUS (0:ATTACHED /1:NOT ATTACHED)           ;
                          ;                                                                     ;
                          ;---------------------------------------------------------------------;
0896                      DELAY   PROC    NEAR
0896  51                        PUSH    CX
0897  53                        PUSH    BX
0898  33 C9                     XOR     CX,CX
089A  E2 FE               DEL00:  LOOP    DEL00
089C  BA 0379                     MOV     DX,PR_STATUS_PORT       ; GET STATUS
089F  EC                          IN      AL,DX
08A0  8A D8                       MOV     BL,AL
08A2  80 E3 FC                    AND     BL,0FCH
08A5  B9 0032                     MOV     CX,50                   ; GET STATUS
08A8  EC                  DEL10:  IN      AL,DX
08A9  24 FC                       AND     AL,0FCH
08AB  3A C3                       CMP     AL,BL
08AD  75 06                       JNE     DEL50
08AF  E2 F7                       LOOP    DEL10
08B1  5B                          POP     BX                      ; PRINTER IS ATTACHED
08B2  59                          POP     CX
08B3  F8                          CLC
08B4  C3                          RET
08B5  5B                  DEL50:  POP     BX                      ; PRINTER ISN'T ATTACHED
08B6  59                          POP     CX
08B7  F9                          STC
08B8  C3                          RET
08B9                      DELAY   ENDP
                          ;-- DEFAULT ---------------------------------------------------------;
                          ; SETS PRINTER CONTROL PARAMETERS TO DEFAULT VALUES.                   ;
                          ;                                                                     ;
                          ;       DEFAULT PARAMETERS :  5/10 CPI (PT-1)                         ;
                          ;                             6/12 CPI (PT-2)                         ;
                          ;                             4.5  LPI                                ;
                          ;                             50   LPP                                ;
                          ;                             SMALL TYPE                              ;
                          ;                                                                     ;
                          ;---------------------------------------------------------------------;
08B9                      DEFAULT PROC    NEAR
08B9  C6 06 0003 R 90             MOV     CPI,90H                 ; CPI DEFAULT SET
08BE  83 3E 0000 R 01             CMP     PRINTER_ID,1
08C3  75 05                       JNE     DEF10
08C5  C6 06 0003 R 78             MOV     CPI,78H                 ; CPI DEFAULT SET
08CA  C6 06 0004 R 14     DEF10:  MOV     LPI,14H                 ; LPI DEFAULT SET
08CF  C7 06 0007 R 0042           MOV     LPP,42H                 ; LPP DEFAULT SET
08D5  C6 06 0026 R 01             MOV     LF_CT,1                 ; SET DEFAULT LINE FEED COUNT ( =1 )
08DA  C7 06 000A R 0002           MOV     CHAR_TYPE,2             ; SET SMALL TYPE CHAR MODE
08E0  C3                  DEF99:  RET
08E1                      DEFAULT ENDP
                          ;-- RESET -----------------------------------------------------------;
                          ; RESETS BIOS CONTROL PARAMETERS AND CLEARS BUFFER                     ;
                          ;                                                                     ;
                          ;---------------------------------------------------------------------;
08E1                      RESET   PROC    NEAR
08E1  50                          PUSH    AX
08E2  33 C0                       XOR     AX,AX                   ; CLEAR AX
08E4  A3 000E R                   MOV     CCP,AX                  ; CLEAR CCP
08E7  A3 0010 R                   MOV     CSP,AX                  ; CLEAR CSP
08EA  A3 0020 R                   MOV     CODEN,AX                ; CLEAR CODE COUNTER
08ED  A2 002C R                   MOV     FLG1,AL                 ; CLEAR FLG1
08F0  A2 002D R                   MOV     FLG2,AL                 ; CLEAR FLG2
08F3  A2 002E R                   MOV     FLG3,AL                 ; CLEAR FLG3
08F6  58                          POP     AX
08F7  C3                          RET
08F8                      RESET   ENDP
                          ;-- CHG-PT1 ---------------------------------------------------------;
                          ; SETS PRINTER CONTROL PARAMETERS FOR PRINTER-1.                       ;
                          ;                                                                     ;
                          ;  CONTROLLED PARAMETERS ARE AS FOLLOWS:                              ;
                          ;                                                                     ;
                          ;               AL = 0 : DEFAULT SET                                  ;
                          ;               AL = 1 : CHANGE CPI  (CHARACTER / INCH)               ;
                          ;               AL = 2 : CHANGE LPI  (LINE / INCH)                    ;
                          ;               AL = 3 : CHANGE LPP  (LINE / PAGE)                    ;
                          ;               AL = 6 : CHANGE TYPE (LARGE / SMALL)                  ;
```

```
08F8                    ;
                        ;-------------------------------------------------------------------;
                        CHG_PT1 PROC    NEAR
08F8  50                        PUSH    AX                      ; SAVE AX
08F9  53                        PUSH    BX                      ; SAVE BX
08FA  0A C0                     OR      AL,AL
08FC  74 13                     JZ      P0                      ; DEFAULT VALUE SET
08FE  FE C8                     DEC     AL
0900  74 15                     JZ      P1                      ; CHANGE CPI
0902  FE C8                     DEC     AL
0904  74 29                     JZ      P2                      ; CHANGE LPI
0906  FE C8                     DEC     AL
0908  74 50                     JZ      P3                      ; CHANGE LPP
090A  3C 03                     CMP     AL,3
090C  74 56                     JZ      P6                      ; CHANGE TYPE
090E  EB 63 90                  JMP     P99
0911  E8 08B9 R         P0:     CALL    DEFAULT                 ; CALL DEFAULT SET ROUTINE
0914  EB 5D 90                  JMP     P99
0917  80 FF 60          P1:     CMP     BH,60H                  ; 7.5/15 CPI
091A  74 0C                     JE      P11
091C  80 FF 6C                  CMP     BH,6CH                  ; 6.7/13.4 CPI
091F  74 07                     JE      P11
0921  80 FF 78                  CMP     BH,78H                  ; 6/12 CPI
0924  74 02                     JE      P11
0926  B7 90                     MOV     BH,90H                  ; IF OTHER, FORCE 5/10 CPI
0928  88 3E 0003 R      P11:    MOV     CPI,BH                  ; STORE TO DATA AREA
092C  EB 45 90                  JMP     P99
092F  C6 06 0026 R 01   P2:     MOV     LF_CT,1                 ;  7.5 LPI CONVERT TO 4.5 LPI
0934  80 FF 10                  CMP     BH,10H
0937  74 10                     JE      P21
0939  FE 06 0026 R              INC     LF_CT                   ;    6 LPI CONVERT TO   3 LPI
093D  80 FF 14                  CMP     BH,14H
0940  74 07                     JE      P21
0942  80 FF 18                  CMP     BH,18H                  ;    5 LPI CONVERT TO   3 LPI
0945  74 02                     JE      P21
0947  B7 1E                     MOV     BH,1EH                  ; IF OTHER, FORCE  3 LPI
0949  88 3E 0004 R      P21:    MOV     LPI,BH
094D  B0 1B                     MOV     AL,1BH                  ; "ESC"
094F  E8 0F0A R                 CALL    FIRE
0952  B0 30                     MOV     AL,30H                  ; "0"
0954  E8 0F0A R                 CALL    FIRE
0957  EB 1A 90                  JMP     P99
095A  89 1E 0024 R      P3:     MOV     N1N2,BX                 ; SET N1N2 VALUE FROM BX
095E  E8 07C0 R                 CALL    F_EM                    ; CALL PAGE LENGTH SET ROUTINE
0961  EB 10 90                  JMP     P99
0964  80 FF 01          P6:     CMP     BH,1                    ; SET CHAR TYPE
0967  77 0A                     JA      P99
0969  FE C7                     INC     BH
096B  8A DF                     MOV     BL,BH
096D  32 FF                     XOR     BH,BH
096F  89 1E 000A R              MOV     CHAR_TYPE,BX
0973  E8 0A34 R         P99:    CALL    GETVAL                  ; RESET CPI , LPI CONTROL VALUE
0976  5B                        POP     BX
0977  58                        POP     AX
0978  C3                        RET
0979                    CHG_PT1 ENDP
                        ;--- CHG_PT2 -----------------------------------------------------;
                        ; SETS PRINTER CONTROL PARAMETERS FOR PRINTER-2.                   ;
                        ;                                                                 ;
                        ;  CONTROLLED PARAMETERS ARE AS FOLLOWS:                          ;
                        ;                                                                 ;
                        ;               AL = 0 : DEFAULT SET                              ;
                        ;               AL = 1 : CHANGE CPI (CHARACTER / INCH)            ;
                        ;               AL = 2 : CHANGE LPI (LINE / INCH)                 ;
                        ;               AL = 3 : CHANGE LPP (LINE / PAGE)                 ;
                        ;               AL = 5 : CHANGE DIRECTION                         ;
                        ;-----------------------------------------------------------------;
0979                    CHG_PT2 PROC    NEAR
0979  0A C0                    OR      AL,AL
097B  74 15                     JZ      XG1                     ; AL=0 ( DEFAULT SET )
097D  FE C8                     DEC     AL
097F  74 17                     JZ      XG2                     ; AL=1 ( CPI CHANGE )
0981  FE C8                     DEC     AL
0983  74 19                     JZ      XG3                     ; AL=2 ( LPI CHANGE )
0985  FE C8                     DEC     AL
0987  74 1B                     JZ      XG4                     ; AL=3 ( LPP CHANGE )
0989  FE C8                     DEC     AL
098B  FE C8                     DEC     AL
098D  74 1B                     JZ      XG6                     ; AL=5 ( PRINT DIRECTION CHANGE )
098F  EB 1C 90                  JMP     XRET
0992  E8 08B9 R         XG1:    CALL    DEFAULT                 ; JUMP TO DEFAULT SET ROUTINE
0995  EB 16 90                  JMP     XRET
0998  E8 09B1 R         XG2:    CALL    XCPI                    ; JUMP TO CPI CHANGE ROUTINE
099B  EB 10 90                  JMP     XRET
099E  E8 09F0 R         XG3:    CALL    XLPI                    ; JUMP TO LPI CHANGE ROUTINE
09A1  EB 0A 90                  JMP     XRET
09A4  E8 0A0E R         XG4:    CALL    XLPP                    ; JUMP TO LPP SET ROUTINE
09A7  EB 04 90                  JMP     XRET
09AA  E8 0A27 R         XG6:    CALL    XDIR                    ; JUMP TO DIRECTION SET ROUTINE
09AD  E8 0A34 R         XRET:   CALL    GETVAL                  ; RESET CPI , LPI CONTROL VALUE
09B0  C3                        RET
09B1                    CHG_PT2 ENDP
                        ;--- XCPI --------------------------------------------------------;
                        ; SETS CPI VALUE TO PRINTER-2.                                    ;
                        ;                                                                 ;
                        ;-----------------------------------------------------------------;
09B1                    XCPI    PROC    NEAR
09B1  B3 4B                     MOV     BL,75
09B3  80 FF 60                  CMP     BH,60H                  ; CHECK BH INDICATES 7.5/15 CPI
09B6  74 15                     JE      XCPF
09B8  B3 43                     MOV     BL,67
09BA  80 FF 6C                  CMP     BH,6CH                  ; CHECK BH INDICATES 6.7/13.4 CPI
```

```
09BD  74 0E                        JE      XCPF
09BF  B3 3C                        MOV     BL,60
09C1  80 FF 78                     CMP     BH,78H                  ; CHECK BH INDICATES 6/12 CPI
09C4  74 07                        JE      XCPF
09C6  B3 32                        MOV     BL,50
09C8  80 FF 90                     CMP     BH,90H                  ; CHECK BH INDICATES 5/10 CPI
09CB  75 22                        JNE     XCP9
09CD  88 3E 0003 R          XCPF:  MOV     CPI,BH                  ; BH VALUE IS STORED
09D1  B0 1B                        MOV     AL,1BH                  ; ESC + 7E + 02 + 00 + 01 + M
09D3  E8 0F0A R                    CALL    FIRE
09D6  B0 7E                        MOV     AL,7EH
09D8  E8 0F0A R                    CALL    FIRE
09DB  B0 02                        MOV     AL,02H
09DD  E8 0F0A R                    CALL    FIRE
09E0  B0 00                        MOV     AL,00H
09E2  E8 0F0A R                    CALL    FIRE
09E5  B0 01                        MOV     AL,01H
09E7  E8 0F0A R                    CALL    FIRE
09EA  8A C3                        MOV     AL,BL
09EC  E8 0F0A R                    CALL    FIRE
09EF  C3                    XCP9:  RET
09F0                        XCPI   ENDP
                            ;-- XLPI ----------------------------------------------------------------;
                            ; SETS LPI VALUE TO PRINTER-2.                                          ;
                            ;                                                                       ;
                            ;-----------------------------------------------------------------------;
09F0                        XLPI   PROC    NEAR
09F0  88 3E 0004 R                 MOV     LPI,BH                  ; BH VALUE IS STORED
09F4  B0 1B                        MOV     AL,1BH                  ; ESC + X + 9 + N1N2
09F6  E8 0F0A R                    CALL    FIRE
09F9  B0 25                        MOV     AL,25H
09FB  E8 0F0A R                    CALL    FIRE
09FE  B0 39                        MOV     AL,39H
0A00  E8 0F0A R                    CALL    FIRE
0A03  B0 00                        MOV     AL,00H
0A05  E8 0F0A R                    CALL    FIRE
0A08  8A C7                        MOV     AL,BH
0A0A  E8 0F0A R                    CALL    FIRE
0A0D  C3                           RET
0A0E                        XLPI   ENDP
                            ;-- XLPP ----------------------------------------------------------------;
                            ; SETS LPP VALUE TO PRINTER-2.                                          ;
                            ;                                                                       ;
                            ;-----------------------------------------------------------------------;
0A0E                        XLPP   PROC    NEAR
0A0E  89 1E 0007 R                 MOV     LPP,BX                  ; BX VALUE IS STORED
0A12  B0 1B                        MOV     AL,1BH                  ; ESC + F + N1 + N2
0A14  E8 0F0A R                    CALL    FIRE
0A17  B0 46                        MOV     AL,46H
0A19  E8 0F0A R                    CALL    FIRE
0A1C  8A C7                        MOV     AL,BH
0A1E  E8 0F0A R                    CALL    FIRE
0A21  8A C3                        MOV     AL,BL
0A23  E8 0F0A R                    CALL    FIRE
0A26  C3                           RET
0A27                        XLPP   ENDP
                            ;-- XDIR ----------------------------------------------------------------;
                            ; SETS DIRECTION MODE TO PRINTER-2.                                     ;
                            ;                                                                       ;
                            ;-----------------------------------------------------------------------;
0A27                        XDIR   PROC    NEAR
0A27  B0 42                        MOV     AL,42H
0A29  80 FF 01                     CMP     BH,1
0A2C  74 02                        JE      XDIR1
0A2E  B0 55                        MOV     AL,55H
0A30  E8 0ECE R             XDIR1: CALL    DIRSET
0A33  C3                           RET
0A34                        XDIR   ENDP
                            ;-- GETVAL --------------------------------------------------------------;
                            ; GETS GRID LINE AND CHAR TYPE CONTROL VALUE                            ;
                            ;                                                                       ;
                            ;-----------------------------------------------------------------------;
0A34                        GETVAL PROC    NEAR
0A34  06                           PUSH    ES
0A35  51                           PUSH    CX
0A36  56                           PUSH    SI
0A37  57                           PUSH    DI
0A38  50                           PUSH    AX
0A39  1E                           PUSH    DS                      ; ES SET FOR MOVSB
0A3A  07                           POP     ES
0A3B  8D 36 0000 R                 LEA     SI,V_VALUE              ; SEARCH V_VALUE TABLE
0A3F  8A 2E 0004 R          GET00: MOV     CH,LPI
0A43  2E: 3A 2C                    CMP     CH,CS:[SI]
0A46  74 05                        JE      GET10
0A48  83 C6 12                     ADD     SI,18
0A4B  EB F2                        JMP     GET00
0A4D  B9 0012               GET10: MOV     CX,18                   ; MOVE TO VALUE AREA
0A50  8D 3E 0040 R                 LEA     DI,GVALUE
0A54  1E                           PUSH    DS
0A55  0E                           PUSH    CS
0A56  1F                           POP     DS
0A57  FC                           CLD
0A58  F3/ A4                       REP     MOVSB
0A5A  1F                           POP     DS
0A5B  8D 36 0048 R                 LEA     SI,H_VALUE              ; SEARCH V_VALUE TABLE
0A5F  8A 2E 0003 R          GET20: MOV     CH,CPI
0A63  2E: 3A 2C                    CMP     CH,CS:[SI]
0A66  74 05                        JE      GET30
0A68  83 C6 06                     ADD     SI,6
0A6B  EB F2                        JMP     GET20
0A6D  B9 0006               GET30: MOV     CX,6                    ; MOVE TO VALUE AREA
0A70  8D 3E 0052 R                 LEA     DI,GVALUE+18
0A74  1E                           PUSH    DS
0A75  0E                           PUSH    CS
```

```
0A76  1F                          POP     DS
0A77  FC                          CLD
0A78  F3/ A4                      REP     MOVSB
0A7A  1F                          POP     DS
0A7B  C7 06 0058 R 0000           MOV     TVALUE+UPPER,0          ; SET CHAR TYPE CONTROL VALUE
0A81  C7 06 005A R 0001           MOV     TVALUE+LOWER,1
0A87  83 3E 000A R 02             CMP     CHAR_TYPE,2
0A8C  74 0C                       JE      GET40
0A8E  C7 06 0058 R 0000           MOV     TVALUE+UPPER,0
0A94  C7 06 005A R 0008           MOV     TVALUE+LOWER,8
0A9A  CD 11                 GET40:INT     11H                     ; SET SYSTEM IDENT
0A9C  24 20                       AND     AL,20H
0A9E  A2 001F R                   MOV     SYSTEM_ID,AL
0AA1  58                          POP     AX
0AA2  5F                          POP     DI
0AA3  5E                          POP     SI
0AA4  59                          POP     CX
0AA5  07                          POP     ES
0AA6  C3                          RET
0AA7                        GETVAL  ENDP
                            ;-- PRINT -----------------------------------------------------------;
                            ; PRINTS CODES IN CODE BUFFER.                                       ;
                            ;       INPUT                                                        ;
                            ;         CODE_BUFFER : CHARACTER STRING                             ;
                            ;         ATTR_BUFFER : CHARACTER SIZE                               ;
                            ;         CCP         : CHARACTER LENGTH                             ;
                            ;                                                                    ;
                            ;--------------------------------------------------------------------;
0AA7                        PRINT   PROC    NEAR
0AA7  1E                          PUSH    DS                      ; SET ES
0AA8  07                          POP     ES
0AA9  8D 1E 00A0 R                LEA     BX,CODE_BUFFER          ; PRINT
0AAD  C7 06 002F R 0001           MOV     CSIZE,0001H
0AB3  E8 0B13 R                   CALL    IMAGE
0AB6  C3                          RET
0AB7                        PRINT   ENDP
                            ;-- ATTR ------------------------------------------------------------;
                            ; PRINTS CODES IN BUFFER WITH ATTRIBUTE.                             ;
                            ;       INPUT                                                        ;
                            ;         ES:BX : CHARACTER STRING                                   ;
                            ;         ES:DI : ATTRIBUTE STRING                                   ;
                            ;         CX    : CHARACTER LENGTH                                   ;
                            ;                                                                    ;
                            ;--------------------------------------------------------------------;
0AB7                        ATTR    PROC    NEAR
0AB7  0B C9                       OR      CX,CX                   ; LAST CODE CHECK
0AB9  74 1D                       JZ      AT99
0ABB  8B F1                       MOV     SI,CX
0ABD  26: 80 78 FF 0D             CMP     BYTE    PTR     ES:[BX+SI-1],0DH
0AC2  74 11                       JE      AT10
0AC4  26: 80 78 FF 0A             CMP     BYTE    PTR     ES:[BX+SI-1],0AH
0AC9  74 0A                       JE      AT10
0ACB  83 3E 0000 R 01             CMP     PRINTER_ID,1            ; WHICH PRINTER IS ATTACHED ?
0AD0  74 07                       JE      AT100
0AD2  EB 1C 90                    JMP     AT200
0AD5  49                    AT10: DEC     CX
0AD6  EB DF                       JMP     ATTR
0AD8  C3                    AT99: RET
                            ;--------------------------------------------------------------------;
                            ; PRINTER-1 IS ATTACHED                                              ;
                            ;--------------------------------------------------------------------;
0AD9  06                    AT100:PUSH    ES                      ; OUT STRING
0ADA  53                          PUSH    BX
0ADB  57                          PUSH    DI
0ADC  51                          PUSH    CX
0ADD  26: 8A 07                   MOV     AL,ES:[BX]
0AE0  E8 019F R                   CALL    CMD_CHK
0AE3  59                          POP     CX
0AE4  5F                          POP     DI
0AE5  5B                          POP     BX
0AE6  07                          POP     ES
0AE7  43                          INC     BX
0AE8  E2 EF                       LOOP    AT100
0AEA  B0 0A                       MOV     AL,0AH                  ; LINE FEED
0AEC  E8 019F R                   CALL    CMD_CHK
0AEF  C3                          RET
                            ;--------------------------------------------------------------------;
                            ; PRINTER-2 IS ATTACHED                                              ;
                            ;--------------------------------------------------------------------;
0AF0  E8 0CBB R             AT200:CALL    MODESET                 ; 3 BYTE MODE SET
0AF3  E8 0CC6 R                   CALL    UGRID                   ; UPPER GRID LINE
0AF6  E8 0EDE R                   CALL    FEED                    ; FEED
0AF9  E8 0D5A R                   CALL    CODEOUT                 ; OUT CODE
0AFC  E8 0EDE R                   CALL    FEED                    ; FEED
0AFF  E8 0D77 R                   CALL    LGRID                   ; LOWER GRID LINE
0B02  8A 16 0048 R                MOV     DL,GVALUE+V_SIZE        ; FEED
0B06  C6 06 0048 R 10             MOV     GVALUE+V_SIZE,16
0B0B  E8 0EDE R                   CALL    FEED
0B0E  88 16 0048 R                MOV     GVALUE+V_SIZE,DL
0B12  C3                          RET
0B13                        ATTR    ENDP
                            ;-- IMAGE -----------------------------------------------------------;
                            ; PRINTS THE CODE STRING                                             ;
                            ;                                                                    ;
                            ;       INPUT                                                        ;
                            ;         ES:BX     : CHARACTER STRING                               ;
                            ;         ES:BX+240 : CHARACTER SIZE                                 ;
                            ;         CCP       : CHARACTER LENGTH                             ;
                            ;         CSIZE     : ATTRIBUTE VALID FLAG (0:INVALID / 1:VALID)   ;
                            ;                                                                    ;
                            ;--------------------------------------------------------------------;
0B13                        IMAGE   PROC    NEAR
0B13  B0 1B                       MOV     AL,1BH                  ; SET LPI 1/9
```

```
0B15  E8 0F0A R                 CALL    FIRE
0B18  B0 30                     MOV     AL,30H
0B1A  E8 0F0A R                 CALL    FIRE
0B1D  8B 36 0058 R              MOV     SI,TVALUE+UPPER         ; PRINT UPPER IMAGE
0B21  E8 0B75 R                 CALL    ROTATE
0B24  83 3E 005A R 01           CMP     TVALUE+LOWER,1          ; EVEN IMAGE ?
0B29  75 17                     JNE     IM10                    ; NO
0B2B  B0 1B                     MOV     AL,1BH                  ; ESC + J + 1
0B2D  E8 0F0A R                 CALL    FIRE
0B30  B0 4A                     MOV     AL,4AH
0B32  E8 0F0A R                 CALL    FIRE
0B35  B0 01                     MOV     AL,01H
0B37  E8 0F0A R                 CALL    FIRE
0B3A  B0 0D                     MOV     AL,0DH
0B3C  E8 0F0A R                 CALL    FIRE
0B3F  EB 10 90                  JMP     IM20
0B42  B0 1B                 IM10:MOV    AL,1BH                  ; THIS SEQUENCE SETS LINE FEED
0B44  E8 0F0A R                 CALL    FIRE                    ;   PITCH FOR ORDINARY USING
0B47  B0 30                     MOV     AL,30H                  ;     VALUE OF ONE-NINTH-INCH.
0B49  E8 0F0A R                 CALL    FIRE                    ;
0B4C  B0 0A                     MOV     AL,0AH                  ; LINE FEED
0B4E  E8 0F0A R                 CALL    FIRE
0B51  8B 36 005A R          IM20:MOV    SI,TVALUE+LOWER         ; PRINT LOWER IMAGE
0B55  E8 0B75 R                 CALL    ROTATE
0B58  83 3E 000A R 01           CMP     CHAR_TYPE,1             ; CHECK CHARACTER TYPE
0B5D  74 05                     JE      IM99                    ; IF SIZE IS LARGE, JUMP
0B5F  B0 0A                     MOV     AL,0AH                  ; IF SIZE IS SMALL,
0B61  E8 0F0A R                 CALL    FIRE                    ;     LF OUT TO PRINTER
0B64  C7 06 000E R 0000     IM99:MOV    CCP,0                   ; CLEAR CURRENT CHARACTER POSITION
0B6A  B9 0474                   MOV     CX,1140                 ; CLEAR IMAGE BUFFER
0B6D  8D 36 00A0 R              LEA     SI,IMAGE_BUFFER         ;
0B71  E8 0E6D R                 CALL    CLEAR                   ;
0B74  C3                        RET
0B75                        IMAGE   ENDP
                            ;-- ROTATE ------------------------------------------------------------;
                            ; ROTATES FONT IMAGE AND OUT                                           ;
                            ;       INPUT                                                          ;
                            ;         ES:BX     : CODE BUFFER                                      ;
                            ;         ES:BX+240 : CHARACTER SIZE                                   ;
                            ;         CCP       : CODE LENGTH                                      ;
                            ;         CSIZE     : ATTRIBUTE VALID FLAG (0:INVALID / 1:VALID)       ;
                            ;-----------------------------------------------------------------------;
0B75                        ROTATE  PROC    NEAR
0B75  53                        PUSH    BX
0B76  FF 36 000E R              PUSH    CCP
0B7A  A0 0057 R                 MOV     AL,GVALUE+H_SPACE       ; (8 + SPACE) * CCP -> AX
0B7D  04 08                     ADD     AL,8
0B7F  8B 16 000E R              MOV     DX,CCP
0B83  F6 E2                     MUL     DL
0B85  E8 0E88 R                 CALL    ESCL
0B88  83 3E 000E R 00       ROT00:CMP   CCP,0
0B8D  7E 13                     JNG     ROT99
0B8F  26: 8B AF 00F0            MOV     BP,ES:240[BX]           ; MOVE CHARACTER SIZE TO BP
0B94  23 2E 002F R              AND     BP,CSIZE
0B98  E8 0BDA R                 CALL    GETFONT                 ; GET FONT INTO FONT WORK
0B9B  0A F6                     OR      DH,DH                   ; HALF ?
0B9D  74 2C                     JZ      HALF
0B9F  EB 07 90                  JMP     NORMAL
0BA2  8F 06 000E R          ROT99:POP   CCP
0BA6  5B                        POP     BX
0BA7  C3                        RET
                            ;-----------------------------------------------------------------------;
                            ;       NORMAL SIZE PRINT                                               ;
                            ;-----------------------------------------------------------------------;
0BA8  8A 26 0057 R          NORMAL: MOV     AH,GVALUE+H_SPACE       ; SPACING
0BAC  E8 0C2A R                 CALL    SPOUT
0BAF  E8 0C42 R                 CALL    HVCONV                  ; LEFT IMAGE OUT
0BB2  83 3E 000E R 00           CMP     CCP,0
0BB7  74 CF                     JZ      ROT00
0BB9  83 C6 10                  ADD     SI,16                   ; RIGHT IMAGE OUT
0BBC  E8 0C42 R                 CALL    HVCONV
0BBF  83 EE 10                  SUB     SI,16
0BC2  8A 26 0057 R              MOV     AH,GVALUE+H_SPACE       ; SPACING
0BC6  E8 0C2A R                 CALL    SPOUT
0BC9  EB BD                     JMP     ROT00
                            ;-----------------------------------------------------------------------;
                            ;       HALF SIZE PRINT                                                 ;
                            ;-----------------------------------------------------------------------;
0BCB  B4 01                 HALF:   MOV     AH,1                    ; SPACING
0BCD  E8 0C2A R                 CALL    SPOUT
0BD0  E8 0C42 R                 CALL    HVCONV                  ; LEFT IMAGE OUT
0BD3  B4 00                     MOV     AH,0                    ; SPACING
0BD5  E8 0C2A R                 CALL    SPOUT
0BD8  EB AE                     JMP     ROT00
0BDA                        ROTATE  ENDP
                            ;-- GETFONT -----------------------------------------------------------;
                            ; GETS FONT TO FONT WORK                                               ;
                            ;                                                                      ;
                            ;-----------------------------------------------------------------------;
0BDA                        GETFONT PROC    NEAR
0BDA  33 D2                     XOR     DX,DX
0BDC  E8 0C23 R                 CALL    GETCODE                 ; GET THE FIRST CODE TO DL
0BDF  80 FA 80                  CMP     DL,80H                  ; ++++++++++++++++++++++++++++++++
0BE2  76 17                     JBE     ONE                     ; +                            +
0BE4  80 FA 9F                  CMP     DL,9FH                  ; + CHECK LEGALITY OF 1ST CODE OF +
0BE7  76 0D                     JBE     TWO                     ; +                            +
0BE9  80 FA DF                  CMP     DL,0DFH                 ; +         TWO BYTES CODE     +
0BEC  76 0D                     JBE     ONE                     ; +                            +
0BEE  80 FA FC                  CMP     DL,0FCH                 ; +                            +
0BF1  76 03                     JBE     TWO                     ; ++++++++++++++++++++++++++++++++
0BF3  EB 06 90                  JMP     ONE                     ;
0BF6  8A F2                 TWO:MOV     DH,DL                   ; GET THE SECOND CODE TO DL
0BF8  E8 0C23 R                 CALL    GETCODE
```

```
0BFB  53                    ONE:    PUSH    BX                      ; GET FONT IMAGE
0BFC  06                            PUSH    ES
0BFD  56                            PUSH    SI
0BFE  B9 0010                       MOV     CX,16                   ; CLEAR FONT AREA
0C01  8D 36 0280 R                  LEA     SI,FONT
0C05  E8 0E6D R                     CALL    CLEAR
0C08  B4 13                         MOV     AH,19                   ; FONT REQUEST
0C0A  80 3E 001F R 00               CMP     SYSTEM_ID,00H
0C0F  74 02                         JE      FONT10
0C11  B4 10                         MOV     AH,16
0C13  32 C0                 FONT10: XOR     AL,AL
0C15  8D 1E 0280 R                  LEA     BX,FONT
0C19  8B CA                         MOV     CX,DX
0C1B  1E                            PUSH    DS
0C1C  07                            POP     ES
0C1D  CD 10                         INT     10H
0C1F  5E                            POP     SI
0C20  07                            POP     ES
0C21  5B                            POP     BX
0C22  C3                            RET
0C23                        GETFONT ENDP
                            ;-- GETCODE ----------------------------------------------------------------;
                            ; GETS CODE AND UPDATE INDEX                                               ;
                            ;                                                                          ;
                            ;----------------------------------------------------------------------------;
0C23                        GETCODE PROC    NEAR
0C23  26: 8A 17                     MOV     DL,ES:[BX]              ; GET CODE INTO DL
0C26  03 DD                         ADD     BX,BP                   ; INDEX UPDATE IF DOUBLE SIZE
0C28  43                            INC     BX                      ; INDEX UPDATE
0C29  C3                            RET
0C2A                        GETCODE ENDP
                            ;-- SPOUT -----------------------------------------------------------------;
                            ; OUTPUTS SPACES BETWEEN CHARACTERS                                        ;
                            ;                                                                          ;
                            ;       INPUT                                                              ;
                            ;         VALUE+H_SPACE : SPACING VALUE                                    ;
                            ;                                                                          ;
                            ;----------------------------------------------------------------------------;
0C2A                        SPOUT   PROC    NEAR
0C2A  02 26 0057 R                  ADD     AH,GVALUE+H_SPACE       ; SPACING VALUE
0C2E  D0 EC                         SHR     AH,1
0C30  8B CD                         MOV     CX,BP                   ; CHAR SIZE
0C32  D2 E4                         SHL     AH,CL
0C34  8A CC                         MOV     CL,AH
0C36  32 C0                         XOR     AL,AL
0C38  0A C9                         OR      CL,CL
0C3A  74 05                         JZ      SP99
0C3C  E8 0F0A R             SP00:   CALL    FIRE                    ; OUT SPACE
0C3F  E2 FB                         LOOP    SP00
0C41  C3                    SP99:   RET
0C42                        SPOUT   ENDP
                            ;-- HVCONV ----------------------------------------------------------------;
                            ; ROTATES FONT IMAGE AND PRINTS IT                                         ;
                            ;                                                                          ;
                            ;       INPUT                                                              ;
                            ;         SI        : FLAG (0:UPPER/EVEN  1:ODD  8:LOWER)                  ;
                            ;         CHAR_TYPE : TYPE (1:SMALL TYPE  2:LARGE TYPE)                    ;
                            ;         BP        : SIZE (0:NORMAL SIZE 1:DOUBLE SIZE)                   ;
                            ;                                                                          ;
                            ;----------------------------------------------------------------------------;
0C42                        HVCONV  PROC    NEAR
0C42  56                            PUSH    SI
0C43  53                            PUSH    BX
0C44  51                            PUSH    CX
0C45  52                            PUSH    DX
0C46  8B 3E 008A R                  MOV     DI,CHAR_TYPE            ; LOAD FONT TO REGISTER
0C4A  8A 84 0280 R                  MOV     AL,FONT[SI]
0C4E  03 F7                         ADD     SI,DI
0C50  8A 8C 0280 R                  MOV     CL,FONT[SI]
0C54  03 F7                         ADD     SI,DI
0C56  8A 94 0280 R                  MOV     DL,FONT[SI]
0C5A  03 F7                         ADD     SI,DI
0C5C  8A 9C 0280 R                  MOV     BL,FONT[SI]
0C60  03 F7                         ADD     SI,DI
0C62  8A A4 0280 R                  MOV     AH,FONT[SI]
0C66  03 F7                         ADD     SI,DI
0C68  8A AC 0280 R                  MOV     CH,FONT[SI]
0C6C  03 F7                         ADD     SI,DI
0C6E  8A B4 0280 R                  MOV     DH,FONT[SI]
0C72  03 F7                         ADD     SI,DI
0C74  8A BC 0280 R                  MOV     BH,FONT[SI]
0C78  03 F7                         ADD     SI,DI
0C7A  BE 0008                       MOV     SI,8
0C7D  D0 E0                 HV00:   SHL     AL,1                    ; LOAD DATA
0C7F  D1 D7                         RCL     DI,1
0C81  D0 E1                         SHL     CL,1
0C83  D1 D7                         RCL     DI,1
0C85  D0 E2                         SHL     DL,1
0C87  D1 D7                         RCL     DI,1
0C89  D0 E3                         SHL     BL,1
0C8B  D1 D7                         RCL     DI,1
0C8D  D0 E4                         SHL     AH,1
0C8F  D1 D7                         RCL     DI,1
0C91  D0 E5                         SHL     CH,1
0C93  D1 D7                         RCL     DI,1
0C95  D0 E6                         SHL     DH,1
0C97  D1 D7                         RCL     DI,1
0C99  D0 E7                         SHL     BH,1
0C9B  D1 D7                         RCL     DI,1
0C9D  50                            PUSH    AX
0C9E  51                            PUSH    CX                      ; FIRE IMAGE
0C9F  8B CD                         MOV     CX,BP
0CA1  41                            INC     CX                      ; LOOP FOR DOUBLE SIZE
```

```
0CA2  8B C7                   HV10:   MOV     AX,DI
0CA4  E8 0F0A R                       CALL    FIRE
0CA7  E2 F9                           LOOP    HV10
0CA9  59                              POP     CX
0CAA  58                              POP     AX
0CAB  4E                              DEC     SI
0CAC  75 CF                           JNZ     HV00
0CAE  29 2E 000E R                    SUB     CCP,BP                  ; CCP UPDATE
0CB2  FF 0E 000E R                    DEC     CCP
0CB6  5A                              POP     DX
0CB7  59                              POP     CX
0CB8  5B                              POP     BX
0CB9  5E                              POP     SI
0CBA  C3                              RET
0CBB                          HVCONV  ENDP
                              ;-- MODESET ----------------------------------------------------------;
                              ; SETS 3 BYTE TRANS MODE                                              ;
                              ;----------------------------------------------------------------------;
0CBB                          MODESET PROC    NEAR
0CBB  B0 1B                           MOV     AL,1BH                  ; ESC + (
0CBD  E8 0F0A R                       CALL    FIRE
0CC0  B0 28                           MOV     AL,28H
0CC2  E8 0F0A R                       CALL    FIRE
0CC5  C3                              RET
0CC6                          MODESET ENDP
                              ;-- UGRID ----------------------------------------------------------;
                              ; PRINTS UPPER GRID LINE                                             ;
                              ;       INPUT                                                        ;
                              ;         ES:DI : ATTRIBUTE BUFFER                                   ;
                              ;         CX    : ATTRIBUTE LENGTH                                   ;
                              ;----------------------------------------------------------------------;
0CC6                          UGRID   PROC    NEAR
0CC6  50                              PUSH    AX
0CC7  51                              PUSH    CX
0CC8  57                              PUSH    DI
0CC9  53                              PUSH    BX
0CCA  C7 06 001C R 0003               MOV     CPIMASK,3               ; SET MASK FOR 13.5 CPI
0CD0  B0 0F                           MOV     AL,0FH                  ; CHECK UPPER ATTRIBUTE EXISTS
0CD2  8B F7                           MOV     SI,DI
0CD4  E8 0E74 R                       CALL    CHECK
0CD7  72 28                           JC      UG99
0CD9  B0 55                           MOV     AL,55H                  ; UNIDIRECTION SET
0CDB  E8 0ECE R                       CALL    DIRSET
0CDE  E8 0E9F R                       CALL    ESCX1                   ; OUT ESC+X+1+N1+N2
0CE1  51                      UG00:   PUSH    CX                      ; CLEAR IMAGE BUFFER
0CE2  8D 36 02A0 R                    LEA     SI,GRID
0CE6  B9 0036                         MOV     CX,54
0CE9  E8 0E6D R                       CALL    CLEAR
0CEC  59                              POP     CX
0CED  BE 0000                         MOV     SI,0                    ; SET VERTICAL GRID (UPPER)
0CF0  E8 0DD2 R                       CALL    VGRID
0CF3  E8 0D06 R                       CALL    HGRID                   ; SET HORIZONTAL GRID
0CF6  E8 0E42 R                       CALL    PGRID                   ; OUT GRID IMAGE
0CF9  47                              INC     DI
0CFA  E2 E5                           LOOP    UG00
0CFC  B0 42                           MOV     AL,42H                  ; BIDIRECTION SET
0CFE  E8 0ECE R                       CALL    DIRSET
0D01  5B                      UG99:   POP     BX
0D02  5F                              POP     DI
0D03  59                              POP     CX
0D04  58                              POP     AX
0D05  C3                              RET
0D06                          UGRID   ENDP
                              ;-- HGRID ----------------------------------------------------------;
                              ; PRINTS HORIZONTAL GRID LINE                                        ;
                              ;----------------------------------------------------------------------;
0D06                          HGRID   PROC    NEAR
0D06  26: 8B 05                       MOV     AX,ES:[DI]              ; GET ATTRIBUTE BYTE
0D09  25 0003                         AND     AX,0003H                ; NO GRID ?
0D0C  74 09                           JZ      HG99
0D0E  48                              DEC     AX                      ; SINGLE ?
0D0F  74 07                           JZ      HG100
0D11  48                              DEC     AX                      ; DOUBLE ?
0D12  74 11                           JZ      HG200
0D14  EB 1C 90                        JMP     HG300                   ; DASHED
0D17  C3                      HG99:   RET
                              ;----------------------------------------------------------------------;
                              ;       SINGLE                                                         ;
                              ;----------------------------------------------------------------------;
0D18  B6 FF                   HG100:  MOV     DH,0FFH                 ; LOAD LINE MASK
0D1A  B2 FF                           MOV     DL,0FFH
0D1C  B7 FF                           MOV     BH,0FFH
0D1E  B4 80                           MOV     AH,080H                 ; LOAD LINE IMAGE
0D20  E8 0D45 R                       CALL    HGSET                   ; SET GRID IMAGE
0D23  EB F2                           JMP     HG99
                              ;----------------------------------------------------------------------;
                              ;       DOUBLE                                                         ;
                              ;----------------------------------------------------------------------;
0D25  B6 FF                   HG200:  MOV     DH,0FFH                 ; LOAD LINE MASK
0D27  B2 FF                           MOV     DL,0FFH
0D29  B7 FF                           MOV     BH,0FFH
0D2B  B4 C0                           MOV     AH,0C0H                 ; LOAD LINE IMAGE
0D2D  E8 0D45 R                       CALL    HGSET                   ; SET GRID IMAGE
0D30  EB E5                           JMP     HG99
                              ;----------------------------------------------------------------------;
                              ;       DASHED                                                         ;
                              ;----------------------------------------------------------------------;
0D32  8A 36 0053 R            HG300:  MOV     DH,GVALUE+H_DASH        ; LOAD LINE MASK
0D36  8A 16 0054 R                    MOV     DL,GVALUE+H_DASH+1
0D3A  8A 3E 0055 R                    MOV     BH,GVALUE+H_DASH+2
0D3E  B4 80                           MOV     AH,080H                 ; LOAD LINE IMAGE
0D40  E8 0D45 R                       CALL    HGSET                   ; SET GRID IMAGE
0D43  EB D2                           JMP     HG99
```

```
0D45                                HGRID   ENDP
                                    ;-- HGSET ---------------------------------------------------------------;
                                    ; SETS GRID LINE IMAGE                                                   ;
                                    ;       INPUT                                                            ;
                                    ;         DH,DL,BH : GRID LINE MASK                                      ;
                                    ;-----------------------------------------------------------------------;
0D45                                HGSET   PROC    NEAR
0D45  33 F6                         HG500:  XOR     SI,SI
0D47  D0 D7                                 RCL     BH,1                    ; SHIFT LINE MASK
0D49  D1 D2                                 RCL     DX,1
0D4B  73 04                                 JNC     SKIP                    ; SET ?
0D4D  08 A4 02A8 R                          OR      GRID[SI],AH             ; YES
0D51  83 C6 03                      SKIP:   ADD     SI,3
0D54  83 FE 36                              CMP     SI,54
0D57  75 EE                                 JNE     HG500
0D59  C3                                    RET
0D5A                                HGSET   ENDP
                                    ;-- CODEOUT -------------------------------------------------------------;
                                    ; PRINTS CODES                                                           ;
                                    ;-----------------------------------------------------------------------;
0D5A                                CODEOUT PROC    NEAR
0D5A  50                                    PUSH    AX
0D5B  51                                    PUSH    CX
0D5C  56                                    PUSH    SI
0D5D  57                                    PUSH    DI
0D5E  B0 DF                                 MOV     AL,0DFH                 ; CHECK ALL SPACE
0D60  8B F3                                 MOV     SI,BX
0D62  E8 0E74 R                             CALL    CHECK
0D65  72 0B                                 JC      CO99
0D67  8B F3                                 MOV     SI,BX                   ; CODE OUT
0D69  26: 8A 04                     CODE00: MOV     AL,ES:[SI]
0D6C  E8 07FA R                             CALL    CHAR0
0D6F  46                                    INC     SI
0D70  E2 F7                                 LOOP    CODE00
0D72  5F                            CO99:   POP     DI
0D73  5E                                    POP     SI
0D74  59                                    POP     CX
0D75  58                                    POP     AX
0D76  C3                                    RET
0D77                                CODEOUT ENDP
                                    ;-- LGRID ---------------------------------------------------------------;
                                    ; PRINTS LOWER GRID LINE                                                 ;
                                    ;       INPUT                                                            ;
                                    ;         ES:DI : ATTRIBUTE BUFFER                                       ;
                                    ;         CX    : ATTRIBUTE LENGTH                                       ;
                                    ;-----------------------------------------------------------------------;
0D77                                LGRID   PROC    NEAR
0D77  50                                    PUSH    AX
0D78  51                                    PUSH    CX
0D79  57                                    PUSH    DI
0D7A  C7 06 001C R 0003                     MOV     CPIMASK,3               ; SET MASK FOR 13.5 CPI
0D80  B0 4C                                 MOV     AL,4CH                  ; CHECK LOWER ATTRIBUTE EXISTS
0D82  8B F7                                 MOV     SI,DI
0D84  E8 0E74 R                             CALL    CHECK
0D87  72 28                                 JC      LG99
0D89  B0 55                                 MOV     AL,55H                  ; UNIDIRECTION SET
0D8B  E8 0ECE R                             CALL    DIRSET
0D8E  E8 0E9F R                             CALL    ESC%1                   ; OUT ESC+X+1+N1+N2
0D91  51                            LG00:   PUSH    CX                      ; CLEAR IMAGE BUFFER
0D92  8D 36 02A0 R                          LEA     SI,GRID
0D96  B9 0036                               MOV     CX,54
0D99  E8 0E6D R                             CALL    CLEAR
0D9C  59                                    POP     CX
0D9D  BE 0009                               MOV     SI,9                    ; SET VERTICAL GRID (LOWER)
0DA0  E8 0DD2 R                             CALL    VGRID
0DA3  E8 0DB5 R                             CALL    UNDER                   ; SET UNDERSCORE
0DA6  E8 0E42 R                             CALL    PGRID                   ; OUT GRID IMAGE
0DA9  47                                    INC     DI
0DAA  E2 E5                                 LOOP    LG00
0DAC  B0 42                                 MOV     AL,42H                  ; BIDIRECTION SET
0DAE  E8 0ECE R                             CALL    DIRSET
0DB1  5F                            LG99:   POP     DI
0DB2  59                                    POP     CX
0DB3  58                                    POP     AX
0DB4  C3                                    RET
0DB5                                LGRID   ENDP
                                    ;-- UNDER ---------------------------------------------------------------;
                                    ; SETS UNDERSCORE IMAGE                                                  ;
                                    ;-----------------------------------------------------------------------;
0DB5                                UNDER   PROC    NEAR
0DB5  26: F6 05 40                          TEST    BYTE    PTR     ES:[DI],40H     ;NO UNDERSCORE ?
0DB9  74 07                                 JZ      UN99
0DBB  8A 26 0047 R                          MOV     AH,GVALUE+V_UNDER
0DBF  E8 0DC3 R                             CALL    UNSET
0DC2  C3                            UN99:   RET
0DC3                                UNDER   ENDP
                                    ;-- UNSET ---------------------------------------------------------------;
                                    ; SETS UNDESCORE IMAGE                                                   ;
                                    ;       INPUT                                                            ;
                                    ;         AH : UNDERSCORE IMAGE                                          ;
                                    ;-----------------------------------------------------------------------;
0DC3                                UNSET   PROC    NEAR
0DC3  33 F6                                 XOR     SI,SI
0DC5  08 A4 02A2 R                  UN500:  OR      GRID+2[SI],AH           ; SET UNDERSCORE
0DC9  83 C6 03                              ADD     SI,3
0DCC  83 FE 36                              CMP     SI,54
0DCF  75 F4                                 JNE     UN500
0DD1  C3                                    RET
0DD2                                UNSET   ENDP
                                    ;-- VGRID ---------------------------------------------------------------;
                                    ; SETS VERTICAL GRID LINE                                                ;
                                    ;       INPUT                                                            ;
                                    ;         ES:DI : ATTRIBUTE BUFFER                                       ;
```

```
0DD2
0DD2  26: 8B 05
0DD5  25 000C
0DD8  74 0D
0DDA  2D 0004
0DDD  74 09
0DDF  2D 0004
0DE2  74 1E
0DE4  EB 42 90
0DE7  C3


0DE8  8A 84 0041 R
0DEC  08 06 02A0 R
0DF0  8A 84 0042 R
0DF4  08 06 02A1 R
0DF8  8A 84 0043 R
0DFC  08 06 02A2 R
0E00  EB E5


0E02  8A 84 0041 R
0E06  08 06 02A0 R
0E0A  08 06 02A3 R
0E0E  8A 84 0042 R
0E12  08 06 02A1 R
0E16  08 06 02A4 R
0E1A  8A 84 0043 R
0E1E  08 06 02A2 R
0E22  08 06 02A5 R
0E26  EB BF


0E28  8A 84 0044 R
0E2C  08 06 02A0 R
0E30  8A 84 0045 R
0E34  08 06 02A1 R
0E38  8A 84 0046 R
0E3C  08 06 02A2 R
0E40  EB A5
0E42


0E42
0E42  51
0E43  06
0E44  32 ED
0E46  8A 0E 0056 R
0E4A  8B C1
0E4C  03 C8
0E4E  03 C8
0E50  80 3E 0052 R 6C
0E55  75 0A
0E57  03 0E 001C R
0E5B  81 36 001C R 0003
0E61  1E
0E62  07
0E63  8D 36 02A8 R
0E67  E8 0EFE R
0E6A  07
0E6B  59
0E6C  C3
0E6D


0E6D
0E6D  C6 04 00
0E70  46
0E71  E2 FA
0E73  C3
0E74


0E74
0E74  51
0E75  0B C9
0E77  74 08
0E79  26: 84 04
0E7C  75 07
0E7E  46
0E7F  E2 F8
0E81  F9
0E82  EB 02 90
0E85  F8
0E86  59
0E87  C3
0E88
```

```
;         SI     : 0:UPPER 9:LOWER                                          ;
;--------------------------------------------------------------------------;
VGRID   PROC    NEAR
        MOV     AX,ES:[DI]              ; GET ATTRIBUTE BYTE
        AND     AX,000CH                ; NO GRID ?
        JZ      VG99
        SUB     AX,4                    ; SINGLE ?
        JZ      VG100
        SUB     AX,4                    ; DOUBLE ?
        JZ      VG200
        JMP     VG300                   ; DASHED
VG99:   RET
;--------------------------------------------------------------------------;
;       SINGLE                                                             ;
;--------------------------------------------------------------------------;
VG100:  MOV     AL,GVALUE+V_SOLID+0[SI] ;SET GRID IMAGE
        OR      GRID+0,AL
        MOV     AL,GVALUE+V_SOLID+1[SI]
        OR      GRID+1,AL
        MOV     AL,GVALUE+V_SOLID+2[SI]
        OR      GRID+2,AL
        JMP     VG99
;--------------------------------------------------------------------------;
;       DOUBLE                                                             ;
;--------------------------------------------------------------------------;
VG200:  MOV     AL,GVALUE+V_SOLID+0[SI] ; SET GRID IMAGE
        OR      GRID+0,AL
        OR      GRID+3,AL
        MOV     AL,GVALUE+V_SOLID+1[SI]
        OR      GRID+1,AL
        OR      GRID+4,AL
        MOV     AL,GVALUE+V_SOLID+2[SI]
        OR      GRID+2,AL
        OR      GRID+5,AL
        JMP     VG99
;--------------------------------------------------------------------------;
;       DASHED                                                             ;
;--------------------------------------------------------------------------;
VG300:  MOV     AL,GVALUE+V_DASH+0[SI]  ; SET GRID IMAGE
        OR      GRID+0,AL
        MOV     AL,GVALUE+V_DASH+1[SI]
        OR      GRID+1,AL
        MOV     AL,GVALUE+V_DASH+2[SI]
        OR      GRID+2,AL
        JMP     VG99
VGRID   ENDP
;-- PGRID ---------------------------------------------------------------;
; PRINTS GRID IMAGE                                                       ;
;-------------------------------------------------------------------------;
PGRID   PROC    NEAR
        PUSH    CX
        PUSH    ES
        XOR     CH,CH                   ; OUT GRID IMAGE
        MOV     CL,GVALUE+H_SIZE
        MOV     AX,CX                   ; CX * 3
        ADD     CX,AX
        ADD     CX,AX
        CMP     GVALUE+H_KEY,6CH        ; 13.5 CPI ?
        JNE     PG00
        ADD     CX,CPIMASK              ; CPI ADJUST
        XOR     CPIMASK,03H
PG00:   PUSH    DS
        POP     ES
        LEA     SI,GRID
        CALL    FIRES
        POP     ES
        POP     CX
        RET
PGRID   ENDP
;-- CLEAR ---------------------------------------------------------------;
; FILLS THE AREA WITH 0                                                   ;
;       INPUT                                                             ;
;         SI : CLEAR ADDRESS                                              ;
;         CX : CLEAR COUNT                                                ;
;-------------------------------------------------------------------------;
CLEAR   PROC    NEAR
        MOV     BYTE    PTR     [SI],0
        INC     SI
        LOOP    CLEAR
        RET
CLEAR   ENDP
;-- CHECK ---------------------------------------------------------------;
; CHECKS STRINGS                                                          ;
;       INPUT                                                             ;
;         AL     : CODE MASK                                              ;
;         CX     : CODE LENGTH                                            ;
;         ES:SI  : CODE BUFFER                                            ;
;-------------------------------------------------------------------------;
CHECK   PROC    NEAR
        PUSH    CX
        OR      CX,CX
        JZ      CH10
CH00:   TEST    ES:[SI],AL              ; ATTRIBUTE EXEISTS ?
        JNZ     CH20
        INC     SI
        LOOP    CH00
CH10:   STC                             ; NO
        JMP     CH99
CH20:   CLC                             ; YES
CH99:   POP     CX
        RET
CHECK   ENDP
;-- ESCL ----------------------------------------------------------------;
; OUTPUTS ESC + L + N1 + N2                                               ;
```

```
                                ;       INPUT                                                   ;
                                ;         AX     : N1N2                                        ;
                                ;-----------------------------------------------------------------;
0E88                            ESCL    PROC    NEAR
0E88  50                                PUSH    AX
0E89  50                                PUSH    AX
0E8A  B0 1B                             MOV     AL,1BH                  ; "ESC"
0E8C  E8 0F0A R                         CALL    FIRE
0E8F  B0 4C                             MOV     AL,4CH                  ; "L"
0E91  E8 0F0A R                         CALL    FIRE
0E94  58                                POP     AX                      ; N1
0E95  E8 0F0A R                         CALL    FIRE
0E98  58                                POP     AX                      ; N2
0E99  86 C4                             XCHG    AL,AH
0E9B  E8 0F0A R                         CALL    FIRE
0E9E  C3                                RET
0E9F                            ESCL    ENDP
                                ;-- ESCX1 -------------------------------------------------------;
                                ; OUTPUTS ESC + X + 1 + N1 + N2                                  ;
                                ;       INPUT                                                   ;
                                ;         CX     : CHARACTER LENGTH                              ;
                                ;-----------------------------------------------------------------;
0E9F                            ESCX1   PROC    NEAR
0E9F  51                                PUSH    CX
0EA0  B0 1B                             MOV     AL,1BH                  ; ESC+X+1
0EA2  E8 0F0A R                         CALL    FIRE
0EA5  B0 25                             MOV     AL,25H
0EA7  E8 0F0A R                         CALL    FIRE
0EAA  B0 31                             MOV     AL,31H
0EAC  E8 0F0A R                         CALL    FIRE
0EAF  A0 0056 R                         MOV     AL,GVALUE+H_SIZE        ; DOT SIZE - CHAR SIZE
0EB2  F6 E1                             MUL     CL
0EB4  80 3E 0052 R 6C                   CMP     GVALUE+H_KEY,6CH        ; 13.5 CPI ?
0EB9  75 05                             JNE     ESCX1X
0EBB  F8                                CLC                             ; AX = CX / 2
0EBC  D1 D9                             RCR     CX,1
0EBE  13 C1                             ADC     AX,CX
0EC0  8B C8                     ESCX1X: MOV     CX,AX
0EC2  8A C5                             MOV     AL,CH                   ; N1+N2
0EC4  E8 0F0A R                         CALL    FIRE
0EC7  8A C1                             MOV     AL,CL
0EC9  E8 0F0A R                         CALL    FIRE
0ECC  59                                POP     CX
0ECD  C3                                RET
0ECE                            ESCX1   ENDP
                                ;-- DIRSET ------------------------------------------------------;
                                ; OUTPUTS ESC + X + DIRECTION (U OR B)                           ;
                                ;       INPUT                                                   ;
                                ;         AL     : DIRECTION                                     ;
                                ;-----------------------------------------------------------------;
0ECE                            DIRSET  PROC    NEAR
0ECE  50                                PUSH    AX                      ; ESC+X+U,B
0ECF  B0 1B                             MOV     AL,1BH
0ED1  E8 0F0A R                         CALL    FIRE
0ED4  B0 25                             MOV     AL,25H
0ED6  E8 0F0A R                         CALL    FIRE
0ED9  58                                POP     AX
0EDA  E8 0F0A R                         CALL    FIRE
0EDD  C3                                RET
0EDE                            DIRSET  ENDP
                                ;-- FEED --------------------------------------------------------;
                                ; EXECUTES LINE FEED                                             ;
                                ;-----------------------------------------------------------------;
0EDE                            FEED    PROC    NEAR
0EDE  B0 0D                             MOV     AL,0DH                  ; CR.
0EE0  E8 0F0A R                         CALL    FIRE
0EE3  B0 1B                             MOV     AL,1BH                  ; ESC+X+5
0EE5  E8 0F0A R                         CALL    FIRE
0EE8  B0 25                             MOV     AL,25H
0EEA  E8 0F0A R                         CALL    FIRE
0EED  B0 35                             MOV     AL,35H
0EEF  E8 0F0A R                         CALL    FIRE
0EF2  B0 00                             MOV     AL,0                    ; N1N2
0EF4  E8 0F0A R                         CALL    FIRE
0EF7  A0 0048 R                         MOV     AL,GVALUE+V_SIZE
0EFA  E8 0F0A R                         CALL    FIRE
0EFD  C3                                RET
0EFE                            FEED    ENDP
                                ;-- FIRES -------------------------------------------------------;
                                ; OUTPUTS STRING DATA TO PRINTER PORT                            ;
                                ;       INPUT                                                   ;
                                ;         ES:SI : ADDRESS OF PRINT STRING                       ;
                                ;         CX    : LENGTH OF STRING                              ;
                                ;       OUTPUT                                                  ;
                                ;         AH    : STATUS                                        ;
                                ;-----------------------------------------------------------------;
0EFE                            FIRES   PROC    NEAR
0EFE  51                                PUSH    CX                      ; SAVE REGS.
0EFF  26: 8A 04                 FIRE00: MOV     AL,ES:[SI]              ; OUT DATA
0F02  E8 0F0A R                         CALL    FIRE
0F05  46                                INC     SI
0F06  E2 F7                             LOOP    FIRE00
0F08  59                                POP     CX
0F09  C3                                RET
0F0A                            FIRES   ENDP
                                ;-- FIRE --------------------------------------------------------;
                                ; OUTPUTS DATA TO PRINTER PORT                                   ;
                                ;       INPUT                                                   ;
                                ;         AL : OUTPUT DATA                                      ;
                                ;       OUTPUT                                                  ;
                                ;         AH : PRINTER STATUS                                   ;
                                ;                                                               ;
                                ;-----------------------------------------------------------------;
```

```
0F0A                            FIRE    PROC    NEAR
0F0A  52                                PUSH    DX
0F0B  51                                PUSH    CX                      ; SAVE REGS.
0F0C  53                                PUSH    BX
0F0D  50                                PUSH    AX
0F0E  BA 0378                           MOV     DX,PR_DATA_PORT         ; OUT DATA IN AL
0F11  EE                                OUT     DX,AL
0F12  B3 01                             MOV     BL,1                    ; BUSY CHECK
0F14  E8 0F54 R                         CALL    STATUS
0F17  F6 C4 08                          TEST    AH,PR_ERROR
0F1A  75 17                             JNZ     ABEND
0F1C  BA 037A                           MOV     DX,PR_CMD_PORT          ; SET STROBE
0F1F  B0 0D                             MOV     AL,0DH
0F21  EE                                OUT     DX,AL                   ;
0F22  B0 0C                             MOV     AL,0CH
0F24  EE                                OUT     DX,AL                   ;
0F25  B3 00                             MOV     BL,0                    ; GET LAST STATUS
0F27  E8 0F54 R                         CALL    STATUS
0F2A  58                                POP     AX                      ; RESTORE REGS.
0F2B  8A 26 0002 R                      MOV     AH,RETURN_CODE
0F2F  5B                                POP     BX
0F30  59                                POP     CX
0F31  5A                                POP     DX
0F32  C3                                RET
0F33  8B 26 0014 R              ABEND:  MOV     SP,SPSAVE               ; ABEND EXIT
0F37  8D 36 0003 R                      LEA     SI,PRINTER_ID+3         ; CLEAR CONTROL AREA
0F3B  B9 005D                           MOV     CX,93
0F3E  E8 0E6D R                         CALL    CLEAR
0F41  8D 36 00A0 R                      LEA     SI,IMAGE_BUFFER         ; CLEAR IMAGE_BUFFER
0F45  B9 0474                           MOV     CX,1140
0F48  E8 0E6D R                         CALL    CLEAR
0F4B  E8 08B9 R                         CALL    DEFAULT                 ; SET DEFAULT VALUE
0F4E  E8 0A34 R                         CALL    GETVAL                  ; RESET CPI , LPI CONTROL VALUE
0F51  E9 00B9 R                         JMP     RETURN
0F54                            FIRE    ENDP
                                ;-- STATUS ------------------------------------------------------------;
                                ; GETS STATUS 1                                                        ;
                                ;                                                                      ;
                                ;       INPUT                                                          ;
                                ;         BL : BUSY WAIT FLAG (0:NOT WAIT / 1:WAIT)                    ;
                                ;                                                                      ;
                                ;       OUTPUT                                                         ;
                                ;         AH           : PRINTER STATUS 1                              ;
                                ;         RETURN_CODE : PRINTER STATUS 1                               ;
                                ;                                                                      ;
                                ;----------------------------------------------------------------------;
0F54                            STATUS  PROC    NEAR
0F54  52                                PUSH    DX
0F55  51                                PUSH    CX
0F56  86 E0                             XCHG    AH,AL
                                ;
0F58  BA 0379                   ST00:   MOV     DX,PR_STATUS_PORT       ; SET STATUS PORT
0F5B  C7 06 0016 R 0004                 MOV     PR_TIME1,4              ; RESET BEEP TIMER
0F61  C7 06 001A R 000B                 MOV     PR_TIME3,11             ; SET WAIT TIME
0F67  33 C9                             XOR     CX,CX
                                ;
0F69  EC                        ST10:   IN      AL,DX                   ; GET STATUS
0F6A  24 FE                             AND     AL,0FEH                 ; RESET TIMEOUT BIT
0F6C  0A DB                             OR      BL,BL                   ; BUSY NO CHECK ?
0F6E  74 13                             JZ      ST99
0F70  A8 80                             TEST    AL,PR_BUSY              ; BUSY ?
0F72  75 0F                             JNZ     ST99                    ; NO
0F74  E8 0FE5 R                         CALL    BEEP                    ; BEEP ROUTINE
0F77  E2 F0                             LOOP    ST10
0F79  FF 0E 001A R                      DEC     PR_TIME3
0F7D  75 EA                             JNZ     ST10
0F7F  0C 01                             OR      AL,PR_TIMEOUT           ; SET TIMEOUT STATUS
0F81  24 F7                             AND     AL,0FFH-PR_ERROR
0F83  34 08                     ST99:   XOR     AL,PR_ERROR
0F85  86 C4                             XCHG    AL,AH                   ; STATUS TO AH
0F87  80 E4 B9                          AND     AH,0B9H                 ; RESET UNUSED FLAG
0F8A  88 26 0002 R                      MOV     RETURN_CODE,AH
0F8E  59                                POP     CX
0F8F  5A                                POP     DX
0F90  C3                                RET
0F91                            STATUS  ENDP
                                ;-- STATUS2 -----------------------------------------------------------;
                                ; GETS STATUS 2                                                        ;
                                ;                                                                      ;
                                ;       OUTPUT                                                         ;
                                ;         AH           : PRINTER STATUS 2                              ;
                                ;         RETURN_CODE : PRINTER STATUS 2                               ;
                                ;                                                                      ;
                                ;----------------------------------------------------------------------;
0F91                            STATUS2 PROC    NEAR
0F91  52                                PUSH    DX
                                ;
0F92  B4 00                             MOV     AH,0                    ; SET PRINTER_ID
0F94  83 3E 0000 R 02                   CMP     PRINTER_ID,2
0F99  74 02                             JE      ST20
0F9B  B4 01                             MOV     AH,1
0F9D  80 CC 80                  ST20:   OR      AH,80H                  ; SET ASF ON
0FA0  80 E4 81                          AND     AH,81H                  ; RESET UNUSED BIT
0FA3  80 3E 0003 R 90                   CMP     CPI,90H                 ; CPI=5 ?
0FA8  74 17                             JE      LPISET
0FAA  80 C4 02                          ADD     AH,02H                  ; CPI=6 ?
0FAD  80 3E 0003 R 78                   CMP     CPI,78H
0FB2  74 0D                             JE      LPISET
0FB4  80 C4 02                          ADD     AH,02H                  ; CPI=6.7 ?
0FB7  80 3E 0003 R 6C                   CMP     CPI,6CH
0FBC  74 03                             JE      LPISET
0FBE  80 C4 02                          ADD     AH,02H                  ; CPI=7.5
0FC1  80 3E 0004 R 1E           LPISET: CMP     LPI,1EH                 ; LPI=4 ?
```

```
0FC6  74 17                     JE      ST999                   ; LPI=5 ?
0FC8  80 C4 08                  ADD     AH,08H
0FCB  80 3E 0004 R 18           CMP     LPI,18H
0FD0  74 0D                     JE      ST999                   ; LPI=6 ?
0FD2  80 C4 08                  ADD     AH,08H
0FD5  80 3E 0004 R 14           CMP     LPI,14H
0FDA  74 03                     JE      ST999                   ; LPI=7.5
0FDC  80 C4 08                  ADD     AH,08H
0FDF  88 26 0002 R              MOV     RETURN_CODE,AH
0FE3  5A                        POP     DX
0FE4  C3                        RET
0FE5                            STATUS2 ENDP
                                ;-- BEEP ------------------------------------------------------------
                                ; BEEP WHEN BUSY                                                    ;
                                ;                                                                   ;
                                ;-------------------------------------------------------------------
0FE5                            BEEP    PROC    NEAR
0FE5  FF 0E 0018 R              DEC     PR_TIME2
0FE9  75 11                     JNZ     BEEP99
0FEB  FF 0E 0016 R              DEC     PR_TIME1
0FEF  75 0B                     JNZ     BEEP99
0FF1  B8 0E07                   MOV     AX,0E07H                ; BEEP
0FF4  CD 10                     INT     10H
                                ;
0FF6  C7 06 0016 R 0001         MOV     PR_TIME1,1
                                ;
0FFC  C3                        BEEP99: RET
                                ;
0FFD                            BEEP    ENDP
1040                            ORG     BEGIN+1040H
1040                            CODE    ENDS
                                END
```

```
**************
**************
*            *
* MODULE 7   *
*            *
**************
**************
```

```
                        ;=== INT 19 ==========================================
                        ; BOOT STRAP LOADER
                        ;       TRACK 0, SECTOR 1 IS READ INTO THE
                        ;       BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
                        ;       AND CONTROL IS TRANSFERRED THERE.
                        ;
                        ;       IF THE DISKETTE IS NOT PRESENT OR HAS A
                        ;       PROBLEM LOADING (E.G., NOT READY), AN INT.
                        ;       18H IS EXECUITED. IF A CARTRIDGE HAS VECTORED
                        ;       INT. 18H TO ITSELF, CONTROL WILL BE PASSED TO
                        ;       THE CARTRIDGE.
                        ;
                        ;       THIS ROUTINE SET INITIAL CARTRIDGE PROCESS
                        ;       AND ALSO CHECK DISKETTE IS BOOTABLE OR NOT
                        ;-----------------------------------------------------
                                ASSUME  CS:CODE,DS:ABS0
= 001E                  LBOOT   EQU     1EH
0000                    BOOT_STRAP      PROC    NEAR
0000  FB                               STI                     ; ENABLE EXTERNAL INTERRUPTS

0001  B4 01                            MOV     AH,1            ; INITIALIZE PRINTER
0003  33 D2                            XOR     DX,DX
0005  CD 17                            INT     INT_17

0007  CD 7B                            INT     7BH             ; GO SYSTEM CARTRIDGE HANDLING

0009  CD 11                    H1:     INT     INT_11          ; CHECK APPLICATION MODE
000B  24 30                            AND     AL,30H          ;
000D  3C 20                            CMP     AL,20H          ; EXTENSION MODE ?
000F  74 15                            JE      H2              ; YES-SKIP FOLLOWING VIDEO SET MODE

0011  B8 0011                          MOV     AX,0011H        ; SET 20X11 COLOR MODE FOR DEFAULT
0014  CD 10                            INT     INT_10          ;
0016  B8 1000                          MOV     AX,1000H        ; SET 20X11 DEFAULT COLOR BLUE
0019  BB 0100                          MOV     BX,0100H        ; FOREGROUND TO BLUE
001C  CD 10                            INT     INT_10          ;
001E  B8 1001                          MOV     AX,1001H        ; INCLUDING BORDER COLOR TO BLUE
0021  BB 0100                          MOV     BX,0100H        ;
0024  CD 10                            INT     INT_10          ;

0026  B8 FF20                  H2:     MOV     AX,KKNINIT      ; INITIALIZE KANA-KAN CONVERSION
0029  CD 78                            INT     78H             ;


                                ASSUME  DS:DATA
002B  E8 0000 E                         CALL    DDS
002E  8B 3E 0302 R                      MOV     DI,JEQUIP_FLAG  ; GET DRIVE CONFIGURATION

0032  80 0E 0336 R 08                   OR      JKB_FLAG,08H    ; FLAG TO KANA-KAN INITIALIZED
0037  B8 0500                           MOV     AX,0500H        ; INITIALIZE KEYBOARD
003A  CD 16                             INT     INT_16          ;

                        ;================================
                                ASSUME  DS:ABS0
003C  2B C0                             SUB     AX,AX           ; ESTABLISH ADDRESSING
003E  8E D8                             MOV     DS,AX
0040  8E C0                             MOV     ES,AX           ; SET DISKETTE BUFFER ADDR
0042  BB 7C00 R                         MOV     BX,OFFSET BOOT_LOCN ;

                        ;----- RESET THE DISK PARAMETER TABLE VECTOR
0045  C7 06 0078 R 0000 E               MOV     WORD PTR DISK_POINTER,OFFSET DISK_BASE
004B  8C 0E 007A R                      MOV     WORD PTR DISK_POINTER+2,CS

                        ;-----  LOAD SYSTEM FROM DISKETTE
004F  33 F6                             XOR     SI,SI           ; RESET DISKETTE INSTALLED FLAG

0051  33 D2                             XOR     DX,DX           ; HEAD 0, DRIVE 0
0053  D1 CF                    H3:     ROR     DI,1            ; DISKETTE DRIVE ATTACHED ?
0055  73 2F                             JNC     H7              ; NO -GOTO NEXT DRIVE CHECK

0057  B9 0004                           MOV     CX,4            ; SET RETRY COUNT
005A  51                        H4:     PUSH    CX              ; SAVE RETRY COUNT
005B  B4 00                             MOV     AH,0            ; RESET THE DISKETTE SYSTEM
005D  CD 13                             INT     INT_13          ;
005F  72 08                             JC      H5              ; IF ERROR, TRY AGAIN

0061  B8 0201                           MOV     AX,0201H        ; READ OP., 1 SECTOR
0064  B9 0001                           MOV     CX,0001H        ; TRACK 0, SECTOR 1
0067  CD 13                             INT     INT_13          ; READ IN THE SINGLE SECTOR
0069  59                        H5:     POP     CX              ; RESTORE RETRY COUNT
006A  73 09                             JNC     H6              ; IF GOOD, GOTO FORMAT CHECK
006C  F6 C4 80                          TEST    AH,80H          ; TIMEOUT ERROR ?
006F  75 15                             JNZ     H7              ; YES-GOTO NEXT DRIVE CHECK
0071  E2 E7                             LOOP    H4              ; DO IT FOR RETRY TIMES
0073  EB 11                             JMP     SHORT H7        ;

                        ;----- CHECK DISKETTE FORMAT
0075  BE FFFF                   H6:     MOV     SI,0FFFFH       ; SET DISKETTE INSTALLED FLAG

0078  81 7F 03 4249                     CMP     DS:WORD PTR BOOT_LOCN1[BX],'BI' ; FORMAT GOOD ?
007D  75 07                             JNE     H7              ; NO -
007F  81 7F 05 4A4D                     CMP     DS:WORD PTR BOOT_LOCN2[BX],'JM'
0084  74 22                             JE      GO_BOOT         ; YES-GOTO BOOT PROGRAM

0086  42                        H7:     INC     DX              ; SET DRIVE 0 FOR NEXT DRIVE

0087  80 FA 04                          CMP     DL,4            ; END OF DRIVES ?
008A  72 C7                             JB      H3              ; NO -

008C  0B F6                             OR      SI,SI           ; ANY DISKETTE INSTALLED ?
008E  74 16                             JZ      H9              ; NO  -

                        ;----- DISKETTE WAS INSTALLED, BUT SYSTEM DISKETTE WAS NOT INSTALLED.
0090  BE 00B4 R                         MOV     SI,OFFSET BOOT_ERR ; DISPLAY ERROR MESSAGE
```

```
0093  B9 003E                        MOV     CX,62                   ;
0096  2E: 8A 04                  H8:   MOV     AL,CS:[SI]              ;
0099  46                              INC     SI                      ;
009A  E8 0000 E                       CALL    PRT_HEX                 ;
009D  E2 F7                           LOOP    H8                      ;

009F  B4 00                           MOV     AH,0                    ; WAIT ANY KEY INPUT
00A1  CD 16                           INT     INT_16                  ;

00A3  E9 0009 R                       JMP     H1                      ; RETRY FROM 1ST DRIVE

                                  ;----- UNABLE TO IPL FROM THE DISKETTE
00A6  CD 18                       H9:   INT     INT_18                  ; GO TO BASIC OR APPLICATION CARTRIDGE

                                  ;----- IPL WAS SUCCESSFUL
00A8                              GO_BOOT:
00A8  59                              POP     CX                      ; ADJUST STACK POINTER

00A9  80 CA 40                        OR      DL,40H                  ; SET DOUBLE TRACK SYSTEM DISKETTE
00AC  88 57 1E                        MOV     DS:BYTE PTR LBOOT[BX],DL    ; OR LOGICAL BOOT DISKETTE
00AF  EA 7C00 ---- R                  JMP     BOOT_LOCN               ; GO BOOT LOCATION

                                  ;--------------------------------
                                  ;       MESSAGE AREA            |
                                  ;--------------------------------
00B4  90 B3 82 B5 82 A2           BOOT_ERR DB     "正しい システム ディスケットを差し込み,"
      20 83 56 83 58 83
      65 83 80 20 83 65
      DE B2 83 58 83 50
      AF 83 67 82 F0 8D
      B7 82 B5 8D 9E 82
      DD 2C
00DA  0A0D                              DW      0A0DH
00DC  89 FC 8D 73 83 4C                 DB      "改行キーを押して下さい"
      81 5B 82 F0 89 9F
      82 B5 82 C4 89 BA
      82 B3 82 A2
00F2                              BOOT_STRAP      ENDP

0100                                  ORG   BEGIN+100H
0100                              CODE    ENDS
                                        END
```

```
                        ;*************************************************************************
                        ;*
                        ;*       KANA-KANJI DATA DEFINITION                                      *
                        ;*                                                                       *
                        ;*************************************************************************
                        ;
0000                    KKDATA  SEGMENT AT 80H
                        ;
= 0000                  DSTART  EQU     $
                        ;
                        ;*************************************************************************
                        ;*       DATA    AEAR                                                    *
                        ;*************************************************************************
                        ;
                        ;*
0000  ??                DSTAT   DB      ?               ;DATA STATUS
= 0001                  HANKAF  EQU     01H             ;HANKAKU
= 0002                  ZEN2F   EQU     02H             ;ZENKAKU
= 0004                  FUNCF   EQU     04H             ;FUNCTION
= 0008                  ZENMENF EQU     08H             ;ZENMEN
= 0010                  KANJIF  EQU     10H             ;KANJI
= 0020                  ZEN1F   EQU     20H             ;ZENKAKU HIGH BYTE
= 0080                  DAKUF   EQU     80H             ;DAKUTEN/HANDAKUTEN
                        ;
0001  ????              KBDSTAT DW      ?               ;KEYBOARD STATAS
= 0000                  ESHIFT  EQU     00H             ;EISU SHIFT
= 0002                  KSHIFT  EQU     02H             ;KATAKANA SHIT
= 0004                  HSHIFT  EQU     04H             ;HIRAGANA SHIFT
= 0001                  ZMODE   EQU     01H             ;ZENKAKU MODE
= 0000                  HMODE   EQU     00H             ;HANKAKU MODE
= 0006                  KBDMSK  EQU     06H             ;KBD STATUS MASK PATTERN
= 0040                  CAPSST  EQU     40H             ;CAPS STATUS
= 0010                  CAPSON  EQU     10H             ;CAPS LOCK
= 00C0                  KKNOCHG EQU     0C0H            ;KANA-KAN REQUIREMENT NOT CHANGED
                        ;
=                       KKKYDCD EQU     CHAR1           ;KEY DATA CODE(6BYTE)
=                       KKINKEY EQU     CHAR1
0003  ??                CHAR1   DB      ?
0004  ??                ATTR1   DB      ?               ;CHARACTER 1
0005  ??                SCAN1   DB      ?               ;ATTRIBUTE 1
0006  ??                CHAR2   DB      ?               ;SCAN CODE 1
0007  ??                ATTR2   DB      ?               ;CHARACTER 2
0008  ??                SCAN2   DB      ?               ;ATTRIBUTE 2
                        ;                               ;SCAN CODE 2
                        ;*---------------------------------------------------------------------*
0009      6C [          KKINBUF DB      108 DUP(?)      ;YOMI INPUT BUFFER(120BYTE)
              ??
                  ]
0075      6C [          KKINBFSV DB     108 DUP(?)      ; YOMI INPUT BUFFER SAVE
              ??
                  ]
                        ;
= 0000                  HANATTR EQU     0               ;HANKAKU ATTRIBUTE
00E1  ??                ZENATTR1 DB     ?               ;HIGH BYTE OF ZENKAKU ATTRIBUTE
= 0001                  ZATTR1S  EQU    01H             ;80*25
= 0080                  ZATTR1N  EQU    80H             ;80*11,40*11
00E2  ??                ZENATTR2 DB     ?               ;LOW BYTE OF ZENKAKU ATTRIBUTE
= 0003                  ZATTR2S  EQU    03H             ;80*25
= 0088                  ZATTR2N  EQU    88H             ;80*11,40*11
                        ;
00E3  ??                ROATCNT DB      ?               ;ROTATE ATTRIBUTE COUNTER
= 0000                  ROT25   EQU     0               ;80*25
= 0001                  ROT11   EQU     1               ;80*11,40*11
                        ;
00E4  ????              SPACE   DW      ?               ;SPACE HANKAKU DATA
                        ;
00E6  ????              KKWCCA  DW      ?               ; BUFFER CONTECUTUAL CURSOR
00E8  ????              KKEOPMAX DW     ?               ; END OF YOMI POINTER ( MAX )
= 000F                  BEOPMAX EQU     15              ;BANGOU MAXIMUM EOP
                        ;
00EA  ????              KKWEOP  DW      ?               ; END OF YOMI POINTER
00EC  ????              KKWEOPSV DW     ?               ; SAVE AREA FOR END OF YOMI POINTER
                        ;
00EE  ????              BASE    DW      ?               ; START COLUMN OF DISPLAY INPUT BUFF.
= 0030                  BBASE25 EQU     48              ;BANGOU 80*25
= 0015                  BBASE11 EQU     21              ;BANGOU 80*11/40*11
                        ;
                        ;
00F0  ??                KKWINST DB      ?               ; INSERT MODE FLAG TABLE
00F1  ????              KKINP   DW      ?               ;DISPLAY POINTER
                        ;
00F3  ????              KKCUSR  DW      ?               ;KANA-KAN CURSOR POSITION
= 183C                  KCSRP25 EQU     183CH           ;KANJI CURSOR POSITION (80*25)
= 0A11                  KCSRP11 EQU     0A11H           ;                      (X *11)
= 1811                  KCSRP4025 EQU   1811H           ;                      (40*25)
                        ;
00F5  ????              KKCHRSP DW      ?               ;CLEAR DATA OF INPUT BUFFER
= 8140                  ZENSP   EQU     8140H           ;ZENKAKU SPACE DATA
                        ;
00F7  ????              KKFORPIT DW     ?               ;MOVE FORWARD POINTER
                        ;
00F9  ??                KKMODE  DB      ?               ;KANA-KANJI MODE
= 0000                  NORMALM EQU     0               ;NORMAL MODE
= 0002                  KANJIM  EQU     2               ;KANJI MODE
= 0004                  SELECTM EQU     4               ;KANJI SELECT MODE
= 0008                  CODEM   EQU     8               ;KANJI BAGOU MODE
                        ;
00FA  ??                TVMODE  DB      ?               ;TV MODE
= 0001                  TV8025  EQU     1               ;TV MODE 80*25
= 0002                  TV8011  EQU     2               ;        80*11
= 0003                  TV4011  EQU     3               ;        40*11
```

```
00FB  ??                    ;
= 0050                      TVLINE  DB      ?               ;NUMBERS OF CHARACTER IN 1 LINE
= 0028                      TVL80   EQU     80              ;80*25
                            TVL40   EQU     40              ;80*11/40*11
                            ;
00FC  ??                    KKOILP  DB      ?               ;OIL POSITION
= 0018                      OILP25  EQU     24              ;80*25 POSITION
= 000A                      OILP11  EQU     10              ;80*11/40*11 POSITION
                            ;
00FD  ??                    KHBLK   DB      ?               ;KOUHO BLOCK NO.
00FE  ??                    KHBLKMX DB      ?               ;MAXIMUM BLOCK NO.
                            ;
00FF  ????                  APKBDST DW      ?               ;SAVE KBD STATUS FOR APPLICATION
                            ;
0101  ??                    DOILP   DB      ?               ;DATA POSITION OF OIL
= 001F                      DOILP25 EQU     31              ;80*25 POSITION
= 000C                      DOILP11 EQU     12              ;80*11 POSITION
= 000C                      DOILP11K EQU    12              ;40*11 POSITION (KANJI MODE)
= 0000                      DOILP11S EQU    00              ;40*11 POSITION (SELECT MODE)
= 0000                      CLIND11P EQU    00              ;40*11 INDICATOR POSITION
0102  ??                    DOILN   DB      ?               ;NUMBER OF OIL DATA
= 0031                      DOILN25 EQU     49              ;80*25 NUMBER
= 0044                      DOILN11 EQU     68              ;80*11 NUMBER
= 001C                      DOILN11K EQU    28              ;40*11 NUMBER (KANJI MODE)
= 0028                      DOILN11S EQU    40              ;40*11 NUMBER (SELECT MODE)
= 000C                      CLIND11N EQU    12              ;40*11 NUMBER OF INDICATOR
                            ;
0103  ??                    BANGOUP DB      ?               ;BANGOU DATA POSITION
= 0027                      BANP25  EQU     39              ;80*25 POSITION
= 000C                      BANP11  EQU     12              ;80*11/40*11 POSITION
                            ;
0104  ??                    BASTERP DB      ?               ;ASTERISK POSITION IN KKOUTBUF
= 002F                      BAST25  EQU     47              ;80*25 POSITION
= 001B                      BAST11  EQU     27              ;80*11/40*11 POSITION
                            ;
0105  ????                  DCRSRP  DW      ?               ;CURSOR POSITION (KKDISP)
0107  ????                  DSTRTP  DW      ?               ;START POSITION  (KKDISP)
0109  ????                  DENDP   DW      ?               ;END POSITON     (KKDISP)
                            ;
                            ;*-----------------------------------------------------------------*
010B      07 [              KKGVNYM DB      7 DUP(?)        ; YOMI WORK
              ??
                ]
                            ;
0112  ????                  KKDICLN DW      ?               ; DICTIONARY INDEX LENGTH
0114  ????                  KKDICFF DW      ?               ; DICTIONARY TOP OFFSET
0116  ????                  KKLENYM DW      ?               ; YOMI LENGTH ON INDEX
                            ;
= 0000                      KKECNRML EQU    0               ; NORMAL
= 0001                      KKECNOTF EQU    1               ; YOMI NOT FOUND
= 0004                      KKECYMLN EQU    4               ; YOMI LENGTH EXCEPTION
= 0008                      KKECYMHR EQU    8               ; HIRAGANA YOMI EXCEPTION
= 0010                      KKECDICT EQU    16              ; DICTIONARY FORMAT ERROR
                            ;
0118  ??                    KKYOMIL DB      ?               ; LENGTH OF YOMI
0119      25 [              KKYOMI  DB      37 DUP(?)       ; YOMI BUFFER
              ??
                ]
                            ;
013E  ????                  KKOHOSUU        DW      ?       ; TOTAL NUMBER OF KOUHO
                            ;
0140                        KKDICADR        LABEL   DWORD
0140  ????                  KKDICOFF        DW      ?       ; DICTIONARY OFFSET
0142  ????                  KKDICSEG        DW      ?       ; DICTIONARY SEGMENT ADDRESS
                            ;
0144  ????                  KKOHOZAN        DW      ?       ; ZAN
0146  ????                  KKOHOCNT        DW      ?       ; DISPLAY KOUHO COUNTER
0148  ????                  KKOHODSP        DW      ?       ; DISPLAY KOUHO PRINTER
                            ;
                            ;*-----------------------------------------------------------------*
014A      44 [              KKHANQUE        DB      17*4 DUP(?)     ; QUEING BUFFER
              ??
                ]
                            ;
018E  ????                  KKHQLEN         DW      ?
                            ;
0190  ??                    KKCHRMD DB      ?
= 0001                      MDZENNUM EQU     01             ; ZENKAKU NUMERIC CHARACTER
= 0002                      MDZENHIR EQU     02             ; ZENKAKU HIRAGANA CHARACTER
= 0003                      MDZENKAT EQU     03             ; ZENKAKU KATAKANA CHARACTER
= 0004                      MDHANNUM EQU     04             ; HANKAKU NUMERIC CHARACTER
= 0005                      MDHANKAT EQU     05             ; HANKAKU KATAKANA CHARACTER
                            ;
0191  ??                    KKYMSTS DB      ?
= 0001                      KKYMSTE EQU     00000001B       ; PROCESS END FLAG
= 0002                      KKYMSTM EQU     00000010B       ; MOVE FORWORD FLAG
= 0004                      KKYMSTD EQU     00000100B       ; DAKUTEN, HANDAKUTEN FLAG
= 0008                      KKYMSTN EQU     00001000B       ; NUMERIC KEY CONVERSION FLAG
= 0080                      KKYMSTER EQU    10000000B       ; ERROR FLAG
                            ;
0192  ??                    KHENFST DB      ?
                            ;
0193  ??                    ATTRMSK DB      ?               ;ATTRIBUTE MASK
= 00FC                      ATMSK25 EQU     0FCH            ;80*25
= 0077                      ATMSK11 EQU     77H             ;80*11/40*11
                            ;
0194  ??                    OLDMODE DB      ?               ;OLD KANA-KAN MODE
                            ;
                            ;*-------*
                            ;
= 0195                      DEND    EQU     $-DSTART
                            ;
```

```
                        ;*-------*
                        ;
0195  ??          KKFLAG  DB      ?
= 00FE            KON     EQU     0FEH
= 0001            KOFF    EQU     01H             ;KANA-KAN ON FLAG
= 0002            CLRF25  EQU     02H             ;KANA-KAN OFF FLAG
= 0004            CUSRST_ON  EQU  04H             ;80*25 COLOR FLAG
= 00FB            CUSRST_OFF EQU  0FBH            ;KANA-KAN CURSOR STATUS ON  (ARI)
                        ;                                OFF (NASHI)
                        ;
                        ;
                        ;**********************************************************************
                        ;*
                        ;*      CONSTANT VALUE                                                *
                        ;*                                                                    *
                        ;**********************************************************************
                        ;
                        ;**********************************************************************
                        ;*      KANA-KAN PARAMATER                                            *
                        ;**********************************************************************
                        ;
= 0000            FUNCAL  EQU     00              ;CHARACTER CODE OF FUNCTION KEY
= 0000            ZENMAH  EQU     00              ;SCAN CODE OF ZENMEN 8-BIT CODE
= 00FF            KANJAH  EQU     0FFH
= 0000            KANEXT  EQU     00              ;SCAN CODE OF KANJI MODE
= 0040            KANENT  EQU     40H             ;CHARACTER CODE OF KANJI EXIT
= 0080            KANCHG  EQU     80H             ;CHARACTER CODE OF KANJI ENTER
= 0020            KANINT  EQU     20H             ;CHARACTER CODE OF KANJI CHANGE
                        ;                         ;CHARACTER CODE OF KANJI INITIAL
= FF00            KEXITX  EQU     0FF00H          ;AX OF KANJI EXIT
= FF40            KENTRX  EQU     0FF40H          ;AX OF KANJI ENTER
= FF80            KCHNGX  EQU     0FF80H          ;AX OF KANJI CHANGE
= FF20            KINITX  EQU     0FF20H          ;AX OF KANJI INITIAL
= FF01            KONX    EQU     0FF01H          ;AX OF KANJI ON
= FF02            KOFFX   EQU     0FF02H          ;AX OF KANJI OFF
                        ;
                        ;**********************************************************************
                        ;*      KEY DATA                                                      *
                        ;**********************************************************************
                        ;
= 6B20            KYKANJX EQU     6B20H           ;SCAN/CHARACTER CODE OF KANJI
= 6B40            KYKANJ2 EQU     6B40H           ;SCAN/CHARACTER CODE OF KANJI (ZEN-LOW)
= 6B81            KYKANJ1 EQU     6B81H           ;SCAN/CHARACTER CODE OF KANJI (ZEN-HIGH)
= 6B38            KYCODEX EQU     6B38H           ;SCAN/CHARACTER CODE OF BANGOU
= 6C20            KYMUHEX EQU     6C20H           ;SCAN/CHARACTER CODE OF MUHENKAN
= 6C40            KYMUHE2 EQU     6C40H           ;SCAN/CHARACTER CODE OF MUHENKAN (ZEN-LOW)
= 6C81            KYMUHE1 EQU     6C81H           ;SCAN/CHARACTER CODE OF MUHENKAN (ZEN-HIGH)
= 6D20            KYHENKX EQU     6D20H           ;SCAN/CHARACTER CODE OF HENKAN
= 6D40            KYHENK2 EQU     6D40H           ;SCAN/CHARACTER CODE OF HENKAN (ZEN-LOW)
= 6D81            KYHENK1 EQU     6D81H           ;SCAN/CHARACTER CODE OF HENKAN (ZEN-HIGH)
= 5200            KYINSTX EQU     5200H           ;SCAN/CHARACTER CODE OF INSERT
= 5300            KYDELTX EQU     5300H           ;SCAN/CHARACTER CODE OF DELETE
= 0E08            KYBACKX EQU     0E08H           ;SCAN/CHARACTER CODE OF BACKSPACE
= 011B            KYESCPX EQU     011BH           ;SCAN/CHARACTER CODE OF ESCAPE
= 7700            KYEEOFX EQU     7700H           ;SCAN/CHARACTER CODE OF ERASE EOF
= 1C0D            KYCRRTX EQU     1C0DH           ;SCAN/CHARACTER CODE OF CARRIER RET.
= 7E0D            KYCRRTY EQU     7E0DH           ;SCAN/CHARACTER CODE OF CARRIER RET.
= 9300            KYENTRX EQU     9300H           ;SCAN/CHARACTER CODE OF ENTER
= 4B00            KYCSRLX EQU     4B00H           ;SCAN/CHARACTER CODE OF CURSOR LEFT
= 4D00            KYCSRRX EQU     4D00H           ;SCAN/CHARACTER CODE OF CURSOR RIGHT
= 1A81            KYDAKUX EQU     1A81H           ;SCAN/CHARACTER CODE OF DAKUTEN (HIGH)
= 1B81            KYHANDX EQU     1B81H           ;SCAN/CHARACTER CODE OF HANDAKUTEN (HIGH)
= 1ADE            KYDAKU1 EQU     1ADEH           ;SCAN/CHARACTER CODE OF DAKUTEN (HANKAKU)
= 1BDF            KYHAND1 EQU     1BDFH           ;SCAN/CHARACTER CODE OF HANDAKUTEN (HANKAKU)
= 1A4A            KYDAKU2 EQU     1A4AH           ;SCAN/CHARACTER CODE OF DAKUTEN (ZENKAKU)
= 1B4B            KYHAND2 EQU     1B4BH           ;SCAN/CHARACTER CODE OF HANDAKUTEN (ZENKAKU)
                        ;
= 001F            CTRL_X  EQU     1FH             ;CHARACTER CODE OF CTRL+X
                        ;
= 0000            RCNOMAL EQU     0               ;NOMAL RETURN CODE
= 0001            RCABNO  EQU     1               ;ABNOMAL RETURN CODE
                        ;
= 0082            ZENHIRA EQU     82H             ;ZENKAKU HIRAGANA HIGH BYTE
= 0083            ZENKATA EQU     83H             ;ZENKAKU KATAKANA HIGH BYTE
= 0081            ZENMARK EQU     81H             ;ZENKAKU MARK     HIGH BYTE
                        ;
= 0700            SPEAKHZ EQU     0700H           ;SPEAKER HZ
= 7FFF            DELAYN  EQU     7FFFH
                        ;
= 0020            BLANK   EQU     20H             ;BLANK DATA CODE
= 002A            ASTER   EQU     2AH             ;ASTERISK DATA CODE
= 00A0            SPECHAR EQU     0A0H            ;SPECIAL CARACTER CODE
= 005E            CAPS    EQU     5EH             ;CAPS DATA CODE
= 0007            BEL     EQU     07H             ;BELL CODE
                        ;
= 0028            DAKUNUM EQU     40              ;NUMBERS OF DAKUTEN CARACTER
= 000A            HANDNUM EQU     10              ;NUMBERS OF HANDAKUTEN CARACTER
                        ;
= 0009            BNUMMIN EQU     9               ;MINIMUM NUMBERS OF BANGOU
= 000F            BNUMMAX EQU     15              ;MAXIMUM NUMBERS OF BANGOU
= 0001            TENMIN  EQU     1               ;MINIMUM TEN
= 005E            TENMAX  EQU     94              ;MAXIMUM TEN
= 0001            KUMIN   EQU     1               ;MINIMUM KU
= 0078            KUMAX   EQU     120             ;MAXIMUM KU
                        ;
                        ;
                        ;**********************************************************************
                        ;*      SCAN CODE                                                     *
                        ;**********************************************************************
                        ;
```

```
= 0002                  SCAN02  EQU     02H
= 000A                  SCAN0A  EQU     0AH
= 000B                  SCAN0B  EQU     0BH
= 000D                  SCAN0D  EQU     0DH
= 0010                  SCAN10  EQU     10H
= 001B                  SCAN1B  EQU     1BH
= 001E                  SCAN1E  EQU     1EH
= 0029                  SCAN29  EQU     29H
= 002B                  SCAN2B  EQU     2BH
= 0035                  SCAN35  EQU     35H
= 0039                  SCAN39  EQU     39H
= 004A                  SCAN4A  EQU     4AH
= 004E                  SCAN4E  EQU     4EH
= 006A                  SCAN6A  EQU     6AH
= 006B                  SCAN6B  EQU     6BH
= 0070                  SCAN70  EQU     70H
= 0072                  SCAN72  EQU     72H
= 007A                  SCAN7A  EQU     7AH
= 007D                  SCAN7D  EQU     7DH
                        ;
= 002F                  NUM19M1 EQU     2FH
= 0025                  NUM0M1  EQU     25H
= 0041                  NUM19M2 EQU     41H
= 0040                  NUM0M2  EQU     40H
                        ;
= 6B00                  SCAN6B0 EQU     6B00H
                        ;**************************************************************
                        ;*      IMMEDIATE VALUE                                       *
                        ;**************************************************************
                        ;
= 0000                  D00     EQU     00H
= 0001                  D01     EQU     01H
= 0002                  D02     EQU     02H
= 0003                  D03     EQU     03H
= 0004                  D04     EQU     04H
= 0005                  D05     EQU     05H
= 0006                  D06     EQU     06H
= 0007                  D07     EQU     07H
= 0008                  D08     EQU     08H
= 0009                  D09     EQU     09H
= 000A                  D0A     EQU     0AH
= 000B                  D0B     EQU     0BH
= 000C                  D0C     EQU     0CH
= 000D                  D0D     EQU     0DH
= 000E                  D0E     EQU     0EH
= 000F                  D0F     EQU     0FH
= 0010                  D10     EQU     10H
= 0020                  D20     EQU     20H
= 0030                  D30     EQU     30H
= 0040                  D40     EQU     40H
= 0050                  D50     EQU     50H
= 0060                  D60     EQU     60H
= 0070                  D70     EQU     70H
= 0080                  D80     EQU     80H
= 0090                  D90     EQU     90H
= 00A0                  DA0     EQU     0A0H
= 00B0                  DB0     EQU     0B0H
= 00C0                  DC0     EQU     0C0H
= 00D0                  DD0     EQU     0D0H
= 00E0                  DE0     EQU     0E0H
= 00F0                  DF0     EQU     0F0H
= 00FF                  DFF     EQU     0FFH
                        ;

                        ;---------------- EQUATES FOR VIDEO INTERFACE ------------

= 0010                  VIDEO           EQU     10H     ; INTERRUPT NO. OF VIDEO I/O
= 0081                  VSETACT         EQU     81H     ; SET ALTERNATE CURSOR TYPE
= 0082                  VSETACP         EQU     82H     ; SET ALTERNATE CURSOR POSITION
= 0083                  VRDACP          EQU     83H     ; READ ALTERNATE CURSOR POSITION
= 0088                  VRDACA          EQU     88H     ; READ ATTRIBUTE AND CHARACTER AT ALT CSR PSN
= 0089                  VWRACA          EQU     89H     ; WRITE ATTRIBUTE AND CHARACTER AT ALT CSR PSN
= 008A                  VWRCA           EQU     8AH     ; WRITE CHARACTER AT ALT CSR PSN
= 008E                  VWRTTYA         EQU     8EH     ; WRITE TTY USING ALT CSR PSN

= 0020                  CURSOR_DISABLE  EQU     20H     ; CURSOR DISABLE BIT


0196                    ;
                        KKDATA  ENDS
                        ;**********************  END **********************************
                        ;
= 0000                          PUBLIC  KKKFDM
                                BIGIN = $
                                ASSUME  CS:CODE,DS:KKDATA
                        ;**************************************************************
                        ;*                                                            *
                        ;* PROGRAM NAME: KKKFDM                                       *
                        ;*                                                            *
                        ;* DESCRIPTIVE NAME: KANA-KANJI MAIN ROUTINE                  *
                        ;*                          (KEYBORAD FUNCTION DISTRIBUTION MONITOR)
                        ;*                                                            *
                        ;* FUNCTION: THIS ROUTINE CLASSIFIES SCAN CODE/CHARACTER INTO GROUPS
                        ;*           THAT ARE THE GRAPHIC, CURSOR OR FUNCTION, THEN BRANCHES
                        ;*           TO EACH PROCESS.                                 *
                        ;*                                                            *
                        ;* LINKAGE: INT 78H                                           *
                        ;*                                                            *
                        ;*    INPUT: AX ( AH - SCAN CODE , AL - CHARACTER )           *
                        ;*                  1 BYTE CODE                               *
                        ;*                  2 BYTE CODE (HIGH BYTE)                   *
                        ;*                  2 BYTE CODE (LOW BYTE)                    *
                        ;*                  FUNCTION KEYS                             *
                        ;*                  8 BIT CODE WITH ZENMEN KEY
```

```
0000                                    ;*
                                        ;*                KANJI FINISHED    (FF00)                *
                                        ;*                KANJI ENTERED     (FF40)                *
                                        ;*                KANJI CHANGED     (FF80)                *
                                        ;*                KANJI INITIALIZED (FF20)                *
                                        ;*                KANJI ON          (FF01)                *
                                        ;*                KANJI OFF         (FF02)                *
                                        ;*                                                      *
                                        ;*    OUTPUT: KANA-KAN COMMON TABLES                    *
                                        ;*                                                      *
                                        ;*    RETURN CODES: NONE                                *
                                        ;*                                                      *
                                        ;*   EXTERNAL REFERENCES:                               *
                                        ;*                                                      *
                                        ;*     ROUTINES: KKINIT - KANA-KAN INITIALIZATION       *
                                        ;*               KKNOML - NORMAL MODE CONTROL           *
                                        ;*               KKZCTL - ZENKOUHO CONTROL              *
                                        ;*               KKBCTL - BANGOU CONTROL                *
                                        ;*               KKWOIL - WRITE OPERATION INFORMATION LINE *
                                        ;*                                                      *
                                        ;*     TABLES: KANA-KAN COMMON TABLES                   *
                                        ;*                                                      *
                                        ;*   REGISTERS: ALL UNCHANGED                           *
                                        ;*                                                      *
                                        ;*   CHANGE ACTIVITY: VERSION 00.00                     *
                                        ;*                                                      *
                                        ;*******************************************************
                                        ;
                                                ASSUME  DS:KKDATA
                                        ;
                                        ;********** KKKFDM **********************************
                                        ;*
                                        ;*      ENTRY OF KKKFDM                                 *
                                        ;*                                                      *
                                        ;*******************************************************
                                        ;
0000                                    KKKFDM          PROC    NEAR
                                        ;
0000  1E                                        PUSH    DS              ;SAVE REGISTERS
0001  06                                        PUSH    ES
0002  55                                        PUSH    BP
0003  57                                        PUSH    DI
0004  56                                        PUSH    SI
0005  52                                        PUSH    DX
0006  51                                        PUSH    CX
0007  53                                        PUSH    BX
0008  50                                        PUSH    AX
0009  BB ---- R                               MOV     BX,SEG KKDATA   ;LOAD KANA-KAN DATA SEGMENT
000C  8E DB                                     MOV     DS,BX           ;GET KANA-KAN DATA SEGMENT
000E  8E C3                                     MOV     ES,BX
0010  3D 6B20                                  CMP     AX,KYKANJX      ;--- MODIFYED FOR KANA-KAN PERFORMANCE UP
0013  74 0A                                     JE      KFD000          ;
0015  3D FF80                                   CMP     AX,KCHNGX       ;
0018  74 05                                     JE      KFD000          ; CHECK SCAN CODE THAT REQUIRE INITIALIZE.
001A  3D FF20                                   CMP     AX,KINITX       ;
001D  75 03                                     JNE     KFD010          ;
001F                                    KFD000:
001F  E8 0219 R                                 CALL    KKINIT          ;KANA-KAN INITIAL PROCESS
0022                                    KFD010:
0022  8A 16 00F9 R                              MOV     DL,KKMODE       ;LOAD KANA-KAN MODE
0026  3D FF02                                   CMP     AX,KOFFX        ;REQUIREMENT IS KANJI OFF?
0029  74 0D                                     JE      KFD012          ; YES. GOTO
002B  3D FF01                                   CMP     AX,KONX         ;REQUIREMENT IS KANJI ON ?
002E  75 15                                     JNE     KFD014          ; NO. GOTO
0030  80 26 0195 R FE                           AND     KKFLAG,KON      ;SET KANJI ON FLAG
0035  EB 56 90                                  JMP     KFDRTN
0038                                    KFD012:
0038  80 0E 0195 R 01                           OR      KKFLAG,KOFF     ;SET KANJI OFF FLAG
003D  80 FA 00                                  CMP     DL,NORMALM      ;NORMAL MODE ?
0040  74 4B                                     JE      KFDRTN          ; YES. GOTO
0042  B8 6B20                                  MOV     AX,KYKANJX      ;LOAD KANJI KEY CODE (FINISH KANJI)
0045                                    KFD014:
0045  88 16 0194 R                              MOV     OLDMODE,DL      ;SAVE KANA-KAN MODE
0049  3D 6B81                                   CMP     AX,KYKANJ1      ;HIGH BYTE OF ZENKAKU KANJI KEY ?
004C  74 3F                                     JE      KFDRTN          ; YES. GOTO
004E  80 FA 00                                  CMP     DL,NORMALM      ;NORMAL MODE ?
0051  74 2F                                     JE      KFD030          ; YES. GOTO
0053  8B D8                                     MOV     BX,AX
0055  E8 132D R                                 CALL    KCALCCW         ;CALCULATE KKWCCA
0058  81 FB 6D81                                CMP     BX,KYHENK1      ;HIGH BYTE OF ZENKAKU HENKAKN KEY ?
005C  74 2F                                     JE      KFDRTN          ; YES. GOTO
005E  81 FB 6C81                                CMP     BX,KYMUHE1      ;HIGH BYTE OF ZENKAKU MUHENKAN KEY ?
0062  74 29                                     JE      KFDRTN          ; YES. GOTO
0064  E8 0097 R                                 CALL    CSCAN           ;CHANGE SCAN CODE
0067  F6 06 0000 R 20                           TEST    DSTAT,ZEN1F     ;HIGH BYTE OF ZENKAKU ?
006C  75 1F                                     JNZ     KFDRTN                  ; YES. GOTO
006E  80 FA 02                                  CMP     DL,KANJIM       ;KANJI MODE ?
0071  74 0A                                     JE      KFD020          ; YES. GOTO
0073  80 FA 04                                  CMP     DL,SELECTM      ;KANJI SELECT MODE ?
0076  74 05                                     JE      KFD020          ; YES. GOTO
0078  E8 0379 R                                 CALL    KKBCTL          ;BANGOU MODE PROCESS
007B  EB 08                                     JMP     SHORT KFD040
007D                                    KFD020:
007D  E8 05EF R                                 CALL    KKZCTL          ;KANJI MODE PROCESS
0080  EB 03                                     JMP     SHORT KFD040
0082                                    KFD030:
0082  E8 033E R                                 CALL    KKNOML          ;NORMAL MODE PROCESS
0085                                    KFD040:
0085  3D 0000                                   CMP     AX,RCNOMAL       ;RETURN CODE IS NORMAL ?
0088  74 03                                     JE      KFDRTN          ; YES. GOTO
008A  E8 0212 R                                 CALL    SPEAKER         ;SOUND SPEAKER
008D                                    KFDRTN:
```

```
008D  58                        POP     AX                      ;RESTOR REGISTER
008E  5B                        POP     BX
008F  59                        POP     CX
0090  5A                        POP     DX
0091  5E                        POP     SI
0092  5F                        POP     DI
0093  5D                        POP     BP
0094  07                        POP     ES
0095  1F                        POP     DS
0096  CF                        IRET                            ;RETURN TO CALLER
0097                    KKKFDM  ENDP
                        ;
                        ;
                        ;
                        ;**** CSCAN ***************************************************************
                        ;*                                                                         *
                        ;*          CHANGE SCAN CODE                                               *
                        ;*                                                                         *
                        ;*                BX : SCAN CODE/CHARACTER                                 *
                        ;*                                                                         *
                        ;***************************************************************************
                        ;
0097                    CSCAN           PROC    NEAR
0097  52                        PUSH    DX
0098  8A 16 0000 R              MOV     DL,DSTAT                ;LOAD DATA STATUS
009C  80 FA 20                  CMP     DL,ZEN1F                ;DATA IS LOW BYTE OF ZENKAKU ?
009F  75 03                     JNE     CSC001                  ; NO. GOTO
00A1                    CSC000:
00A1  E9 0143 R                 JMP     CSC080
00A4                    CSC001:
00A4  E8 0166 R                 CALL    HANDAKU                 ;DAKUTEN OR HANDAKUTEN PROCESS
00A7  3C 00                     CMP     AL,0                    ;DAKUTEN OR HANDAKUTEN CHARACTER ?
00A9  74 03                     JE      CSC002                  ; NO. GOTO
00AB  E9 015D R                 JMP     CSC090
00AE                    CSC002:
00AE  80 FB 00                  CMP     BL,FUNCAL               ;DATA IS FUNCTION KEY ?
00B1  74 68                     JE      CSC010                  ; YES. GOTO
00B3  80 FF 00                  CMP     BH,ZENMAH               ;DATA IS ZENMEN KEY ?
00B6  74 67                     JE      CSC020                  ; YES. GOTO
00B8  80 FF FF                  CMP     BH,KANJAH               ;DATA IS KANJI REQUIREMENT ?
00BB  74 66                     JE      CSC030                  ; YES. GOTO
00BD  80 FB 82                  CMP     BL,ZENHIRA              ;DATA IS HIRAGANA KEY ?
00C0  74 65                     JE      CSC040                  ; YES. GOTO
00C2  80 FB 83                  CMP     BL,ZENKATA              ;DATA IS KATAKANA KEY ?
00C5  74 60                     JE      CSC040                  ; YES. GOTO
00C7  80 F3 81                  CMP     BL,ZENMARK              ;DATA IS HIGH BYTE OF ZENKAKU ?
00CA  74 5B                     JE      CSC040                  ; YES. GOTO
00CC  80 FF 02                  CMP     BH,SCAN02
00CF  72 05                     JB      CSC003
00D1  80 FF 0D                  CMP     BH,SCAN0D               ;DATA IS GRAPHIC (HANKAKU) ?
00D4  76 3C                     JBE     CSC009                  ; YES. GOTO
00D6                    CSC003:
00D6  80 FF 10                  CMP     BH,SCAN10
00D9  72 05                     JB      CSC004
00DB  80 FF 1B                  CMP     BH,SCAN1B               ;DATA IS GRAPHIC (HANKAKU) ?
00DE  76 32                     JBE     CSC009                  ; YES. GOTO
00E0                    CSC004:
00E0  80 FF 1E                  CMP     BH,SCAN1E
00E3  72 05                     JB      CSC005
00E5  80 FF 29                  CMP     BH,SCAN29               ;DATA IS GRAPHIC (HANKAKU) ?
00E8  76 28                     JBE     CSC009                  ; YES. GOTO
00EA                    CSC005:
00EA  80 FF 2B                  CMP     BH,SCAN2B
00ED  72 05                     JB      CSC006
00EF  80 FF 35                  CMP     BH,SCAN35               ;DATA IS GRAPHIC (HANKAKU) ?
00F2  76 1E                     JBE     CSC009                  ; YES. GOTO
00F4                    CSC006:
00F4  80 FF 39                  CMP     BH,SCAN39               ;DATA IS GRAPHIC (HANKAKU) ?
00F7  74 19                     JE      CSC009                  ; YES. GOTO
00F9  80 FF 4A                  CMP     BH,SCAN4A               ;DATA IS GRAPHIC (HANKAKU) ?
00FC  74 14                     JE      CSC009                  ; YES. GOTO
00FE  80 FF 4E                  CMP     BH,SCAN4E               ;DATA IS GRAPHIC (HANKAKU) ?
0101  74 0F                     JE      CSC009                  ; YES. GOTO
0103  80 FF 6A                  CMP     BH,SCAN6A               ;DATA IS GRAPHIC (HANKAKU) ?
0106  74 0A                     JE      CSC009                  ; YES. GOTO
0108                    CSC007:
0108  80 FF 70                  CMP     BH,SCAN70
010B  72 0E                     JB      CSC010
010D  80 FF 7D                  CMP     BH,SCAN7D               ;DATA IS GRAPHIC (HANKAKU) ?
0110  77 09                     JA      CSC010                  ; NO. GOTO
                        ;*-------------------------------------------------------------------------*
                        ;*      HANKAKU PROCESS                                                    *
                        ;*-------------------------------------------------------------------------*
0112                    CSC009:
0112  80 FB 1F                  CMP     BL,CTRL_X               ;CTRL+X KEY ?
0115  76 04                     JBE     CSC010                  ; YES. GOTO
0117  B0 01                     MOV     AL,HANKAF               ;HANKAKU FLAG ON
0119  EB 14                     JMP     SHORT CSC060
011B                    CSC010:
011B  B0 04                     MOV     AL,FUNCF                ;FUNCTION FLAG ON
011D  EB 10                     JMP     SHORT CSC060
011F                    CSC020:
011F  B0 08                     MOV     AL,ZENMENF              ;ZENMEN FLAG ON
0121  EB 0C                     JMP     SHORT CSC060
0123                    CSC030:
0123  B0 10                     MOV     AL,KANJIF               ;KANJI FLAG ON
0125  EB 08                     JMP     SHORT CSC060
0127                    CSC040:
0127  B0 20                     MOV     AL,ZEN1F                ;ZENKAKU 1 FLAG ON
0129  8A 26 00E1 R              MOV     AH,ZENATTR1             ;SET ZENKAKU-1 ATTRIBUTE
012D  EB 02                     JMP     SHORT CSC070
012F                    CSC060:
012F  B4 00                     MOV     AH,HANATTR              ;SET HANKAKU ATTRIBUTE
```

```
                                CSC070:
                                        MOV     CHAR1,BL
0131                                    MOV     ATTR1,AH        ;SET HANKAKU CHARACTER
0131  88 1E 0003 R                      MOV     SCAN1,BH        ;SET ATTRIBUTE
0135  88 26 0004 R                      MOV     DSTAT,AL        ;SET SCAN CODE
0139  88 3E 0005 R                      JMP     CSC090          ;SET DATA STATUS
013D  A2 0000 R                 ;*-------------------------------------------------------------------*
0140  EB 1B 90                  ;*      ZENKAKU PROCESS                                              *
                                ;*-------------------------------------------------------------------*
                                CSC080:                                                             *
0143                                    CALL    HANDAKU
0143  E8 0166 R                         CMP     AL,0            ;DAKUTEN OR HANDAKUTEH PROCESS
0146  3C 00                             JNE     CSC090          ;DAKUTEN OR HANDAKUTEN ?
0148  75 13                             MOV     CHAR2,BL        ; YES. GOTO
014A  88 1E 0006 R                      MOV     AL,ZENATTR2     ;SET ZENKAKU CHARACTER
014E  A0 00E2 R                         MOV     ATTR2,AL
0151  A2 0007 R                         MOV     SCAN2,BH        ;SET ZENKAKU-2 ATTRIBUTE
0154  88 3E 0008 R                      MOV     DSTAT,ZEN2F     ;SET SCAN CODE
0158  C6 06 0000 R 02                   ;                       ;ZENKAKU FLAG ON
                                ;*-------------------------------------------------------------------*
                                ;*      RETURN TO CALLER                                             *
                                ;*-------------------------------------------------------------------*
                                CSC090:                                                             *
015D                                    MOV     AH,D02
015D  B4 02                             INT     16H
015F  CD 16                             MOV     KBDSTAT,AX      ;READ KBD STATUS
0161  A3 0001 R                         POP     DX              ;SET KANA-KAN KBD STATUS
0164  5A                                RET                     ;RETURN TO CALLER
0165  C3                        CSCAN   ENDP
0166                            ;**** HANDAKU ***************************************************
                                ;*
                                ;*      DAKUTEN/HANDAKUTEN PROCESS                                   *
                                ;*                                                                   *
                                ;*********************************************************************
0166                            HANDAKU PROC
0166  8B 3E 00E6 R                       MOV     DI,KKWCCA       ;LOAD CURSOR POINTER IN KKINBUF
016A  83 EF 06                          SUB     DI,D06          ;CURSOR POINTER IS AT TOP ?
016D  72 78                             JB      HAN080          ; YES. GOTO
016F  8A A5 0009 R                      MOV     AH,KKINBUF[DI]          ;LOAD HIGH BYTE OF ZENKAKU
0173  8A 85 000A R                      MOV     AL,KKINBUF[DI]+1        ;LOAD ATTRIBUTE-1
0177  3A 06 00E1 R                      CMP     AL,ZENATTR1     ;DATA IS ZENKAKU ?
017B  75 6A                             JNE     HAN080          ; NO. GOTO
017D  80 FC 82                          CMP     AH,ZENHIRA      ;HIRAGANA ?
0180  74 05                             JE      HAN000          ; YES. GOTO
0182  80 FC 83                          CMP     AH,ZENKATA      ;KATAKANA ?
0185  75 60                             JNE     HAN080          ; NO. GOTO
0187                            HAN000:
0187  8A 85 000C R                      MOV     AL,KKINBUF[DI]+3        ;LOAD LOW BYTE OF ZENKAKU
018B  81 FB 1ADE                        CMP     BX,KYDAKU1
018F  74 1C                             JE      HAN040
0191  81 FB 1A4A                        CMP     BX,KYDAKU2      ;DAKUTEN ?
0195  74 16                             JE      HAN040          ; YES. GOTO
0197  81 FB 1BDF                        CMP     BX,KYHAND1
019B  74 06                             JE      HAN005
019D  81 FB 1B4B                        CMP     BX,KYHAND2      ;HANDAKUTEN ?
01A1  75 44                             JNE     HAN080          ; NO. GOTO
                                ;*-------------------------------------------------------------------*
                                ;*      HANDAKUTEN PROCESS                                           *
                                ;*-------------------------------------------------------------------*
01A3                            HAN005:
01A3  BE 0208 R                         MOV     SI,OFFSET HAGYOH        ;LOAD HA GYOU DATA OFFSET
01A6  B9 000A                           MOV     CX,HANDNUM      ;LOAD NUMBERS OF HANDAKUTEN CHARACTER
01A9  B6 02                             MOV     DH,D02
01AB  EB 08                             JMP     SHORT HAN050    ;GOTO CARACTER CHECK
                                ;*-------------------------------------------------------------------*
                                ;*      DAKUTEN PROCESS                                              *
                                ;*-------------------------------------------------------------------*
01AD                            HAN040:
01AD  BE 01EA R                         MOV     SI,OFFSET KAGYOH        ;LOAD KA GYOU DATA OFFSET
01B0  B9 0028                           MOV     CX,DAKUNUM      ;LOAD NUMBERS OF DAKUTEN CHARACTER
01B3  B6 01                             MOV     DH,D01
01B5                            HAN050:
01B5  2E: 3A 04                         CMP     AL,CS: [SI]     ;CHECK DAKUTEN CHARACTER ?
01B8  74 05                             JE      HAN060          ; YES. GOTO
01BA  46                                INC     SI
01BB  E2 F8                             LOOP    HAN050
01BD  EB 28                             JMP     SHORT HAN080    ;UNMATCH. GOTO
01BF                            HAN060:
01BF  02 C6                             ADD     AL,DH           ;MAKE DAKUTEN/HANDAKUTEN CHARACTER
01C1  88 26 0003 R                      MOV     CHAR1,AH        ;SET HIGH BYTE OF ZENKAKU
01C5  A2 0006 R                         MOV     CHAR2,AL        ;SET LOW BYTE OF ZENKAKU
01C8  A0 00E1 R                         MOV     AL,ZENATTR1
01CB  A2 0004 R                         MOV     ATTR1,AL        ;SET ATTRIBUTE-1
01CE  A0 00E2 R                         MOV     AL,ZENATTR2
01D1  A2 0007 R                         MOV     ATTR2,AL        ;SET ATTRIBUTE-2
01D4  8A 85 000B R                      MOV     AL,KKINBUF[DI]+2        ;LOAD SCAN CODE
01D8  A2 0005 R                         MOV     SCAN1,AL        ;SET SCAN CODE
01DB  A2 0008 R                         MOV     SCAN2,AL        ;SET SCAN CODE
01DE  C6 06 0000 R 82                   MOV     DSTAT,ZEN2F+DAKUF       ;SET ZENKAKU DAKUTEN/HANDAKUTEN FLAG
01E3  B0 01                             MOV     AL,D01          ;SET RETURN CODE OF DAKUTEN/HANDAKUTEN
01E5  EB 02                             JMP     SHORT HANRTN
01E7                            HAN080:
01E7  32 C0                             XOR     AL,AL           ;SET RETURN CODE
01E9                            HANRTN:
01E9  C3                                RET                     ;RETURN TO CALLER
01EA                            HANDAKU ENDP
                                ;*-------------------------------------------------------------------*
                                ;*      DAKUTEN/HANDAKUTEN CHARACTER                                 *
                                ;*-------------------------------------------------------------------*
01EA  A9 AB AD AF B1            KAGYOH: DB      0A9H,0ABH,0ADH,0AFH,0B1H
01EF  B3 B5 B7 B9 BB            SAGYOH: DB      0B3H,0B5H,0B7H,0B9H,0BBH
01F4  BD BF C2 C4 C6            TAGYOH: DB      0BDH,0BFH,0C2H,0C4H,0C6H
01F9  4A 4C 4E 50 52            KAGYOK: DB      4AH,4CH,4EH,50H,52H
01FE  54 56 58 5A 5C            SAGYOK: DB      54H,56H,58H,5AH,5CH
```

```
0203  5E 60 63 65 67          5EH,60H,63H,65H,67H
0208  CD D0 D3 D6 D9  TAGYOH: DB      0CDH,0D0H,0D3H,0D6H,0D9H
020D  6E 71 74 77 7A  HAGYOH: DB      6EH,71H,74H,77H,7AH
                      HAGYOK: DB
                      ;**** SPEAKER ************************************************
                      ;*
                      ;*      SOUND SPEAKER FOR INVALID OPERATION                    *
                      ;*                                                             *
                      ;*******************************************************************
0212                  SPEAKER PROC
0212  B0 07                    MOV     AL,BEL
0214  B4 0E                    MOV     AH,0EH
0216  CD 10                    INT     10H             ;SPEAKER ON
0218  C3                      RET                     ;RETURN TO CALLER
0219                  SPEAKER ENDP
                      ;*******************************************************************
                      ;*                                                                 *
                      ;* PROGRAM NAME: KKINIT                                            *
                      ;*                                                                 *
                      ;* DESCRIPTIVE NAME: KANA-KAN INITIALIZATION                       *
                      ;*                                                                 *
                      ;* FUNCTION: THIS ROUTINE IS INITIALIZATION FOR KANA-KANJI HENKAN. *
                      ;*           IT SETS TELEVISION MODE.                              *
                      ;*                                                                 *
                      ;* LINKAGE: CALL                                                   *
                      ;*                                                                 *
                      ;*   INPUT: NONE                                                   *
                      ;*                                                                 *
                      ;*   OUTPUT: KANA-KAN COMMON TABLES                                *
                      ;*                                                                 *
                      ;*   RETURN CODES: NONE                                            *
                      ;*                                                                 *
                      ;*  EXTERNAL REFERENCES:                                           *
                      ;*                                                                 *
                      ;*    ROUTINES: NONE                                               *
                      ;*                                                                 *
                      ;*    TABLES: KANA-KAN COMMON TABLES                               *
                      ;*                                                                 *
                      ;*  REGISTERS: BX,CX,DX,DI - WORK                                  *
                      ;*             ALL OTHERS UNCHANGED                                *
                      ;*                                                                 *
                      ;*  CHANGE ACTIVITY: VERSION 00.00                                 *
                      ;*                                                                 *
                      ;*******************************************************************
                      ;
                      ;********************* KKINIT ************************************
                      ;*                                                                 *
                      ;*      ENTRY OF KKINIT                                            *
                      ;*                                                                 *
                      ;*******************************************************************
0219                  KKINIT  PROC    NEAR
0219  50                      PUSH    AX
021A  50                      PUSH    AX
                      ;*-----------------------------------------------------------------*
                      ;*      GET TV MODE                                                *
                      ;*-----------------------------------------------------------------*
021B  B4 0F                   MOV     AH,0FH
021D  CD 10                   INT     10H             ;READ DISPLAY STATUS
021F  8B D0                   MOV     DX,AX
0221  CD 11                   INT     11H             ;READ EQUIPMENT
0223  5B                      POP     BX
0224  24 30                   AND     AL,030H         ;IGNORE MEANINGLESS BIT

0226  3C 20                   CMP     AL,020H         ;SYSTEM MODE IS 5550 EM ?
0228  74 0D                   JE      INI002          ; YES. GOTO
022A  80 FE 50                CMP     DH,80           ;TV MODE IS 80*11 ?
022D  74 04                   JE      INI000          ; YES. GOTO
022F  B2 03                   MOV     DL,TV4011       ;LOAD 40*11 MODE
0231  EB 23                   JMP     SHORT INI006
0233                  INI000:
0233  B2 02                   MOV     DL,TV8011       ;LOAD 80*11 MODE
0235  EB 1F                   JMP     SHORT INI006
0237                  INI002:
0237  80 26 0195 R FD          AND     KKFLAG,0FDH     ;
023C  80 FA 0B                CMP     DL,0BH          ;COLOR  GRAPHIC MODE ?
023F  74 0E                   JE      INI004          ; YES. GOTO
0241  80 FA 0E                CMP     DL,0EH          ;COLOR TEXT MODE ?
0244  75 05                   JNE     INI003          ; NO. GOTO
0246  80 0E 0195 R 02         OR      KKFLAG,CLRF25   ;SET 80*25 COLOR FLAG
024B                  INI003:
024B  B2 01                   MOV     DL,TV8025       ;LOAD 80*25 MODE
024D  EB 07                   JMP     SHORT INI006
024F                  INI004:
024F  80 0E 0195 R 02         OR      KKFLAG,CLRF25   ;SET 80*25 COLOR FLAG
0254  B2 03                   MOV     DL,TV4011       ;LOAD 40*25 MODE
0256                  INI006:
0256  81 FB FF20              CMP     BX,KINITX       ;REQUIREMENT IS INITIAL ?
025A  74 09                   JE      INI008          ; YES. GOTO
025C  38 16 00FA R            CMP     TVMODE,DL       ;CHANGE TV MODE ?
0260  75 03                   JNE     INI008          ; YES. GOTO
0262  E9 033C R               JMP     INIRTN
0265                  INI008:
0265  BF 0000 R               MOV     DI,OFFSET DSTART ;LOAD KANA-KAN CONTROL TABLE OFFSET
0268  B9 0195                 MOV     CX,DEND         ;CAPACITY OF KANA-KAN CONTROL TABLE
026B  32 C0                   XOR     AL,AL           ;LOAD 00H
026D                  INI009:
026D  AA                      STOSB                   ;CLEAR KANA-KAN CONTROL TABLE
026E  E2 FD                   LOOP    INI009
0270  80 26 0195 R 03         AND     KKFLAG,03H      ;CLEAR KKFLAG WITHOUT KANA-KAN ON/OFF FLAG
0275  88 16 00FA R            MOV     TVMODE,DL       ;SET TV MODE
0279  C7 06 00E4 R 0020       MOV     SPACE,BLANK     ;SET HANKAKU SPACE DATA
027F  80 FA 03                CMP     DL,TV4011       ;TV MODE IS 40*11 ?
0282  74 3F                   JE      TV4011L         ; YES. GOTO
0284  80 FA 02                CMP     DL,TV8011       ;TV MODE IS 80*11 ?
```

```
0287  74 7B                        JE      TV8011L         ; YES. GOTO
                                ;*-------------------------------------------------------------------*
                                ;*      TV MODE = 80*25                                              *
                                ;*-------------------------------------------------------------------*
0289
0289  C6 06 00FB R 50           TV8025L:
                                        MOV     TVLINE,TVL80    ;SET NUMBERS OF CARACTER IN LINE
028E  C6 06 00FC R 18                   MOV     KKOILP,OILP25   ;SET OIL POSITION
0293  C7 06 00F3 R 1811                 MOV     KKCUSR,KCSRP4025 ;SET KANJI CURSOR POSITION
0299  C6 06 00E1 R 01                   MOV     ZENATTR1,ZATTR15        ;SET ZENKAKU ATTRIBUTE 1
029E  C6 06 00E2 R 03                   MOV     ZENATTR2,ZATTR25        ;SET ZENKAKU ATTRIBUTE 2
02A3  C6 06 0193 R FC                   MOV     ATTRMSK,ATMSK25 ;SET ATTRIBUTE MASK PATTERN
02A8  C6 06 00E3 R 00                   MOV     ROATCNT,ROT25   ;SET ROTATE COUNTER
02AD  C6 06 0101 R 0C                   MOV     DOILP,DOILP11   ;SET DATA POSITION OF OIL
02B2  C6 06 0102 R 44                   MOV     DOILN,DOILN11   ;SET NUMBERS OF OIL DATA
02B7  C6 06 0103 R 0C                   MOV     BANGOUP,BANP11  ;SET BANGOU DATA POSITION
02BC  C6 06 0104 R 1B                   MOV     BASTERP,BAST11  ;SET ASTERISK POSITION
02C1  EB 79                             JMP     SHORT INIRTN
                                ;*-------------------------------------------------------------------*
                                ;*      TV MODE = 40*11 , 40*25                                      *
                                ;*-------------------------------------------------------------------*
02C3
02C3  C6 06 00FB R 28           TV4011L:
                                        MOV     TVLINE,TVL40    ;SET NUMBERS OF CARACTER IN LINE
02C8  F6 06 0195 R 02                   TEST    KKFLAG,CLRF25   ;5550EM COLOR GRAPHIC MODE (40*25) ?
02CD  74 3A                             JZ      INI010          ; NO. GOTO
                                ;*----- 40*25
02CF  C6 06 00FC R 18                   MOV     KKOILP,OILP25   ;SET OIL POSITION
02D4  C7 06 00F3 R 1811                 MOV     KKCUSR,KCSRP4025        ;SET KANJI CURSOR POSITION
02DA  C6 06 00E1 R 01                   MOV     ZENATTR1,ZATTR15        ;SET ZENKAKU ATTRIBUTE 1
02DF  C6 06 00E2 R 03                   MOV     ZENATTR2,ZATTR25        ;SET ZENKAKU ATTRIBUTE 2
02E4  C6 06 0193 R FC                   MOV     ATTRMSK,ATMSK25 ;SET ATTRIBUTE MASK PATTERN
02E9  C6 06 00E3 R 00                   MOV     ROATCNT,ROT25   ;SET ROTATE COUNTER
02EE  C6 06 0101 R 0C                   MOV     DOILP,DOILP11   ;SET DATA POSITION OF OIL
02F3  C6 06 0102 R 44                   MOV     DOILN,DOILN11   ;SET NUMBERS OF OIL DATA
02F8  C6 06 0103 R 0C                   MOV     BANGOUP,BANP11  ;SET BANGOU DATA POSITION
02FD  C6 06 0104 R 1B                   MOV     BASTERP,BAST11  ;SET ASTERISK POSITION
0302  EB 38                             JMP     SHORT INIRTN
                                ;*-------------------------------------------------------------------*
                                ;*      TV MODE = 80*11                                              *
                                ;*-------------------------------------------------------------------*
0304
0304  C6 06 00FB R 50           TV8011L:
                                        MOV     TVLINE,TVL80    ;SET NUMBERS OF CARACTER IN LINE
0309                            INI010:
0309  C6 06 00FC R 0A                   MOV     KKOILP,OILP11   ;SET OIL POSITION
030E  C7 06 00F3 R 0A11                 MOV     KKCUSR,KCSRP11  ;SET KANJI CURSOR POSITION
0314  C6 06 00E1 R 80                   MOV     ZENATTR1,ZATTR1N        ;SET ZENKAKU ATTRIBUTE 1
0319  C6 06 00E2 R 88                   MOV     ZENATTR2,ZATTR2N        ;SET ZENKAKU ATTRIBUTE 2
031E  C6 06 0193 R 77                   MOV     ATTRMSK,ATMSK11 ;SET ATTRIBUTE MASK PATTERN
0323  C6 06 00E3 R 01                   MOV     ROATCNT,ROT11   ;SET ROTATE COUNTER
0328  C6 06 0101 R 0C                   MOV     DOILP,DOILP11   ;SET DATA POSITION OF OIL
032D  C6 06 0102 R 44                   MOV     DOILN,DOILN11   ;SET NUMBERS OF OIL DATA
0332  C6 06 0103 R 0C                   MOV     BANGOUP,BANP11  ;SET BANGOU DATA POSITION
0337  C6 06 0104 R 1B                   MOV     BASTERP,BAST11  ;SET ASTERISK POSITION
                                ;*-------------------------------------------------------------------*
033C                            INIRTN:
033C  58                                POP     AX              ;POP SCAN/CHARACTER CODE
033D  C3                                RET
033E                            KKINIT  ENDP
                                ;*********************************************************************
                                ;*                                                                   *
                                ;* PROGRAM NAME: KKNOML                                              *
                                ;*                                                                   *
                                ;* DESCRIPTIVE NAME: NORMAL MODE CONTROL CSECT                       *
                                ;*                                                                   *
                                ;* FUNCTION: THIS ROUTINE SETS AX TO QUEUING BUFFER IN NORMAL MODE.  *
                                ;*           WHEN KANJI KEY IS PRESSED, IT INITIALIZES KANA-KAN MODE.*
                                ;*                                                                   *
                                ;* LINKAGE: CALL                                                     *
                                ;*                                                                   *
                                ;*   INPUT: KANA-KAN COMMON TABLES                                   *
                                ;*                                                                   *
                                ;*   OUTPUT: QUEUING BUFFER                                          *
                                ;*                                                                   *
                                ;*   RETURN CODES: (AX)                                              *
                                ;*                                                                   *
                                ;*      0 - SUCCESSFUL                                               *
                                ;*      1 - INVALID OPERATION                                        *
                                ;*                                                                   *
                                ;*  EXTERNAL REFERENCES:                                             *
                                ;*                                                                   *
                                ;*    ROUTINES: KKZINT - ZENKOUHO INITIALIZATION                     *
                                ;*              KKWOIL - CHANGE INDICATOR                            *
                                ;*                                                                   *
                                ;*    TABLES: KANA-KAN COMMON TABLES                                 *
                                ;*                                                                   *
                                ;*  REGISTERS: AX - RETURN CODE                                      *
                                ;*             BX - WORK                                             *
                                ;*             ALL OTHERS UNCHANGED                                  *
                                ;*                                                                   *
                                ;*  CHANGE ACTIVITY: VERSION 00.00                                   *
                                ;*                                                                   *
                                ;*                                                                   *
                                ;*********************************************************************
                                ;
                                ;********************* KKNOML *****************************************
                                ;*                                                                   *
                                ;*      ENTRY OF KKNOML                                              *
                                ;*                                                                   *
                                ;*********************************************************************
033E                            KKNOML  PROC
033E  F6 06 0195 R 01                   TEST    KKFLAG,KOFF     ;DURRING KANJI OFF ?
0343  75 14                             JNZ     NOR000          ; YES. GOTO
0345  3D FF40                           CMP     AX,KENTRX
0348  74 22                             JE      NOR010
034A  3D FF80                           CMP     AX,KCHNGX
```

```
034D  74 1D                  JE      NOR010          ;ENTER KANJI MODE ?
034F  3D 6B20                CMP     AX,KYKANJX      ; YES. GOTO
0352  74 18                  JE      NOR010          ;ENTER KANJI MODE ?
0354  3D 6B40                CMP     AX,KYKANJ2      ; YES. GOTO
0357  74 13                  JE      NOR010
0359                 NOR000:
0359  80 FC FF               CMP     AH,KANJAH       ;OTHER KANJI REQUIREMENT ?
035C  74 18                  JE      NOR020          ; YES. GOTO
035E  3D 6B00                CMP     AX,SCAN6B0
0361  74 05                  JE      NOR001
0363  80 FC 6B               CMP     AH,SCAN6B       ;KANJI KEY ?
0366  74 0E                  JE      NOR020          ; YES. GOTO
0368                 NOR001:
0368  CD 79                  INT     79H             ;BUFFER QUEUING
036A  EB 0A                  JMP     SHORT NOR020
036C                 NOR010:
036C  B3 80                  MOV     BL,D80
036E  E8 1139 R              CALL    KKWOIL          ;INITIALIZE INDICATOR
0371  E8 0CA2 R              CALL    KKZINT          ;KANJI MODE INITIALIZTION
0374  EB 02                  JMP     SHORT NORRTN
0376                 NOR020:
0376  33 C0                  XOR     AX,AX           ;SET NORMAL RETURN CODE
0378                 NORRTN:
0378  C3                     RET                     ;RETURN TO CALLER
0379                 KKNOML  ENDP
                     ;*****************************************************************
                     ;*                                                               *
                     ;* PROGRAM NAME: KKBCTL                                          *
                     ;*                                                               *
                     ;* DESCRIPTIVE NAME: BANGOU CONTROL CSECT                        *
                     ;*                                                               *
                     ;* FUNCTION: THIS ROUTINE BRANCHES TO EACH PROCESS THAT IS THE   *
                     ;*           GRAPHIC, CURSOR OR FUNCTION.                        *
                     ;*                                                               *
                     ;* LINKAGE: CALL                                                 *
                     ;*                                                               *
                     ;*   INPUT: KANA-KAN COMMON TABLES                               *
                     ;*                                                               *
                     ;*   OUTPUT: NONE                                                *
                     ;*                                                               *
                     ;*   RETURN CODES: (AX)                                          *
                     ;*                                                               *
                     ;*      0 - SUCCESSFUL                                           *
                     ;*      1 - INVALID OPERATION                                    *
                     ;*                                                               *
                     ;*  EXTERNAL REFERENCES:                                         *
                     ;*                                                               *
                     ;*    ROUTINES: KKBGRP - BANGOU GRAPHIC KEYS                     *
                     ;*              KKCSR  - BANGOU CURSOR KEYS                      *
                     ;*              KKBFNC - BANGOU FUNCTION KEYS                    *
                     ;*                                                               *
                     ;*    TABLES: KANA-KAN COMMON TABLES                             *
                     ;*                                                               *
                     ;*  REGISTERS: AX - RETURN CODE                                  *
                     ;*             ALL OTHERS UNCHANGED                              *
                     ;*                                                               *
                     ;*  CHANGE ACTIVITY: VERSION 00.00                               *
                     ;*                                                               *
                     ;*                                                               *
                     ;*****************************************************************
                     ;
                     ;********************** KKBCTL ***********************************
                     ;*                                                               *
                     ;*      ENTRY OF KKBCTL                                          *
                     ;*                                                               *
                     ;*****************************************************************
0379                 KKBCTL  PROC
0379  F6 06 0000 R 83        TEST    DSTAT,HANKAF+ZEN2F+DAKUF        ;GRAPHIC KEY ?
037E  74 05                  JZ      BCT010          ; NO. GOTO
0380  E8 039F R              CALL    KKBGRP          ;GRAPHIC KEY PROCESS
0383  EB 19                  JMP     SHORT BCTRTN
0385                 BCT010:
0385  8A 26 0005 R           MOV     AH,SCAN1
0389  A0 0003 R              MOV     AL,CHAR1        ;LOAD SCAN CODE/CHARACTER
038C  3D 4B00                CMP     AX,KYCSRLX      ;CURSOR LEFT KEY ?
038F  74 0A                  JE      BCT020          ; YES. GOTO
0391  3D 4D00                CMP     AX,KYCSRRX      ;CURSOR RIGHT KEY ?
0394  74 05                  JE      BCT020          ; YES. GOTO
0396  E8 03DE R              CALL    KKBFNC          ;FUNCTION KEY PROCESS
0399  EB 03                  JMP     SHORT BCTRTN
039B                 BCT020:
039B  E8 0C2E R              CALL    KKCSR           ;CUSOR KEY PROCESS
039E                 BCTRTN:
039E  C3                     RET                     ;RETURN TO CALLER
039F                 KKBCTL  ENDP
                     ;*****************************************************************
                     ;*                                                               *
                     ;* PROGRAM NAME: KKBGRP                                          *
                     ;*                                                               *
                     ;* DESCRIPTIVE NAME: BANGOU GRAPHIC KEYS                         *
                     ;*                                                               *
                     ;* FUNCTION: THIS ROUTINE DISPLAYS NUMERICAL DATA AND STACKS IT  *
                     ;*           IN INPUT BUFFER.                                    *
                     ;*                                                               *
                     ;* LINKAGE: CALL                                                 *
                     ;*                                                               *
                     ;*   INPUT: KANA-KAN COMMON TABLES                               *
                     ;*                                                               *
                     ;*   OUTPUT: KANA-KAN COMMON TABLES                              *
                     ;*                                                               *
                     ;*   RETURN CODES: (AX)                                          *
                     ;*                                                               *
                     ;*      0 - SUCCESSFUL                                           *
                     ;*      1 - INVALID OPERATION                                    *
```

```
                        ;*
                        ;*  EXTERNAL REFERENCES:                                    *
                        ;*                                                          *
                        ;*    ROUTINES: KKGRP - GRAPHIC KEYS PROCESS                *
                        ;*                                                          *
                        ;*    TABLES: KANA-KAN COMMON TABLES                        *
                        ;*                                                          *
                        ;*  REGISTERS: AX - RETURN CODE                             *
                        ;*             ALL OTHERS UNCHANGED                         *
                        ;*                                                          *
                        ;*  CHANGE ACTIVITY: VERSION 00.00                          *
                        ;*                                                          *
                        ;*                                                          *
                        ;************************************************************
                        ;
                        ;********************* KKBGRP *******************************
                        ;*                                                          *
                        ;*      ENTRY OF KKBGRP                                     *
                        ;*                                                          *
                        ;************************************************************
 039F                   KKBGRP  PROC
 039F  A0 0005 R                MOV     AL,SCAN1        ;LOAD SCAN CODE
 03A2  3C 02                    CMP     AL,SCAN02
 03A4  72 34                    JB      BGR050
 03A6  3C 0A                    CMP     AL,SCAN0A       ;NUMERICAL DATA (1<->9) ?
 03A8  77 04                    JA      BGR010          ; NO. GOTO
                        ;*----------------------------------------------------------*
                        ;*      MAKE DISPLAY DATA ( 0 <--> 9 )                      *
                        ;*----------------------------------------------------------*
 03AA  04 2F                    ADD     AL,NUM19M1      ;MAKE 1<-->9
 03AC  EB 1A                    JMP     SHORT BGR040
 03AE                   BGR010:
 03AE  3C 0B                    CMP     AL,SCAN0B       ;NUMERICAL DATA (0) ?
 03B0  75 04                    JNE     BGR020          ; NO. GOTO
 03B2  04 25                    ADD     AL,NUM0M1       ;MAKE 0
 03B4  EB 12                    JMP     SHORT BGR040
 03B6                   BGR020:
 03B6  3C 72                    CMP     AL,SCAN72
 03B8  72 08                    JB      BGR030
 03BA  3C 7A                    CMP     AL,SCAN7A       ;NUMERICAL DATA OF TEN-KEY (1<->9) ?
 03BC  77 1C                    JA      BGR050          ; NO. GOTO
 03BE  2C 41                    SUB     AL,NUM19M2      ;MAKE 1<-->9
 03C0  EB 06                    JMP     SHORT BGR040
 03C2                   BGR030:
 03C2  3C 70                    CMP     AL,SCAN70       ;NUMERICAL DATA OF TEN-KEY (0)
 03C4  75 14                    JNE     BGR050          ; NO. GOTO
 03C6  2C 40                    SUB     AL,NUM0M2       ;MAKE 0
                        ;*----------------------------------------------------------*
                        ;*      STACK AND DISPLAY                                   *
                        ;*----------------------------------------------------------*
 03C8                   BGR040:
 03C8  A2 0003 R                MOV     CHAR1,AL        ;SET NUMERICAL DATA
 03CB  C6 06 0004 R 00          MOV     ATTR1,HANATTR   ;SET HANKAKU ATTRIBUTE
 03D0  C6 06 0000 R 01          MOV     DSTAT,HANKAF    ;SET HANKAKU STATUS
 03D5  E8 1209 R                CALL    KKGRP           ;GRAPHIC KEY PROCESS
 03D8  EB 03                    JMP     SHORT BGRRTN
                        ;*----------------------------------------------------------*
                        ;*      ABNORMAL RETURN                                     *
                        ;*----------------------------------------------------------*
 03DA                   BGR050:
 03DA  B8 0001                  MOV     AX,001          ;SET ABNORMAL RETURN CODE
 03DD                   BGRRTN:
 03DD  C3                       RET                     ;RETURN TO CALLER
 03DE                   KKBGRP  ENDP
                        ;************************************************************
                        ;*                                                          *
                        ;* PROGRAM NAME: KKBFNC                                     *
                        ;*                                                          *
                        ;* DESCRIPTIVE NAME: BANGOU FUNCTION KEY                    *
                        ;*                                                          *
                        ;* FUNCTION: THIS ROUTINE TREATS ALL FUNCTION KEYS.         *
                        ;*           THESE KEYS ARE THE KANJI, ESCAPE, BACK, DELETE, INSERT,  *
                        ;*           ERASE EOF. THE OTHERS ARE INVALID.             *
                        ;*                                                          *
                        ;* LINKAGE: CALL                                            *
                        ;*                                                          *
                        ;*   INPUT: KANA-KAN COMMON TABLES                          *
                        ;*                                                          *
                        ;*   OUTPUT: KANA-KAN COMMON TABLE                          *
                        ;*           KANJI QUEUING                                  *
                        ;*                                                          *
                        ;*   RETURN CODES: (AX)                                     *
                        ;*                                                          *
                        ;*       0 - SUCCESSFUL                                     *
                        ;*       1 - INVALID OPERATION                              *
                        ;*                                                          *
                        ;*  EXTERNAL REFERENCES:                                    *
                        ;*                                                          *
                        ;*    ROUTINES: KKZINT - ZENKOUHO INITIALIZATION            *
                        ;*              KKBINT - BANGOU INITIALIZATION              *
                        ;*              KKBACK - BACK KEY PROCESS                   *
                        ;*              KKDELT - DELETE KEY PROCESS                 *
                        ;*              KKINST - INSERT KEY PROCESS                 *
                        ;*              KKEEOF - ERASE EOF KEY PROCESS              *
                        ;*              KKWOIL - WRITE OIL                          *
                        ;*                                                          *
                        ;*    TABLES: KANA-KAN COMMON TABLES                        *
                        ;*                                                          *
                        ;*  REGISTERS: AX - RETURN CODE                             *
                        ;*             ALL OTHERS UNCHANGED                         *
                        ;*                                                          *
                        ;*  CHANGE ACTIVITY: VERSION 00.00                          *
                        ;*                                                          *
                        ;*                                                          *
```

```
                                ;*****************************************************************
                                ;
                                ;*********************** KKBFNC ***********************************
                                ;*
                                ;*       ENTRY OF KKBFNC                                         *
                                ;*                                                               *
                                ;*****************************************************************
03DE                            KKBFNC  PROC
03DE  56                                PUSH    SI              ;SAVE REGISTER
03DF  52                                PUSH    DX
03E0  51                                PUSH    CX
03E1  53                                PUSH    BX
03E2  8A 26 0005 R                      MOV     AH,SCAN1        ;LOAD INPUT DATA
03E6  A0 0003 R                         MOV     AL,CHAR1
03E9  3D FF00                           CMP     AX,KEXITX
03EC  74 43                             JE      BFN010
03EE  3D FF80                           CMP     AX,KCHNGX
03F1  74 3E                             JE      BFN010
03F3  3D 6B20                           CMP     AX,KYKANJX      ;EXIT KANJI MODE ?
03F6  74 39                             JE      BFN010          ; YES. GOTO
03F8  3D 6B40                           CMP     AX,KYKANJ2      ;EXIT KANJI MODE ?
03FB  74 34                             JE      BFN010          ; YES. GOTO
03FD  80 FC 6B                          CMP     AH,SCAN6B       ;KANJI BANGOU KEY ?
0400  74 71                             JE      BFN090          ; YES. GOTO
0402  3D 011B                           CMP     AX,KYESCPX      ;ESCAPE KEY ?
0405  74 39                             JE      BFN020          ; YSE. GOTO
0407  3D 6D20                           CMP     AX,KYHENKX      ;HENKAN KEY ?
040A  74 3C                             JE      BFN030          ; YES. GOTO
040C  3D 6D40                           CMP     AX,KYHENK2      ;HENKAN KEY ?
040F  74 37                             JE      BFN030          ; YES. GOTO
0411  83 3E 00EA R 00                   CMP     KKWEOP,D00      ;BANGOU EMPTY ?
0416  74 4F                             JE      BFN080          ; YES. GOTO
0418  3D 0E08                           CMP     AX,KYBACKX      ;BACK SPACE KEY ?
041B  74 30                             JE      BFN040          ; YES. GOTO
041D  3D 5300                           CMP     AX,KYDELTX      ;DELETE KEY ?
0420  74 30                             JE      BFN050          ; YES. GOTO
0422  3D 5200                           CMP     AX,KYINSTX      ;INSERT KEY ?
0425  74 33                             JE      BFN060          ; YES. GOTO
0427  3D 7700                           CMP     AX,KYEEOFX      ;ERASE EOF KEY ?
042A  74 33                             JE      BFN070          ; YES. GOTO
                                ;*-----------------------------------------------------------------*
                                ;*       INVALID KEY PROCESS                                     *
                                ;*-----------------------------------------------------------------*
042C  B8 0001                           MOV     AX,D01          ;SET ABNOMAL RETURN CODE
042F  EB 44                             JMP     SHORT BFNRTN
                                ;*-----------------------------------------------------------------*
                                ;*       KANJI KEY PROCESS                                       *
                                ;*-----------------------------------------------------------------*
                                ;
0431                            BFN010:
0431  E8 047A R                         CALL    RINST
0434  B3 01                             MOV     BL,D01
0436  E8 1139 R                         CALL    KKWOIL          ;CLEAR OIL
0439  C6 06 00F9 R 00                   MOV     KKMODE,NORMALM  ;CHANG BANGOU MODE INTO NORMAL MODE
043E  EB 33                             JMP     SHORT BFN090
                                ;*-----------------------------------------------------------------*
                                ;*       ESCAPE KEY PROCESS                                      *
                                ;*-----------------------------------------------------------------*
0440                            BFN020:
0440  E8 047A R                         CALL    RINST
0443  E8 0CA2 R                         CALL    KKZINT          ;ZENKOUHO INITIALIZATION
0446  EB 2B                             JMP     SHORT BFN090
                                ;*-----------------------------------------------------------------*
                                ;*       HENKAN KEY PROCESS                                      *
                                ;*-----------------------------------------------------------------*
0448                            BFN030:
0448  E8 0480 R                         CALL    BHENKAN         ;BANGOU HENKAN
044B  EB 28                             JMP     SHORT BFNRTN
                                ;*-----------------------------------------------------------------*
                                ;*       BACK SPACE KEY PROCESS                                  *
                                ;*-----------------------------------------------------------------*
044D                            BFN040:
044D  E8 0E04 R                         CALL    KKBACK
0450  EB 21                             JMP     SHORT BFN090
                                ;*-----------------------------------------------------------------*
                                ;*       DELETE KEY PROCESS                                      *
                                ;*-----------------------------------------------------------------*
0452                            BFN050:
0452  E8 047A R                         CALL    RINST
0455  E8 0E5A R                         CALL    KKDELT
0458  EB 19                             JMP     SHORT BFN090
                                ;*-----------------------------------------------------------------*
                                ;*       INSERT KEY PROCESS                                      *
                                ;*-----------------------------------------------------------------*
045A                            BFN060:
045A  E8 047A R                         CALL    RINST
045D  EB 14                             JMP     SHORT BFN090
                                ;*-----------------------------------------------------------------*
                                ;*       ERASE EOF KEY PROCESS                                   *
                                ;*-----------------------------------------------------------------*
045F                            BFN070:
045F  E8 047A R                         CALL    RINST
0462  E8 0DAF R                         CALL    KKEEOF
0465  EB 0C                             JMP     SHORT BFN090
                                ;*-----------------------------------------------------------------*
                                ;*       BUFFER QUEUING                                          *
                                ;*-----------------------------------------------------------------*
0467                            BFN080:
0467  3D 6C20                           CMP     AX,KYMUHEX      ;MUHENKAN KEY ?
046A  74 07                             JE      BFN090          ; YES. GOTO
046C  3D 6C40                           CMP     AX,KYMUHE2      ;MUHENKAN KEY ?
046F  74 02                             JE      BFN090          ; YES. GOTO
0471  CD 79                             INT     79H
```

```
0473
0473  33 C0
0475
0475  5B
0476  59
0477  5A
0478  5E
0479  C3
047A

047A
047A  33 C0
047C  E8 0CF2 R
047F  C3
0480

0480
0480  8B 1E 00EA R
0484  80 FB 09
0487  72 79
0489  80 FB 0F
048C  77 74
048E  BE 0009 R

0491  E8 0506 R
0494  8A D0
0496  E8 0506 R
0499  B6 0A
049B  F6 E6
049D  02 D0
049F  80 FA 01
04A2  72 5E
04A5  80 FA 5E
04A7  77 59

04A9  E8 0506 R
04AC  8A C8
04AE  32 ED
04B0  83 FB 00
04B3  74 15
04B5  E8 0506 R
04B8  F6 E6
04BA  03 C8
04BC  83 FB 00
04BF  74 09
04C1  E8 0506 R
04C4  B6 64
04C6  F6 E6
04C8  03 C8
04CA
04CA  83 F9 01
04CD  72 33
04CF  83 F9 78
04D2  77 2E

04D4  F6 C1 01
04D7  74 11

04D9  FE C1
04DB  E8 050E R
04DE  80 C2 3F
04E1  80 FA 7F
04E4  72 0A
04E6  FE C2
04E8  EB 06

04EA
04EA  E8 050E R
04ED  80 C2 9E

04F0
04F0  B4 FF
04F2  8A C1
04F4  CD 79
04F6  8A C2
04F8  CD 79

04FA  E8 047A R
04FD  E8 051E R
```

```
;*----------------------------------------------------------------------*
;*      NORMAL RETURN                                                   *
;*----------------------------------------------------------------------*
BFN090:
        XOR     AX,AX           ;SET NOMAL RETURN CODE
BFNRTN:
        POP     BX              ;RESTORE REGISTER
        POP     CX
        POP     DX
        POP     SI
        RET                     ;RETURN TO CALLER
KKBFNC  ENDP
;***** RINST ***********************************************************
;*                                                                      *
;*      RESET INSERT MODE                                               *
;*                                                                      *
;************************************************************************
RINST   PROC
        XOR     AX,AX
        CALL KKINST
        RET
RINST   ENDP
;***** BHENKAN **********************************************************
;*                                                                      *
;*      BANGOU HENKAN                                                   *
;*                                                                      *
;************************************************************************
BHENKAN PROC
        MOV     BX,KKWEOP
        CMP     BL,BNUMMIN      ;
        JB      BHE030
        CMP     BL,BNUMMAX      ;9 <= BANGOU NUM. => F ?
        JA      BHE030          ; NO. GOTO
        MOV     SI,OFFSET KKINBUF       ;LOAD INPUT BUFFER OFFSET
;*----------------------------------------------------------------------*
;*      CHECK TEN                                                       *
;*----------------------------------------------------------------------*
        CALL    CHRLD           ;LOAD TEN CHARACTER 1
        MOV     DL,AL
        CALL    CHRLD           ;LOAD TEN CHARACTER 2
        MOV     DH,10
        MUL     DH
        ADD     DL,AL           ;CHANGE DECIMAL TEN INTO BINARY
        CMP     DL,TENMIN
        JB      BHE030
        CMP     DL,TENMAX       ;1 <= TEN => 94 ?
        JA      BHE030          ; NO. GOTO
;*----------------------------------------------------------------------*
;*      CHECK KU                                                        *
;*----------------------------------------------------------------------*
        CALL    CHRLD           ;LOAD KU CHARACTER 1
        MOV     CL,AL
        XOR     CH,CH
        CMP     BX,D00          ;KU FINISHED ?
        JE      BHE000          ; YES. GOTO
        CALL    CHRLD           ;LOAD KU CHARACTER 2
        MUL     DH
        ADD     CX,AX
        CMP     BX,D00          ;KU FINISHED ?
        JE      BHE000          ; YES. GOTO
        CALL    CHRLD           ;LOAD KU CHARACTER 3
        MOV     DH,100
        MUL     DH
        ADD     CX,AX           ;CHANGE DECIMAL KU INTO BINARY
BHE000:
        CMP     CX,KUMIN        ;
        JB      BHE030
        CMP     CX,KUMAX        ;01 <= KU => 120 ?
        JA      BHE030          ; NO. GOTO
;*----------------------------------------------------------------------*
;*      CHANGE KU/TEN INTO INTERNAL CODE                                *
;*----------------------------------------------------------------------*
        TEST    CL,D01          ;KU IS EVEN ?
        JZ      BHE010          ; YES. GOTO
;*----------------------------------------------------------------------*
;*      ODD KU                                                          *
;*----------------------------------------------------------------------*
        INC     CL
        CALL    CHGKU
        ADD     DL,3FH
        CMP     DL,7FH
        JB      BHE020
        INC     DL
        JMP     SHORT BHE020
;*----------------------------------------------------------------------*
;*      EVEN    KU                                                      *
;*----------------------------------------------------------------------*
BHE010:
        CALL    CHGKU
        ADD     DL,9EH
;*----------------------------------------------------------------------*
;*      BUFFER QUEUING                                                  *
;*----------------------------------------------------------------------*
BHE020:
        MOV     AH,KANJAH
        MOV     AL,CL           ;SET INTERNAL CODE
        INT     79H             ;BUFFER QUEUING
        MOV     AL,DL           ;SET INTERNAL CODE
        INT     79H             ;BUFFER QUEUING
;*----------------------------------------------------------------------*
;*      INITIALIZE BANGOU MODE                                          *
;*----------------------------------------------------------------------*
        CALL    RINST
        CALL    KKBINT
```

```
0500  EB 03                 JMP     SHORT BHERTN
                      ;*-----------------------------------------------------------*
                      ;*      INVALID BANGOU CODE                                  *
                      ;*-----------------------------------------------------------*
0502                  BHE030:
0502  B8 0001               MOV     AX,D01          ;SET ABNORMAL RETURN CODE
0505                  BHERTN:
0505  C3                    RET                     ;RETURN TO CALLER
0506                  BHENKAN ENDP
                      ;**** CHRLD ***************************************************
                      ;*
                      ;*      LOAD KU/TEN CHARACTER AND CHANGE INTO DECIMAL        *
                      ;*                                                           *
                      ;*************************************************************
0506                  CHRLD   PROC
0506  83 EB 03              SUB     BX,D03
0509  8A 00                 MOV     AL,[BX][SI]     ;LOAD CHARACTER
050B  2C 30                 SUB     AL,30H          ;CHANGE INTO DECIMAL
050D  C3                    RET
050E                  CHRLD   ENDP
                      ;**** CHGKU ***************************************************
                      ;*
                      ;*      CHANGE KU INTO INTERNAL CODE(HIGH BYTE)              *
                      ;*                                                           *
                      ;*************************************************************
050E                  CHGKU   PROC
050E  D0 C9                 ROR     CL,1
0510  80 F9 1E              CMP     CL,1EH          ;01 <= KU => 62 ?
0513  77 05                 JA      CHG010          ; NO. GOTO
0515  80 C1 80              ADD     CL,80H          ;SET 81H <-> 9FH
0518  EB 03                 JMP     SHORT CHGRTN
051A                  CHG010:
051A  80 C1 C0              ADD     CL,0C0H         ;SET 0E0H <-> 0FCH
051D                  CHGRTN:
051D  C3                    RET
051E                  CHGKU   ENDP
                      ;*************************************************************
                      ;*                                                           *
                      ;* PROGRAM NAME: KKBINT                                      *
                      ;*                                                           *
                      ;* DESCRIPTIVE NAME: BANGOU INITIALIZATION                   *
                      ;*                                                           *
                      ;* FUNCTION: THIS ROUTINE DISPLAYS BANGOU OIL AND SETS BANGOU MODE. *
                      ;*                                                           *
                      ;* LINKAGE: CALL                                             *
                      ;*                                                           *
                      ;*   INPUT: KANA-KAN COMMON TABLES                           *
                      ;*                                                           *
                      ;*   OUTPUT: KANA-KAN COMMON TABLES                          *
                      ;*                                                           *
                      ;*   RETURN CODES: (AX)                                      *
                      ;*                                                           *
                      ;*      0 - SUCCESSFUL                                       *
                      ;*      1 - INVALID OPERATION                                *
                      ;*                                                           *
                      ;*  EXTERNAL REFERENCES:                                     *
                      ;*                                                           *
                      ;*    ROUTINES: KKWOIL - WRITE OIL                           *
                      ;*              KKDISP - DISPLAY DATA                        *
                      ;*                                                           *
                      ;*    TABLES: KANA-KAN COMMON TABLES                         *
                      ;*                                                           *
                      ;*  REGISTERS: AX - RETURN CODE                              *
                      ;*             ALL OTHERS UNCHANGED                          *
                      ;*                                                           *
                      ;*  CHANGE ACTIVITY: VERSION 00.00                           *
                      ;*                                                           *
                      ;*                                                           *
                      ;*************************************************************
                      ;
                      ;********************** KKBINT ***********************************
                      ;*                                                           *
                      ;*      ENTRY OF KKBINT                                      *
                      ;*                                                           *
                      ;*************************************************************
051E                  KKBINT  PROC
051E  56                    PUSH    SI              ;SAVE REGISTER
051F  57                    PUSH    DI
0520  52                    PUSH    DX
0521  51                    PUSH    CX
0522  53                    PUSH    BX
0523  C6 06 00F9 R 08       MOV     KKMODE,CODEM    ;SET BANGOU MODE
0528                  BIN000:
0528  BF 0000               MOV     DI,D00          ;LOAD COLUMN IN OIL
052B  C7 06 00EE R 0015     MOV     BASE,BBASE11    ;SET BANGOU BASE POSITION OF 80*11/40*11
0531                  BIN005:
0531  B8 00A0               MOV     AX,SPECHAR
0534  A3 00F5 R             MOV     KKCHRSP,AX              ;SET DEFAULT DATA OF INPUT BUFFER
0537  C7 06 00E8 R 000F     MOV     KKEOPMAX,BEOPMAX        ;SET MAXIMUM EOP
053D  E8 12FC R             CALL    KINBFCLR        ;CLEAR INPUT BUFFER
                      ;*-----------------------------------------------------------*
                      ;*      WRITE OIL                                            *
                      ;*-----------------------------------------------------------*
0540  E8 1538 R             CALL    CLOIL           ;CLEAR OIL
0543  BE 057D R             MOV     SI,OFFSET BMARK ;LOAD BANGOU DATA OFFSET
0546  B9 000F 90            MOV     CX,BNUMS        ;LOAD NUMBERS OF BANGOU DATA
054A  B4 00                 MOV     AH,HANATTR      ;LOAD HANKAKU ATTRIBUTE
054C  32 DB                 XOR     BL,BL
054E                  BIN010:
054E  2E: 8A 04             MOV     AL,CS:[SI]
0551  E8 1139 R             CALL    KKWOIL
0554  46                    INC     SI
0555  E2 F7                 LOOP    BIN010          ;FINISH BANGOU DATA SETING ?
0557  A0 0104 R             MOV     AL,BASTERP      ;LOAD ASTERISK POSITION
```

```
055A  32 E4                     XOR     AH,AH
055C  8B F8                     MOV     DI,AX
055E  B0 2A                     MOV     AL,ASTER
0560  E8 1139 R                 CALL    KKWOIL          ;SET ASTERISK DATA
                        ;*--------------------------------------------------------------------*
                        ;*      SET CURSOR                                                    *
                        ;*--------------------------------------------------------------------*
0563  B8 FFFF                   MOV     AX,0FFFFH
0566  8B 16 00EE R             MOV     DX,BASE         ;LOAD BAGOU BASE POSITION
056A  89 16 0105 R             MOV     DCRSRP,DX
056E  A3 0107 R                MOV     DSTRTP,AX
0571  A3 0109 R                MOV     DENDP,AX
0574  E8 10ED R                CALL    KKDISP          ;DISPLAY CURSOR
0577  5B                       POP     BX              ;RESTORE REGISTER
0578  59                       POP     CX
0579  5A                       POP     DX
057A  5F                       POP     DI
057B  5E                       POP     SI
057C  C3                       RET                     ;RETURN TO CALLER
057D  BA B0 C4 DE 3D 3D BMARK  DB      0BAH,0B0H,0C4H,0DEH,3DH,3DH,3DH,3EH,20H
      3D 3E 20
0586  A0 A0 A0 A0 A0 2A        DB      0A0H,0A0H,0A0H,0A0H,0A0H,2AH
= 000F                  BNUMS   EQU     15              ;NUMBER OF BMARK
058C                    KKBINT  ENDP
                        ;*********************************************************************
                        ;*                                                                   *
                        ;* PROGRAM NAME: KKZCTL                                              *
                        ;*                                                                   *
                        ;* DESCRIPTIVE NAME: KEY CONTROL FOR KANA-KAN ZENKOUHO MODE          *
                        ;*                                                                   *
                        ;* FUNCTION:                                                         *
                        ;*                                                                   *
                        ;*      THIS MODULE ANALYZE INPUT KEY FOR KANA-KANJI CONVERSION,     *
                        ;*      AND CALLS EACH FUNCTION ROUTINES.                            *
                        ;*                                                                   *
                        ;* LINKAGE:      CALLED BY KKKFDM                                    *
                        ;*                                                                   *
                        ;*   INPUT:                                                          *
                        ;*           KANA-KAN COMMON TABLES                                  *
                        ;*                                                                   *
                        ;*   OUTPUT:                                                         *
                        ;*           LINK EACH FUNCTION MODULE                               *
                        ;*                                                                   *
                        ;*           BP : FUNCTION KEY NUMBER                                *
                        ;*                                                                   *
                        ;*                                                                   *
                        ;*   RETURN CODES: (AX)                                              *
                        ;*                                                                   *
                        ;*      0 - SUCCESSFUL                                               *
                        ;*      1 - INVALID OPERATION                                        *
                        ;*                                                                   *
                        ;*  EXTERNAL REFERENCES:                                             *
                        ;*                                                                   *
                        ;*    ROUTINES:                                                      *
                        ;*                                                                   *
                        ;*      KKZGRP ... GRAPHIC KEY DISPLAY                               *
                        ;*      KKCSR  ... CURSOR KEY PROCESS                                *
                        ;*      KKZFNC ... FUNCTION KEY CONTROL                              *
                        ;*                                                                   *
                        ;*    TABLES:                                                        *
                        ;*                                                                   *
                        ;*      KANA-KAN COMMON TABLES                                       *
                        ;*                                                                   *
                        ;*  REGISTERS: AX - RETURN CODE                                      *
                        ;*             BX,CX,DX,DI,SI,BP - WORK                              *
                        ;*             ALL OTHERS UNCHANGED                                  *
                        ;*                                                                   *
                        ;*  CHANGE ACTIVITY: VERSION 00.00                                   *
                        ;*                                                                   *
                        ;*                                                                   *
                        ;*********************************************************************
                        ;
                        ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        ;+      INPUT FUNCTION KEY CODE & PROC.ENTRY ADDRESS TABLE            +
                        ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        KKCCDTB:
                                DW      KYCSRLX,KZCCSR  ; CURSOR LEFTE
                                DW      KYCSRRX,KZCCSR  ; CURSOR RIGHT
                                DW      KYKANJX,KZCFNC  ; KANJI
                                DW      KYKANJ2,KZCFNC  ; KANJI
                                DW      KEXITX,KZCFNC   ; KANJI EXIT ( = KANJI )
                                DW      KYCODEX,KZCFNC  ; KANJI BANGOU
                                DW      KCHNGX,KZCFNC   ; KANJI MODE CHANGE ( = KANJI BANGOU )
                                DW      KYHENKX,KZCFNC  ; HENKAN
                                DW      KYHENK2,KZCFNC  ; HENKAN (ZENKAKU)
                                DW      KYMUHEX,KZCFNC  ; MUHENKAN
                                DW      KYMUHE2,KZCFNC  ; MUHENKAN (ZENKAKU)
                                DW      KYESCPX,KZCFNC  ; ESCAPE
                                DW      KYINSTX,KZCFNC  ; INSERT
                                DW      KYDELTX,KZCFNC  ; DELETE
                                DW      KYBACKX,KZCFNC  ; BACKSPACE
                                DW      KYEEOFX,KZCFNC  ; ERASE EOF
                                DW      KYCRRTX,KZCFNC  ; CARRIER RETURN
                                DW      KYCRRTY,KZCFNC  ; CARRIER RETURN
                                DW      KYENTRX,KZCFNC  ; ENTER
                        KKCCDTBE DW     0FFFFH,KZCGRP
                        KKCCDTBL EQU     KKCCDTBE-KKCCDTB ; LENGTH OF KKCCDTB
                        FLGNL   EQU     00H     ; NULL
                        FLGVK   EQU     01H     ; NOT QUEING KEY CODE: IF THERE AER NO YOMI CHAR.
                        FLGRI   EQU     02H     ; RESET INSERT MODE: WHEN THIS FUNC. KEY INPUT
                        FLGNQ   EQU     04H     ; NOT FUNCTION, NOT QUEUING
                        KPRCSTS:
                                DB      FLGNL           ; CURSOR LEFTE
                                DB      FLGNL           ; CURSOR RIGHT
058C
058C  4B00 063D R
0590  4D00 063D R
0594  6B20 060D R
0598  6B40 060D R
059C  FF00 060D R
05A0  6B38 060D R
05A4  FF80 060D R
05A8  6D20 060D R
05AC  6D40 060D R
05B0  6C20 060D R
05B4  6C40 060D R
05B8  011B 060D R
05BC  5200 060D R
05C0  5300 060D R
05C4  0E08 060D R
05C8  7700 060D R
05CC  1C0D 060D R
05D0  7E0D 060D R
05D4  9300 060D R
05D8  FFFF 0641 R
= 004C
= 0000
= 0001
= 0002
= 0004
05DC
05DC  00
05DD  00
```

```
05DE  01                DB      FLGNL+FLGVK         ; KANJI
05DF  01                DB      FLGNL+FLGVK         ; KANJI
05E0  01                DB      FLGNL+FLGVK         ; KANJI EXIT ( = KANJI )
05E1  01                DB      FLGNL+FLGVK         ; KANJI BANGOU
05E2  01                DB      FLGNL+FLGVK         ; KANJI MODE CHANGE ( = KANJI BANGOU )
05E3  04                DB      FLGNL+FLGNQ         ; HENKAN
05E4  04                DB      FLGNL+FLGNQ         ; HENKAN (ZENKAKU)
05E5  04                DB      FLGNL+FLGNQ         ; MUHENKAN
05E6  04                DB      FLGNL+FLGNQ         ; MUHENKAN (ZENKAKU)
05E7  00                DB      FLGNL               ; ESCAPE
05E8  02                DB      FLGNL+FLGRI         ; INSERT
05E9  00                DB      FLGNL               ; DELETE
05EA  02                DB      FLGNL+FLGRI         ; BACKSPACE
05EB  00                DB      FLGNL               ; ERASE EOF
05EC  02                DB      FLGNL+FLGRI         ; CARRIER RETURN
05ED  02                DB      FLGNL+FLGRI         ; CARRIER RETURN
05EE  03                DB      FLGNL+FLGVK+FLGRI   ; ENTER
05EF                    KKZCTL  PROC    NEAR
05EF  2B ED                     SUB     BP,BP               ; CLEAR FUNC.KEY TABLE POINTER
05F1  A0 0003 R                 MOV     AL,KKINKEY          ; GET INPUT KEY (CHR. CODE)
05F4  8A 26 0005 R              MOV     AH,KKINKEY+2        ; GET INPUT KEY (SCAN CODE)
05F8                    KZC010:
05F8  2E: 3B 86 058C R          CMP     AX,WORD PTR KKCCDTB[BP]
                                                            ; COMPARE INPUT KEY WITH FUNC.KEY
05FD  74 08                     JE      KZC020
05FF  83 C5 04                  ADD     BP,4                ; RETURN THE POINTER
0602  83 FD 4C                  CMP     BP,KKCCDTBL         ; SCAN TERMINATE CHECK
0605  75 F1                     JNE     KZC010              ; NO... REREAT SCAN
0607                    KZC020:
0607  2E: FF 96 058E R          CALL    WORD PTR KKCCDTB[BP][2]
                                                            ; BRANCH TO EACH FUNCTION
060C  C3                        RET
                        ;*************************************************************************
                        ;*      FUNCTON KEY CONTROL                                              *
                        ;*************************************************************************
060D                    KZCFNC:
060D  D1 ED                     SHR     BP,1                ; BP <-- BP / 4
060F  D1 ED                     SHR     BP,1
0611  2E: 8A 9E 05DC R          MOV     BL,BYTE PTR KPRCSTS[BP]
0616  83 3E 00EA R 00           CMP     KKWEOP,0
061B  75 0F                     JNE     KZCFNC_CHK
061D  F6 C3 01                  TEST    BL,FLGVK
0620  75 0A                     JNZ     KZCFNC_CHK
0622                    KZCFNC_QUEING:
0622  F6 C3 04                  TEST    BL,FLGNQ
0625  75 02                     JNZ     QUEURTN
0627  CD 79                     INT     79H                 ; QUEING INPUT FUNCTION KEY CODE
0629                    QUEURTN:
0629  33 C0                     XOR     AX,AX
062B  C3                        RET
062C                    KZCFNC_CHK:
062C  F6 C3 02                  TEST    BL,FLGRI
062F  75 05                     JNZ     KZCFNC_GO
0631  33 C0                     XOR     AX,AX
0633  E8 0CF2 R                 CALL    KKINST              ; RESET INSERT MODE
0636                    KZCFNC_GO:
0636  83 ED 02                  SUB     BP,2                ; BP <-- BP - 2
0639  E8 072B R                 CALL    KKZFNC
063C  C3                        RET
                        ;*************************************************************************
                        ;*      CURSOR KEY MOTION                                                *
                        ;*************************************************************************
063D                    KZCCSR:
063D  E8 0C2E R                 CALL    KKCSR               ; LINK CURSOR PROC. ROUTINE
0640  C3                        RET
                        ;*************************************************************************
                        ;*      GRAPHIC KEY                                                      *
                        ;*************************************************************************
0641                    KZCGRP:
0641  F6 06 0000 R 83           TEST    DSTAT,HANKAF+ZEN2F+DAKUF
0646  74 04                     JZ      KZCGRP_CHK
0648  E8 065D R                 CALL    KKZGRP              ; GRAPHIC KEY INPUT OPERATION
064B  C3                        RET
064C                    KZCGRP_CHK :
064C  83 3E 00EA R 00           CMP     KKWEOP,0
0651  75 05                     JNE     KZCGRP_RET
0653                    KZCGRP_QUEING:
0653  CD 79                     INT     79H                 ; QUEING CURSOR KEY CODE
0655  33 C0                     XOR     AX,AX
0657  C3                        RET
0658                    KZCGRP_RET:
0658  B8 0001                   MOV     AX,1
065B  C3                        RET
065C                    KKZCTL  ENDP
                        ;*************************************************************************
                        ;*                                                                       *
                        ;* PROGRAM NAME: KKZGRP                                                  *
                        ;*                                                                       *
                        ;* DESCRIPTIVE NAME: KEY CONTROL FOR KANA-KAN ZENKOUHO MODE              *
                        ;*                                                                       *
                        ;* FUNCTION:                                                             *
                        ;*                                                                       *
                        ;*      THIS MODULE ANALYZE INPUT KEY FOR KANA-KANJI CONVERSION,         *
                        ;*      AND CALLS EACH FUNCTION ROUTINES.                                *
                        ;*                                                                       *
                        ;*                                                                       *
                        ;* LINKAGE:     CALLED BY KKZCTL                                         *
                        ;*                                                                       *
                        ;*   INPUT: KANA-KAN COMMON TABLES                                       *
                        ;*                                                                       *
                        ;*   OUTPUT: KANA-KAN COMMON TABLES                                      *
                        ;*                                                                       *
                        ;*   RETURN CODES: (AX)                                                  *
                        ;*                                                                       *
```

```
065C  03

065D
065D  A0 00F9 R
0660  3C 02
0662  75 04
0664  E8 1209 R
0667  C3
0668
0668  8A 1E 0005 R
066C  B0 01
066E  E8 16CF R
0671  84 C0
0673  74 0D
0675  B0 02
0677  E8 16CF R
067A  84 C0
067C  74 04
067E
067E  B8 0001
0681  C3
0682
0682  80 E3 0F
0685  74 F7
0687  80 EB 01
068A  3A 1E 0146 R
068E  76 02
0690  EB EC
0692
0692  C6 06 00F9 R 02
0697  8A C3
0699  98
069A  8B D8
069C  03 1E 0148 R
06A0  3B 1E 013E R
06A4  73 1C
06A6  4B
06A7  D1 E3
06A9  1E
06AA  56
06AB  C5 36 0140 R
06AF  B4 FF
06B1  8A 00
06B3  0C 80
06B5  24 BF
06B7  CD 79
06B9  8A 40 01
06BC  CD 79
06BE  5E
06BF  1F
06C0  EB 15
06C2
06C2  8B 0E 018E R
06C6  2B DB
06C8
06C8  8A 87 014A R
06CC  8A A7 014B R
06D0  CD 79
06D2  43
06D3  43
06D4  49
06D5  75 F1
06D7
06D7  F6 06 0191 R 02
06DC  75 05
06DE  E8 0CA2 R
06E1  EB 23
06E3
06E3  A0 000A R
06E6  E8 1341 R
06E9  50
06EA  B8 0000
06ED  E8 162F R
06F0  58
06F1  29 06 00EA R
06F5  29 06 00F7 R
```

```
;*      0 - SUCCESSFUL                                                  *
;*      1 - INVALID OPERATION                                           *
;*                                                                      *
;*  EXTERNAL REFERENCES:                                                *
;*                                                                      *
;*    ROUTINES:                                                         *
;*                                                                      *
;*      KKGRP ... GRAPHIC KEY DISPLAY                                   *
;*      KKCSR ... CURSOR KEY PROCESS                                    *
;*      KKZFNC ... FUNCTION KEY CONTROL                                 *
;*                                                                      *
;*    TABLES: KANA-KAN COMMON TABLES                                    *
;*                                                                      *
;*  REGISTERS: AX - RETURN CODE                                         *
;*             BX,CX,SI - WORK                                          *
;*             ALL OTHERS UNCHANGED                                     *
;*                                                                      *
;*  CHANGE ACTIVITY: VERSION 00.00                                      *
;*                                                                      *
;*                                                                      *
;************************************************************************
;
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;+      INTERNAL CONSTANT DATA                                          +
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
MUL03   DB      3
;************************************************************************
;*      GRAPHIC KEY INPUT OPERATION                                     *
;************************************************************************
KKZGRP  PROC    NEAR
        MOV     AL,KKMODE       ; GET CURRENT MODE
        CMP     AL,KANJIM
        JNZ     KZG300
        CALL    KKGRP           ; KANJI MODE
        RET
KZG300:                         ; SELECT MODE
        MOV     BL,KKINKEY+2    ; GET KEY SCAN CODE
        MOV     AL,1
        CALL    CDCHCK1         ; NUMERIC KEY CODE ?(TEN KEY)
        TEST    AL,AL           ; TEST RESULT
        JZ      KZG320
        MOV     AL,2
        CALL    CDCHCK1         ; NUMERIC KEY CODE ?(SHIFT EISUU)
        TEST    AL,AL           ; TEST RETRUN
        JZ      KZG320
KZG310:
        MOV     AX,1            ; ERROR RETURN
        RET
KZG320:
        AND     BL,0FH          ;
        JZ      KZG310          ; RETURN ZERO
        SUB     BL,01H          ;GET VALUE ... 1 --- 9
        CMP     BL,BYTE PTR KKOHOCNT    ;
        JBE     KZG325
        JMP     SHORT KZG310    ; RETURN IF ABOVE
KZG325:
        MOV     KKMODE,KANJIM   ; CHANGE TO JANJI MODE
        MOV     AL,BL
        CBW                     ; BYTE TO WORD CONVERSION
        MOV     BX,AX
        ADD     BX,KKOHODSP     ; CALCULATE CODE SELECT POINTER
        CMP     BX,KKOHOSUU     ; IS POINTER LAST ?
        JAE     KZG330          ; YES...
        DEC     BX
        SHL     BX,1            ; BX*2
        PUSH    DS
        PUSH    SI
        LDS     SI,DWORD PTR KKDICADR   ; GET OFFSET & SEGMENT
        MOV     AH,0FFH
        MOV     AL,DS:[SI][BX]      ; GET FIRST BYTE KANJI CODE
        OR      AL,10000000B
        AND     AL,10111111B
        INT     79H                 ; BUFFER QUEING
        MOV     AL,DS:[SI][BX][1]   ; GET SECOND BYTE KANJI CODE
        INT     79H                 ; BUFFER QUEING
        POP     SI
        POP     DS
        JMP     SHORT KZG350
KZG330:
        MOV     CX,KKHQLEN          ; GET QUEING COUNT
        SUB     BX,BX
KZG340:
        MOV     AL,KKHANQUE[BX]     ; GET CHARACTER CODE TO AL REG
        MOV     AH,KKHANQUE[1][BX] ; GET SCAN CODE TO AH REG
        INT     79H                 ; BUFFER QUEING
        INC     BX                  ; RENEW QUEING SUB BUFFER POINTER
        INC     BX
        DEC     CX
        JNZ     KZG340
KZG350:
        TEST    KKYMSTS,KKYMSTM     ; TEST MOVE FORWARD FLAG
        JNZ     KZG360
        CALL    KKZINT              ; INITIALIZE OPERATOR INFORM. LINE
        JMP     SHORT KZG370
KZG360:
        MOV     AL,KKINBUF+1    ; GET FIRST CHARACTER ATTRIBUTE
        CALL    KCALELM         ; CALCULATE ELEMENT LENGTH
        PUSH    AX
        MOV     AX,0
        CALL    DTSHFT          ; MOVE FORWARD KEY INPUT BUFFER
        POP     AX
        SUB     KKWEOP,AX
        SUB     KKFORPIT,AX
```

```
06F9  77 E8                    JA      KZG360          ; INITIALIZE O.I.L.
06FB  E8 1291 R                CALL    KOUTINIT        ; DISPLAY CURSOR TO EOP POSITION
06FE  E8 1355 R                CALL    KCSRDSP
0701  C6 06 0192 R 01          MOV     KHENFST,01H
0706                    KZG370:
0706  2B C0                    SUB     AX,AX
0708  C3                      RET
0709                    KKZGRP  ENDP
                        ;*****************************************************************
                        ;*                                                               *
                        ;* PROGRAM NAME: KKZFNC                                          *
                        ;*                                                               *
                        ;* DESCRIPTIVE NAME: KANA-KAN MAIN FUNCTION                      *
                        ;*                                                               *
                        ;* FUNCTION:                                                     *
                        ;*      THIS MODULE HAS THE FOLLOWING FUNCTION;                  *
                        ;*        . KANA - KANJI CONVERSION                              *
                        ;*        . DISPLAY SELECT NUMBER AND KANJI                      *
                        ;*        . EDIT INPUT BUFFER ( INSERT,DELETE,BACKSPACE,ERASE-EOF )*
                        ;*        . REGISTERD YOMI & BANGOU                              *
                        ;*                                                               *
                        ;* LINKAGE:                                                      *
                        ;*                                                               *
                        ;*   INPUT:                                                      *
                        ;*           BP : FUNCTION KEY NUMBER                            *
                        ;*           KANA-KAN COMMON TABLES                              *
                        ;*                                                               *
                        ;*   OUTPUT:                                                     *
                        ;*           KANA-KAN COMMON TABLES                              *
                        ;*                                                               *
                        ;*   RETURN CODES: (AX)                                          *
                        ;*                                                               *
                        ;*      0 - SUCCESSFUL                                           *
                        ;*      1 - INVALID OPERATION                                    *
                        ;*                                                               *
                        ;*   EXTERNAL REFERENCES:                                        *
                        ;*                                                               *
                        ;*     ROUTINES:                                                 *
                        ;*           KKKNDR          KKBINT          KKZINT          KKBACK  *
                        ;*           KKDELT          KKINST          KKEEOF          KKGRP   *
                        ;*           KKWOIL          KKDISP          KOUTINIT        *
                        ;*           CDCHCK1         CDCHCK2         KKOHOEDT        *
                        ;*           KCALELM         KCSRDSP                         *
                        ;*                                                               *
                        ;*     TABLES: KANA-KAN COMMON TABLES                            *
                        ;*                                                               *
                        ;*   REGISTERS: AX - RETURN CODE                                 *
                        ;*              BX,CX,DX,DI,SI,BP - WORK                         *
                        ;*              ALL OTHERS UNCHANGED                             *
                        ;*                                                               *
                        ;*   CHANGE ACTIVITY: VERSION 00.00                              *
                        ;*                                                               *
                        ;*****************************************************************
0709  0735 R               FNCBR:  DW      FNCKANJ         ; KANJI
070B  0735 R                      DW      FNCKANJ         ; KANJI
070D  0735 R                      DW      FNCKANJ         ; KANJI EXIT
070F  0742 R                      DW      FNCCODE         ; KANJI BANGO
0711  0735 R                      DW      FNCKANJ         ; KANJI MODE CHENGE
0713  0894 R                      DW      FNCHENK         ; HENKAN
0715  0894 R                      DW      FNCHENK         ; HENKAN
0717  0B33 R                      DW      FNCMUHE         ; MUHENKAN
0719  0B33 R                      DW      FNCMUHE         ; MUHENKAN
071B  0B75 R                      DW      FNCESCP         ; ESCAPE
071D  0B94 R                      DW      FNCINST         ; INSERT
071F  0BAE R                      DW      FNCDELT         ; DELETE
0721  0BB9 R                      DW      FNCBACK         ; BACKSPACE
0723  0BC4 R                      DW      FNCEEOF         ; ERASE-EOF
0725  0BCF R                      DW      FNCCRRT         ; CARRIER RETURN
0727  0BCF R                      DW      FNCCRRT         ; CARRIER RETURN
0729  0BF4 R                      DW      FNCENTR         ; ENTER
072B                    KKZFNC  PROC    NEAR
072B  B8 0001                   MOV     AX,1
072E  D1 E5                     SHL     BP,1                    ; BP = 2
0730  2E: FF A6 0709 R          JMP     WORD PTR FNCBR[BP]      ; BRANCH TO EACH FUNCTION
0735                    KKZFNC  ENDP
                                        ;*****************************
                                        ;*                           *
                                        ;*          KANJI            *
                                        ;*                           *
                                        ;*****************************
0735                    FNCKANJ PROC
0735  C6 06 00F9 R 08          MOV     KKMODE,NORMALM  ; CHANGE TO NORMAL MODE
073A  B3 01                     MOV     BL,1
073C  E8 1139 R                 CALL    KKWOIL          ; CLEAR OPERATOR INFORMATION LIME
073F  2B C0                     SUB     AX,AX
0741  C3                        RET
0742                    FNCKANJ ENDP
                                        ;*****************************
                                        ;*                           *
                                        ;*       KANJI - BANGOU      *
                                        ;*                           *
                                        ;*****************************
0742                    FNCCODE PROC
0742  C6 06 00F9 R 08          MOV     KKMODE,CODEM    ; CHANGE TO CODE MODE
0747  E8 051E R                 CALL    KKBINT          ; LINK KKBINT MODULE
074A  2B C0                     SUB     AX,AX
074C  C3                        RET
074D                    FNCCODE ENDP
                                        ;*****************************
                                        ;*                           *
                                        ;*          HENKAN           *
                                        ;*                           *
                                        ;*****************************
```

```
074D
074D   A7  00
074F   B1  00
0751   A8  00
0753   B2  00
0755   A9  00
0757   B3  00
0759   AA  00
075B   B4  00
075D   AB  00
075F   B5  00
0761   B6  00
0763   B6  DE
0765   B7  00
0767   B7  DE
0769   B8  00
076B   B8  DE
076D   B9  00
076F   B9  DE
0771   BA  00
0773   BA  DE
0775   BB  00
0777   BB  DE
0779   BC  00
077B   BC  DE
077D   BD  00
077F   BD  DE
0781   BE  00
0783   BE  DE
0785   BF  00
0787   BF  DE
0789   C0  00
078B   C0  DE
078D   C1  00
078F   C1  DE
0791   AF  00
0793   C2  00
0795   C2  DE
0797   C3  00
0799   C3  DE
079B   C4  00
079D   C4  DE
079F   C5  00
07A1   C6  00
07A3   C7  00
07A5   C8  00
07A7   C9  00
07A9   CA  00
07AB   CA  DE
07AD   CA  DF
07AF   CB  00
07B1   CB  DE
07B3   CB  DF
07B5   CC  00
07B7   CC  DE
07B9   CC  DF
07BB   CD  00
07BD   CD  DE
07BF   CD  DF
07C1   CE  00
07C3   CE  DE
07C5   CE  DF
07C7   CF  00
07C9   D0  00
07CB   D1  00
07CD   D2  00
07CF   D3  00
07D1   AC  00
07D3   D4  00
07D5   AD  00
07D7   D5  00
07D9   AE  00
07DB   D6  00
07DD   D7  00
07DF   D8  00
07E1   D9  00
07E3   DA  00
07E5   DB  00
07E7   FF  00
07E9   DC  00
07EB   FF  00
07ED   FF  00
07EF   A6  00
07F1   DD  00
07F3
= 0053
07F3   DE  00
07F5   DF  00
07F7   B0  00
07F9
= 0056
07F9   30  00
07FB   31  00
07FD   32  00
07FF   33  00
0801   34  00
0803   35  00
```

```
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;+       PRIVATE CONSTANT  < FNCHENK >                                        +
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
;M          ZENKAKU CODE CONVERSION TABLE                                    M
;MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
KZENCONV:
        DB      0A7H,00H        ; A ( SHOUJI )
        DB      0B1H,00H        ; A
        DB      0A8H,00H        ; I ( SHOUJI )
        DB      0B2H,00H        ; I
        DB      0A9H,00H        ; U ( SHOUJI )
        DB      0B3H,00H        ; U
        DB      0AAH,00H        ; E ( SHOUJI )
        DB      0B4H,00H        ; E
        DB      0ABH,00H        ; O ( SHOUJI )
        DB      0B5H,00H        ; O
        DB      0B6H,00H        ; KA
        DB      0B6H,0DEH       ; GA ( KA + DAKUTEN )
        DB      0B7H,00H        ; KI
        DB      0B7H,0DEH       ; GI ( KI + DAKUTEN )
        DB      0B8H,00H        ; KU
        DB      0B8H,0DEH       ; GU ( KU + DAKUTEN )
        DB      0B9H,00H        ; KE
        DB      0B9H,0DEH       ; GE ( KE + DAKUTEN )
        DB      0BAH,00H        ; KO
        DB      0BAH,0DEH       ; GO ( KO + DAKUTEN )
        DB      0BBH,00H        ; SA
        DB      0BBH,0DEH       ; ZA ( SA + DAKUTEN )
        DB      0BCH,00H        ; SI
        DB      0BCH,0DEH       ; JI ( SI + DAKUTEN )
        DB      0BDH,00H        ; SU
        DB      0BDH,0DEH       ; ZU ( SU + DAKUTEN )
        DB      0BEH,00H        ; SE
        DB      0BEH,0DEH       ; ZE ( SE + DAKUTEN )
        DB      0BFH,00H        ; SO
        DB      0BFH,0DEH       ; ZO ( SO + DAKUTEN )
        DB      0C0H,00H        ; TA
        DB      0C0H,0DEH       ; DA ( TA + DAKUTEN )
        DB      0C1H,00H        ; TI
        DB      0C1H,0DEH       ; ZI ( TI + DAKUTEN )
        DB      0AFH,00H        ; TU ( HATUON )
        DB      0C2H,00H        ; TU
        DB      0C2H,0DEH       ; ZU ( TU + DAKUTEN )
        DB      0C3H,00H        ; TE
        DB      0C3H,0DEH       ; DE ( TE + DAKUTEN )
        DB      0C4H,00H        ; TO
        DB      0C4H,0DEH       ; DO ( TO + DAKUTEN )
        DB      0C5H,00H        ; NA
        DB      0C6H,00H        ; NI
        DB      0C7H,00H        ; NU
        DB      0C8H,00H        ; NE
        DB      0C9H,00H        ; NO
        DB      0CAH,00H        ; HA
        DB      0CAH,0DEH       ; VA ( HA + DAKUTEN )
        DB      0CAH,0DFH       ; PA ( HA + HANDAKUTEN )
        DB      0CBH,00H        ; HI
        DB      0CBH,0DEH       ; VI ( HI + DAKUTEN )
        DB      0CBH,0DFH       ; PI ( HI + HANDAKUTEN )
        DB      0CCH,00H        ; HU
        DB      0CCH,0DEH       ; VU ( HU + DAKUTEN )
        DB      0CCH,0DFH       ; PU ( HU + HANDAKUTEN )
        DB      0CDH,00H        ; HE
        DB      0CDH,0DEH       ; VE ( HE + DAKUTEN )
        DB      0CDH,0DFH       ; PE ( HE + HANDAKUTEN )
        DB      0CEH,00H        ; HO
        DB      0CEH,0DEH       ; VO ( HO + DAKUTEN )
        DB      0CEH,0DFH       ; PO ( HO + HANDAKUTEN )
        DB      0CFH,00H        ; MA
        DB      0D0H,00H        ; MI
        DB      0D1H,00H        ; MU
        DB      0D2H,00H        ; ME
        DB      0D3H,00H        ; MO
        DB      0ACH,00H        ; YA ( SHOJI )
        DB      0D4H,00H        ; YA
        DB      0ADH,00H        ; YU ( SHOJI )
        DB      0D5H,00H        ; YU
        DB      0AEH,00H        ; YO ( SHOJI )
        DB      0D6H,00H        ; YO
        DB      0D7H,00H        ; LA
        DB      0D8H,00H        ; LI
        DB      0D9H,00H        ; LU
        DB      0DAH,00H        ; LE
        DB      0DBH,00H        ; LO
        DB      0FFH,00H        ; WA ( SHOJI )
        DB      0DCH,00H        ; WA
        DB      0FFH,00H        ; I
        DB      0FFH,00H        ; E
        DB      0A6H,00H        ; O
        DB      0DDH,00H        ; UN
DAKTENDATA:
KZENDAKU EQU    (DAKTENDATA-KZENCONV)/2
        DB      0DEH,00H        ; DAKUTEN
        DB      0DFH,00H        ; HANDAKUTEN
        DB      0B0H,00H        ; CHOUON
NUMEDATA:
KZENNUMR EQU    (NUMEDATA-KZENCONV)/2
        DB      30H,00H         ; 0
        DB      31H,00H         ; 1
        DB      32H,00H         ; 2
        DB      33H,00H         ; 3
        DB      34H,00H         ; 4
        DB      35H,00H         ; 5
```

```
0805  36 00                        DB      36H,00H                 ; 6
0807  37 00                        DB      37H,00H                 ; 7
0809  38 00                        DB      38H,00H                 ; 8
080B  39 00                        DB      39H,00H                 ; 9
                        ;**************************************************************************
                        ;*          HANKAKU CODE CONVERSION TABLE                                 *
                        ;**************************************************************************
080D                    KHANCONV:
080D  F5 00                        DB      0F5H,00H                ; 30    0    ( NUMERIC )
080F  F6 00                        DB      0F6H,00H                ; 31    1    ( NUMERIC )
0811  F7 00                        DB      0F7H,00H                ; 32    2    ( NUMERIC )
0813  F8 00                        DB      0F8H,00H                ; 33    3    ( NUMERIC )
0815  F9 00                        DB      0F9H,00H                ; 34    4    ( NUMERIC )
0817  FA 00                        DB      0FAH,00H                ; 35    5    ( NUMERIC )
0819  FB 00                        DB      0FBH,00H                ; 36    6    ( NUMERIC )
081B  FC 00                        DB      0FCH,00H                ; 37    7    ( NUMERIC )
081D  FD 00                        DB      0FDH,00H                ; 38    8    ( NUMERIC )
081F  FE 00                        DB      0FEH,00H                ; 39    9    ( NUMERIC )
0821  9F 00                        DB      09FH,00H                ; A7    A    ( SHOUON )
0823  A1 00                        DB      0A1H,00H                ; A8    I    ( SHOUON )
0825  A3 00                        DB      0A3H,00H                ; A9    U    ( SHOUON )
0827  A5 00                        DB      0A5H,00H                ; AA    E    ( SHOUON )
0829  A7 00                        DB      0A7H,00H                ; AB    O    ( SHOUON )
082B  E1 00                        DB      0E1H,00H                ; AC    YA   ( YOUON )
082D  E3 00                        DB      0E3H,00H                ; AD    YU   ( YOUON )
082F  E4 00                        DB      0E4H,00H                ; AE    YO   ( YOUON )
0831  C1 00                        DB      0C1H,00H                ; AF    TU   ( HATUON )
0833  F4 00                        DB      0F4H,00H                ; B0    -    ( CHOUON )
0835  A0 00                        DB      0A0H,00H                ; B1    A
0837  A2 00                        DB      0A2H,00H                ; B2    I
0839  A4 00                        DB      0A4H,00H                ; B3    U
083B  A6 00                        DB      0A6H,00H                ; B4    E
083D  A8 00                        DB      0A8H,00H                ; B5    O
083F  A9 04                        DB      0A9H,KKYMSTD            ; B6    KA
0841  AB 04                        DB      0ABH,KKYMSTD            ; B7    KI
0843  AD 04                        DB      0ADH,KKYMSTD            ; B8    KU
0845  AF 04                        DB      0AFH,KKYMSTD            ; B9    KE
0847  B1 04                        DB      0B1H,KKYMSTD            ; BA    KO
0849  B3 04                        DB      0B3H,KKYMSTD            ; BB    SA
084B  B5 04                        DB      0B5H,KKYMSTD            ; BC    SI
084D  B7 04                        DB      0B7H,KKYMSTD            ; BD    SU
084F  B9 04                        DB      0B9H,KKYMSTD            ; BE    SE
0851  BB 04                        DB      0BBH,KKYMSTD            ; BF    SO
0853  BD 04                        DB      0BDH,KKYMSTD            ; C0    TA
0855  BF 04                        DB      0BFH,KKYMSTD            ; C1    TI
0857  C2 04                        DB      0C2H,KKYMSTD            ; C2    TU
0859  C4 04                        DB      0C4H,KKYMSTD            ; C3    TE
085B  C6 04                        DB      0C6H,KKYMSTD            ; C4    TO
085D  C6 00                        DB      0C6H,00H                ; C5    NA
085F  C8 00                        DB      0C8H,00H                ; C6    NI
0861  CA 00                        DB      0CAH,00H                ; C7    NU
0863  CB 00                        DB      0CBH,00H                ; C8    NE
0865  CC 00                        DB      0CCH,00H                ; C9    NO
0867  CD 04                        DB      0CDH,KKYMSTD            ; CA    HA
0869  D0 04                        DB      0D0H,KKYMSTD            ; CB    HI
086B  D3 04                        DB      0D3H,KKYMSTD            ; CC    HU
086D  D6 04                        DB      0D6H,KKYMSTD            ; CD    HE
086F  D9 04                        DB      0D9H,KKYMSTD            ; CE    HO
0871  DC 00                        DB      0DCH,00H                ; CF    MA
0873  DD 00                        DB      0DDH,00H                ; D0    MI
0875  DE 00                        DB      0DEH,00H                ; D1    MU
0877  DF 00                        DB      0DFH,00H                ; D2    ME
0879  E0 00                        DB      0E0H,00H                ; D3    MO
087B  E2 00                        DB      0E2H,00H                ; D4    YA
087D  E3 00                        DB      0E3H,00H                ; D5    YU
087F  E4 00                        DB      0E4H,00H                ; D6    YO
0881  E7 00                        DB      0E7H,00H                ; D7    LA
0883  E8 00                        DB      0E8H,00H                ; D8    LI
0885  E9 00                        DB      0E9H,00H                ; D9    LU
0887  EA 00                        DB      0EAH,00H                ; DA    LE
0889  EB 00                        DB      0EBH,00H                ; DB    LO
088B  DC 00                        DB      0DCH,00H                ; DC    WA
088D  F1 00                        DB      0F1H,00H                ; DD    UN
088F  F2 00                        DB      0F2H,00H                ; DE    DAKUTEN
0891  F3 00                        DB      0F3H,00H                ; DF    HANDAKUTEN
0893  07                  DATA07  DB      7
                        =0001   NUMYOMFG        EQU     00000001B
                        =0002   NUMSKIP         EQU     00000010B
0894                    FNCHENK         PROC    NEAR
0894  C6 06 00F9 R 04              MOV     KKMODE,SELECTM  ; CHANGE MODE TO SELECT MODE
0899  F6 06 0192 R 01              TEST    KHENFST,01H
089E  75 03                        JNZ     HEN0010
08A0  E9 0AF8 R                    JMP     HEN3900
08A3                    HEN0010:
08A3  B8 0000                      MOV     AX,0            ; KKINBUF (KEY INPUT BUFFER) INDEX
08A6  8B F0                        MOV     SI,AX           ; KKHANQUE (QUEING BUFFER) POINTER
08A8  8B F8                        MOV     DI,AX           ; KKYOMI (YOMI CODE BUFFER) POINTER
08AA  8B E8                        MOV     BP,AX           ; KZENCONV (ZENKAKU CONVERSION TABLE)
08AC  8B D8                        MOV     BX,AX           ; OR KHANCONV ( HANKAKU CONV. TABLE )
08AE  A3 018E R                    MOV     KKHQLEN,AX
08B1  A2 0191 R                    MOV     KKYMSTS,AL
08B4  A2 0118 R                    MOV     KKYOMIL,AL
08B7  8B D0                        MOV     DX,AX
08B9  C6 06 0190 R FF              MOV     KKCHRMD,0FFH
08BE                    HEN0020:
08BE  8A 84 000A R                 MOV     AL,KKINBUF[SI][1]       ; GET INPUT KEY ATTRIBUTE
08C2  84 06 00E1 R                 TEST    AL,ZENATTR1
08C6  75 03                        JNZ     HEN1000
08C8  E9 09D9 R                    JMP     HEN2000
                        ;-------------------------------------------------------
                        ;----------  ZENKAKU CODE CONVERSION   -----------------
                        ;-------------------------------------------------------
08CB                    HEN1000:
08CB  8A A4 0009 R                 MOV     AH,KKINBUF[SI]
```

```
08CF  8A 84 000C R
08D3  3D 8140
08D6  74 4E
08D8  F6 06 0191 R 08
08DD  75 56
08DF  3D 814A
08E2  74 2C
08E4  3D 814B
08E7  74 27
08E9  B3 2D
08EB  3D 817C
08EE  74 09
08F0  B3 B0
08F2  3D 815B
08F5  74 02
08F7  EB 44
08F9
08F9  3E: C6 86 0119 R F4
08FF  8A C3
0901  8A A4 000B R
0905  E8 0B2A R
0908  80 26 0191 R FB
090D  E9 09C0 R
0910
0910  8B D8
0912  8A 36 0190 R
0916  80 FE FF
0919  74 02
091B  B6 02
091D
091D  81 EB 814A
0921  83 C3 53
0924  EB 68
0926
0926  83 C6 06
0929
0929  80 3E 0118 R 00
092E  75 05
0930  80 0E 0191 R 80
0935
0935  80 0E 0191 R 03
093A  E9 09C4 R
093D
093D  8B D8
093F  B0 01
0941  E8 1707 R
0944  84 C0
0946  74 14
0948  B0 02
094A  E8 1707 R
094D  84 C0
094F  74 1B
0951  B0 03
0953  E8 1707 R
0956  84 C0
0958  74 1A
095A  EB CD
095C
095C  B6 01
095E  81 EB 824F
0962  83 C3 56
0965  80 0E 0191 R 0A

096A  EB 15
096C
096C  B6 02
096E  81 EB 829F
0972  EB 0D
0974
0974  B6 03
0976  81 FB 837F
097A  72 01
097C  4B
097D
097D  81 EB 8340
0981
0981  80 3E 0190 R FF
0986  74 06
0988  38 36 0190 R
098C  75 A7
098E
098E  88 36 0190 R
0992  8B C3
0994  05 009F
0997  3E: 88 86 0119 R
099C  D1 E3
099E  2E: 8B 87 074D R
09A3  50
09A4  8A A4 000B R
09A8  E8 0B2A R
09AB  58
09AC  84 E4
09AE  74 10
09B0  47
09B1  47
09B2  B0 1A
09B4  80 FC DF
09B7  75 02
09B9  B0 1B
09BB
09BB  86 E0
09BD  E8 0B2A R
09C0
09C0  FE 06 0118 R
```

```
        MOV     AL,KKINBUF[SI][3]       ; GET ZENKAKU CODE
        CMP     AX,8140H                ; SPACE " " KEY
        JZ      HEN1090
        TEST    KKYMSTS,KKYMSTN
        JNZ     HEN1110
        CMP     AX,814AH                ; DAKUTEN KEY
        JZ      HEN1020
        CMP     AX,814BH                ; HANDAKUTEN KEY
        JZ      HEN1020
        MOV     BL,2DH
        CMP     AX,817CH                ; MINUS "-" KEY
        JZ      HEN1010
        MOV     BL,0B0H
        CMP     AX,815BH                ; CHOUON KEY
        JZ      HEN1010
        JMP     SHORT HEN1200           ; THE OTHER KEY
HEN1010:
        MOV     KKYOMI[BP],0F4H         ; SET CHOUON YOMI CODE
        MOV     AL,BL                   ;
        MOV     AH,KKINBUF[SI][2]       ; GET CHARACTER CODE & SCAN CODE
        CALL    QUEBUF                  ; SET CODES
        AND     KKYMSTS,0FFH-KKYMSTD    ; DAKUTEN,HANDAKUTEN BIT OFF
        JMP     HEN1270
HEN1020:
        MOV     BX,AX
        MOV     DH,KKCHRMD      ; GET PEVIOUS CHARACTER MODE
        CMP     DH,0FFH
        JE      HEN1021
        MOV     DH,MDZENHIR     ; SET ZENKAKU HIRAGANA MODE
HEN1021:
        SUB     BX,814AH
        ADD     BX,KZENDAKU     ; CALCULATE OFFSET OF CONVERSION TABLE
        JMP     SHORT HEN1250
HEN1090:
        ADD     SI,6
HEN1100:
        CMP     KKYOMIL,0
        JNZ     HEN1110
        OR      KKYMSTS,KKYMSTER        ; TURN ON ERROR BIT
HEN1110:
        OR      KKYMSTS,KKYMSTE+KKYMSTM
        JMP     HEN1280                 ; SET END & MOVE FORWORD FLAGS
HEN1200:
        MOV     BX,AX           ; BX <--- 2 BYTE KANJI CODE
        MOV     AL,1
        CALL    CDCHCK2         ; ZENKAKU NUMERIC CODE ?
        TEST    AL,AL
        JZ      HEN1210         ; JUMP IF YES
        MOV     AL,2
        CALL    CDCHCK2         ; ZENKAKU HIRAGANA CODE ?
        TEST    AL,AL
        JZ      HEN1220         ; JUMP IF YES
        MOV     AL,3
        CALL    CDCHCK2         ; ZENKAKU KATAKANA CODE ?
        TEST    AL,AL
        JZ      HEN1230         ; JUMP IF YES
        JMP     SHORT HEN1100           ; THE OTHER CODE ( CAN'T CONV. YOMI )
HEN1210:
        ;*******  ZENKAKU NUMERIC  *******
        MOV     DH,MDZENNUM
        SUB     BX,824FH
        ADD     BX,KZENNUMR     ; CALCULATE OFFSET OF CONVERSION TABLE
        OR      KKYMSTS,KKYMSTM+KKYMSTN
                                ; SEY END FLAG & MOVE FORWORD FLAG
        JMP     SHORT HEN1240
HEN1220:
        ;*******  ZENKAKU HIRAGANA  *******
        MOV     DH,MDZENHIR
        SUB     BX,829FH        ; CALCULATE OFFSET OF CONVERSION TABLE
        JMP     SHORT HEN1240
HEN1230:
        ;*******  ZENKAKU KATAKANA  *******
        MOV     DH,MDZENKAT
        CMP     BX,837FH
        JB      HEN1231
        DEC     BX
HEN1231:
        SUB     BX,8340H
HEN1240:
        CMP     KKCHRMD,0FFH
        JZ      HEN1250         ;
        CMP     KKCHRMD,DH      ; COMPARE CURRENT MODE WITH PREVIOUS
        JNZ     HEN1110
HEN1250:
        MOV     KKCHRMD,DH
        MOV     AX,BX
        ADD     AX,9FH
        MOV     KKYOMI[BP],AL   ; SET YOMI CONVERSION CODE
        SHL     BX,1                    ; BX * 2
        MOV     AX,WORD PTR KZENCONV[BX]  ; GET ZEN. KATAKANA & DAKUTEN
        PUSH    AX
        MOV     AH,KKINBUF[SI][2]       ; GET KEY SCAN CODE
        CALL    QUEBUF                  ; SET ONE
        POP     AX
        TEST    AH,AH
        JZ      HEN1270
        INC     DI
        INC     DI
        MOV     AL,1AH          ; SET DAKUTEN SCAN CODE
        CMP     AH,0DFH         ; HANDAKUTEN CODE ?
        JNZ     HEN1260
        MOV     AL,1BH
HEN1260:
        XCHG    AH,AL
        CALL    QUEBUF          ; SET DAKUTEN,HANDAKUTEN CODE
HEN1270:
        INC     KKYOMIL         ; INCREMENT YOMI LENGTH
```

```
09C4
09C4  F6 06 0191 R 02          HEN1280:  TEST    KKYMSTS,KKYMSTM
09C9  74 04                              JZ      HEN1290
09CB  89 36 00F7 R                       MOV     KKFORPIT,SI     ; SET MOVE FORWORD POINTER
09CF                           HEN1290:
09CF  83 C6 06                           ADD     SI,6            ; RENEW KKINBUF POINTER
09D2  83 C7 02                           ADD     DI,2            ; RENEW KKHANQUE POINTER
09D5  45                                 INC     BP              ; RENEW KKYOMI POINTER
09D6  E9 0AA9 R                          JMP     HEN3000
                               ;-----------------------------------------------------
                               ;----------    HANKAKU CODE CONVERSION    ------------
                               ;-----------------------------------------------------
09D9                           HEN2000:
09D9  2B DB                              SUB     BX,BX
09DB  8A 9C 0009 R                       MOV     BL,KKINBUF[SI]  ; GET HANKAKU CHARACTER CODE
09DF  80 FB 20                           CMP     BL,20H          ; SPACE " " KEY ?
09E2  74 34                              JE      HEN2090
09E4  F6 06 0191 R 08                    TEST    KKYMSTS,KKYMSTN
09E9  75 3C                              JNZ     HEN2110
09EB  80 FB 2D                           CMP     BL,2DH          ; MINUS KEY ?
09EE  74 17                              JE      HEN2010
09F0  80 FB B0                           CMP     BL,0B0H         ; CHOUON KEY ?
09F3  74 12                              JE      HEN2010
09F5  B7 01                              MOV     BH,1
09F7  80 FB DE                           CMP     BL,0DEH         ; DAKUTEN KEY ?
09FA  74 33                              JE      HEN2120
09FC  B7 02                              MOV     BH,2
09FE  80 FB DF                           CMP     BL,0DFH         ; HAN-DAKUTEN KEY ?
0A01  74 2C                              JE      HEN2120
0A03  B7 00                              MOV     BH,0
0A05  EB 42                              JMP     SHORT HEN2200
0A07                           HEN2010:
0A07  E8 0B22 R                          CALL    SETQUEBF                ; SET CHARACTER CODE & SCAN CODE
0A0A  3E: C6 86 0119 R F4                MOV     KKYOMI[BP],0F4H         ; SET CHOUON YOMI CODE
0A10  80 26 0191 R FB                    AND     KKYMSTS,0FFH-KKYMSTD    ; DAKUTEN,HANDAKUTEN BIT OFF
0A15  EB 7C 90                           JMP     HEN2270
0A18                           HEN2090:
0A18  83 C6 03                           ADD     SI,3
0A1B                           HEN2100:
0A1B  80 3E 0118 R 00                    CMP     KKYOMIL,0
0A20  75 85                              JNZ     HEN2110
0A22  80 0E 0191 R 80                    OR      KKYMSTS,KKYMSTER        ; TURN ON ERROR BIT
0A27                           HEN2110:
0A27  80 0E 0191 R 03                    OR      KKYMSTS,KKYMSTE+KKYMSTM
0A2C  EB 69 90                           JMP     HEN2280                 ; SET END FLAG & MOVE FORWORD FLAG
0A2F                           HEN2120:
0A2F  F6 06 0191 R 04                    TEST    KKYMSTS,KKYMSTD
0A34  74 0B                              JZ      HEN2130
0A36  4D                                 DEC     BP
0A37  3E: 00 BE 0119 R                   ADD     KKYOMI[BP],BH           ; ADJUST PREVIOUS YOMI CODE
0A3C  E8 0B22 R                          CALL    SETQUEBF                ; SET CHARACTER CODE & SCAN CODE
0A3F  EB 56                              JMP     SHORT HEN2280
0A41                           HEN2130:
0A41  2A FF                              SUB     BH,BH
0A43  81 EB 009D                         SUB     BX,9DH
0A47  EB 37                              JMP     SHORT HEN2260
0A49                           HEN2200:
0A49  B0 03                              MOV     AL,3
0A4B  E8 16CF R                          CALL    CDCHCK1         ; HANKAKU NUMERIC CODE ?
0A4E  84 C0                              TEST    AL,AL
0A50  74 0B                              JZ      HEN2210         ; JUMP IF YES
0A52  B0 04                              MOV     AL,4
0A54  E8 16CF R                          CALL    CDCHCK1         ; HANKAKU KATAKANA CODE ?
0A57  84 C0                              TEST    AL,AL
0A59  74 0E                              JZ      HEN2220         ; JMP IF YES
0A5B  EB BE                              JMP     SHORT HEN2100   ; THE OTHER CODE ( CAN'T CONV. YOMI )
0A5D                           HEN2210:  ;******  HANKAKU NUMERIC  ******
0A5D  B6 04                              MOV     DH,MDHANNUM
0A5F  83 EB 30                           SUB     BX,30H          ; CALCULATE OFFSET OF CONVERSION TABLE
0A62  80 0E 0191 R 0A                    OR      KKYMSTS,KKYMSTM+KKYMSTN
                                                                 ; SET END FLAG & MOVE FORWORD FLAG
0A67  EB 06                              JMP     SHORT HEN2240
0A69                           HEN2220:  ;******  HANKAKU KATAKANA  ******
0A69  B6 05                              MOV     DH,MDHANKAT
0A6B  81 EB 009D                         SUB     BX,9DH
0A6F                           HEN2240:
0A6F  80 3E 0190 R FF                    CMP     KKCHRMD,0FFH
0A74  74 06                              JZ      HEN2250
0A76  38 36 0190 R                       CMP     KKCHRMD,DH      ; COMPARE CURRENT MODE WITH PREVIOUS
0A7A  75 AB                              JNZ     HEN2110
0A7C                           HEN2250:
0A7C  88 36 0190 R                       MOV     KKCHRMD,DH
0A80                           HEN2260:
0A80  D1 E3                              SHL     BX,1                    ; BX * 2
0A82  2E: 8B 87 080D R                   MOV     AX,WORD PTR KHANCONV[BX]
0A87  3E: 88 86 0119 R                   MOV     KKYOMI[BP],AL           ; SET YOMI CONVERSION CODE
0A8C  08 26 0191 R                       OR      KKYMSTS,AH              ; SET DAKUTEN,HANDAKUTEN CODE
0A90  E8 0B22 R                          CALL    SETQUEBF                ; SET CHARACTER CODE & SCAN CODE
0A93                           HEN2270:
0A93  FE 06 0118 R                       INC     KKYOMIL         ; INCREMENT YOMI LENGTH
0A97                           HEN2280:
0A97  F6 06 0191 R 02                    TEST    KKYMSTS,KKYMSTM
0A9C  74 04                              JZ      HEN2290
0A9E  89 36 00F7 R                       MOV     KKFORPIT,SI     ; SET MOV FORWORD POINTER
0AA2                           HEN2290:
0AA2  83 C6 03                           ADD     SI,3            ; RENEW KKINBUF POINTER
0AA5  83 C7 02                           ADD     DI,2            ; RENEW KKHANQUE POINTER
0AA8  45                                 INC     BP              ; RENEW KKYOMI POINTER
0AA9                           HEN3000:
0AA9  A1 00EA R                          MOV     AX,KKWEOP
0AAC  2B C6                              SUB     AX,SI
0AAE  7E 0A                              JLE     HEN3010         ; ALL INPUT CHARACTER CONVERTED ?
0AB0  F6 06 0191 R 01                    TEST    KKYMSTS,KKYMSTE ; YES ...
0AB5  75 03                              JNZ     HEN3010         ; ENDFLAG CHECK
```

```
0AB7  E9 08BE R                 JMP     HEN0020         ; WHEN NOT END, REPEAT PROC.
                        ;-----------------------------------------------------------------
                        ;********          LINK TO DICTIONARY ACCESS ROUTINE,        **********
                        ;********                AND  EDIT CANDIDATE FORMAT.         **********
                        ;-----------------------------------------------------------------
0ABA                    HEN3010:
0ABA  F6 06 0191 R 80           TEST    KKYMSTS,KKYMSTER
0ABF  75 58                     JNZ     HEN4020
0AC1  BE 0118 R                 MOV     SI,OFFSET KKYOMIL
0AC4  1E                        PUSH    DS
0AC5  E8 0E89 R                 CALL    KKKNDR          ; LINK KANA-KAN CONVERSION ROUTINE
0AC8  1F                        POP     DS
0AC9  06                        PUSH    ES
0ACA  8F 06 0142 R              POP     KKDICSEG        ; SAVE SEGMENT OF DICTIONARY
0ACE  89 3E 0140 R              MOV     KKDICOFF,DI     ; SAVE OFFSET OF DICTIONARY
0AD2  A8 08                     TEST    AL,00001000B    ; INVALID YOMI CODE ?
0AD4  75 43                     JNZ     HEN4020         ; YES... ERROR RETURN
0AD6  41                        INC     CX
0AD7  89 0E 013E R              MOV     KKOHOSUU,CX     ; SAVE TOTAL KOUHO COUNT
0ADB  8B C1                     MOV     AX,CX
0ADD  2E: F6 36 0893 R          DIV     DATA07          ; CALCURATE MAX KOUHO BLOCK NUMBER
0AE2  84 E4                     TEST    AH,AH
0AE4  74 04                     JZ      HEN3020
0AE6  32 E4                     XOR     AH,AH
0AE8  EB 02                     JMP     SHORT HEN3030
0AEA                    HEN3020:
0AEA  FE C8                     DEC     AL
0AEC                    HEN3030:
0AEC  86 C4                     XCHG    AL,AH
0AEE  A3 00FD R                 MOV     WORD PTR KHBLK,AX ; SET INITIAL KOUHO BLOCK NUMBER
0AF1  33 C0                     XOR     AX,AX
0AF3  A2 0192 R                 MOV     KHENFST,AL      ;
0AF6  EB 1B                     JMP     SHORT HEN4000
0AF8                    HEN3900:
0AF8  A0 00FD R                 MOV     AL,KHBLK        ; CHECK END BLOCK NUMBER
0AFB  3A 06 00FE R              CMP     AL,KHBLKMX
0AFF  74 08                     JE      HEN3910         ; JUMP IF END BLOCK
0B01  FE 06 00FD R              INC     KHBLK           ; NEXT BLOCK
0B05  33 C0                     XOR     AX,AX
0B07  EB 0A                     JMP     SHORT HEN4000
0B09                    HEN3910:
0B09  83 3E 0144 R 01           CMP     KKOHOZAN,1      ;
0B0E  75 06                     JNE     HEN4010
0B10  B8 0001                   MOV     AX,1
0B13                    HEN4000:
0B13  E8 1396 R                 CALL    KKOHOEDT        ; EDIT OUTPUT BUFFER
0B16                    HEN4010:
0B16  2B C0                     SUB     AX,AX
0B18  C3                        RET
0B19                    HEN4020:
0B19  C6 06 00F9 R 02           MOV     KKMODE,KANJIM   ; CHANGE TO KANJI MODE
0B1E  B8 0001                   MOV     AX,1
0B21  C3                        RET
                        ;****************************************************************
                        ;*  INTERNAL SUB ROUTINE                                        *
                        ;*                  <SETQUEBF> , <QUEBUF>                       *
                        ;****************************************************************
0B22                    SETQUEBF:
0B22  8A 84 0009 R              MOV     AL,KKINBUF[SI]
0B26  8A A4 000B R              MOV     AH,KKINBUF[SI][2]       ; GET CHR. CODE & SCAN CODE
0B2A                    QUEBUF:
0B2A  89 85 014A R              MOV     WORD PTR KKHANQUE[DI],AX    ; SET ONE TO QUEING BUFFER
0B2E  FF 06 018E R              INC     KKHQLEN
0B32  C3                        RET
0B33                    FNCHENK ENDP
                                                ;********************************
                                                ;*                              *
                                                ;*          MUHENKAN            *
                                                ;*                              *
                                                ;********************************
0B33                    FNCMUHE PROC
0B33  80 3E 00F9 R 02           CMP     KKMODE,KANJIM   ; CURRENT MODE CHECK
0B38  74 1B                     JE      MUH020
                        ;       ********  SELECT MODE  ********
0B3A                    MUH010:
0B3A  A1 0148 R                 MOV     AX,KKOHODSP
0B3D  3B 06 013E R              CMP     AX,KKOHOSUU     ; DISPLAY LAST KOUHO NUMBER ?
0B41  74 0B                     JE      MUH011
0B43  80 3E 00FD R 00           CMP     KHBLK,0         ; CHECK FIRST KOUHO BLOCK
0B48  74 27                     JE      MUH050          ;
0B4A  FE 0E 00FD R              DEC     KHBLK           ; PREVIOUS BLOCK
0B4E                    MUH011:
0B4E  33 C0                     XOR     AX,AX
0B50  E8 1396 R                 CALL    KKOHOEDT        ; EDIT KOUHO
0B53  EB 1C                     JMP     SHORT MUH050
0B55                    MUH020:
0B55  2B ED                     SUB     BP,BP
0B57                    MUH030:
0B57  3B 2E 00EA R              CMP     BP,KKWEOP
0B5B  7D 11                     JGE     MUH040
0B5D  3E: 8A 86 0009 R          MOV     AL,KKINBUF[BP]          ; GET CHARACTER CODE
0B62  3E: 8A A6 000B R          MOV     AH,KKINBUF[BP][2]       ; GET KEY SCAN CODE
0B67  CD 79                     INT     79H                     ; QUEING
0B69  83 C5 03                  ADD     BP,3                    ; RENEW POINTER
0B6C  EB E9                     JMP     SHORT MUH030
0B6E                    MUH040:
0B6E  E8 0CA2 R                 CALL    KKZINT          ; INITIALIZE ZENKOHO MODE
0B71                    MUH050:
0B71  2B C0                     SUB     AX,AX
0B73  C3                        RET
0B74                    FNCMUHE  ENDP
                                                ;********************************
                                                ;*                              *
                                                ;*          ESCAPE              *
```

```
                                        ;*
                                        ;*************************************
                                                                            *
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;+      PRIVATE CONSTANT  < FNCESCP >                                       +
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
0B74  03                DIV03   DB      3
                        ;++++++++++++++++++++++++++++++++++++++++++++++++++++++
0B75                    FNCESCP PROC
0B75  80 3E 00F9 R 04           CMP     KKMODE,SELECTM
0B7A  74 05                     JE      ESC020
                        ;       ********  KANJI MODE  ********
0B7C                    ESC010:
0B7C  E8 0CA2 R                 CALL    KKZINT          ; CLEAR YOMI INPUT BUFFER
0B7F  EB 10                     JMP     SHORT ESC040
                        ;       ********  SELECT MODE  ********
0B81                    ESC020:
0B81  C6 06 00F9 R 02           MOV     KKMODE,KANJIM
0B86  E8 1291 R                 CALL    KOUTINIT        ;
0B89  E8 1355 R                 CALL    KCSRDSP         ; DISPLAY CURSOR
0B8C  C6 06 0192 R 01           MOV     KHENFST,01H
0B91                    ESC040:
0B91  2B C0                     SUB     AX,AX
0B93  C3                        RET
0B94                    FNCESCP ENDP
                                                ;*************************************
                                                ;*                                   *
                                                ;*          INSERT                   *
                                                ;*                                   *
                                                ;*************************************
0B94                    FNCINST PROC
0B94  80 3E 00F9 R 02           CMP     KKMODE,KANJIM   ; CHECK CURRENT MODE
0B99  75 12                     JNE     FNCINST_RET
0B9B  F6 06 00F0 R 01           TEST    KKWINST,01H
0BA0  74 04                     JZ      INS010
0BA2  33 C0                     XOR     AX,AX
0BA4  EB 04                     JMP     SHORT INS020
0BA6                    INS010:
0BA6  8D 06 0003 R              LEA     AX,KKINKEY
0BAA                    INS020:
0BAA  E8 0CF2 R                 CALL    KKINST
0BAD                    FNCINST_RET:
0BAD  C3                        RET
0BAE                    FNCINST ENDP
                                                ;*************************************
                                                ;*                                   *
                                                ;*          DELETE                   *
                                                ;*                                   *
                                                ;*************************************
0BAE                    FNCDELT PROC
0BAE  80 3E 00F9 R 02           CMP     KKMODE,KANJIM   ; CHECK CURRENT MODE
0BB3  75 03                     JNE     FNCDELT_RET
0BB5  E8 0E5A R                 CALL    KKDELT
0BB8                    FNCDELT_RET:
0BB8  C3                        RET
0BB9                    FNCDELT ENDP
                                                ;*************************************
                                                ;*                                   *
                                                ;*          BACKSPACE                *
                                                ;*                                   *
                                                ;*************************************
0BB9                    FNCBACK PROC
0BB9  80 3E 00F9 R 02           CMP     KKMODE,KANJIM   ; CHECK CURRENT MODE
0BBE  75 03                     JNE     FNCBACK_RET
0BC0  E8 0E04 R                 CALL    KKBACK
0BC3                    FNCBACK_RET:
0BC3  C3                        RET
0BC4                    FNCBACK ENDP
                                                ;*************************************
                                                ;*                                   *
                                                ;*          ERASE EOF                *
                                                ;*                                   *
                                                ;*************************************
0BC4                    FNCEEOF PROC
0BC4  80 3E 00F9 R 02           CMP     KKMODE,KANJIM   ; CHECK CURRENT MODE
0BC9  75 03                     JNE     FNCEEOF_RET
0BCB  E8 0DAF R                 CALL    KKEEOF
0BCE                    FNCEEOF_RET:
0BCE  C3                        RET
0BCF                    FNCEEOF ENDP
                                                ;*************************************
                                                ;*                                   *
                                                ;*       CARRIER RETURN              *
                                                ;*                                   *
                                                ;*************************************
0BCF                    FNCCRRT PROC    NEAR
0BCF  80 3E 00F9 R 02           CMP     KKMODE,KANJIM
0BD4  74 04                     JE      CRR010
0BD6  B8 0001                   MOV     AX,1            ; ERROR RETURN
0BD9  C3                        RET
0BDA                    CRR010:
0BDA  A1 00EA R                 MOV     AX,KKWEOP       ; INPUT BUFFER END POINTER CHECK
0BDD  0B C0                     OR      AX,AX           ;
0BDF  74 10                     JZ      CRR020          ;
0BE1  A3 00EC R                 MOV     KKWEOPSV,AX     ; SAVE END OF POINTER
0BE4  8B C8                     MOV     CX,AX           ; GET COPY COUNT
0BE6  1E                        PUSH    DS
0BE7  07                        POP     ES              ; GET DESTINATION SEGMENT
0BE8  BE 0009 R                 MOV     SI,OFFSET KKINBUF       ; SI <--- KKINBUF  ( SOURCE )
0BEB  BF 0075 R                 MOV     DI,OFFSET KKINBFSV      ; DI <--- KKINBFSB ( DESTINATION )
0BEE  FC                        CLD
0BEF  F3/ A4                    REP MOVSB
0BF1                    CRR020:
0BF1  2B C0                     SUB     AX,AX
```

```
0BF3  C3                        RET
0BF4                    FNCCRRT ENDP
                                                ;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                                ;%                              %
                                                ;%      ENTER                   %
                                                ;%                              %
                                                ;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0BF4                    FNCENTR PROC    NEAR
0BF4  80 3E 00F9 R 02           CMP     KKMODE,KANJIM
0BF9  74 04                     JZ      ENT010
0BFB  B8 0001                   MOV     AX,1            ; ERROR RETURN
0BFE  C3                        RET
0BFF                    ENT010:
0BFF  1E                        PUSH    DS
0C00  07                        POP     ES
0C01  8B 16 00EC R              MOV     DX,KKWEOPSV
0C05  85 D2                     TEST    DX,DX
0C07  74 22                     JZ      ENT030
0C09  BE 0075 R                 MOV     SI,OFFSET KKINBFSV
0C0C                    ENT020:
0C0C  BF 0003 R                 MOV     DI,OFFSET KKKYDCD
0C0F  8A 44 01                  MOV     AL,DS:[SI][1]
0C12  E8 1341 R                 CALL    KCALELM
0C15  8B C8                     MOV     CX,AX
0C17  2B D1                     SUB     DX,CX
0C19  FC                        CLD
0C1A  F3/ A4                    REP MOVSB               ; COPY TO KKKYDCD FROM KKINBFSV
0C1C  56                        PUSH    SI
0C1D  52                        PUSH    DX
0C1E  8D 06 0003 R              LEA     AX,KKKYDCD
0C22  E8 1209 R                 CALL    KKGRP           ; INPUT BUFFER EDIT & DISPLAY
0C25  5A                        POP     DX
0C26  5E                        POP     SI
0C27  0B D2                     OR      DX,DX           ; CHARACTER CHANGE END ?
0C29  75 E1                     JNZ     ENT020
0C2B                    ENT030:
0C2B  2B C0                     SUB     AX,AX
0C2D  C3                        RET
0C2E                    FNCENTR ENDP
                        ;%%%%%%%%%%%%%%%%%%%
```

```
0C5A  2B C0                   SUB     AX,AX
0C5C  3E: 8A 86 0007 R        MOV     AL,KKINBUF[BP][-2]
                                      ; GET PREVIOUS CHARACTER ATTRIBUTE
0C61  22 06 00E1 R            AND     AL,ZENATTR1     ; ZENKAKU BIT CHECK
0C65  74 04                   JZ      KCS020
0C67  B0 FE                   MOV     AL,-2           ; ZENKAKU
0C69  EB 1F                   JMP     SHORT KCS040
0C6B                  KCS020:
0C6B  B0 FF                   MOV     AL,-1           ; HANKAKU
0C6D  EB 1B                   JMP     SHORT KCS040
                      ;*****************************************************************
                      ;       CURSOR RIGHT                                            *
                      ;                                                               *
                      ;*****************************************************************
0C6F                  KCS100:
0C6F  8B 2E 00E6 R            MOV     BP,KKWCCA       ; GET CONTECTUAL CURSOR ADDRESS
0C73  3B 2E 00EA R            CMP     BP,KKWEOP       ; IF CONTECTUAL CURSOR IS ZERO
0C77  73 22                   JAE     KCS200          ; THEN RETURN
0C79  3E: 8A 86 000A R        MOV     AL,KKINBUF[1][BP]
                                      ; GET PREVIOUS CHARACTER ATTRIBUTE
0C7E  22 06 00E1 R            AND     AL,ZENATTR1     ; ZENKAKU BIT CHECK
0C82  74 04                   JZ      KCS030
0C84  B0 02                   MOV     AL,2            ; ZENKAKU
0C86  EB 02                   JMP     SHORT KCS040
0C88                  KCS030:
0C88  B0 01                   MOV     AL,1            ; HANKAKU
0C8A                  KCS040:
0C8A  98                      CBW
0C8B  01 06 0105 R            ADD     DCRSRP,AX       ; CALCULATE NEW X-POSITION
0C8F  B8 FFFF                 MOV     AX,0FFFFH
0C92  A3 0107 R               MOV     DSTRTP,AX       ; SET PARAMETER FOR DISPLAY (KKDISP)
0C95  A3 0109 R               MOV     DENDP,AX
0C98  E8 10ED R               CALL    KKDISP          ; CURSOR MOTION
0C9B                  KCS200:
0C9B  33 C0                   XOR     AX,AX           ; RETURN CODE = 0
0C9D  C3                      RET
0C9E                  KCS210:
0C9E  B8 0001                 MOV     AX,1            ; RETURN CODE = ABNORMAL
0CA1  C3                      RET
0CA2                  KKCSR   ENDP
                      ;*****************************************************************
                      ;*                                                               *
                      ;* PROGRAM NAME: KKZINT                                          *
                      ;*                                                               *
                      ;* DESCRIPTIVE NAME:  INITIALIZE ZEN KOUHO MODE                  *
                      ;*                                                               *
                      ;* FUNCTION:                                                     *
                      ;*        INITIALIZE TABLE FOR KANJI MODE                        *
                      ;*                                                               *
                      ;* LINKAGE:  CALLED BY KKKFDM                                    *
                      ;*           ( JUST INPUT KANJI KEY )                            *
                      ;*                                                               *
                      ;*   INPUT:    KANA-KAN COMMON TABLES                          *
                      ;*                                                               *
                      ;*   OUTPUT:   KANA-KAN COMMON TABLES                          *
                      ;*                                                               *
                      ;*   RETURN CODES: (AX)                                        *
                      ;*                                                               *
                      ;*      0 - SUCCESSFUL                                         *
                      ;*      1 - INVALID OPERATION                                  *
                      ;*                                                               *
                      ;*  EXTERNAL REFERENCES:                                       *
                      ;*                                                               *
                      ;*    ROUTINES: KKWOIL                                         *
                      ;*              KKDISP                                         *
                      ;*                                                               *
                      ;*    TABLES: KANA-KAN COMMON TABLES                           *
                      ;*                                                               *
                      ;*  REGISTERS: AX - RETURN CODE                                *
                      ;*             ALL OTHERS UNCHANGED                            *
                      ;*                                                               *
                      ;*  CHANGE ACTIVITY: VERSION 00.00                             *
                      ;*                                                               *
                      ;*****************************************************************
0CA2                  KKZINT  PROC    NEAR
0CA2  C6 06 00F9 R 02         MOV     KKMODE,KANJIM   ; TURN TO KANJI MODE
0CA7  80 3E 00FA R 01         CMP     TVMODE,TV8025   ; VIDEO MODE CHECK
0CAC  74 1E                   JE      KZI200          ; change 100 to 200
0CAE  80 3E 00FA R 02         CMP     TVMODE,TV8011   ;
0CB3  74 17                   JE      KZI200
0CB5  80 3E 00FA R 03         CMP     TVMODE,TV4011   ;
0CBA  74 10                   JE      KZI200
0CBC  EB 31                   JMP     SHORT KZI900
                      ;**********************  INITIALIZE  80 X 25  TYPE  VIDEO  **********
0CBE                  KZI100:
0CBE  C7 06 00E8 R 0066       MOV     KKEOPMAX,34*3
0CC4  C7 06 00EE R 002C       MOV     BASE,44
0CCA  EB 0E                   JMP     SHORT KZI800
                      ;********  INITIALIZE  80 X 11  AND  40 X 11 TYPE  VIDEO  **********
0CCC                  KZI200:
0CCC  C7 06 00E8 R 0042       MOV     KKEOPMAX,22*3
0CD2  C7 06 00EE R 0011       MOV     BASE,17
0CD8  EB 00                   JMP     SHORT KZI800
                      ;**********  INITIALIZE  COMMON AREA  **************************
0CDA                  KZI800:
0CDA  B4 00                   MOV     AH,HANATTR
0CDC  B0 A0                   MOV     AL,SPECHAR
0CDE  A3 00F5 R               MOV     KKCHRSP,AX      ; SET SPECIAL CHARACTER FOR YOMI SP.
0CE1  E8 12FC R               CALL    KINBFCLR        ; KEY INPUT BUFFER (KKINBUF) CLEAR
0CE4  E8 1291 R               CALL    KOUTINIT        ; OUTPUT BUFFER (KKOUTBUF) INITIAL
0CE7  E8 1355 R               CALL    KCSRDSP         ; DISPLAY CURSOR
```

```
0CEA  C6 06 0192 R 01          MOV     KHENFST,01H
0CEF                        KZI900:
0CEF  2B C0                    SUB     AX,AX
0CF1  C3                       RET
0CF2                        KKZINT  ENDP
                            ;%%%
```

```
0D3A                    INPRC_050:
0D3A  A1 00E6 R                 MOV     AX,KKWCCA       ; GET CURRENT CURSOR POSITION
0D3D  03 C2                     ADD     AX,DX           ;
0D3F  3B 06 00E8 R              CMP     AX,KKEOPMAX     ; CHECK CURSOR POSITION
0D43  77 33                     JA      INPRC_120       ; ERROR RETURN
0D45  E8 1573 R                 CALL    DTINST          ; INSERT DATA FROM TABLE TO INPUT BUFFE
0D48  8B C2                     MOV     AX,DX           ; GET INSERT CHARACTER ELEMENT (3 OR 6)
0D4A  B1 03                     MOV     CL,INDEX
0D4C  F6 F1                     DIV     CL
0D4E  32 E4                     XOR     AH,AH
0D50  03 06 00F3 R              ADD     AX,KKCUSR
0D54  A3 0105 R                 MOV     DCRSRP,AX       ; SET CURSOR POSITION
0D57  E8 16A3 R                 CALL    SETSTRCL        ; SET START POSITION
0D5A  E8 16B0 R                 CALL    SETENDCL        ; SET END POSITION
0D5D                    INPRC_090:
0D5D  E8 10ED R                 CALL    KKDISP          ; CALL DISPLAY ROUTINE
0D60  A1 00EA R                 MOV     AX,KKWEOP       ; GET END POSITION
0D63  3B 06 00E8 R              CMP     AX,KKEOPMAX     ; CHECK END POSITION
0D67  72 06                     JB      INPRC_100       ;
0D69  A1 00E8 R                 MOV     AX,KKEOPMAX     ; GET BUFFER END POSITION
0D6C  A3 00EA R                 MOV     KKWEOP,AX       ; RESET END POSITION
0D6F                    INPRC_100:
0D6F  01 16 00E6 R              ADD     KKWCCA,DX       ; UPDATE CURRENT CURSOR POSITION
0D73  B8 0000                   MOV     AX,0H           ; SET RETURN CODE(NORMAL)
0D76  EB 03                     JMP     SHORT INPRC_990
0D78                    INPRC_120:
0D78  B8 0001                   MOV     AX,01H          ; SET RETURN CODE ( ERROR )
0D7B                    INPRC_990:
0D7B  C3                        RET                     ; RETURN TO CALLER
0D7C                    INSTPRC ENDP
                        ;*****<OFINST>*****************************************************
                        ;*                                                               *
                        ;*      RESET INSERT_MODE,WHEN INSERT KEY IS PRESSED ON INSERT_MODE  *
                        ;*                                                               *
                        ;*****<OFINST>*****************************************************
0D7C                    OFINST          PROC    NEAR
0D7C  80 26 00F0 R FE           AND     KKWINST,0FEH    ; RESET INSERT MODE FLAG
                        ;
                        ;       CMP     TVMODE,TV8025
                        ;       JNE     OFINS010
                        ;       TEST    KKFLAG,CLRF25   ; TV MODE IS 80*25 COLOR ?
                        ;       JNZ     OFINS010        ; YES. GOTO
                        ;       MOV     DSTRTP,0FFFFH
                        ;       CALL    KKDISP          ; DISPLAY CURSOR
                        ;       RET                     ; RETURN TO CALLER
                        ;OFINS010:
0D81  B0 2A                     MOV     AL,ASTER
0D83  E8 0D92 R                 CALL    SETMARK         ; DISPLAY INSERTION MARK
0D86  C3                        RET
0D87                    OFINST  ENDP
                        ;*****<SETINST>***************************************************
                        ;*                                                               *
                        ;*      SET INSERT_MODE,WHEN INSERT KEY IS PRESSED ON NOT INSERT_MODE *
                        ;*                                                               *
                        ;*****<SETINST>***************************************************
0D87                    SETINST         PROC    NEAR
0D87  80 0E 00F0 R 01           OR      KKWINST,01H     ; SET INSERT MODE
                        ;       CMP     TVMODE,TV8025
                        ;       JNE     STINS010
                        ;       TEST    KKFLAG,CLRF25   ; TV MODE IS 80*25 COLOR ?
                        ;       JNZ     STINS010        ; YES. GOTO
                        ;       MOV     DSTRTP,0FFFFH
                        ;       CALL    KKDISP          ; DISPLAY CURSOR
                        ;       RET                     ; RETURN TO CALLER
                        ;STINS010:
0D8C  B0 5E                     MOV     AL,CAPS
0D8E  E8 0D92 R                 CALL    SETMARK         ; DISPLAY INSERTION MARK
0D91  C3                        RET                     ; RETURN TO CALLER
0D92                    SETINST ENDP
                        ;*****<SETMARK>***************************************************
                        ;*                                                               *
                        ;*      DISPLAY INSERT MARK OR ASTER MARK                        *
                        ;*                                                               *
                        ;*****<SETMARK>***************************************************
0D92                    SETMARK PROC    NEAR
0D92  B4 00                     MOV     AH,HANATTR      ; GET HANKAKU ATTRIBUTE
0D94  8B 1E 00E8 R              MOV     BX,KKEOPMAX
0D98  89 87 0009 R              MOV     WORD PTR KKINBUF[BX],AX   ; SET CHARACTER CODE
0D9C  89 1E 00F1 R              MOV     KKINP,BX        ; SET DISPLAY POINTER
0DA0  8B C3                     MOV     AX,BX
0DA2  E8 16BA R                 CALL    GETCULM         ; CALCULATE COLUMN COUNT
0DA5  A3 0107 R                 MOV     DSTRTP,AX       ; SET COLUMN POSITION
0DA8  A3 0109 R                 MOV     DENDP,AX
0DAB  E8 10ED R                 CALL    KKDISP
0DAE  C3                        RET                     ; RETURN TO CALLER
0DAF                    SETMARK ENDP                    ;
                        ;*****************************************************************
                        ;*                                                               *
                        ;* PROGRAM NAME: KKEEOF                                          *
                        ;*                                                               *
                        ;* DESCRIPTIVE NAME: KANA-KAN ERASE EOF PROCESS                  *
                        ;*                                                               *
                        ;* FUNCTION: WHEN ERASE EOF KEY IS PRESSED, THIS ROUTINE ERASE THE *
                        ;*           CHARACTERS AFTER THE CURSOR POSITION AND SET SPECIAL *
                        ;*           CHARACTER.                                          *
                        ;*                                                               *
                        ;* LINKAGE: CALL                                                 *
                        ;*                                                               *
                        ;*    INPUT: KANA-KAN COMMON TABLES                              *
                        ;*                                                               *
                        ;*    OUTPUT: KANA-KAN COMMON TABLES                             *
                        ;*                                                               *
                        ;*    RETURN CODES: (AX)                                         *
                        ;*                                                               *
```

```
                        ;*      0 - SUCCESSFUL                                          *
                        ;*      1 - INVALID OPERATION                                   *
                        ;*                                                              *
                        ;*  EXTERNAL REFERENCES:                                        *
                        ;*                                                              *
                        ;*    ROUTINES: KKDISP - DISPLAY YOMI AND BANGOU                *
                        ;*              EOPCHECK - CHECK END POSITION OF CHARACTER       *
                        ;*                                                              *
                        ;*    TABLES: KANA-KAN COMMON TABLES                            *
                        ;*                                                              *
                        ;*  REGISTERS: AX - RETURN CODE                                 *
                        ;*             ALL OTHERS UNCHANGED                             *
                        ;*                                                              *
                        ;*  CHANGE ACTIVITY: VERSION 00.00                              *
                        ;*                                                              *
                        ;*                                                              *
                        ;****************************************************************
= 0006                  ZENLEN  EQU     06H             ;ZENKAKU DATA LENGTH
                        ;********************* KKEEOF *******************************
                        ;*
                        ;*      ENTRY OF KKEEOF                                         *
                        ;*                                                              *
                        ;****************************************************************
0DAF                    KKEEOF  PROC    NEAR
0DAF  53                        PUSH    BX              ;SAVE REGISTER
0DB0  51                        PUSH    CX
0DB1  52                        PUSH    DX
0DB2  56                        PUSH    SI
0DB3  57                        PUSH    DI
0DB4                    KKEE_020:
0DB4  A1 00E6 R                 MOV     AX,KKWCCA       ; GET CURRENT CURSOR POSITION
0DB7  3B 06 00EA R              CMP     AX,KKWEOP       ; CHECK CURRENT CURSOR POSITION
0DBB  73 15                     JAE     KKEE_090        ; CURSOR > DATA END , GOTO RETURN
0DBD  E8 0DF2 R                 CALL    POINTST         ; SET PARAMETERS FOR SETCHR ROUTINE
0DC0  E8 0DDB R                 CALL    SETCHR          ; SET CHARACTERS AFTER THE CURSOR
0DC3                    KKEE_060:
0DC3  E8 16A3 R                 CALL    SETSTRCL        ; SET START POSITION
0DC6  E8 16B0 R                 CALL    SETENDCL        ; SET END POSITION
0DC9  E8 10ED R                 CALL    KKDISP          ; CALL  DISPLAY ROUTINE
0DCC  A1 00E6 R                 MOV     AX,KKWCCA       ; GET CURSOR POSITION
0DCF  A3 00EA R                 MOV     KKWEOP,AX       ; SET NEW END POSITION
0DD2                    KKEE_090:
0DD2  B8 0000                   MOV     AX,0H           ; SET RETURN CODE ( NORMAL )
0DD5  5F                        POP     DI              ; RESTORE REGISTERS
0DD6  5E                        POP     SI
0DD7  5A                        POP     DX
0DD8  59                        POP     CX
0DD9  5B                        POP     BX
0DDA                    KKEE_990:
0DDA  C3                        RET                     ; RETURN TO CALLER
0DDB                    KKEEOF  ENDP
                        ;******<SETCHR>*************************************************
                        ;*
                        ;*      SET SPECIAL CHARACTER AFTER THE CURSOR                  *
                        ;*      INPUT:                                                  *
                        ;*            CX -- LOOP COUNT                                  *
                        ;*            DX -- START POSITION TO SET CHR.                  *
                        ;*                                                              *
                        ;******<SETCHR>*************************************************
0DDB                    SETCHR  PROC    NEAR
0DDB  8D 3E 0009 R              LEA     DI,KKINBUF      ; GET OFFSET OF INPUT BUFFER
0DDF  03 FA                     ADD     DI,DX           ; SI POINTS START POSITION TO SET
0DE1                    ST_010:
0DE1  83 F9 00                  CMP     CX,0H           ; LOOP END ?
0DE4  76 0B                     JBE     ST_990          ; YES,GOTO RETURN
0DE6  FC                        CLD
0DE7  A1 00F5 R                 MOV     AX,KKCHRSP      ; GET SPECIAL CHARACTER CODE
0DEA  AB                        STOSW                   ; SET CHRACTER CODE
0DEB  32 C0                     XOR     AL,AL
0DED  AA                        STOSB                   ; CLEAR SCAN CODE AREA
0DEE  49                        DEC     CX
0DEF  EB F0                     JMP     SHORT ST_010            ; LOOP PROC.
0DF1                    ST_990:
0DF1  C3                        RET                     ; RETURN TO CALLER
0DF2                    SETCHR  ENDP
                        ;******<POINTST>*************************************************
                        ;*
                        ;*      SET START & END POSITION AND LOOP_CNT TO BE SET SPECIAL CHR.*
                        ;*      OUTPUT:                                                 *
                        ;*            CX -- LOOP COUNT                                  *
                        ;*            DX -- START POSITION TO SET CHR.                  *
                        ;*                                                              *
                        ;******<POINTST>*************************************************
0DF2                    POINTST PROC    NEAR            ;
0DF2  8B 16 00E6 R              MOV     DX,KKWCCA       ; GET CURRENT CURSOR POSITION
0DF6  A1 00EA R                 MOV     AX,KKWEOP       ; GET END POSITION
0DF9  2B C2                     SUB     AX,DX
0DFB  B1 03                     MOV     CL,INDEX        ;
0DFD  F6 F1                     DIV     CL
0DFF  32 E4                     XOR     AH,AH           ; CLEAR AH
0E01  8B C8                     MOV     CX,AX           ; SET LOOP_CNT.
0E03  C3                        RET                     ; RETURN TO CALLER
0E04                    POINTST ENDP
                        ;****************************************************************
                        ;*
                        ;* PROGRAM NAME: KKBACK                                         *
                        ;*                                                              *
                        ;* DESCRIPTIVE NAME: KANAKAN BACK SPACE PROCESS ROUTINE         *
                        ;*                                                              *
                        ;* FUNCTION: WHEN BACK SPACE KEY IS PRESSED,THIS ROUTINE DELETES*
                        ;*           THE CHARACTER FRONT OF THE CURSOR AND SHIFTS CHARACTERS*
                        ;*           AFTER THE CURSOR POSITION LEFT                     *
                        ;*                                                              *
```

```
;* LINKAGE: CALL KKBACK                                                  *
;*                                                                       *
;*   INPUT: KANA-KAN COMMON TABLES                                       *
;*                                                                       *
;*   OUTPUT: KANA-KAN COMMON TABLES                                      *
;*                                                                       *
;*   RETURN CODES: (AX)                                                  *
;*                                                                       *
;*      0 - SUCCESSFUL                                                   *
;*      1 - INVALID OPERATION                                            *
;*                                                                       *
;*  EXTERNAL REFERENCES:                                                 *
;*                                                                       *
;*    ROUTINES: KKDISP -- DISPLAY DATA OF KANAKAN INPUT BUFFER           *
;*              DTSHFT -- SHIFT DATA FO INPUT BUFFER AND CHECK           *
;*                        END POSITION                                   *
;*                                                                       *
;*    TABLES: KANA-KAN COMMON TABLES                                     *
;*                                                                       *
;*  REGISTERS: AX - RETURN CODE                                          *
;*             ALL OTHERS UNCHANGED                                      *
;*                                                                       *
;*  CHANGE ACTIVITY: VERSION 00.00                                       *
;*                                                                       *
;************************************************************************
;
;
;*****<KKBACK>***********************************************************
;*                                                                       *
;*      WHEN BACK SPACE KEY IS PRESSED, THIS ROUTINE IS CALLED           *
;*                                                                       *
;*****<KKBACK>***********************************************************
0E04                    KKBACK  PROC    NEAR
0E04  53                        PUSH    BX              ; SAVE REGISTERS
0E05  51                        PUSH    CX
0E06  52                        PUSH    DX
0E07  56                        PUSH    SI
0E08  57                        PUSH    DI
0E09                    KKBK_020:                       ;
0E09  8D 36 0089 R              LEA     SI,KKINBUF      ; GET OFFSET OF INPUT BUFFER
0E0D  A1 00E6 R                 MOV     AX,KKWCCA       ; GET CURRENT CURSOR POSITION
0E10  3D 0000                   CMP     AX,0H           ; CURSOR = BUFFER HEAD ?
0E13  75 05                     JNZ     KKBK_070        ; NO,GOTO...
0E15                    KKBK_040:                       ; CURSOR IS BUFFER HEAD
0E15  BB 0000                   MOV     BX,0
0E18  EB 0E                     JMP     SHORT KKBK_137
0E1A                    KKBK_070:                       ; CURSOR ISN'T BUFFER HEAD
0E1A  8B 1E 00E6 R              MOV     BX,KKWCCA       ; GET CURRENT CURSOR POSITION
0E1E  03 DE                     ADD     BX,SI           ;
0E20  8A 47 FE                  MOV     AL,BYTE PTR DS:[BX-2]
                                                        ; GET ATTR. OF PREVIOUS DATA
0E23  E8 1341 R                 CALL    KCALELM         ; GET DATA LENGTH
0E26  8B D8                     MOV     BX,AX
0E28                    KKBK_137:
0E28  E8 0E3E R                 CALL    BKSHFT
0E2B  E8 16B0 R                 CALL    SETENDCL        ; SET END POSITION
0E2E  E8 10ED R                 CALL    KKDISP          ; CALL DISPLAY ROUTINE
0E31                    KKBK_150:
0E31  29 16 00EA R              SUB     KKWEOP,DX       ; UPDATE END POINT
0E35                    KKBK_160:
0E35  B8 0000                   MOV     AX,0            ; SET RETURN CODE (NORMAL)
0E38  5F                        POP     DI              ; RESTORE REGISTERS
0E39  5E                        POP     SI
0E3A  5A                        POP     DX
0E3B  59                        POP     CX
0E3C  5B                        POP     BX
0E3D                    KKBK_990:
0E3D  C3                        RET                     ; RETURN TO CALLER
0E3E                    KKBACK  ENDP
;*****<BKSHFT>**********************************************************
;*                                                                       *
;*      SHIFT    PROCESS OF BACK SPACE PROC. ROUTINE                     *
;*                                                                       *
;*****<BKSHFT>**********************************************************
0E3E                    BKSHFT  PROC    HEAR            ;
0E3E  A1 00E6 R                 MOV     AX,KKWCCA       ; SET PARAMETER TO DTSHFT
0E41  2B C3                     SUB     AX,BX           ; AX CONTAIN DATA POSITION
0E43  E8 162F R                 CALL    DTSHFT          ; SHIFT DATA & CHECK EOP
0E46  29 1E 00E6 R              SUB     KKWCCA,BX       ; UPDATE CURRENT CURSOR POSITION
0E4A  8B C3                     MOV     AX,BX           ; GET DATA LENGTH
0E4C  B1 03                     MOV     CL,INDEX        ;
0E4E  F6 F1                     DIV     CL              ;
0E50  32 E4                     XOR     AH,AH           ; CLEAR AH
0E52  29 06 0105 R              SUB     DCRSRP,AX       ; SET CURSOR POSITION
0E56  E8 16A3 R                 CALL    SETSTRCL        ; SET START POSITION
0E59  C3                        RET
0E5A                    BKSHFT  ENDP
;************************************************************************
;*                                                                       *
;* PROGRAM NAME: KKDELT                                                  *
;*                                                                       *
;* DESCRIPTIVE NAME: KANAKAN DELETE PROCESS ROUTINE                      *
;*                                                                       *
;* FUNCTION: WHEN DELETE KEY IS PRESSED, THIS ROUTINE DELETS             *
;*           THE CHARCTER ON THE CURSOR POSITION AND SHFT CHARACTERS      *
;*           AFTER THE CURSOR LEFT                                       *
;*                                                                       *
;* LINKAGE: CALL KKDELT                                                  *
;*                                                                       *
;*   INPUT: KANA-KAN COMMON TABLES                                       *
;*                                                                       *
;*   OUTPUT: KANA-KAN COMMON TABLES                                      *
;*                                                                       *
;*   RETURN CODES: (AX)                                                  *
```

```
                        ;*                                                                    *
                        ;*      0 - SUCCESSFUL                                                *
                        ;*      1 - INVALID OPERATION                                         *
                        ;*                                                                    *
                        ;*  EXTERNAL REFERENCES:                                              *
                        ;*                                                                    *
                        ;*    ROUTINES: KKDISP -- DISPLAY DATA OF KANAKAN INPUT BUFFER        *
                        ;*              DTSHFT -- SHIFT DATA OF INPUT BUFFER AND CHECK        *
                        ;*                        END POSITION                                *
                        ;*                                                                    *
                        ;*    TABLES: KANA-KAN COMMON TABLES                                  *
                        ;*                                                                    *
                        ;*  REGISTERS: AX - RETURN CODE                                       *
                        ;*             ALL OTHERS UNCHANGED                                   *
                        ;*                                                                    *
                        ;*  CHANGE ACTIVITY: VERSION 00.00                                    *
                        ;*                                                                    *
                        ;**************************************************************************
                        ;
                        ;
                        ;*****<KKDELT>*************************************************************
                        ;*                                                                    *
                        ;*      THIS ROUTINE DELETES A CHARACTER ON THE CURSOR                *
                        ;*                                                                    *
                        ;*****<KKDELT>*************************************************************
0E5A                    KKDELT  PROC    NEAR            ;
0E5A  53                        PUSH    BX              ; SAVE REGISTERS
0E5B  51                        PUSH    CX
0E5C  52                        PUSH    DX
0E5D  56                        PUSH    SI
0E5E  57                        PUSH    DI
0E5F  A1 00E6 R                 MOV     AX,KKWCCA       ; GET CURRENT CURSOR POSITION
0E62  3B 06 00E8 R              CMP     AX,KKEOPMAX     ; CHECK BUFFER END
0E66  73 18                     JAE     KKDL_200
0E68  3B 06 00EA R              CMP     AX,KKWEOP       ; CHECK CURSOR POINT
0E6C  73 12                     JAE     KKDL_110        ; NOP
0E6E  E8 162F R                 CALL    DTSHFT          ; SHIFT DATA & CHECK EOP
0E71  E8 16A3 R                 CALL    SETSTRCL        ; SET START POSITION
0E74  E8 16B0 R                 CALL    SETENDCL        ; SET END POSITION
0E77  E8 10ED R                 CALL    KKDISP          ; CALL DISPLAY ROUTINE
0E7A                    KKDL_100:
0E7A  29 16 00EA R              SUB     KKWEOP,DX       ; UPDATE ENDPOSITION
0E7E  EB 00                     JMP     SHORT KKDL_110  ;
0E80                    KKDL_200:
0E80                    KKDL_110:
0E80  B8 0000                   MOV     AX,0H           ; SET RETURN CODE
0E83  5F                        POP     DI              ; RESTORE REGISTERS
0E84  5E                        POP     SI
0E85  5A                        POP     DX
0E86  59                        POP     CX
0E87  5B                        POP     BX
0E88                    KKDL_990:
0E88  C3                        RET                     ; RETURN TO CALLER
0E89                    KKDELT  ENDP
                        ;**************************************************************************
                        ;* PROGRAM NAME: KKKNDR                                               *
                        ;*                                                                    *
                        ;* DISCRIPTIVE NAME: READ DICTIONARY                                  *
                        ;*                                                                    *
                        ;* FUNCTION: ACCESS TANKANJI DICTIONARY                               *
                        ;*                                                                    *
                        ;* LINKAGE                                                            *
                        ;*   INPUT:                                                           *
                        ;*           1. LENGTH OF YOMI ( DS,SI ) + 0                          *
                        ;*           2. YOMI           ( DS,SI ) + 1                          *
                        ;*   OUTPUT:                                                          *
                        ;*           1. NUMBER OF CANDIDATES ( CX )                           *
                        ;*           2. CANDIDATDS           ( ES,DI )                        *
                        ;*           3. RETURN CODE          ( AX )                           *
                        ;*                                                                    *
                        ;*   RETURN CODES: (AX)                                               *
                        ;*                                                                    *
                        ;*     0 - SUCCESSFUL                                                 *
                        ;*     1 - YOMI NOT FOUND                                             *
                        ;*     4 - YOMI LENGTH EXCEPTION                                      *
                        ;*     8 - HIRAGANA CODE EXCEPTION                                    *
                        ;*    16 - DICTIONARY FORMAT EXCEPTION                                *
                        ;*                                                                    *
                        ;* EXTERNAL REFERENCES:                                               *
                        ;*   ROUTINES: NON                                                    *
                        ;*   TABLES:   TANKANJI DICTIONARY                                    *
                        ;*                                                                    *
                        ;* REGISTERS: AX - RETURN CODE                                        *
                        ;*            CX - NUMBER OF CANDIDATES                               *
                        ;*            ( ES,DI ) - FIRST CANDIDATE TOP ADDRESS                 *
                        ;*            ALL OTHERS UNCHANGED                                    *
                        ;*                                                                    *
                        ;* RESTRICTIONS:                                                      *
                        ;*     YOMI LENGTH INDICATION:                                        *
                        ;*           DICTIONARY CONTAINS YOMI LENGTH INFORMATIONS              *
                        ;*         BY FOLLOWING CONVENTION ,                                  *
                        ;*          1. 81H INDICATES THAT YOMI LENGTH IS 1                    *
                        ;*          2. 80H INDICATES THAT NOLENGTH 1 YOMI IS CONTAINED IN     *
                        ;*             THIS DICTIONARY. AFTER 80H, NEXT YOMI IS CONTAINED.    *
                        ;*          3. IF YOMI LENGTH IS GRATER THAN 2 OR EQUAL TO 2 THEN     *
                        ;*             LENGTH IS NOT CONTAINED, BUT YOMI DELIMITTER INDICATE  *
                        ;*             THAT LAST CHARACTER OF THAT YOMI.                      *
                        ;*                                                                    *
                        ;* CHANGE ACTIVITY: VERSION 00.00                                     *
                        ;*           KJ CODE COMPACTION REJECTED                              *
                        ;*                                                                    *
                        ;**************************************************************************
                        ;DIC    LABEL   BYTE            ; 1 BYTE MEMORY ON DICTIONARY
```

```
= 01E8
= 00C0
= 00BF
= 00F8
= 00FE
0E89

0E89  52
0E8A  53
0E8B  56
0E8C  8C D8
0E8E  8E C0
0E90  B9 0008
0E93  8D 3E 0118 R
0E97  F3/ A4

0E99  E8 0EBE R
0E9C  85 C0
0E9E  75 18
0EA0  E8 0ED7 R
0EA3  85 C0
0EA5  75 11
0EA7  C7 06 013E R 0000
0EAD  E8 0F11 R
0EB0  8B 0E 013E R
0EB4  85 C0
0EB6  74 02
0EB8  2B C9
0EBA
0EBA  5E
0EBB  5B
0EBC  5A
0EBD  C3
0EBE

0EBE
0EBE  2B C0
0EC0  A0 0118 R
0EC3  3D 0001
0EC6  7C 0B
0EC8  3D 0007
0ECB  7F 06
0ECD  8B C8
0ECF  2B C0
0ED1  EB 03
0ED3  B8 0004
0ED6  C3
0ED7

0ED7
0ED7  57
0ED8  56
0ED9  BE 0000
0EDC  BF FFFF

0EDF  47
0EE0  2B C0
0EE2  8A 85 0119 R
0EE6  3D 009F
0EE9  72 20
0EEB  3D 00FE
0EEE  77 1B
0EF0  24 7F
0EF2  88 85 010B R
0EF6  E0 E7

0EF8  2B C0
0EFA  A0 0118 R
0EFD  0C 80
0EFF  A2 010B R

0F02  80 8D 010B R 80
0F07  2B C0
0F09  EB 03
0F0B  B8 0008
0F0E  5E
0F0F  5F
0F10  C3
0F11

0F11
```

```
;DICW    LABEL   WORD            ; 2 BYTE MEMORY ON DICTIONARY
;DICD    LABEL   DWORD           ; 4 BYTE MEMORY ON DICTIONARY
TOP      EQU     7AH*4           ; DUMMY INTERRUPT ( PARAMETER )
C0H      EQU     11000000B
BFH      EQU     10111111B
F8H      EQU     11111000B
FEH      EQU     11111110B
KKKNDR   PROC    NEAR
;*****************************************************************
;*                                                               *
;*              ( M A I N   O F   K K K N D R )                  *
;*                                                               *
;*****************************************************************
;*-------------------------------( P R E S E T )-----------------*
         PUSH    DX
         PUSH    BX
         PUSH    SI
         MOV     AX,DS
         MOV     ES,AX
         MOV     CX,8
         LEA     DI,KKYOMIL
         REP     MOVSB
;*--------------( E N D   O F   P R E S E T )-------------------*
         CALL    LENCK           ; LIMIT CHECK OF YOMI LENGTH
         TEST    AX,AX           ;       AX --- RETURN CODE
         JNZ     ERREND          ;
         CALL    HRGCK           ; HIRAGANA CODE LIMIT CHECK
         TEST    AX,AX           ;       AX --- RETURN CODE
         JNZ     ERREND          ;
         MOV     KKOHOSUU,0000H
         CALL    ACCDICT         ; ACCESS DICTIONARY
         MOV     CX,KKOHOSUU     ;
         TEST    AX,AX           ;       AX --- RETURN CODE
         JZ      ENDMAIN
ERREND:  SUB     CX,CX           ; CANDIDATE NUMBER IS ZERO IF ERROR
ENDMAIN:
         POP     SI
         POP     BX
         POP     DX
         RET
KKKNDR   ENDP
;*****************************************************************
;* PROCEDURE NAME: LENCK                                         *
;* FUNCTION: LIMIT CHECK OF YOMI LENGTH                          *
;*                                                               *
;*****************************************************************
LENCK    PROC    NEAR
         SUB     AX,AX
         MOV     AL,KKYOMIL      ; ( AX ) -- GIVEN LENGTH OF YOMI
         CMP     AX,1            ; LOWER LIMIT CHECK
         JL      LENCKER
         CMP     AX,7            ; UPPER LIMIT CHECK
         JG      LENCKER
         MOV     CX,AX           ; ( CX ) -- VALID GIVEN LENGTH OF YOMI
         SUB     AX,AX
         JMP     SHORT LENCK09   ; YOMI LENGTH IS VALID
LENCKER: MOV     AX,KKECYMLN     ; YOMI LENGTH EXCEPTION
LENCK09: RET
LENCK    ENDP
;*****************************************************************
;* PROCEDURE NAME: HRGCK                                         *
;* FUNCTION: HIRAGANA CODE CHECK                                 *
;*           AND GENERATE DICTIONARY-LIKE YOMI ID                *
;*****************************************************************
HRGCK    PROC    NEAR
         PUSH    DI              ;
         PUSH    SI              ;
         MOV     SI,0            ;
         MOV     DI,-1           ;
;*****************************************************************
HRG03:   INC     DI              ; NEXT HIRAGANA CODE
         SUB     AX,AX           ;
         MOV     AL,KKYOMI[DI]   ;
         CMP     AX,009FH        ; LOWER LIMIT CHECK
         JB      HRGEXCP         ;
         CMP     AX,00FEH        ; UPPER LIMIT CHECK
         JA      HRGEXCP         ;
         AND     AL,7FH          ; RESET DELIMITTER
         MOV     KKGVNYM[DI],AL  ; COPY A HIRAGANA CODE TO WORK AREA
         LOOPNZ  HRG03
;*****************************************************************
         SUB     AX,AX           ;
         MOV     AL,KKYOMIL      ;
         OR      AL,80H          ; YOMI ID FLAG ON
         MOV     KKGVNYM,AL      ; YOMI ID IN DICTIONARY
                                 ;   FIRST HIRAGANA IS OVERIDED BY
                                 ;   LENGTH OF YOMI
         OR      KKGVNYM[DI],80H ; DELIMITTER  OF YOMI ON
         SUB     AX,AX           ;
         JMP     SHORT HRG09     ; HIRAGANA CODE IS VALID
HRGEXCP: MOV     AX,KKECYMHR     ; EXCEPTION
HRG09:   POP     SI              ;
         POP     DI              ;
         RET
HRGCK    ENDP
;*****************************************************************
;* PROCEDURE NAME: ACCDICT                                       *
;* FUNCTION: DICTIONARY ACCESS                                   *
;*                                                               *
;*****************************************************************
ACCDICT PROC    NEAR
;*****************************************************************
;*              FETCH TOP ADDRESS OF DICTIONARY                  *
;*                AND  LENGTH OF INDEX
```

```
0F11  E8 10D1 R                 ;%%%
```

```
0FAE                    NDX000:
0FAE  03 D9                     ADD     BX,CX           ; ( ES,BX ) --> POINTER IN INDEX
0FB0  26: 8B 07                 MOV     AX,ES:[BX]      ; FETCH POINTER FROM INDEX
0FB3  86 E0                     XCHG    AH,AL
0FB5  8B F8                     MOV     DI,AX           ; DI --- POINTER DATA OF DICTIONARY
0FB7  2B C0                     SUB     AX,AX
0FB9  03 3E 0112 R              ADD     DI,KKDICLN      ; DICTIONARY INDEX LENGTH
0FBD  03 3E 0140 R              ADD     DI,KKDICOFF     ; ( ES,DI ) --> YOMI IN DICTIONARY DATA
0FC1  5E                NDXNEXT: POP    SI
0FC2  5B                        POP     BX
0FC3  C3                        RET
0FC4                    NDXSRC  ENDP
                        ;*************************************************************************
                        ;* PROCEDURE NAME: YMSRCH                                                *
                        ;* FUNCTION: YOMI SEARCH IN DATA                                         *
                        ;*                                                                       *
                        ;*************************************************************************
                                                ;
0FC4                    YMSRCH  PROC    NEAR
0FC4  52                        PUSH    DX              ;
0FC5  56                        PUSH    SI              ;
0FC6  80 3E 0118 R 01           CMP     KKYOMIL,1       ; YOMI LENGTH COMPARISON
0FCB  75 0B                     JNE     YMSR05          ;
                        ;*************************************************************************
                        ;*                                                                       *
                        ;*************************************************************************
                                                GIVEN YOMI LENGTH = 1
0FCD  26: 80 3D 81              CMP     BYTE PTR ES:[DI],81H  ; YOMI ID COMPARE
0FD1  75 52                     JNE     YMSRER
0FD3  E8 1058 R                 CALL    SETCAND         ; GET FIRST CANDIDATE OFFSET AND
0FD6  EB 50                     JMP     SHORT YMSR99
                        ;*************************************************************************
                        ;*                                                                       *
                        ;*************************************************************************
                                                GIVEN YOMI LENGTH NOT 1
0FD8  26: 80 3D 80      YMSR05: CMP     BYTE PTR ES:[DI],80H
0FDC  75 03                     JNE     YMSRLOOP
0FDE  83 C7 01                  ADD     DI,1            ; POINT TO NEXT ID
0FE1  26: 80 3D 81      YMSRLOOP: CMP   BYTE PTR ES:[DI],81H  ; CF. YOMI LENGTH INDICATION ( L=1 )
0FE5  75 05                     JNE     YMSR07
0FE7  E8 103E R                 CALL    YMNEXT          ; SEARCH NEXT ENTRY ( YOMI )
0FEA  EB F5                     JMP     SHORT YMSRLOOP  ;
0FEC  26: 80 3D 80      YMSR07: CMP     BYTE PTR ES:[DI],80H  ; CF. YOMI LENGTH INDICATION
0FF0  75 05                     JNE     YMSR08          ;
0FF2  B8 0001                   MOV     AX,KKECNOTF     ; EXCEPTION ' YOMI NOT FOUND '
0FF5  EB 31                     JMP     SHORT YMSR99
0FF7  26: 80 3D 81      YMSR08: CMP     BYTE PTR ES:[DI],81H  ; CF. YOMI LENGTH INDICATION ( L=1 )
0FFB  74 28                     JE      YMSRER          ; EXCEPTION ' YOMI NOT FOUND '
0FFD  8B D7             YMSRLOP2: MOV   DX,DI           ; SAVE YOMI ID
0FFF  E8 102D R                 CALL    YOMIC           ; COMPARE GIVEN YOMI AND ON DICTIONARY
1002  76 13                     JBE     YMSR09          ;
                        ;*----------------------------( GIVEN YOMI > DICTIONARY YOMI )-----*
1004  8B FA                     MOV     DI,DX           ; RETRIEVE YOMI ID ADDRESS
1006  E8 103E R                 CALL    YMNEXT          ; SEARCH NEXT YOMI ENTRY
1009  26: 80 3D 81              CMP     BYTE PTR ES:[DI],81H  ; CF. YOMI LENGTH IN
100D  74 16                     JE      YMSRER          ; EXCEPTION ' YOMI NOT FOU
100F  26: 80 3D 80              CMP     BYTE PTR ES:[DI],80H  ; CF. YOMI LENGTH IN
1013  74 10                     JE      YMSRER          ; EXCEPTION ' YOMI NOT FOU
1015  EB E6                     JMP     SHORT YMSRLOP2  ; CONTINUE SEARCH YOMI AND COMPARE
                        ;*----------------------( END OF GIVEN YOMI > DICTIONARY YOMI )-----*
1017  8B FA             YMSR09: MOV     DI,DX           ; RETRIEVE YOMI ID
1019  E8 102D R                 CALL    YOMIC           ; COMPARE GIVEN YOMI AND ON DICTIONARY
101C  75 07                     JNE     YMSRER          ; GIVEN YOMI < DICTIONARY YOMI
101E  8B FA                     MOV     DI,DX           ; RETRIEVE YOMI ID
1020  E8 1058 R                 CALL    SETCAND         ; SET CANDIDATES TO OUTPUT BUFFER
1023  EB 03                     JMP     SHORT YMSR99
                        ;*-------------------------------------------------------------------*
1025  B8 0001          YMSRER: MOV     AX,KKECNOTF     ; EXCEPTION  ' YOMI NOT FOUND '
1028  8B FA             YMSR99: MOV     DI,DX           ; RETRIEVE YOMI ID ADDRESS
102A  5E                        POP     SI              ;
102B  5A                        POP     DX              ;
102C  C3                        RET
102D                    YMSRCH  ENDP
                        ;*************************************************************************
                        ;* PROCEDURE NAME: YOMIC                                                 *
                        ;* FUNCTION: YOMI COMPARISON ( GIVEN YOMI AND ON DICTIONARY )            *
                        ;*                                                                       *
                        ;*************************************************************************
102D                    YOMIC   PROC    NEAR
102D  2B C9                     SUB     CX,CX           ;
102F  2B F6                     SUB     SI,SI           ;
1031  8D 36 010B R              LEA     SI,KKGVNYM      ; KKGVNYM CONTAINS DICTIONARY LIKE YOMI
1035  46                       INC     SI              ; COMPARE FROM SECOND CHARACTER OF YOMI
1036  8A 0E 0118 R              MOV     CL,KKYOMIL      ; GIVEN YOMI LENGTH
103A  49                        DEC     CX              ; ####
103B  F3/ A6                    REPE    CMPSB
103D  C3                        RET
103E                    YOMIC   ENDP
                        ;*************************************************************************
                        ;* PROCEDURE NAME: YMNEXT                                                *
                        ;* FUNCTION: SEARCH NEXT YOMI IN DATA                                    *
                        ;*                                                                       *
                        ;*************************************************************************
103E                    YMNEXT  PROC    NEAR
103E  E8 10C1 R                 CALL    LENGYM          ; GET YOMI LENGTH ON DICTIONARY
1041  3D 0000                   CMP     AX,0
1044  75 01                     JNE     YMSKP
1046  40                       INC     AX
1047  03 F8             YMSKP:  ADD     DI,AX           ; ES+DI--> NEXT OF LAST CHARACTER OF
1049  26: F6 05 80      YMNXLOOP: TEST  BYTE PTR ES:[DI],80H  ; CHECK DELIMITTER
104D  75 05                     JNZ     YMNX04
                        ;*************************************************************************
                        ;*                                                                       *
                        ;*************************************************************************
                                                NON LAST CANDIDATE
104F  83 C7 02                  ADD     DI,2            ; POINT TO NEXT CANDIDATE
1052  EB F5                     JMP     SHORT YMNXLOOP
```

```
                                ;
                                ;*****************************************************************
                                ;*                          LAST CANDIDATE                       *
                                ;*****************************************************************
                                ;
1054  83 C7 02                  YMNX04: ADD     DI,2            ; POINT TO SECOND CHARACTER OF YOMI
1057  C3                              RET
1058                            YMNEXT  ENDP
                                ;
                                ;*****************************************************************
                                ;* PROCEDURE NAME: SETCAND                                       *
                                ;* FUNCTION: SET CANDIDATES TO OUTPUT BUFFER                     *
                                ;*                                                               *
                                ;*****************************************************************
1058                            SETCAND PROC    NEAR
1058  2B C0                             SUB     AX,AX           ;
105A  A0 0118 R                         MOV     AL,KKYOMIL      ; ( AX ) GIVEN YOMI LENGTH
105D  3D 0002                           CMP     AX,2
1060  7C 01                             JL      SETSKP
1062  48                              DEC     AX              ; CF. YOMI LENGTH IUNDICATE
1063  03 F8                   SETSKP: ADD     DI,AX           ; ( ES,DI ) POINT TOP OF CANDIDATES
1065  89 3E 0140 R                      MOV     KKDICOFF,DI     ; SAVE OFFSET OUTPUT OF THIS ROUTINE
1069  8C 06 0142 R                      MOV     KKDICSEG,ES     ; SAVE SEGMENT OUTPUT OF THIS ROUTINE
106D  2B F6                             SUB     SI,SI           ; TOP OF OUTPUT BUFFER
106F  26: F6 05 80              SETCLOOP: TEST  BYTE PTR ES:[DI],80H  ; IF LAST CANDIDATE FOR THAT YOMI
1073  75 05                             JNZ     SETC04          ;   THEN JUMP
                                ;*****************************************************************
                                ;*                          NON LAST CANDIDATE                   *
                                ;*****************************************************************
1075  E8 10C9 R                         CALL    SET2BKJ
1078  EB F5                             JMP     SHORT SETCLOOP
                                ;*****************************************************************
                                ;*                          LAST CANDIDATE                       *
                                ;*****************************************************************
107A                            SETC04:
107A  E8 10C9 R                         CALL    SET2BKJ
107D  2B C0                             SUB     AX,AX           ; NORMAL RETURN
107F  C3                                RET
1080                            SETCAND ENDP
                                ;*****************************************************************
                                ;* PROCEDURE NAME: YMLENC                                        *
                                ;* FUNCTION: COUNT LENGTH OF YOMI ON DICTIONARY                  *
                                ;*                                                               *
                                ;*****************************************************************
1080                            YMLENC  PROC    NEAR
1080  57                                PUSH    DI
1081  51                                PUSH    CX
1082  53                                PUSH    BX
1083  B8 0000                           MOV     AX,0
1086  26: 80 3D 81                      CMP     BYTE PTR ES:[DI],81H  ; CF. YOMI LENGTH INDICATION
108A  74 1E                             JE      YMLC1
108C  26: 80 3D 80                      CMP     BYTE PTR ES:[DI],80H  ; CF. YOMI LENGTH INDICATION
1090  74 20                             JE      YMLC2
1092  B9 0002                           MOV     CX,2
1095  26: F6 05 80              YMLREP: TEST    BYTE PTR ES:[DI],80H  ; IF LAST YOMI CHARACTER
1099  75 09                             JNZ     YMLREPE         ;   THEN JUMP
1098  83 F9 07                          CMP     CX,7            ; YOMI LENGTH LIMIT CHECK
109E  7F 1A                             JG      YMLERR          ; YOMI LENGTH ERROR
10A0  41                                INC     CX              ; COUNT YOMI LENGTH
10A1  47                                INC     DI              ; POINT TO NEXT CHARACTER OF YOMI
10A2  EB F1                             JMP     SHORT YMLREP
10A4  89 0E 0116 R              YMLREPE: MOV    KKLENYM,CX      ; SET YOMI LENGTH >= 2
10A8  EB 13                             JMP     SHORT YMLEND
10AA  C7 06 0116 R 0001         YMLC1:  MOV     KKLENYM,1       ; SET YOMI LENGTH IS  1
10B0  EB 0B                             JMP     SHORT YMLEND
10B2  C7 06 0116 R 0000         YMLC2:  MOV     KKLENYM,0       ; SET YOMI LENGTH IS 0
10B8  EB 03                             JMP     SHORT YMLEND
10BA  B8 0010                   YMLERR: MOV     AX,KKECDICT     ; EXCEPTION ' DICTIONARY FORMAT ERROR '
10BD                            YMLEND:
10BD  5B                                POP     BX
10BE  59                                POP     CX
10BF  5F                                POP     DI
10C0  C3                                RET
10C1                            YMLENC  ENDP
                                ;*****************************************************************
10C1                            LENGYM  PROC
10C1  E8 1080 R                         CALL    YMLENC          ;
10C4  A1 0116 R                         MOV     AX,KKLENYM      ;
10C7  48                                DEC     AX              ; AX = YOMI LENGTH ON DICTIONARY
10C8  C3                                RET
10C9                            LENGYM  ENDP
                                ;*****************************************************************
10C9                            SET2BKJ PROC
10C9  83 C7 02                          ADD     DI,2            ; POINT TO NEXT CANDIDATE
10CC  FF 06 013E R                      INC     KKOHOSUU        ; INCREMENT CANDIDATE NUMBER
10D0  C3                                RET
10D1                            SET2BKJ ENDP
                                ;*****************************************************************
10D1                            GETDICT PROC
10D1  53                                PUSH    BX
10D2  50                                PUSH    AX
10D3  1E                                PUSH    DS
10D4  2B DB                             SUB     BX,BX
10D6  8E DB                             MOV     DS,BX           ; DS CONTAINS ZERO
10D8  BB 01E8                           MOV     BX,TOP          ; TOP -- INT 7A * 4
10DB  C4 3F                             LES     DI,DS:[BX]      ; GET PARAMETER FOR DICTIONARY ( SEG & OFF )
10DD  26: 8B 05                         MOV     AX,ES:[DI]      ; DICTIONARY INDEX LENGTH --- AX
10E0  86 E0                             XCHG    AH,AL           ;
10E2  1F                                POP     DS              ; RETRIEVE DATA SEGMENT
10E3  A3 0112 R                         MOV     KKDICLN,AX      ; DICTIONARY INDEX LENGTH
10E6  89 3E 0140 R                      MOV     KKDICOFF,DI     ; DICTIONARY TOP OFFSET
10EA  58                                POP     AX
10EB  5B                                POP     BX
10EC  C3                                RET
```

```
10ED                        GETDICT ENDP
                            ;**************************************************************
                            ;*                                                            *
                            ;* PROGRAM NAME: KKDISP                                       *
                            ;*                                                            *
                            ;* DESCRIPTIVE NAME: DISPLAY YOMI AND BAGOU                   *
                            ;*                                                            *
                            ;* FUNCTION: THIS ROUTINE DISPLAYS DATA, AND SETS CURSOR      *
                            ;*           AT APPOINTED POSITION ON DISPLAY.                *
                            ;*                                                            *
                            ;* LINKAGE: CALL                                              *
                            ;*                                                            *
                            ;*   INPUT: KANA-KAN COMMON TABLES                            *
                            ;*                                                            *
                            ;*   OUTPUT: KANA-KAN COMMON TABLES                           *
                            ;*                                                            *
                            ;*   RETURN CODES: (AX)                                       *
                            ;*                                                            *
                            ;*      0 - SUCCESSFUL                                        *
                            ;*      1 - INVALID OPERATION                                 *
                            ;*                                                            *
                            ;*  EXTERNAL REFERENCES:                                      *
                            ;*                                                            *
                            ;*    ROUTINES: NONE                                          *
                            ;*                                                            *
                            ;*    TABLES: KANA-KAN COMMON TABLES                          *
                            ;*                                                            *
                            ;*  REGISTERS: AX - RETURN CODE                               *
                            ;*             ALL OTHERS UNCHANGED                           *
                            ;*                                                            *
                            ;*  CHANGE ACTIVITY: VERSION 00.00                            *
                            ;*                                                            *
                            ;*                                                            *
                            ;**************************************************************
                            ;
                            ;******************** KKDISP **********************************
                            ;*                                                            *
                            ;*      ENTRY OF KKDISP                                       *
                            ;*                                                            *
                            ;**************************************************************
10ED                        KKDISP  PROC
10ED  56                            PUSH    SI              ;SAVE REGISTER
10EE  52                            PUSH    DX
10EF  53                            PUSH    BX
                            ;*----------------------------------------------------------*
                            ;*      DISPLAY DATA                                        *
                            ;*----------------------------------------------------------*
10F0                        DIS010:
10F0  8B 16 0107 R                  MOV     DX,DSTRTP       ;LOAD START POSITION
10F4  83 FA FF                      CMP     DX,0FFFFH       ;ONLY CURSOR DISPLAY ?
10F7  74 1D                         JE      DIS020          ; YES. GOTO
10F9  8B 36 00F1 R                  MOV     SI,KKINP        ;LOAD INPUT BUFFER POINTER
10FD  8A 36 00FC R                  MOV     DH,KKOILP
1101                        DIS012:
1101  8A 84 0009 R                  MOV     AL,KKINBUF[SI]  ;LOAD DISPLAY DATA
1105  E8 1557 R                     CALL    WDISP           ;DISPLAY DATA
1108  A1 0109 R                     MOV     AX,DENDP        ;LOAD END POSITION
110B  3A C2                         CMP     AL,DL           ;FINISH DISPLAY
110D  74 07                         JE      DIS020          ;YES.GOTO
110F  FE C2                         INC     DL              ;CURSOR POSITION 1 UP
1111  83 C6 03                      ADD     SI,D03          ;RENEW INPUT BUFFER POINTER(+3)
1114  EB EB                         JMP     SHORT DIS012
                            ;*----------------------------------------------------------*
                            ;*      SET CURSOR                                          *
                            ;*----------------------------------------------------------*
1116                        DIS020:
1116  8B 16 0105 R                  MOV     DX,DCRSRP
111A  89 16 00F3 R                  MOV     KKCUSR,DX       ;SAVE NEW KANA-KAN CURSOR POSITION
111E  8A 36 00FC R                  MOV     DH,KKOILP       ;LOAD OIL POSITION
1122  B4 82                         MOV     AH,VSETACP
1124  CD 10                         INT     VIDEO           ;SET CURSOR POSITION
1126  52                            PUSH    DX              ; PUSH DX
1127  B4 83                         MOV     AH,VRDACP       ;
1129  CD 10                         INT     VIDEO           ; READ CURSOR TYPE
112B  80 E5 DF                      AND     CH,NOT CURSOR_DISABLE; CURSOR DISABLE BIT OFF
112E  B4 81                         MOV     AH,VSETACT      ;
1130  CD 10                         INT     VIDEO           ;SET CURSOR TYPE
1132  5A                            POP     DX              ; RESTORE DX
                            ;*----------------------------------------------------------*
                            ;*      RETURN                                              *
                            ;*----------------------------------------------------------*
1133  33 C0                         XOR     AX,AX           ;SET NOMAL RETURN CODE
1135  5B                            POP     BX              ;RESTORE REGISTER
1136  5A                            POP     DX
1137  5E                            POP     SI
1138  C3                            RET                     ;RETURN TO CALLER
1139                        KKDISP  ENDP
                            ;**************************************************************
                            ;*                                                            *
                            ;* PROGRAM NAME: KKWOIL                                       *
                            ;*                                                            *
                            ;* DESCRIPTIVE NAME: DISPLAY OPERATER INFORMATION ILNE        *
                            ;*                                                            *
                            ;* FUNCTION: 1. WRITE OPERATOR INFORMATION LINE               *
                            ;*           2. CLEAR OPERATOR INFORMATION LINE               *
                            ;*           3. INITIALIZE INDICATOR                          *
                            ;*                                                            *
                            ;* LINKAGE: CALL                                              *
                            ;*                                                            *
                            ;*   INPUT: BL = 0   - WRITE OIL                              *
                            ;*               1   - CLEAR OIL                              *
                            ;*               80H - INITIALIZE INDICATOR                   *
                            ;*          AH = CHARACTER ( ONLY BL=0 )                      *
                            ;*          AL = ATTRIBUTE ( ONLY BL=0 )                      *
```

```
                        ;*          DI = POINTER IN OIL ( ONLY BL=0 )                  *
                        ;*                                                             *
                        ;*   OUTPUT: DI = DI + 2 ( ONLY WRITE OIL )                    *
                        ;*                                                             *
                        ;*   RETURN CODES: (AX)                                        *
                        ;*                                                             *
                        ;*      0 - SUCCESSFUL                                         *
                        ;*      1 - INVALID OPERATION                                  *
                        ;*                                                             *
                        ;*   EXTERNAL REFERENCES:                                      *
                        ;*                                                             *
                        ;*    ROUTINES: NONE                                           *
                        ;*                                                             *
                        ;*    TABLES: KANA-KAN COMMON TABLES                           *
                        ;*                                                             *
                        ;*   REGISTERS: AX - RETURN CODE                               *
                        ;*              ALL OTHERS UNCHANGED                           *
                        ;*                                                             *
                        ;*   CHANGE ACTIVITY: VERSION 00.00                            *
                        ;*                                                             *
                        ;*                                                             *
                        ;*************************************************************
                        ;
                        ;********************* KKWOIL ********************************
                        ;*                                                             *
                        ;*      ENTRY OF KKWOIL                                        *
                        ;*                                                             *
                        ;*************************************************************
1139                    KKWOIL  PROC
1139  52                        PUSH    DX              ;SAVE REGISTER
113A  51                        PUSH    CX
113B  53                        PUSH    BX
113C  80 FB 00                  CMP     BL,D00          ;REQUIREMENT IS TO WRITE OIL ?
113F  74 0C                     JE      WOI010          ; YES. GOTO
1141  80 FB 01                  CMP     BL,D01          ;REQUIREMENT IS TO CLEAR OIL ?
1144  74 0C                     JE      WOI020          ; YES. GOTO
1146  80 FB 80                  CMP     BL,D80          ;REQUIREMENT IS TO INITIALIZE INDICATOR ?
1149  74 19                     JE      WOI040          ; YES. GOTO
114B  EB 2D                     JMP     SHORT WOIRTN
                        ;*-------------------------------------------------------------*
                        ;*      WRITE OIL                                              *
                        ;*-------------------------------------------------------------*
114D                    WOI010:
114D  E8 1180 R                 CALL    WROIL           ;WRITE OIL
1150  EB 28                     JMP     SHORT WOIRTN
                        ;*-------------------------------------------------------------*
                        ;*      CLEAR OIL                                              *
                        ;*-------------------------------------------------------------*
1152                    WOI020:
1152  E8 1538 R                 CALL    CLOIL           ;CLEAR OIL
1155  32 C0                     XOR     AL,AL
1157  B4 07                     MOV     AH,D07
1159  CD 16                     INT     16H             ;INDICATOR ON
115B  8B 16 00FF R              MOV     DX,APKBDST      ;LOAD KBD STATUS FOR APPLICATION
115F  E8 11EB R                 CALL    WKBDST          ;RESET KBD STATUS FOR APPLICATION
1162  EB 16                     JMP     SHORT WOIRTN
                        ;*-------------------------------------------------------------*
                        ;*      INITIALIZE INDICATOR                                   *
                        ;*-------------------------------------------------------------*
1164                    WOI040:
1164  B4 02                     MOV     AH,D02
1166  CD 16                     INT     16H             ;READ KBD STATUS
1168  A3 00FF R                 MOV     APKBDST,AX      ;SAVE KBD STATUS FOR APPLICATION
116B  8B D0                     MOV     DX,AX           ;LOAD KBD STATUS
116D  80 E6 F9                  AND     DH,0F9H
1170  80 CE 04                  OR      DH,HSHIFT       ;CHANGE INDCATOR INTO HIRAGANA
1173  89 16 0001 R              MOV     KBDSTAT,DX      ;SET KBD STATUS FOR APPLICATION
1177  E8 11EB R                 CALL    WKBDST          ;WRITE KBD STATUS
                        ;*-------------------------------------------------------------*
                        ;*      RETURN                                                 *
                        ;*-------------------------------------------------------------*
117A                    WOIRTN:
117A  33 C0                     XOR     AX,AX           ;SET NOMAL RETURN CODE
117C  5B                        POP     BX              ;RESTORE REGISTER
117D  59                        POP     CX
117E  5A                        POP     DX
117F  C3                        RET                     ;RETURN TO CALLER
1180                    KKWOIL  ENDP
                        ;***** WROIL ***************************************************
                        ;*                                                             *
                        ;*   WRITE OPERATOR INFOMATION LINE                            *
                        ;*                                                             *
                        ;*************************************************************
1180                    WROIL   PROC
1180  50                        PUSH    AX
1181  E8 12EC R                 CALL    ERASE_CURSOR    ; ERASE CURSOR
1184  80 3E 00FA R 03            CMP     TVMODE,TV4011   ;TV MODE IS 40*11 ?
1189  75 50                     JNE     WRO010          ; NO. GOTO
118B  83 FF 00                  CMP     DI,D00          ;FIRST CALL ?
118E  75 4B                     JNE     WRO010          ; NO. GOTO
1190  80 3E 00F9 R 04            CMP     KKMODE,SELECTM  ;KANA-KAN MODE IS SELECT ?
1195  74 2D                     JE      WRO000          ; YES. GOTO
1197  80 3E 0194 R 04            CMP     OLDMODE,SELECTM ;OLD KANA-KAN MODE IS SELECT ?
119C  75 1A                     JNE     WRO0001         ; NO. GOTO
119E  32 C0                     XOR     AL,AL
11A0  B4 07                     MOV     AH,D07
11A2  CD 16                     INT     16H             ;INDICATOR ON
11A4  C6 06 0101 R 00           MOV     DOILP,CLIND11P  ;SET DATA POSITION OF INDICATOR
11A9  C6 06 0102 R 0C           MOV     DOILN,CLIND11N  ;SET NUMBER OF INDICATOR DATA
11AE  E8 1538 R                 CALL    CLOIL           ;CLEAR OIL
11B1  8B 16 0001 R              MOV     DX,KBDSTAT      ;LOAD KANA-KAN KBD STATUS
11B5  E8 11EB R                 CALL    WKBDST          ;DISPLAY INDICATOR
11B8                    WRO0001:
11B8  C6 06 0101 R 0C           MOV     DOILP,DOILP11K  ;SET DATA POSITION OF OIL
```

```
11BD  C6 06 0102 R 1C           MOV     DOILN,DOILN11K  ;SET NUMBER OF OIL DATA
11C2  EB 17                     JMP     SHORT WR0010
11C4
11C4  80 3E 0194 R 04   WR0000: CMP     OLDMODE,SELECTM ;OLD KANA-KAN MODE IS SELECT ?
11C9  74 06                     JE      WR0002          ; YES. GOTO
11CB  B0 02                     MOV     AL,D02
11CD  B4 07                     MOV     AH,D07
11CF  CD 16                     INT     16H             ;INDICATOR OFF
11D1                    WR0002:
11D1  C6 06 0101 R 00           MOV     DOILP,DOILP11S  ;SET DATA POSITION OF OIL
11D6  C6 06 0102 R 28           MOV     DOILN,DOILN11S  ;SET NUMBERS OF OIL DATA
11DB                    WR0010:
11DB  8A 16 0101 R              MOV     DL,DOILP        ;LOAD DATA POSITION OF OIL
11DF  8A 36 00FC R              MOV     DH,KKOILP       ;LOAD OIL POSITION
11E3  03 D7                     ADD     DX,DI           ;ADD OIL POINTER
11E5  58                        POP     AX
11E6  E8 1557 R                 CALL    WDISP           ;DISPLAY DATA
11E9  47                        INC     DI              ;OIL POINTER 1 UP
11EA  C3                        RET                     ;RETURN TO CALLER
11EB                    WROIL   ENDP
                        ;**** WKBDST ********************************************************
                        ;*                                                                  *
                        ;*   WRITE KEYBOARD STATUS                                          *
                        ;*                                                                  *
                        ;*                DX : KBD STATUS                                   *
                        ;*                                                                  *
                        ;********************************************************************
                        ;
11EB                    WKBDST  PROC
11EB  32 C0                     XOR     AL,AL
11ED  F6 C6 01                  TEST    DH,ZMODE        ;ZENKAKU MODE ?
11F0  74 02                     JZ      WKB010
11F2  0C 01                     OR      AL,ZMODE        ;ZENKAKU MODE ON
11F4                    WKB010:
11F4  80 E6 06                  AND     DH,KBDMSK
11F7  D0 C6                     ROL     DH,1
11F9  0A C6                     OR      AL,DH           ;SET SHIFT
11FB  F6 C2 40                  TEST    DL,CAPSST       ;CAPS LOCK ?
11FE  74 02                     JZ      WKB020
1200  0C 10                     OR      AL,CAPSON       ;CAPS LOCK ON
1202                    WKB020:
1202  0C C0                     OR      AL,KKNOCHG      ;KANA-KAN REQUIREMENT NOT CHANGED
1204  B4 05                     MOV     AH,D05
1206  CD 16                     INT     16H             ;WRITE KBD STATUS
1208  C3                        RET                     ;RETURN TO CALLER
1209                    WKBDST  ENDP
                        ;********************************************************************
                        ;*    S U B R O U T I N E                                           *
                        ;*                                                                  *
                        ;*      NAME :  KKGRP                                               *
                        ;*                                                                  *
                        ;*                                                                  *
                        ;*                                                                  *
                        ;*                                                                  *
                        ;*                                                                  *
                        ;*                                                                  *
                        ;********************************************************************
1209                    KKGRP   PROC    NEAR
1209  A0 0000 R                 MOV     AL,DSTAT        ; GET INPUT CHARACTER STATUS
120C  24 82                     AND     AL,DAKUF+ZEN2F
120E  3C 82                     CMP     AL,DAKUF+ZEN2F  ; DAKU/HANDAKU TEN & ZENKAKU CHARACTER
1210  74 33                     JE      KGR300
1212  A0 00F0 R                 MOV     AL,KKWINST      ; GET INSERT MODE FLAG
1215  A8 01                     TEST    AL,00000001B    ; INSERT MODE CHECK
1217  74 09                     JZ      KGR100
                        ;******  INSERT  MODE  ******
1219  8D 06 0003 R              LEA     AX,KKKYDCD
121D  E8 0CF2 R                 CALL    KKINST          ; CHARACTER INSERT
1220  EB 52                     JMP     SHORT KGR900
                        ;******  NOT INSERT MODE  ******
1222                    KGR100:
1222  8B 1E 00E6 R              MOV     BX,KKWCCA
1226  3B 1E 00EA R              CMP     BX,KKWEOP
122A  74 03                     JE      KGR110
                        ;       MOV     AL,KKINBUF[BX][1]
                        ;       AND     AL,01H
                        ;       INC     AL
                        ;       MUL     MUL03
                        ;       MOV     BX,AX
                        ;
                        ;       MOV     AX,KKWCCA
                        ;       SUB     KKWEOP,BX       ; ADJUST END OF POINTER
122C  E8 0E5A R                 CALL    KKDELT
122F                    KGR110:
122F  8D 06 0003 R              LEA     AX,KKINKEY
1233  8A 16 00F0 R              MOV     DL,KKWINST      ; SAVE INSERT MODE FLAG
1237  80 0E 00F0 R 80           OR      KKWINST,80H     ; SET DUMMY INSERT FLAG
123C  E8 0CF2 R                 CALL    KKINST
123F  88 16 00F0 R              MOV     KKWINST,DL      ; RESTORE INSERT MODE FLAG
1243  EB 2F                     JMP     SHORT KGR900
                        ;******  DAKU/HANDAKU TEN MODE  ******
1245                    KGR300:
1245  8B 1E 00E6 R              MOV     BX,KKWCCA
1249  83 EB 06                  SUB     BX,6
124C  89 1E 00F1 R              MOV     KKINP,BX        ; SET DISPLAY POINTER
1250  B9 0006                  MOV     CX,6
1253  8D 36 0003 R              LEA     SI,KKINKEY
1257                    KGR310:
1257  8A 04                     MOV     AL,DS:[SI]                  ; GET DAK/HANDAKU TEN CODE
1259  88 87 0009 R              MOV     BYTE PTR KKINBUF[BX],AL ; SET CHARACTER CODE
125D  43                        INC     BX
125E  46                        INC     SI
125F  49                        DEC     CX
```

```
1260  75 F5                    JNZ     KGR310
1262  A1 0105 R                MOV     AX,DCRSRP
1265  2C 02                    SUB     AL,2
1267  A3 0107 R                MOV     DSTRTP,AX        ; SET DISPLAY START POSITION
126A  04 01                    ADD     AL,1
126C  A3 0109 R                MOV     DENDP,AX         ; SET DISPLAY END POSITION
126F  E8 10ED R                CALL    KKDISP
1272  2B C0                    SUB     AX,AX
1274                    KGR900:
1274  C3                       RET
1275                    KKGRP   ENDP
                        ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        ;+          PRIVATE CONSTANT DATA ( FOR KOUTINIT )                    +
                        ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1275                    BASE_FOMAT:
1275  127B R                   DW      TV8025_FMT
1277  127B R                   DW      TV8011_FMT
1279  1287 R                   DW      TV4011_FMT
127B                    TV8025_FMT:
127B                    TV8011_FMT:
127B  03 3D                    DB      03,3DH           ;" = "
127D  01 3E                    DB      01,3EH           ;" > "
127F  01 20                    DB      01,20H           ;"   "
1281  96                       DB      11*2+80H         ; SPECIAL CODE
1282  01 2A                    DB      01,2AH           ;" * "
1284  28 20                    DB      40,20H           ;"   "
1286  FF                       DB      0FFH
1287                    TV4011_FMT:
1287  03 3D                    DB      03,3DH           ;" = "
1289  01 3E                    DB      01,3EH           ;" > "
128B  01 20                    DB      01,20H           ;"   "
128D  96                       DB      11*2+80H         ; SPECIAL CODE
128E  01 2A                    DB      01,2AH           ;" * "
1290  FF                       DB      0FFH
                        ;*****************************************************************
                        ;*    S U B R O U T I N E                                        *
                        ;*                                                               *
                        ;*      NAME : KOUTINIT                                          *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*****************************************************************
1291                    KOUTINIT        PROC    NEAR
1291  06                       PUSH    ES               ; SAVE REGISTERS
1292  57                       PUSH    DI
1293  56                       PUSH    SI
1294  52                       PUSH    DX
1295  53                       PUSH    BX
1296  55                       PUSH    BP
1297  A0 00FA R                MOV     AL,TVMODE        ; GET VIDEO MODE
129A  98                       CBW
129B  8B E8                    MOV     BP,AX
129D  4D                       DEC     BP
129E  D1 E5                    SHL     BP,1             ; BP X 2
12A0  2E: 8B AE 1275 R         MOV     BP,WORD PTR BASE_FOMAT[BP]
12A5  1E                       PUSH    DS
12A6  07                       POP     ES
12A7  B3 00                    MOV     BL,0
12A9  2B FF                    SUB     DI,DI
12AB  2B F6                    SUB     SI,SI
12AD  32 ED                    XOR     CH,CH
12AF                    OINT010:
12AF  2E: 8A 0A                MOV     CL,CS:[BP][SI]
12B2  80 F9 FF                 CMP     CL,0FFH
12B5  74 2E                    JE      OINT060
12B7  F6 C1 80                 TEST    CL,10000000B     ; LSB = 1 : ZENKAKU
12BA  74 16                    JZ      OINT040          ;       0 : HANKAKU
                        ; ***************************************************
                        ; *********        DISPLAY INPUT BUFFER    *************
                        ; ***************************************************
12BC                    OINT020:
12BC  80 E1 7F                 AND     CL,7FH
12BF  74 EE                    JZ      OINT010
12C1  56                       PUSH    SI
12C2  8D 36 0009 R             LEA     SI,KKINBUF       ; GET OFFSET OF INPUT BUFFER
12C6  FC                       CLD
12C7                    OINT030:
12C7  AD                       LODSW
12C8  E8 1139 R                CALL    KKWOIL           ; DISPLAY INPUT BUFFER
12CB  46                       INC     SI
12CC  E2 F9                    LOOP    OINT030
12CE  5E                       POP     SI
12CF  46                       INC     SI
12D0  EB DD                    JMP     SHORT OINT010
                        ; ***************************************************
                        ; *********        DISPLAY O.I.L FORMAT    *************
                        ; ***************************************************
12D2                    OINT040:
12D2  80 E1 7F                 AND     CL,7FH
12D5  74 D8                    JZ      OINT010
12D7  46                       INC     SI
12D8                    OINT050:
12D8  2E: 8A 02                MOV     AL,CS:[BP][SI]
12DB  B4 00                    MOV     AH,HANATTR       ; GET TO AX HANKAKU CODE
12DD  E8 1139 R                CALL    KKWOIL
12E0  E2 F6                    LOOP    OINT050
12E2  46                       INC     SI
12E3  EB CA                    JMP     SHORT OINT010
12E5                    OINT060:
12E5  5D                       POP     BP
```

```
12E6  5B                        POP     BX
12E7  5A                        POP     DX
12E8  5E                        POP     SI
12E9  5F                        POP     DI
12EA  07                        POP     ES
12EB  C3                        RET
12CC                    KOUTINIT ENDP

                        ;--------------------------------------------------------------
                        ;
                        ;       ERASE_CURSOR    ERASE CURSOR
                        ;
                        ;       THIS ROUTINE ERASES CURSOR
                        ;
                        ;
                        ;       INPUT           NONE
                        ;       OUTPUT          NONE
                        ;       VOLATILE        AX
                        ;
                        ;--------------------------------------------------------------

12EC                    ERASE_CURSOR    PROC    NEAR

12EC  51                        PUSH    CX              ; SAVE CX
12ED  52                        PUSH    DX              ; SAVE DX

12EE  B4 83                     MOV     AH,VRDACP       ; READ ALTERNATE CURSOR POSITION
12F0  CD 10                     INT     VIDEO           ;

12F2  5A                        POP     DX              ; RESTORE DX

12F3  80 CD 20                  OR      CH,CURSOR_DISABLE; SET CURSOR DISABLE BIT ON
12F6  B4 81                     MOV     AH,VSETACT      ;
12F8  CD 10                     INT     VIDEO           ; SET ALTERNATE CURSOR TYPE

12FA  59                        POP     CX              ; RESTORE CX
12FB  C3                        RET

12FC                    ERASE_CURSOR    ENDP

                        ;*************************************************************
                        ;*    S U B R O U T I N E                                    *
                        ;*                                                           *
                        ;*       NAME : KINBFCLR                                     *
                        ;*                                                           *
                        ;*                                                           *
                        ;*                                                           *
                        ;*                                                           *
                        ;*                                                           *
                        ;*                                                           *
                        ;*************************************************************
12FC                    KINBFCLR        PROC    NEAR
12FC  06                        PUSH    ES
12FD  57                        PUSH    DI
12FE  51                        PUSH    CX
12FF  B8 0000                   MOV     AX,0
1302  A2 00F0 R                 MOV     KKHINST,AL
1305  A3 00E6 R                 MOV     KKHCCA,AX
1308  A3 00EA R                 MOV     KKHEOP,AX
130B  FC                        CLD
130C  1E                        PUSH    DS
130D  07                        POP     ES
130E  BF 0009 R                 MOV     DI,OFFSET KKINBUF
1311  8B 0E 00E8 R              MOV     CX,KKEOPMAX
1315                    IBCL020:                        ; INPUT BUFFER SPACE CLEAR
1315  A1 00F5 R                 MOV     AX,KKCHRSP
1318  AB                        STOSW                   ; SET SPECIAL SPACE CODE
1319  32 C0                     XOR     AL,AL
131B  AA                        STOSB                   ; CLEAR SCAN CODE AREA
131C  83 E9 03                  SUB     CX,3
131F  75 F4                     JNZ     IBCL020
1321  B4 00                     MOV     AH,HANATTR
1323  B0 2A                     MOV     AL,ASTER
1325  AB                        STOSW                   ; SET ' * ' CHARACTER CODE
1326  32 C0                     XOR     AL,AL
1328  AA                        STOSB
1329  59                        POP     CX
132A  5F                        POP     DI
132B  07                        POP     ES
132C  C3                        RET
132D                    KINBFCLR ENDP
                        ;*************************************************************
                        ;*    S U B R O U T I N E                                    *
                        ;*                                                           *
                        ;*       NAME : KCALCCH                                      *
                        ;*                                                           *
                        ;*                                                           *
                        ;*                                                           *
                        ;*                                                           *
                        ;*                                                           *
                        ;*                                                           *
                        ;*************************************************************
132D
132D  A1 00F3 R                 KCALCCH PROC    NEAR
1330  25 00FF                   MOV     AX,KKCUSR       ; GET CURSOR POSIYION
1333  2B 06 00EE R              AND     AX,00FFH        ; GET X-POSITION
1337  2E: F6 26 065C R          SUB     AX,BASE
133C  A3 00E6 R                 MUL     MUL03
133F  C3                        MOV     KKHCCA,AX       ; CALCULATE KKHCCA
                                RET
```

```
1340                    KCALCCW ENDP
                        ;*****************************************************************
                        ;*    S U B R O U T I N E                                        *
                        ;*                                                               *
                        ;*      NAME : KCALELM                                           *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*****************************************************************
1340  03                ELMIMVAL  DB    3
1341                    KCALELM PROC    NEAR
1341  51                        PUSH    CX
1342  22 06 00E1 R              AND     AL,ZENATTR1     ; GET ZENKAKU/HANKAKU ATTRIBUTE
1346  8A 0E 00E3 R              MOV     CL,ROATCNT      ; GET DATA ROTATE COUNT
134A  D2 C0                     ROL     AL,CL
134C  04 01                     ADD     AL,1            ; CALCULATE ELEMENT OF INPUT BUFFER
134E  2E: F6 26 1340 R          MUL     ELMIMVAL        ;    ZENKAKU ... 6
1353  59                        POP     CX              ;    HANKAKU ... 3
1354  C3                        RET
1355                    KCALELM ENDP
                        ;*****************************************************************
                        ;*    S U B R O U T I N E                                        *
                        ;*                                                               *
                        ;*      NAME : KCSRDSP                                           *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*                                                               *
                        ;*****************************************************************
1355                    KCSRDSP PROC    NEAR
1355  A1 00EA R                 MOV     AX,KKWEOP
1358  2E: F6 36 1340 R          DIV     ELMIMVAL
135D  2A E4                     SUB     AH,AH
135F  03 06 00EE R              ADD     AX,BASE         ; CALUCULATE CURSOR POSITION ( COLUMN )
1363  A3 0105 R                 MOV     DCRSRP,AX       ; SET CURSOR POSITION
1366  C7 06 0107 R FFFF         MOV     DSTRTP,0FFFFH
136C  E8 10ED R                 CALL    KKDISP          ; DISPLAY CURSOR
136F  C3                        RET
1370                    KCSRDSP ENDP
                        ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        ;+      PRIVATE CONSTANT  < KKOHOEDT >                            +
                        ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1370                    KKBFCAA:
1370  001C 001C                 DW      0028*1,0028         ; TV8025
1374  001C 001C                 DW      0028*1,0028         ; TV8011
1378  0000 001C                 DW      0000*1,0028         ; TV4011
137C  0000 0024                 DW      0000*1,0036         ; TV8025
1380                    DECAJST:
1380  FF FF                     DB      0FFH,0FFH         ; DUMMY
1382  12 05                     DB      12H,05H
1384  56 02                     DB      56H,02H
1386  28 01                     DB      28H,01H
1388  64 00                     DB      64H,00H
138A  32 00                     DB      32H,00H
138C  16 00                     DB      16H,00H
138E  08 00                     DB      08H,00H
1390  04 00                     DB      04H,00H
1392  02 00                     DB      02H,00H
1394  01 00                     DB      01H,00H
                        ;*****************************************************************:*
                        ;*    SUBROUTINE  :  KKOHOEDT                                    *
                        ;*                                                               *
                        ;*       EDIT DISPLAY BUFFER FOR KANJI SELECT                    *
                        ;*                                                               *
                        ;*         PARAM ...  AX  =  0 : CALCULATE KKOHODSP              *
                        ;*                        =  1 : SET KKOHODSP FROM KKOHOSUU      *
                        ;*****************************************************************
1396                    KKOHOEDT        PROC    NEAR
1396  85 C0                     TEST    AX,AX           ;
1398  74 08                     JZ      KHED010
139A  A1 013E R                 MOV     AX,KKOHOSUU     ;
139D  A3 0148 R                 MOV     KKOHODSP,AX     ; SET DISPLAY POINTER
13A0  EB 15                     JMP     SHORT KHED020
13A2                    KHED010:
13A2  A0 00FD R                 MOV     AL,KHBLK
13A5  2E: F6 26 0893 R          MUL     DATA07
13AA  A3 0148 R                 MOV     KKOHODSP,AX     ; CALCULATE DISPLAY POINTER
13AD  8B 1E 013E R              MOV     BX,KKOHOSUU
13B1  2B D8                     SUB     BX,AX           ; CALCULATE KKOHOZAN
13B3  89 1E 0144 R              MOV     KKOHOZAN,BX
                        ;*****************        EDIT DISPLAY BUFFER        ****************
13B7                    KHED020:
13B7  33 DB                     XOR     BX,BX
13B9  A0 0190 R                 MOV     AL,KKCHRMD
13BC  3C 01                     CMP     AL,MDZENNUM
13BE  74 04                     JE      KHED021
13C0  3C 04                     CMP     AL,MDHANNUM     ; NUMERIC YOMI ?
13C2  75 06                     JNE     KHED022         ; NO. COTO
13C4                    KHED021:
13C4  FF 0E 0144 R              DEC     KKOHOZAN        ; KOUHO ZAN 1 DOWN
13C8  B7 01                     MOV     BH,NUMYOMFO     ; SET NUMERIC YOMI FLAG
13CA                    KHED022:
13CA  A0 00FA R                 MOV     AL,TVMODE       ; TV MODE CHECK
13CD  98                        CBW
13CE  8B E8                     MOV     BP,AX
13D0  4D                        DEC     BP
13D1  83 3E 018E R 1A           CMP     KKHQLEN,26
13D6  7E 03                     JLE     KHED030
```

```
13D8  BD 0003                    MOV     BP,3
13DB                        KHED030:
13DB  D1 E5                      SHL     BP,1
13DD  D1 E5                      SHL     BP,1                    ; BP X 4
13DF  2E: 8B BE 1370 R           MOV     DI,DWORD PTR KKBFCAA[BP]    ;GET EDITION BUFFER POINT
13E4  2E: 8B 8E 1372 R           MOV     CX,DWORD PTR KKBFCAA[BP][2] ;GET LENG. OF CANDIDATA AREA
13E9  1E                         PUSH    DS
13EA  07                         POP     ES                      ; DS <--- ES
13EB  FC                         CLD
13EC  8A 26 00E1 R               MOV     AH,ZENATTR1             ; SET FIRST CODE
13F0  B0 81                      MOV     AL,81H                  ; SET FIRST CODE
13F2  E8 1139 R                  CALL    KKWOIL
13F5  8A 26 00E2 R               MOV     AH,ZENATTR2             ; SET SECOND CODE
13F9  B0 79                      MOV     AL,79H                  ; SET SECOND CODE
13FB  E8 1139 R                  CALL    KKWOIL
13FE  C7 06 0146 R 0000          MOV     KKOHOCNT,0
1404  B6 00                      MOV     DH,HANATTR              ; SELECT NUMBER
1406  B2 31                      MOV     DL,31H                  ; SELECT NUMBER
1408  8B 2E 0148 R               MOV     BP,KKOHODSP
140C  D1 E5                      SHL     BP,1
140E                        KHED040:
140E  83 3E 0144 R 01            CMP     KKOHOZAN,1
1413  74 4E                      JE      KHED050
1415                        KHED045:
1415  A1 00E4 R                  MOV     AX,SPACE
1418  E8 1139 R                  CALL    KKWOIL                  ; SET HANKAKU SPACE
141B  49                         DEC     CX
141C  8B C2                      MOV     AX,DX
141E  E8 1139 R                  CALL    KKWOIL                  ; SET SELECT NUMBER
1421  49                         DEC     CX
1422  52                         PUSH    DX
1423  1E                         PUSH    DS
1424  56                         PUSH    SI
1425  C5 36 0140 R               LDS     SI,DWORD PTR KKDICADR ; GET OFFSET & SEGMENT ADDR.
1429  3E: 8B 12                  MOV     DX,DS:[SI][BP]  ; GET KOUHO KANJI CODE
142C  86 D6                      XCHG    DL,DH
142E  80 CE 80                   OR      DH,10000000B
1431  80 E6 BF                   AND     DH,10111111B    ; EDIT UPER  CODE
1434  5E                         POP     SI
1435  1F                         POP     DS
1436  8A C6                      MOV     AL,DH
1438  8A 26 00E1 R               MOV     AH,ZENATTR1
143C  E8 1139 R                  CALL    KKWOIL                  ; SET UPER KANJI CODE & ATTRIBUTE
143F  49                         DEC     CX
1440  8A C2                      MOV     AL,DL
1442  8A 26 00E2 R               MOV     AH,ZENATTR2
1446  E8 1139 R                  CALL    KKWOIL                  ; SET LOWER KANJI CODE & ATTRIBUTE
1449  5A                         POP     DX
144A  49                         DEC     CX
144B  FF 0E 0144 R               DEC     KKOHOZAN
144F  42                         INC     DX                      ; NEXT SELECT NUMBER
1450  FF 06 0146 R               INC     KKOHOCNT
1454  45                         INC     BP                      ; RENEW KOUHO POINTER
1455  45                         INC     BP
1456  83 3E 0146 R 07            CMP     KKOHOCNT,7              ; IF SELECT NUMBER >= 7
145B  7C B1                      JL      KHED040                 ;   THEN REPEAT
145D  FF 0E 0146 R               DEC     KKOHOCNT                ; ADJUST KOUHO COUNT
1461  EB 50                      JMP     SHORT KHED080
1463                        KHED050:
1463  F6 C7 01                   TEST    BH,NUMYOMFG
1466  74 18                      JZ      KHED052
1468  F6 C7 02                   TEST    BH,NUMSKIP
146B  75 0D                      JNZ     KHED051
146D  FF 06 0144 R               INC     KKOHOZAN                ; FORCED ADJUST "ZAN"
1471  FF 0E 0146 R               DEC     KKOHOCNT                ; FORCED ADJUST KOUHO COUNT
1475  80 CF 02                   OR      BH,NUMSKIP
1478  EB 94                      JMP     SHORT KHED040
147A                        KHED051:
147A  FF 0E 0144 R               DEC     KKOHOZAN                ; KOUHO ZAN 1 DOWN
147E  EB 33                      JMP     SHORT KHED080
1480                        KHED052:
1480  8B C1                      MOV     AX,CX
1482  2D 0002                   SUB     AX,2
1485  3B 06 018E R               CMP     AX,KKHQLEN
1489  7D 02                      JGE     KHED060
148B  EB 26                      JMP     SHORT KHED080
148D                        KHED060:
148D  FF 0E 0144 R               DEC     KKOHOZAN
1491  A1 00E4 R                  MOV     AX,SPACE
1494  E8 1139 R                  CALL    KKWOIL                  ; SET HANKAKU SPACE
1497  49                         DEC     CX
1498  8B C2                      MOV     AX,DX
149A  E8 1139 R                  CALL    KKWOIL                  ; SET SELECT NUMBER
149D  49                         DEC     CX
149E  51                         PUSH    CX
149F  8B 0E 018E R               MOV     CX,KKHQLEN
14A3  BE 014A R                  MOV     SI,OFFSET KKHANQUE
14A6                        KHED070:
14A6  AD                         LODSW
14A7  B4 00                      MOV     AH,HANATTR              ; HANKAKU ATTRIBUTE SET
14A9  E8 1139 R                  CALL    KKWOIL                  ; COPY HANKAKU YOMI DATA
14AC  E2 F8                      LOOP    KHED070
14AE  59                         POP     CX
14AF  2B 0E 018E R               SUB     CX,KKHQLEN              ; CALCULATE COLUMN COUNT
14B3                        KHED080:
14B3  41                         INC     CX
14B4  41                         INC     CX
14B5  A1 00E4 R                  MOV     AX,SPACE                ; GET HANKAKU SPACE CODE
14B8                        KHED081:
14B8  E8 1139 R                  CALL    KKWOIL                  ; SPACE CLEAR
14BB  E2 FB                      LOOP    KHED081
14BD  FF 06 0146 R               INC     KKOHOCNT
14C1                        KHED090:
```

```
14C1  8A 26 00E1 R                  MOV     AH,ZENATTR1
14C5  B0 81                         MOV     AL,81H
14C7  E8 1139 R                     CALL    KKHOIL          ; SET "]" FIRST CODE
14CA  8A 26 00E1 R                  MOV     AH,ZENATTR1
14CE  B0 7A                         MOV     AL,7AH
14D0  E8 1139 R                     CALL    KKHOIL          ; SET "]" SECOND CODE
14D3  8A 26 00E1 R                  MOV     AH,ZENATTR1     ; SET "ZAN" KANJI CODE & ATTRIBUTE
14D7  B0 8E                         MOV     AL,8EH          ; SET "ZAN" KANJI CODE & ATTRIBUTE
14D9  E8 1139 R                     CALL    KKHOIL
14DC  8A 26 00E2 R                  MOV     AH,ZENATTR2
14E0  B0 63                         MOV     AL,63H
14E2  E8 1139 R                     CALL    KKHOIL
14E5  8B 16 0144 R                  MOV     DX,KKOHOZAH
14E9  81 E2 03FF                    AND     DX,03FFH
14ED  BD 0014                       MOV     BP,10*2
14F0  2B C0                         SUB     AX,AX
14F2                        KHED100:
14F2  D1 EA                         SHR     DX,1
14F4  73 0B                         JNC     KHED110
14F6  2E: 02 86 1380 R              ADD     AL,BYTE PTR DECAJST[BP]
14FB  27                            DAA
14FC  2E: 12 A6 1381 R              ADC     AH,BYTE PTR DECAJST[1][BP]
1501                        KHED110:
1501  4D                            DEC     BP
1502  4D                            DEC     BP
1503  75 ED                         JNZ     KHED100
1505  8B D0                         MOV     DX,AX
1507  B9 0404                       MOV     CX,0404H
150A                        KHED120:                        ; *** ZERO SUPPRESS ***
150A  F6 C6 F0                      TEST    DH,0F0H
150D  75 0F                         JNZ     KHED130
150F  A1 00E4 R                     MOV     AX,SPACE        ; SET SPACE CODE INSTEAD OF ZERO
1512  E8 1139 R                     CALL    KKHOIL
1515  D3 E2                         SHL     DX,CL           ; GET TO PH-REGISTER
1517  FE CD                         DEC     CH
1519  80 FD 01                      CMP     CH,1
151C  75 EC                         JNZ     KHED120
151E                        KHED130:                        ; *** SET NUMBER DATA(S)
151E  B4 00                         MOV     AH,HANATTR
1520  8A C6                         MOV     AL,DH
1522  24 F0                         AND     AL,0F0H
1524  D2 E8                         SHR     AL,CL
1526  0C 30                         OR      AL,30H          ; ADJUST TO DISPLAY CODE
1528  E8 1139 R                     CALL    KKHOIL          ; SET NUMBER
152B  D3 E2                         SHL     DX,CL
152D  FE CD                         DEC     CH
152F  75 ED                         JNZ     KHED130
1531  2B C0                         SUB     AX,AX
1533  C3                            RET
1534                        KHED900:
1534  B8 0001                       MOV     AX,1
1537  C3                            RET
1538                        KKOHOEDT ENDP
                            ;**** CLOIL ***************************************************
                            ;*                                                           *
                            ;*   CLEAR OPERATOR INFOMATION LINE                          *
                            ;*                                                           *
                            ;***************************************************************
1538                        CLOIL   PROC
                            ;       MOV     AH,81H
                            ;       MOV     CX,2000H
                            ;       INT     10H
1538  E8 12EC R                     CALL    ERASE_CURSOR    ; ERASE CURSOR
153B  8B 16 00F3 R                  MOV     DX,KKCUSR       ;LOAD KANA-KAN CURSOR POSITION
153F  8A 16 0101 R                  MOV     DL,DOILP        ;LOAD TOP POSITION OF OIL
1543  8A 36 00FC R                  MOV     DH,KKOILP       ;LOAD OIL POSITION
1547  B0 20                         MOV     AL,BLANK        ;LOAD BLANK DATA
1549  8A 0E 0102 R                  MOV     CL,DOILN        ;LOAD NUMBER OF OIL
154D  32 ED                         XOR     CH,CH
154F                        CL0010:
154F  E8 1557 R                     CALL    WDISP           ;CLEAR OIL
1552  FE C2                         INC     DL
1554  E2 F9                         LOOP    CL0010
1556  C3                            RET                     ;RETURN TO CALLER
1557                        CLOIL   ENDP
                            ;**** WDISP ***************************************************
                            ;*                                                           *
                            ;*   WRITE CHARACTER                                         *
                            ;*                                                           *
                            ;***************************************************************
1557                        WDISP   PROC
1557  56                            PUSH    SI              ;SAVE REGISTER
1558  57                            PUSH    DI
1559  55                            PUSH    BP
155A  51                            PUSH    CX
155B  53                            PUSH    BX
155C  50                            PUSH    AX
155D  50                            PUSH    AX
155E  B4 82                         MOV     AH,82H
1560  CD 10                         INT     10H             ;SET ALTERNATE CURSOR
1562  58                            POP     AX
1563  B3 0F                         MOV     BL,0FH          ;SET COLOR (IN GRAPHIS MODE)
1565  B9 0001                       MOV     CX,D01
1568  B4 8A                         MOV     AH,8AH
156A  CD 10                         INT     10H             ;WRITE DISPLAY
156C  58                            POP     AX              ;RESTORE REGISTER
156D  5B                            POP     BX
156E  59                            POP     CX
156F  5D                            POP     BP
1570  5F                            POP     DI
1571  5E                            POP     SI
1572  C3                            RET                     ;RETURN TO CALLER
```

```
1573                    WDISP   ENDP
                        ;*************************************************************
                        ;*                                                           *
                        ;* PROGRAM NAME: DTINST                                      *
                        ;*                                                           *
                        ;* DESCRIPTIVE NAME: DATA INSERT PROCESS ROUTINE             *
                        ;*                                                           *
                        ;* FUNCTION: THIS ROUTIN INSERTS A CHARACTER FRONT OF THE CURSOR *
                        ;*           POSITION AND MODIFIES END POSITION              *
                        ;*                                                           *
                        ;* LINKAGE:                                                  *
                        ;*                                                           *
                        ;*   INPUT: BX-- OFFSET OF DATA POSITION TO INSERT           *
                        ;*          DX-- DATA LENGTH TO INSERT                       *
                        ;*                                                           *
                        ;*   OUTPUT: KANA-KAN COMMON TABLES                          *
                        ;*                                                           *
                        ;*   RETURN CODES: (AX)                                      *
                        ;*                                                           *
                        ;*      0 - SUCCESSFUL                                       *
                        ;*      1 - INVALID OPERATION                                *
                        ;*                                                           *
                        ;*  EXTERNAL REFERENCES:                                     *
                        ;*                                                           *
                        ;*    ROUTINES: EOPCHECK-- CHECK AND MODIFY END POSITION OF  *
                        ;*                         CHARACTERS                        *
                        ;*                                                           *
                        ;*    TABLES: KANA-KAN COMMON TABLES                         *
                        ;*                                                           *
                        ;*  REGISTERS: AX - RETURN CODE                              *
                        ;*             ALL OTHERS UNCHANGED                          *
                        ;*                                                           *
                        ;*  CHANGE ACTIVITY: VERSION 00.00                           *
                        ;*                                                           *
                        ;*************************************************************
                        ;
                        ;
                        ;*****<DTINST>**************************************************
                        ;*                                                           *
                        ;*         WHEN NOT INSERT KEY PRESSED ON INSERT MODE        *
                        ;*            THIS SUBROUTINE IS CALLED                      *
                        ;*                                                           *
                        ;*              BX: OFFSET OF DATA POSITION TO INSERT        *
                        ;*                                                           *
                        ;*****<DTINST>**************************************************
1573                    DTINST  PROC    NEAR
1573  53                        PUSH    BX              ; SAVE REGISTERS
1574  51                        PUSH    CX
1575  52                        PUSH    DX
1576  56                        PUSH    SI
1577  57                        PUSH    DI
1578                    DTI_010:
1578  E8 1593 R                 CALL    XMTDATA         ; TRANSMIT DATA OF INPUT BUFFER
157B                    DTI_020:
157B  8B F3                       MOV     SI,BX           ; GET OFFSET VALUE OF DATA TO INSERT
157D  8D 3E 0009 R                LEA     DI,KKINBUF      ; GET OFFSET OF INPUT BUFFER
1581  03 3E 00E6 R                ADD     DI,KKHCCA       ; DI POINTS CURSOR POSITION
1585  8B CA                       MOV     CX,DX           ; GET DATA LENGTH
1587  FC                          CLD                     ; SET DIRECTION FLAG
1588  F3/ A4                      REP     MOVSB           ; INSERT DATA
158A                    DTI_030:
158A                    DTI_040:
158A  B8 0000                     MOV     AX,0H           ; SET RETURN CODE(NORMAL)
158D  5F                          POP     DI              ; RESTORE REGISTERS
158E  5E                          POP     SI
158F  5A                          POP     DX
1590  59                          POP     CX
1591  5B                          POP     BX
1592                    DT_990:
1592  C3                        RET                     ; RETURN TO CALLER
1593                    DTINST  ENDP
                        ;
                        ;*****<XMTDATA>************************************************
                        ;*                                                           *
                        ;*       TRANSMIT DATA FROM INPUT BUFFER TO ITSELF           *
                        ;*                                                           *
                        ;*                INPUT   DX:DATA LENGTH TO INSERT           *
                        ;*                                                           *
                        ;*****<XMTDATA>************************************************
                        ;
1593                    XMTDATA         PROC    NEAR
1593  A1 00EA R                 MOV     AX,KKHEOP       ; GET END POSITION OF CHARACTERS
1596  3B 06 00E6 R              CMP     AX,KKHCCA       ; CURSOR POS. >= END POS. ?
159A  76 24                     JBE     XMT_030         ; NO,...
159C  03 C2                     ADD     AX,DX           ; YES,...
159E  3B 06 00E8 R              CMP     AX,KKEOPMAX     ; CHECK BOUNDARY
15A2  76 03                     JNA     XMT_020         ; IF END POINTER =< END OF BUFFER

15A4                    XMT_010:                        ; ELSE
15A4  E8 15C5 R           CALL    MODEOP          ; MODIFY END POSITION
15A7                    XMT_020:
15A7  8B 0E 00EA R              MOV     CX,KKHEOP       ; GET DATA LENGTH OF XMTION
15AB  2B 0E 00E6 R              SUB     CX,KKHCCA
15AF  8D 36 0009 R              LEA     SI,KKINBUF      ; SET POINTER OF MOVEMENT AT FIRST
15B3  03 36 00EA R              ADD     SI,KKHEOP
15B7  4E                        DEC     SI
15B8  8B FE                     MOV     DI,SI           ; SET POINTER OF MOVEMENT THE POINT
15BA  03 FA                     ADD     DI,DX
15BC  FD                        STD                     ; SET DIRECTION FLAG
15BD  F3/ A4                    REP     MOVSB           ; XMT DATA
15BF  FC                        CLD                     ; RESET DIRECTION FLAG
15C0                    XMT_030:
15C0  01 16 00EA R              ADD     KKHEOP,DX       ; UPDATE END POSITION OF DATA
```

```
15C4                                    XMT_990:
15C4  C3                                        RET
15C5                                    XMTDATA ENDP
                                        ;*****<MODEOP>**************************************************************
                                        ;*                                                                        *
                                        ;*                    MODIFY END POINTER                                  *
                                        ;*                                                                        *
                                        ;*****<MODEOP>**************************************************************
15C5                                    MODEOP  PROC    NEAR
15C5  8D 36 0009 R                              LEA     SI,KKINBUF      ; GET OFFSET OF INPUT BUFFER
15C9  03 36 00EA R                              ADD     SI,KKHEOP
15CD  83 EE 02                                  SUB     SI,2            ; SI POINTS ATTR OF LAST CHARACTER
15D0                                    MOD_020:
15D0  83 FA 06                                  CMP     DX,ZENLEN       ; CHECK DATA LENGTH OF INSERTION
15D3  75 29                                     JNZ     MOD_070         ; IF HANKAKU DATA
15D5                                    MOD_030:                        ; ELSE (ZENKAKU)
15D5  E8 1612 R                                 CALL    GETATRI         ; GET ATTR OF LAST CHARACTER
15D8  74 04                                     JZ      MOD_040         ; LAST CHR. = HANKAKU ?
15DA  B0 06                                     MOV     AL,ZENLEN       ; NO,LAST CHR. = ZENKAKU
15DC  EB 2E                                     JMP     SHORT MOD_990   ; GOTO RETURN
15DE                                    MOD_040:                        ; YES,LAST CHR. = HANKAKU
15DE  A1 00EA R                                 MOV     AX,KKHEOP       ; GET END POSITION OF CHARACTERS
15E1  3B 06 00E8 R                              CMP     AX,KKEOPMAX     ; CHECK BOUNDARY.
15E5  74 04                                     JZ      MOD_050
15E7  B0 03                                     MOV     AL,INDEX        ;
15E9  EB 21                                     JMP     SHORT MOD_990   ; GOTO RETURN
15EB                                    MOD_050:
15EB  83 EE 03                                  SUB     SI,INDEX        ; SI POINTS LAST 2ND CHARACTER ATTR.
15EE  E8 1612 R                                 CALL    GETATRI         ; GET ATTR. OF DATA THAT SI POINTS
15F1  75 04                                     JNZ     MOD_060
15F3  B0 06                                     MOV     AL,ZENLEN
15F5  EB 15                                     JMP     SHORT MOD_990   ; GOTO RETURN
15F7                                    MOD_060:
15F7  E8 1619 R                                 CALL    CLEAR3          ; CLEAR LAST 3 BYTE OF INPUT BUFFER
15FA  B0 09                                     MOV     AL,ZENLEN+INDEX ; UPDATE END POINTER
15FC  EB 0E                                     JMP     SHORT MOD_990   ; GOTO RETURN
15FE                                    MOD_070:
15FE  E8 1612 R                                 CALL    GETATRI         ; GETATTR
1601  75 04                                     JNZ     MOD_080         ; IF LAST CHARACTER IS HANKAKU
1603  B0 03                                     MOV     AL,INDEX
1605  EB 05                                     JMP     SHORT MOD_990
1607                                    MOD_080:                        ; ELSE (ZENKAKU)
1607  E8 1619 R                                 CALL    CLEAR3          ; CLEAR SPACE AREA
160A  B0 06                                     MOV     AL,ZENLEN       ; UPDATE END POINTER
160C                                    MOD_990:
160C  98                                        CBW
160D  29 06 00EA R                              SUB     KKHEOP,AX
1611  C3                                        RET
1612                                    MODEOP  ENDP
                                        ;*****<GETATRI>************************************************************
                                        ;*                                                                        *
                                        ;*         SI: OFFSET OF DATA ATTRIBUTE                                   *
                                        ;*                                                                        *
                                        ;*****<GETATRI>************************************************************
1612                                    GETATRI PROC    NEAR
1612  8A 04                                     MOV     AL,BYTE PTR DS:[SI]
                                                                        ; GET ATTRIBTEOP OF CHARACTER
1614  22 06 00E1 R                              AND     AL,ZENATTR1     ; ZENKAKU OR HANKAKU
1618  C3                                        RET
1619                                    GETATRI ENDP
                                        ;*****<CLEAR3>*************************************************************
                                        ;*                                                                        *
                                        ;*       THIS ROUTINE THE CLEAR SPACE AREA (3BYTE)                        *
                                        ;*                                                                        *
                                        ;*              SI: POINTS THE AREA FOR CLEAR CHAR                        *
                                        ;*                                                                        *
                                        ;*****<CLEAR3>*************************************************************
1619                                    CLEAR3  PROC    NEAR
1619  50                                        PUSH    AX
161A  8D 3E 0009 R                              LEA     DI,KKINBUF      ; GET OFFSET VALUE OF INPUT BUFFER
161E  03 3E 00E8 R                              ADD     DI,KKEOPMAX     ;
1622  83 EF 03                                  SUB     DI,INDEX        ; SI POINTS THE TOP OF CLEARING AREA
1625  A1 00F5 R                                 MOV     AX,KKCHRSP
1628  FC                                        CLD
1629  AB                                        STOSW                   ; SET SPACE CODE
162A  32 C0                                     XOR     AL,AL
162C  AA                                        STOSB
162D  58                                        POP     AX
                                                                        ;
162E  C3                                        RET                     ; RETURN TO CALLER
162F                                    CLEAR3  ENDP
                                        ;*************************************************************************
                                        ;*                                                                       *
                                        ;* PROGRAM NAME: DTSHFT                                                  *
                                        ;*                                                                       *
                                        ;* DESCRIPTIVE NAME: SHIFT DATA & MODIFY END POSITION                    *
                                        ;*                                                                       *
                                        ;* FUNCTION: WHEN DELETE KEY OR BACK SPACE KEY IS PRESSED,               *
                                        ;*           THIS ROUTINE SHIFT CHARACTERS AFTER THE POSITION            *
                                        ;*           POINTED BY AX REGISTER                                      *
                                        ;*                                                                       *
                                        ;* LINKAGE:                                                              *
                                        ;*                                                                       *
                                        ;*   INPUT: AX -- POINTS DATA TO BE DELETE                               *
                                        ;*                                                                       *
                                        ;*   OUTPUT: DX -- LENGTH OF DATA TO BE DELETED                          *
                                        ;*                                                                       *
                                        ;*   RETURN CODES: (AX)                                                  *
                                        ;*                                                                       *
                                        ;*      0 - SUCCESSFUL                                                   *
                                        ;*      1 - SUCCESSFUL                                                   *
                                        ;*                                                                       *
                                        ;*  EXTERNAL REFERENCES:                                                 *
```

```
                        ;*
                        ;*    ROUTINES: EOPCHECK -- CHECK END POSITION AND SET HANKAKU      *
                        ;*                          SPACE OR PSECIAL CHR.                   *
                        ;*                                                                  *
                        ;*    TABLES:                                                       *
                        ;*                                                                  *
                        ;*  REGISTERS: AX - RETURN CODE                                     *
                        ;*             ALL OTHERS UNCHANGED                                 *
                        ;*                                                                  *
                        ;*  CHANGE ACTIVITY: VERSION 00.00                                  *
                        ;*                                                                  *
                        ;*******************************************************************
                        ;
                        ;*****<DTSHFT>*****************************************************
                        ;*                                                                  *
                        ;*      THIS ROUTINE SHIFTS DATA OF INPUT BUFFER LEFT               *
                        ;*      INPUT: AX -- DATA POSITION TO BE DELETED                    *
                        ;*                                                                  *
                        ;*****<DTSHFT>*****************************************************
162F                    DTSHFT  PROC    NEAR                    ;
162F  53                        PUSH    BX                      ; SAVE REGISTERS
1630  51                        PUSH    CX
1631  56                        PUSH    SI
1632  57                        PUSH    DI
1633  3B 06 00EA R              CMP     AX,KKWEOP               ; CHECK POINTER
1637  73 31                    JAE     DTS_020                  ; AX >= NED POSITION
1639  8D 36 0009 R              LEA     SI,KKINBUF              ; GET OFFSET OF INPUT BUFFER
163D  03 F0                    ADD     SI,AX                   ; SI POINTS DATA TO BE DELETED
163F  8A 44 01                 MOV     AL,BYTE PTR DS:[SI+1]
                                                               ; GET ATTR. OF DATA
1642  E8 1341 R                CALL    KCALELM                 ; CHECK ZENKAKU OR HANKAKU
1645  8B D0                    MOV     DX,AX                   ; SAVE DATA LENGTH
1647  8B FE                    MOV     DI,SI                   ; DI POINTS DESTINATION POSITION
1649  03 F2                    ADD     SI,DX                   ; SI POINTS SOURCE POSITION
164B  8D 0E 0009 R              LEA     CX,KKINBUF              ; GET OFFSET OF INPUT BUFFER
164F  03 0E 00EA R              ADD     CX,KKWEOP               ; GET OFFSET OF DATA TO BE DELETED
1653  2B CE                    SUB     CX,SI                   ; CX = REPEAT CNT.
1655  83 F9 00                 CMP     CX,0H                   ; CHECK REPEAT CNT.
1658  7C 10                    JL      DTS_020                  ; REPEAT CNT. < 0
165A  74 03                    JZ      DTS_010                  ; REPEAT CNT. = 0
165C  FC                       CLD                             ; SET DIRECTION FLAG
165D  F3/ A4                   REP     MOVSB                   ; TRANSFER DATA OF INPUT BUFFER
165F                    DTS_010:
165F  29 16 00EA R             SUB     KKWEOP,DX               ; SET PARAMETER FOR EOPCHECK
1663  E8 1672 R                CALL    EOPCHECK                ; CHECK EOP
1666  01 16 00EA R             ADD     KKWEOP,DX               ; RESET PARAMETER TO RETURN
166A                    DTS_020:
166A  B8 0000                  MOV     AX,0H                   ; SET RETURN CODE ( NORMAL )
166D  5F                       POP     DI                      ; RESTORE REGISTERS
166E  5E                       POP     SI
166F  59                       POP     CX
1670  5B                       POP     BX
1671  C3                       RET                             ; RETURN TO CALLER
1672                    DTSHFT  ENDP                            ;
                        ;*******************************************************************
                        ;*                                                                  *
                        ;* PROGRAM NAME: EOPCHECK                                           *
                        ;*                                                                  *
                        ;* DESCRIPTIVE NAME: CHECK END POSITION AND SET SPACE OR SPECIAL    *
                        ;*                                                                  *
                        ;* FUNCTION:  CHECK END POSITION AND SET SPACE OR SPECIAL           *
                        ;*            CHARACTER                                             *
                        ;*                                                                  *
                        ;* LINKAGE:                                                         *
                        ;*                                                                  *
                        ;*   INPUT: DX-- <>0: LENGTH OF DATA TO BE DELETED                  *
                        ;*               =0: INSERT PROCESS                                 *
                        ;*                                                                  *
                        ;*   OUTPUT: NONE                                                   *
                        ;*                                                                  *
                        ;*   RETURN CODES: (AX)                                             *
                        ;*                                                                  *
                        ;*      0 - SUCCESSFUL                                              *
                        ;*      1 - SUCCESSFUL                                              *
                        ;*                                                                  *
                        ;*  EXTERNAL REFERENCES:                                            *
                        ;*                                                                  *
                        ;*    ROUTINES: NONE                                                *
                        ;*                                                                  *
                        ;*    TABLES:                                                       *
                        ;*                                                                  *
                        ;*  REGISTERS: AX - RETURN CODE                                     *
                        ;*             ALL OTHERS UNCHANGED                                 *
                        ;*                                                                  *
                        ;*  CHANGE ACTIVITY: VERSION 00.00                                  *
                        ;*                                                                  *
                        ;*******************************************************************
                        ;
                        ;*****<EOPCHECK>***************************************************
                        ;*                                                                  *
                        ;*      CHECK END POINTER AND CLEAR FIELD OF SPACE AREA              *
                        ;*      INPUT:  DX: <> 0 :LENGTH OF DELETE DATA                     *
                        ;*                  = 0 : INSERT PROCESS                            *
                        ;*                                                                  *
                        ;*****<EOPCHECK>***************************************************
1672                    EOPCHECK        PROC    NEAR
1672  56                        PUSH    SI                      ; SAVE REGISTER
1673  A1 00EA R                 MOV     AX,KKWEOP               ; GET END POSITION
1676  3B 06 00E8 R              CMP     AX,KKEOPMAX             ; CHECK BUFFER END
167A  73 1A                    JAE     EOP_990                 ; GOTO RETURN
167C  8D 36 0009 R              LEA     SI,KKINBUF              ; GET OFFSET OF INPUT BUFFER
1680  03 F0                    ADD     SI,AX                   ; SI POINTS INSERT AREA
```

```
1682  8B C2                         MOV     AX,DX
1684  B1 03                         MOV     CL,INDEX        ;
1686  F6 F1                         DIV     CL
1688  84 C0                         TEST    AL,AL           ; CHECK
168A  75 03                         JNZ     EOP_090
168C  B8 0001                       MOV     AX,1
168F                          EOP_090:
168F  8B C8                         MOV     CX,AX
1691                          EOP_100:
1691  E8 169A R                     CALL    SETSPC          ; SET HANKAKU SPACE
1694  E2 FB                         LOOP    EOP_100
1696                          EOP_990:
1696  33 C0                         XOR     AX,AX           ; RETURN CODE ( NORMAL )
1698  5E                            POP     SI              ; RESTORE REGISTER
1699  C3                            RET                     ; RETURN TO CALLER
169A                          EOPCHECK        ENDP
                              ;******<SETSPC>*****************************************************
                              ;*                                                                 *
                              ;*      SET HANKAKU SPACE AFTER END DATA POSITION                  *
                              ;*              SI: POINTS HEAD POSITION TO BE INSERT              *
                              ;*                                                                 *
                              ;******<SETSPC>*****************************************************
169A                          SETSPC  PROC    NEAR            ; SET SPACE CODE
169A  A1 00F5 R                     MOV     AX,KKCHRSP      ; GET SPECIAL CHARACTER CODE
169D  FC                            CLD
169E  AB                            STOSW
169F  32 C0                         XOR     AL,AL
16A1  AA                            STOSB
16A2  C3                            RET                     ; RETURN TO CALLER
16A3                          SETSPC  ENDP                  ;
                              ;*****************************************************************
                              ;*                                                               *
                              ;* PROGRAM NAME   ( SUBROUTINE BLOCK )                           *
                              ;*                :    SETSTRCL                                  *
                              ;*                :    SETENDCL                                  *
                              ;*                :    GETCULM                                   *
                              ;* STATUS:                                                       *
                              ;*                                                               *
                              ;* FUNCTION:  CALCULATE COLUMN POSITION : FROM POINTER OF INPUT BUFF.*
                              ;* LINKAGE:                                                      *
                              ;*                                                               *
                              ;*   INPUT:                                                      *
                              ;*                                                               *
                              ;*                                                               *
                              ;*   OUTPUT:                                                     *
                              ;*                                                               *
                              ;*   RETURN CODES: NON                                           *
                              ;*                                                               *
                              ;*  EXTERNAL REFERENCES                                          *
                              ;*                                                               *
                              ;*    ROUTINES: NONE                                             *
                              ;*                                                               *
                              ;*    TABLES:                                                    *
                              ;*                                                               *
                              ;*  REGISTERS: AX - RESULT                                       *
                              ;*             ALL OTHERS UNCHANGED                              *
                              ;*                                                               *
                              ;*  CHANGE ACTIVITY: VERSION 00.00                               *
                              ;*                                                               *
                              ;*****************************************************************
                              ;
                              ;******<SETSTRCL>****************************************************
                              ;*                                                                 *
                              ;*      SET START POSITION FOR DISPLAY ROUTINE                     *
                              ;*                                                                 *
                              ;******<SETSTRCL>****************************************************
16A3                          SETSTRCL PROC   NEAR            ;
16A3  A1 00E6 R                     MOV     AX,KKHCCA       ; GET START POSITION
16A6  A3 00F1 R                     MOV     KKINP,AX        ; SET START POINT FOR DISPLAY
16A9  E8 16BA R                     CALL    GETCULM         ;
16AC  A3 0107 R                     MOV     DSTRTP,AX       ; SET START COLUMN FOR DISPLAY
16AF  C3                            RET                     ;
16B0                          SETSTRCL ENDP
                              ;******<SETENDCL>****************************************************
                              ;*                                                                 *
                              ;*      SET END POSITION FOR DISPLAY ROUTINE                       *
                              ;*                                                                 *
                              ;******<SETENDCL>****************************************************
16B0                          SETENDCL PROC   NEAR            ;
16B0  A1 00EA R                     MOV     AX,KKHEOP       ; GET END POSITION
16B3  E8 16BA R                     CALL    GETCULM         ;
16B6  A3 0109 R                     MOV     DENDP,AX        ; SET END POINT (COLUMN) FOR DISPLAY
16B9  C3                            RET                     ;
16BA                          SETENDCL ENDP
                              ;******<GETCULM>****************************************************
                              ;*                                                                 *
                              ;*      CALCULATE COLUMN POSITION                                  *
                              ;*                                                                 *
                              ;******<GETCULM>****************************************************
16BA                          GETCULM PROC    NEAR            ;
16BA  51                            PUSH    CX
16BB  B1 03                         MOV     CL,INDEX        ;
16BD  F6 F1                         DIV     CL
16BF  32 E4                         XOR     AH,AH           ; CLEAR AH
16C1  03 06 00EE R                  ADD     AX,BASE         ; GET COLUMN POSITION
16C5  59                            POP     CX
16C6  C3                            RET                     ;
16C7                          GETCULM ENDP
                              ;******<CDCHCK1>****************************************************
                              ;*                                                                 *
                              ;*      CODE REASONABLE CHECK 1                                    *
                              ;*                                                                 *
                              ;*      INPUT : BL:CODE                                            *
                              ;*              AL:RANGE TYPE                                      *
```

```
                        ;*                                                                    *
                        ;*****<CDCHCK1>*******************************************************
                        ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
16C7                    RNGTBL1:
16C7  72                        DB      72H             ; TEN KEY SCAN CODE
16C8  7A                        DB      7AH             ;
16C9  02                        DB      02H             ; GRAPHIC KEY SCAN CODE
16CA  0A                        DB      0AH             ;
16CB  30                        DB      30H             ; NUMERIC CODE
16CC  39                        DB      39H             ;
16CD  A7                        DB      0A7H            ; HANKAKU KATAKANA CODE
16CE  DF                        DB      0DFH            ;
16CF                    RNGTBLL1:
= 0004                  RNGMAX1 EQU     (RNGTBLL1-RNGTBL1)/2
= 0001                  RNGMIN1 EQU     01H             ; RANGE MINIMUM
                        ;**************************************************************************
16CF                    CDCHCK1 PROC    NEAR            ;
16CF  53                        PUSH    BX              ; SAVE REGISTERS
16D0  51                        PUSH    CX              ;
16D1  55                        PUSH    BP              ;
16D2  3C 01                     CMP     AL,RNGMIN1      ; CHECK RANGE TYPE
16D4  72 1E                     JC      CD1_050         ; AL<RANGE MIN.,THEN ERROR RETURN
16D6  3C 04                     CMP     AL,RNGMAX1      ;
16D8  77 1A                     JA      CD1_050         ; AL>RANGE MAX.,THEN ERROR RETURN
16DA  32 E4                     XOR     AH,AH
16DC  FE C8                     DEC     AL
16DE  8B E8                     MOV     BP,AX
16E0  D1 E5                     SHL     BP,1
16E2  2E: 8B 96 16C7 R          MOV     DX,WORD PTR RNGTBL1[BP]
                                                        ; GET RANGE (DH=MAX.,DL=MIN.)
16E7  3A F3                     CMP     DH,BL           ; ? CODE =< MAX.
16E9  72 09                     JC      CD1_050         ; NO,THEN....
16EB  3A DA                     CMP     BL,DL           ; ? CODE >= MIN.
16ED  72 05                     JC      CD1_050         ; NO,THEN....
16EF  B8 0000                   MOV     AX,0H           ; SET RETURN CODE (NORMAL)
16F2  EB 03                     JMP     SHORT CD1_990   ; GOTO RETURN
16F4                    CD1_050:                        ;
16F4  B8 0001                   MOV     AX,01H          ; SET RETURN CODE (ERROR)
16F7                    CD1_990:                        ;
16F7  5D                        POP     BP              ; RESTORE REGISTERS
16F8  59                        POP     CX              ;
16F9  5B                        POP     BX              ;
16FA  C3                        RET                     ; RETURN TO CALLER
16FB                    CDCHCK1 ENDP                    ;
                        ;*****<CDCHCK2>*******************************************************
                        ;*                                                                    *
                        ;*      CODE REASONABLE CHECK 2 (WORD)                                *
                        ;*                                                                    *
                        ;*      INPUT : BX:CODE                                               *
                        ;*              AL:RANGE TYPE                                         *
                        ;*                                                                    *
                        ;*****<CDCHCK2>*******************************************************
                        ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
16FB                    RNGTBL2:
16FB  824F                      DW      824FH           ; 2 BYTE NUMERIC CODE
16FD  8258                      DW      8258H           ;
16FF  829F                      DW      829FH           ; 2 BYTE HIRAGANA CODE
1701  82F1                      DW      82F1H           ;
1703  8340                      DW      8340H           ; 2 BYTE KATAKANA CODE
1705  8393                      DW      8393H           ;
                        ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1707                    RNGTBL2:
= 0006                  RNGMAX2 EQU     (RNGTBLL2-RNGTBL2)/2
= 0001                  RNGMIN2 EQU     01H             ; RANGE MINIMUM (WORD TYPE)
                        ;**************************************************************************
1707                    CDCHCK2 PROC    NEAR            ;
1707  53                        PUSH    BX              ; SAVE REGISTERS
1708  51                        PUSH    CX              ;
1709  55                        PUSH    BP              ;
170A  3C 01                     CMP     AL,RNGMIN2      ; CHECK RANGE TYPE
170C  72 25                     JC      CD2_050         ; AL<RANGE MIN.,THEN ERROR RETURN
170E  3C 06                     CMP     AL,RNGMAX2      ;
1710  77 21                     JA      CD2_050         ; AL>RANGE MAX.,THEN ERROR RETURN
1712  32 E4                     XOR     AH,AH
1714  FE C8                     DEC     AL
1716  8B E8                     MOV     BP,AX
1718  B1 02                     MOV     CL,02H
171A  D3 E5                     SHL     BP,CL
171C  2E: 8B 96 16FB R          MOV     DX,WORD PTR RNGTBL2[BP]
1721  3B DA                     CMP     BX,DX           ; ? CODE >= MIN.
1723  72 0E                     JC      CD2_050         ; NO,THEN....
1725  2E: 8B 96 16FD R          MOV     DX,WORD PTR RNGTBL2[BP+2]
                                                        ; GET RANGE (DX=MAX.)
172A  3B D3                     CMP     DX,BX           ; ? CODE =< MAX.
172C  72 05                     JC      CD2_050         ; NO,THEN....
172E  B8 0000                   MOV     AX,0H           ; SET RETURN CODE (NORMAL)
1731  EB 03                     JMP     SHORT CD2_990   ; GOTO RETURN
1733                    CD2_050:                        ;
1733  B8 0001                   MOV     AX,01H          ; SET RETURN CODE (ERROR)
1736                    CD2_990:                        ;
1736  5D                        POP     BP              ; RESTORE REGISTERS
1737  59                        POP     CX              ;
1738  5B                        POP     BX              ;
1739  C3                        RET                     ; RETURN TO CALLER
173A                    CDCHCK2 ENDP
173A                            ORG     $
1800                            ORG     BIGIN+1800H
1800                    CODE    ENDS
                                END
```

Appendix A.

This page intentionally left blank.

```
                ;--------------------------------------------------------------
                ; THIS SUBROUTINE SETS DS TO POINT TO THE BIOS DATA AREA
                ; INPUT: NONE
                ; OUTPUT: DS IS SET
                ;--------------------------------------------------------------
                        ASSUME  CS:CODE,DS:DATA
0000            DDS     PROC    NEAR
0000  50                PUSH    AX
0001  B8 0040           MOV     AX,40H
0004  8E D8              MOV     DS,AX
0006  58                POP     AX
0007  C3                RET
0008            DDS     ENDP
                ;--- INT 1A ------------------------------------------------------
                ; TIME_OF_DAY/SOUND SOURCE SELECT
                ;       THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ.
                ;       AN INTERFACE FOR SETTING THE MULTIPLEXER FOR
                ;       AUDIO SOURCE IS ALSO PROVIDED
                ;
                ; INPUT
                ;   (AH) = 0    READ THE CURRENT CLOCK SETTING
                ;               RETURNS CX = HIGH PORTION OF COUNT
                ;                       DX = LOW PORTION OF COUNT
                ;                       AL = 0 IF TIMER HAS NOT PASSED 24 HOURS
                ;                       SINCE LAST READ. <> 0 IF ON ANOTHER DAY
                ;   (AH) = 1    SET THE CURRENT CLOCK
                ;       CX = HIGH PORTION OF COUNT
                ;       DX = LOW PORTION OF COUNT
                ;   (AH) = 80H  SET UP SOUND MULTIPLEXER
                ;       AL =(SOURCE OF SOUND) --> "AUDIO OUT" OR RF MODULATOR
                ;            00 = 8253 CHANNEL 2
                ;            01 = CASSETTE INPUT
                ;            02 = "AUDIO IN" LINE ON I/O CHANNEL
                ;            03 = COMPLEX SOUND GENERATOR CHIP
                ;
                ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SEC
                ;       (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
                ;--------------------------------------------------------------
                        ASSUME  CS:CODE,DS:DATA
0008            TIME_OF_DAY     PROC    FAR
0008  FB                STI                     ; INTERRUPTS BACK ON
0009  1E                PUSH    DS              ; SAVE SEGMENT
000A  E8 0000 R           CALL    DDS
000D  80 FC 80            CMP     AH,80H          ; AH=80
0010  74 2E              JE      T4A             ; MUX_SET-UP
0012  0A E4               OR      AH,AH           ; AH=0
0014  74 07               JZ      T2              ; READ_TIME
0016  FE CC               DEC     AH              ; AH=1
0018  74 16               JZ      T3              ; SET_TIME
001A  FB          T1:     STI                     ; INTERRUPTS BACK ON
001B  1F                  POP     DS              ; RECOVER SEGMENT
001C  CF                  IRET                    ; RETURN TO CALLER
001D  FA          T2:     CLI                     ; NO TIMER INTERRUPTS WHILE READING
001E  A0 0070 R           MOV     AL,TIMER_OFL
0021  C6 06 0070 R 00     MOV     TIMER_OFL,0     ; GET OVERFLOW, AND RESET THE FLAG
0026  8B 0E 006E R        MOV     CX,TIMER_HIGH
002A  8B 16 006C R        MOV     DX,TIMER_LOW
002E  EB EA               JMP     T1              ; TOD_RETURN
0030  FA          T3:     CLI                     ; NO INTERRUPTS WHILE WRITING
0031  89 16 006C R        MOV     TIMER_LOW,DX
0035  89 0E 006E R        MOV     TIMER_HIGH,CX   ; SET THE TIME
0039  C6 06 0070 R 00     MOV     TIMER_OFL,0     ; RESET OVERFLOW
003E  EB DA               JMP     T1              ; TOD_RETURN
0040  51          T4A:    PUSH    CX              ;
0041  B1 05               MOV     CL,5            ;
0043  D2 E0               SAL     AL,CL           ; SHIFT PARM BITS LEFT 5 POSITIONS
0045  86 C4               XCHG    AL,AH           ; SAVE PARM
0047  E4 61               IN      AL,PORT_B       ; GET CURRENT PORT SETTINGS
0049  24 9F               AND     AL,10011111B    ; ISOLATE MUX BITS
004B  0A C4               OR      AL,AH           ; COMBINE PORT BITS/PARM BITS
004D  E6 61               OUT     PORT_B,AL       ; SET PORT TO NEW VALUE
004F  59                  POP     CX
0050  EB C8               JMP     T1              ; TOD_RETURN
0052            TIME_OF_DAY     ENDP
                ;--------------------------------------------------------------
                ;THIS ROUTINE WILL READ TIMER1.  THE VALUE READ IS RETURNED IN AX.
                ;--------------------------------------------------------------
0052            READ_TIME PROC NEAR
0052  B0 40               MOV     AL,40H          ; LATCH TIMER1
0054  E6 43               OUT     TIM_CTL,AL
0056  50                  PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
0057  58                  POP     AX
0058  E4 41               IN      AL,TIMER+1      ; READ LSB
005A  8A E0               MOV     AH,AL           ; SAVE IT IN HIGH BYTE
005C  50                  PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
005D  58                  POP     AX
005E  E4 41               IN      AL,TIMER+1      ; READ MSB
0060  86 C4               XCHG    AL,AH           ; PUT BYTES IN PROPER ORDER
0062  C3                  RET
0063            READ_TIME       ENDP
                ;--------------------------------------------------------------
                ;REAL_VECTOR_SETUP
                ;
                ;       THIS ROUTINE WILL INITIALIZE THE INTERRUPT 48 VECTOR TO
                ;       POINT AT THE REAL INTERRUPT ROUTINE.
                ;--------------------------------------------------------------
0063            REAL_VECTOR_SETUP       PROC    NEAR
0063  50                PUSH    AX              ; SAVE THE SCAN CODE
0064  53                PUSH    BX
0065  06                PUSH    ES
0066  33 C0             XOR     AX,AX           ; INITIALIZE TO POINT AT VECTOR
                                                  ; SECTOR(0)
0068  8E C0               MOV     ES,AX
```

```
006A  BB 0120                         MOV     BX,48H*4H       ; POINT AT INTERRUPT 48
006D  26: C7 07 0000 E                MOV     WORD PTR ES:[BX],OFFSET KEY62_INT ; MOVE IN OFFSET OF
                                                              ; ROUTINE
0072  43                              INC     BX              ; ADD 2 TO BX
0073  43                              INC     BX
0074  0E                              PUSH    CS              ; GET CODE SEGMENT OF BIOS (SEGMENT
                                                              ; RELOCATEABLE)
0075  58                              POP     AX
0076  26: 89 07                       MOV     WORD PTR ES:[BX],AX ; MOVE IN SEGMENT OF ROUTINE
0079  07                              POP     ES
007A  5B                              POP     BX
007B  58                              POP     AX
007C  C3                              RET
007D                            REAL_VECTOR_SETUP       ENDP
                                ;--------------------------------------------------------------------
                                ;KB_NOISE
                                ;       THIS ROUTINE IS CALLED WHEN GENERAL BEEPS ARE REQUIRED FROM
                                ;       THE SYSTEM.
                                ;INPUT
                                ;       BX=LENGH OF THE TONE
                                ;       CX=CONTAINS THE FREQUENCY
                                ;OUTPUT
                                ;       ALL REGISTERS ARE MAINTAINED.
                                ;HINTS
                                ;       AS CX GETS LARGER THE TONE PRODUCED GETS LOWER IN PITCH.
                                ;
                                ;--------------------------------------------------------------------
007D                            KB_NOISE        PROC    NEAR
007D  FB                                STI
007E  50                                PUSH    AX
007F  53                                PUSH    BX
0080  51                                PUSH    CX
0081  E4 61                             IN      AL,061H         ; GET CONTROL INFO
0083  50                                PUSH    AX              ; SAVE
0084                            LOOP01:
0084  24 FC                             AND     AL,0FCH         ; TURN OFF TIMER GATE AND SPEAKER
                                                                ; DATA
0086  E6 61                             OUT     061H,AL         ; OUTPUT TO CONTROL
0088  51                                PUSH    CX              ; HALF CYCLE TIME FOR TONE
0089  E2 FE                       LOOP02: LOOP    LOOP02          ; SPEAKER OFF
008B  0C 02                             OR      AL,2            ; TURN ON SPEAKER BIT
008D  E6 61                             OUT     061H,AL         ; OUTPUT TO CONTROL
008F  59                                POP      CX
0090  51                                PUSH     CX             ; RETRIEVE FREQUENCY
0091  E2 FE                       LOOP03: LOOP    LOOP03          ; ANOTHER HALF CYCLE
0093  4B                                DEC     BX              ; TOTAL TIME COUNT
0094  59                                POP      CX             ; RETRIEVE  FREQ.
0095  75 ED                             JNZ     LOOP01          ; DO ANOTHER CYCLE
0097  58                                POP     AX              ; RECOVER CONTROL
0098  E6 61                             OUT     061H,AL         ; OUTPUT THE CONTROL
009A  59                                POP     CX
009B  5B                                POP     BX
009C  58                                POP     AX
009D  C3                                RET
009E                            KB_NOISE            ENDP
                                ;-----------------------------------------------;
                                ;  EASE OF USE REVECTOR ROUTINE - CALLED THROUGH
                                ;  INT 18H WHEN CASSETTE BASIC IS INVOKED (NO DISKETTE
                                ;  NO CARTRIDGES)
                                ;  KEYBOARD VECTOR IS RESET TO POINT TO "NEW_INT_48"
                                ;  PLAN TO TRANSFER KBDIO
                                ;  BASIC VECTOR IS SET TO POINT TO E000:0
                                ;-----------------------------------------------
009E                            BAS_ENT PROC    FAR
                                        ASSUME  DS:ABS0
009E  2B C0                             SUB     AX,AX
00A0  8E D8                             MOV     DS,AX           ;SET ADDRESSING
00A2  C7 06 0120 R 00F9 R               MOV     KEY62_PTR,OFFSET NEW_INT_48
00A8  A3 0060 R                         MOV     BASIC_PTR,AX    ; SET INT 18=E000:0  JX BASIC
00AB  C7 06 0062 R E000                 MOV     BASIC_PTR+2,0E000H
00B1  CD 18                             INT     18H             ; GO TO BASIC
00B3                            BAS_ENT ENDP
                                ;-----------------------------------------------
                                ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM
                                ; CHANNEL 0 OF THE 8253 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ
                                ; AND THE DIVISOR IS 65536, RESULTING IN APPROX. 18.2 INTERRUPTS
                                ; EVERY SECOND.
                                ;
                                ; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS SINCE POWER
                                ;  ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
                                ;  INTERRUPTS MISSED WHILE INTS. WERE DISABLED ARE TAKEN CARE OF
                                ;  BY THE USE OF TIMER 1 AS A OVERFLOW COUNTER
                                ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT
                                ;  OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE DISKETTE
                                ;  MOTOR, AND RESET THE MOTOR RUNNING FLAGS
                                ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH
                                ;  INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A ROUTINE
                                ; AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.
                                ;-----------------------------------------------
00B3                                    ASSUME  DS:DATA
                                TIMER_INT       PROC    FAR
00B3  FB                                STI                     ; INTERRUPTS BACK ON
00B4  1E                                PUSH    DS
00B5  50                                PUSH    AX
00B6  52                                PUSH    DX              ; SAVE MACHINE STATE
00B7  E8 0000 R                         CALL    DDS
00BA  FF 06 006C R                      INC     TIMER_LOW       ; INCREMENT TIME
00BE  75 04                             JNZ     T4              ; TEST_DAY
00C0  FF 06 006E R                      INC     TIMER_HIGH      ; INCREMENT HIGH WORD OF TIME
00C4                            T4:                             ; TEST_DAY
00C4  83 3E 006E R 18                   CMP     TIMER_HIGH,018H ; TEST FOR COUNT EQUALLING 24 HOURS
00C9  75 15                             JNZ     T5              ; DISKETTE_CTL
00CB  81 3E 006C R 00B0                 CMP     TIMER_LOW,0B0H
```

```
00D1  75 0D                     JNZ     T5                      ; DISKETTE_CTL
                        ;------ TIMER HAS GONE 24 HOURS
00D3  2B C0                     SUB     AX,AX
00D5  A3 006E R                 MOV     TIMER_HIGH,AX
00D8  A3 006C R                 MOV     TIMER_LOW,AX
00DB  C6 06 0070 R 01           MOV     TIMER_OFL,1
                        ;------ TEST FOR DISKETTE TIME OUT
00E0                    T5:                                     ; LOOP TILL ALL OVERFLOWS TAKEN
                                                                ; CARE OF
00E0  FE 0E 0040 R              DEC     MOTOR_COUNT
00E4  75 09                     JNZ     T6                      ; RETURN IF COUNT NOT OUT
00E6  80 26 003F R F0           AND     MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
00EB  B0 80                     MOV     AL,FDC_RESET            ; TURN OFF MOTOR, DO NOT RESET FDC
00ED  E6 F2                     OUT     NEC_CTL,AL              ; TURN OFF THE MOTOR
00EF  CD 1C             T6:     INT     1CH                     ; TRANSFER CONTROL TO A USER
                                                                ; ROUTINE
00F1  B0 20                     MOV     AL,EOI
00F3  E6 20                     OUT     020H,AL                 ; END OF INTERRUPT TO 8259
00F5  5A                        POP     DX
00F6  58                        POP     AX
00F7  1F                        POP     DS
00F8  CF                        IRET                            ; RESET MACHINE STATE
00F9                    TIMER_INT       ENDP                    ; RETURN FROM INTERRUPT
                        ;--------------------------------------------------------------------
                        ;NEW_INT48
                        ;       THIS ROUTINE IS THE INTERRUPT 48 HANDLER WHEN THE MACHINE IS
                        ;       FIRST POWERED ON AND CASSETTE BASIC IS GIVEN CONTROL.  IT
                        ;       HANDLES THE FIRST KEYSTROKES ENTERED FROM THE KEYBOARD AND
                        ;       PERFORMS "SPECIAL" ACTIONS AS FOLLOWS:
                        ;               IF CTRL-ESC IS THE FIRST SEQUENCE "LOAD CAS1:,R" IS
                        ;                  EXECUTED GIVING THE USER THE ABILITY TO BOOT
                        ;                  FROM CASSETTE
                        ;       AFTER THESE KEYSTROKES OR AFTER ANY OTHER KEYSTROKES THE
                        ;       INTERRUPT 48 VECTOR IS CHANGED TO POINT AT THE REAL
                        ;       INTERRUPT 48 ROUTINE.
                        ;--------------------------------------------------------------------
00F9                    NEW_INT_48 PROC FAR
00F9  3C 01                     CMP     AL,1                    ; IS THIS AN ESCAPE KEY?
00FB  74 10                     JE      ESC_KEY                 ; JUMP IF AL=ESCAPE KEY
00FD  3C 1D                     CMP     AL,29                   ; ELSE, IS THIS A CONTROL KEY?
00FF  74 06                     JE      CTRL_KEY                ; JUMP IF AL=CONTROL KEY
0101  E8 0063 R                 CALL    REAL_VECTOR_SETUP ; OTHERWISE, INITIALIZE REAL
                                                                ; INT 48 VECTOR
0104  CD 48                     INT     48H                     ; PASS THE SCAN CODE IN AL
0106                    ESC_ONLY:
0106  CF                       IRET                            ; RETURN TO INTERRUPT 48H
0107                    CTRL_KEY:
0107  80 0E 0017 R 04          OR      KB_FLAG,04H             ; TURN ON CTRL SHIFT IN KB_FLAG
010C  CF                        IRET                            ; RETURN TO INTERRUPT
010D                    ESC_KEY:
010D  F6 06 0017 R 04           TEST    KB_FLAG,04H             ; HAS CONTROL SHIFT OCCURED ?
0112  74 F2                     JE      ESC_ONLY                ; NO, ESC ONLY
                        ;CONTROL ESCAPE HAS OCCURED, PUT MESSAGE IN BUFFER FOR CASSETTE
                        ; LOAD
0114  C6 06 0017 R 00           MOV     KB_FLAG,0               ; ZERO OUT CONTROL STATE
0119  1E                        PUSH    DS
011A  07                        POP     ES                      ; INITIALIZE ES FOR BIOS DATA
011B  1E                        PUSH    DS                      ; SAVE OLD DS
011C  0E                        PUSH    CS                      ; POINT DS AT CODE SEGMENT
011D  1F                        POP     DS
011E  BE 0138 R                 MOV     SI,OFFSET CAS_LOAD ; GET MESSAGE
0121  B9 000E 90                MOV     CX,CAS_LENGTH   ; LENGTH OF CASSETTE MESSAGE
0125  33 C0                     XOR     AX,AX
0127  8A 04             T_LOOP: MOV     AL,[SI]                 ; GET ASCII CHARACTER FROM MESSAGE
0129  CD 79                     INT     79H                     ; PUT IN KEYBOARD BUFFER
012B  46                        INC     SI
012C  E2 F9                     LOOP    T_LOOP
012E  B8 1C0D                   MOV     AX,1C0DH
0131  CD 79                     INT     79H
0133  1F                        POP     DS                      ; RETRIEVE BIOS DATA SEGMENT
                        ;--------------------------------------------------------------------
                        ;***NOTE***
                        ;       IT IS ASSUMED THAT THE LENGTH OF THE CASSETTE MESSAGE IS
                        ;       LESS THAN OR EQUAL TO THE LENGTH OF THE BUFFER.  IF THIS IS
                        ;       NOT THE CASE THE BUFFER WILL EVENTUALLY CONSUME MEMORY.
                        ;--------------------------------------------------------------------
0134  E8 0063 R                 CALL    REAL_VECTOR_SETUP
0137  CF                        IRET
                        ;------ MESSAGE FOR OUTPUT WHEN CONTROL-ESCAPE IS ENTERED AS FIRST
                        ;       KEY SEQUENCE
0138                    CAS_LOAD        LABEL   BYTE
0138  4C 4F 41 44 20 22         DB 'LOAD "CAS1:",R'
      43 41 53 31 3A 22
      2C 52
= 000E                  CAS_LENGTH EQU $ - CAS_LOAD
0146                    NEW_INT_48  ENDP
0300                            ORG     BEGIN+300H
0300                    CODE    ENDS
                                END
```

```
;%
;% PROGRAM NAME = RAS.ASM
;%
;% DESCRIPTIVE NAME = POWER ON SELF TEST
;%
;% FUNCTION = POWER ON SELF TESTS (POSTS)
;%
;% NOTES = NONE
;%
;% DEPENDENCIES = NONE
;%
;% RESTRICTION = NONE
;%
;% PROGRAM TYPE = PROCEDURE
;%
;% PROCESSOR = 8088
;%
;% MODULE SIZE = VALUE OF LABEL "POST_END"
;%
;% ATTRIBUTES =  SERIALLY RE-USABLE
;%
;% LINKAGE = RESET_FLAG : 0000 : POST - POWER ON ENTRY
;%                        1234 : POST - "CTRL"+"ALT"+"DEL" ENTRY
;%                        3412 : POST - "CTRL"+"ALT"+"INS" ENTRY
;%                        4321 : DIAG. CARTRIDGE ENTRY
;%
;% INPUT = NONE
;%
;% OUTPUT = NONE
;%
;% EXIT-NORMAL = INT 19 TO BOOT LOADER, OR
;%               INT 80 TO DIAG. CARTRIDGE, OR
;%               JMP (FAR INDIRECT) TO PCjr CARTRIDGE
;%
;% EXIT-ERROR = NORMAL MODE:
;%               JMP (NEAR) TO E_MSG OR SAME AS EXIT-NORMAL
;%              SERVICE MODE:
;%               JMP (NEAR) TO E_MSG
;%
;% EXTERNAL REFERENCES = VIDEO_PARMS : CRTC PARAMETER TABLE
;%                       DISK_BASE   : DISKETTE DRIVE PARM. TABLE
;%                       EXTAB       : EXTENDED SCAN CODE TABLE
;%
;% EXTERNAL ROUTINES = DDS           : SET DS AS 40H SUBROUTINE
;%                     READ_HALF_BIT : CASSETTE SUBROUTINE
;%                     SEEK          : SEEK SUBROUTINE
;%                     VIDEO_IO      : INT 10H
;%                     DISKETTE_IO   : INT 13H
;%                     KEYBOARD_IO   : INT 16H
;%                     BOOT_STRAP    : INT 19H
;%
;% DATA AREA = 00000 : 8088 INTERRUPT LOCATIONS
;%             00300 : TEMPORARY STACK AREA DURING POST
;%             00400 : ROM BIOS DATA AREA
;%             00500 : EXTRA DATA AREA
;%
;% CONTROL BLOCKS = 00400-7 : RS-232C ADAPTER ADDRESSES
;%                  00408-F : PRINTER ADAPTER ADDRESSES
;%                  00410-1 : INSTALLED HARDWARE
;%                  00413-4 : USABLE MEMORY SIZE IN K BYTES
;%                  00415-6 : REAL MEMORY SIZE IN K BYTES
;%                  0041A-B : KBD BUFFER HEAD POINTER
;%                  0041C-D : KBD BUFFER END POINTER
;%                  0043E   : DISKETTE SEEK STATUS
;%                  00462   : CURRENT PAGE BEING DISPLAYED
;%                  0046B   : CASSETE LAST INPUT VALUE
;%                  00472-3 : WARM START INDICATOR
;%                  00474-7 : DISKETTE SCRACH PADS
;%                  00478-B : PRINTER TIMEOUT VALUE
;%                  0047C-F : RS-232 TIMEOUT VALUE
;%                  00480-3 : KBD BUFFER POINTERS
;%                  00484   : INTERRUPT HAPPEND FLAG
;%                  00505   : MFG CHECK POINT VALUE
;%                  00518   : POST ERROR VALUE
;%                  00702   : DISKETTE DRIVES' CONFIGURATION
;%
;% TABLES = Z0      : SX-08 DATA
;%          Z_0     :    "
;%          ZI      : BRANCH DATA
;%          Z2      :    "
;%          EX_0    :    "
;%          SYSTRS  : SX-08 DATA
;%          PPD     : PRINTER DATA
;%
```

```
0000                    CODE    SEGMENT PUBLIC
                                ASSUME  CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK

                                PUBLIC  POST,POST_END
                                PUBLIC  RESET,D11,DUMMY_RETURN,PRT_HEX,BEEP,VECTOR_TABLE

                                EXTRN   DDS:NEAR
                                EXTRN   READ_HALF_BIT:NEAR
                                EXTRN   SEEK:NEAR
                                EXTRN   PRINT_SCREEN:FAR                ; INT 05H
                                EXTRN   TIMER_INT:FAR                   ;     08H
                                EXTRN   KB_INT:FAR                      ;     09H
                                EXTRN   DISK_INT:FAR                    ;     0EH
                                EXTRN   VIDEO_IO:NEAR                   ;     10H
                                EXTRN   EQUIPMENT:FAR                   ;     11H
                                EXTRN   MEMORY_SIZE_DETERMINE:FAR       ;     12H
                                EXTRN   DISKETTE_IO:FAR                 ;     13H
                                FXTRN   RS232 IO:FAR                    ;     14H
                                EXTRN   CASSETTE_IO:FAR                 ;     15H
                                EXTRN   KEYBOARD_IO:FAR                 ;     16H
                                EXTRN   PRINTER_IO:FAR                  ;     17H
                                EXTRN   BAS_ENT:FAR                     ;     18H
                                EXTRN   BOOT_STRAP:NEAR                 ;     19H
                                EXTRN   TIME_OF_DAY:FAR                 ;     1AH
                                EXTRN   VIDEO_PARMS:BYTE                ;     1DH
                                EXTRN   DISK_BASE:BYTE                  ;     1EH
                                EXTRN   KBDNMI:FAR                      ;     41H
                                EXTRN   KEY62_INT:FAR                   ;     48H
                                EXTRN   EXTAB:BYTE                      ;     49H
                                EXTRN   KKKFDM:FAR                      ;     78H
                                EXTRN   BUFFER_QUEING:FAR               ;     79H

0000                    POST    LABEL   FAR
= 0000                  BEGIN   EQU     $
                        ;--------------------------------------------------------------------
                        ;       RETURN POINTERS FOR RTNS CALLED BEFORE STACK INITIALIZED    :
                        ;--------------------------------------------------------------------
                        ;----- DISABLE MEMORY REGISTER DECODERS AND ENABLE VRAM1 DECODER
0000  01                Z0      DB      001H            ;  START REG ADDR
```

```
0001  60                        DB      060H            ;  MASK DATA
0002  0A                        DB      10              ;  PARAMETER LENGTH
0003  00                        DB      000H            ;  01 - CARTRIDGE ROM 2 (D0000-)
0004  00                        DB      000H            ;  02 - CARTRIDGE ROM 3 (D8000-)
0005  00                        DB      000H            ;  03 - CARTRIDGE ROM 4 (E0000-)
0006  00                        DB      000H            ;  04 - CARTRIDGE ROM 5 (E8000-)
0007  00                        DB      000H            ;  05 - CARTRIDGE ROM 6 (F0000-)
0008  00                        DB      000H            ;  06 - CARTRIDGE ROM 7 (F8000-)
0009  00                        DB      000H            ;  07 - KANJI ROM
000A  00                        DB      000H            ;  08 - PROGRAMMABLE RAM

000B  B7                        DB      0B7H            ;  09 - VRAM1 (B8000-BFFFF)
000C  00                        DB      000H            ;  0A - VRAM2
000D  0086 R                    DW      L0_1            ; RETURN ADDR
                        ;----- ENABLE BASE ROM DECODER
000F  00                        DB      000H            ;  START REG ADDR
0010  23                        DB      023H            ;  MASK DATA
0011  01                        DB      1               ;  PARAMETER LENGTH
0012  BC                        DB      0BCH            ;  00 - BASE ROM (E0000-FFFFF)
0013  0089 R                    DW      L0_2            ; RETURN ADDR
                        ;----- INITIALIZE I/O REGISTER DECODERS
0015  80                        DB      080H            ;  START REG ADDR
0016  00                        DB      000H            ;  MASK DATA
0017  14                        DB      20              ;  PARAMETER LENGTH
0018  80                        DB      080H            ;  80 - 8259 PIC CS       (20H-27H)
0019  80                        DB      080H            ;  81 - 8253 PIT CS       (40H-47H)
001A  80                        DB      080H            ;  82 - 8255 PPI CS       (60H-67H)
001B  80                        DB      080H            ;  83 - NMI CTRL CS       (A0H-A7H)
001C  80                        DB      080H            ;  84 - SOUND CHIP CS     (C0H-C7H)
001D  9E                        DB      09EH            ;  85 - UPD765 FDC CS     (F0H-F7H)
001E  80                        DB      080H            ;  86 - JOYSTICK READ  (200H-207H)
001F  80                        DB      080H            ;  87 - JOYSTICK WRITE (200H-207H)
0020  80                        DB      080H            ;  88 - PARALLEL PRINTR(378H-37FH)
0021  80                        DB      080H            ;  89 - 8250 COMM PORT (2F8H-2FFH)
0022  80                        DB      080H            ;  8A - 46505 CRTC     (3D0H-3D7H)
0023  00                        DB      000H            ;  8B - RESERVED
0024  80                        DB      080H            ;  8C - PCJR VGA             (3DAH)
0025  80                        DB      080H            ;  8D - NATIVE VGA           (3DAH)
0026  80                        DB      080H            ;  8E - EXT. VGA             (3DDH)
0027  80                        DB      080H            ;  8F - LIGHT PEN GATE (3DAH,3DEH)
0028  80                        DB      080H            ;  90 - CRT/CPU PAGE REG 2  (3D9H)
0029  80                        DB      080H            ;  91 - CRT/CPU PAGE REG    (3DFH)
002A  00                        DB      000H            ;  92 - MODEM CTRL REG (3F8H-3FFH)
002B  80                        DB      080H            ;  93 - EXTERNAL BUS CONTROL REG
002C  008C R                    DW      L0              ; RETURN ADDR
                        ;----- DISABLE SX-03 DECODER
002E  8E                Z_0     DB      08EH            ;  START REG ADDR
002F  00                        DB      000H            ;  MASK DATA
0030  01                        DB      1               ;  PARAMETER LENGTH
0031  00                        DB      000H            ;  8E - SX-03 EXT. VIDEO    (3DDH)
0032  0177 R                    DW      L11             ; RETURN ADDR
                        ;-------------------------------------------------------------------
0034  018A R            Z1      DW      L13             ; RETURN ADDR OF ROS CHECKSUM TEST
                        ;-------------------------------------------------------------------
                        ;----- ALLOCATE PROGRAMMABLE RAM
0036  08                        DB      008H            ;  START REG ADDR
0037  63                        DB      063H            ;  MASK DATA
0038  01                        DB      1               ;  PARAMETER LENGTH
0039  A0                        DB      0A0H            ;  (00000-1FFFF)
003A  01FD R                    DW      MCONF3          ; RETURN ADDR
003C  08                        DB      008H            ;  START REG ADDR
003D  63                        DB      063H            ;  MASK DATA
003E  01                        DB      1               ;  PARAMETER LENGTH
003F  A4                        DB      0A4H            ;  (20000-3FFFF)

0040  01FD R                    DW      MCONF3          ; RETURN ADDR
0042  08                        DB      008H            ;  START REG ADDR
0043  63                        DB      063H            ;  MASK DATA
0044  01                        DB      1               ;  PARAMETER LENGTH
0045  A8                        DB      0A8H            ;  (40000-5FFFF)
0046  01FD R                    DW      MCONF3          ; RETURN ADDR
0048  08                        DB      008H            ;  START REG ADDR
0049  63                        DB      063H            ;  MASK DATA
004A  01                        DB      1               ;  PARAMETER LENGTH
004B  AC                        DB      0ACH            ;  (60000-7FFFF)
004C  01FD R                    DW      MCONF3          ; RETURN ADDR
                        ;-------------------------------------------------------------------
004E  020E R            Z2      DW      L16             ; RETURN ADDR OF 2K RAM TEST
                        ;-------------------------------------------------------------------
0050  0DCE R            EX_0    DW      OFFSET  EB0     ; RETURN ADDR OF E_MSG
0052  0DCE R                    DW      OFFSET  EB0     ;          "
0054  0DA7 R                    DW      OFFSET  TOTLTPO ;          "
                        ;-----------------------------------
                        ;       MESSAGE AREA FOR POST       ;
                        ;-----------------------------------
0056  20 4B 42          F3B             DB      ' KB'   ; MEMORY SIZE PROMPT
0059  45 52 52 4F 52    ERROR_ERR       DB      'ERROR' ; GENERAL ERROR PROMPT
005E  41                MEM_ERR         DB      'A'     ; MEMORY ERROR MESSAGE
005F  42                KEY_ERR         DB      'B'     ; KEYBOARD ERROR MESSAGE
0060  43                CASS_ERR        DB      'C'     ; CASSETTE ERROR MESSAGE
0061  44                COM1_ERR        DB      'D'     ; SERIAL PORT ERROR MESSAGE (2FXH)
0062  45                COM2_ERR        DB      'E'     ; SERIAL PORT ERROR MESSAGE (3FXH)
0063  46                ROM_ERR         DB      'F'     ; OPTIONAL GENERIC BIOS ROM ERROR
0064  47                CART_ERR        DB      'G'     ; CARTRIDGE ERROR MESSAGE
0065  48                DISK_ERR        DB      'H'     ; DISKETTE ERROR MESSAGE
0066  4A                PRT_ERR         DB      'J'     ; PARARELL PRINTER ERROR MESSAGE
0067  4B                KFNT_ERR        DB      'K'     ; KANJI-FONT ROM ERROR MESSAGE
0068  4C                INVC_ERR        DB      'L'     ; INVALID COMBINATION OF CARTRIDGE
                        ;
0069                    IMASKS  LABEL   BYTE            ; INTERRUPT MASKS FOR 8259
                                                        ; INTERRUPT CONTROLLER
0069  EF                        DB      0EFH            ; MODE4             ; RS232C INTR MASK
006A  F7                        DB      0F7H            ; RS232C INTR MASK
```

```
                        ;=============================================================
                        ;       POST ENTRY POINT                                      :
                        ;=============================================================
006B                    RESET   LABEL   FAR
006B                    START:
                        ;-------------------------------------------------------------
                        ; SETUP                                                       :
                        ;       DISABLE NMI & MASKABLE INTS, CLEAR MEMORY MAPPING,    :
                        ;       ENABLE BASE ROM, AND SET UP I/O MAPPING               :
                        ;-------------------------------------------------------------
006B  B0 00                     MOV     AL,0            ; DISABLE NMI
006D  E6 A0                     OUT     NMI_PORT,AL     ;
006F  E4 A0                     IN      AL,NMI_PORT     ; RESET NMI F/F
0071  FA                        CLI                     ; DISABLES MASKABLE INTERRUPTS

0072  B0 FF                     MOV     AL,0FFH         ; SEND 'FF' TO MFG_TESTER
0074  E6 10                     OUT     MFG_PORT,AL     ;
0076  B0 00                     MOV     AL,0            ; CLEAR MFG_PORT 11 & 12
0078  E6 11                     OUT     MFG_PORT+1,AL   ;
007A  E6 12                     OUT     MFG_PORT+2,AL   ;

007C  8C C8                     MOV     AX,CS           ; SET UP STACK SEGMENT & POINTER
007E  8E D0                     MOV     SS,AX           ;
0080  BC 0000 R                 MOV     SP,OFFSET Z0    ;
0083  E9 04CA R                 JMP     S8SETJ          ; CLEAR MEMORY MAPPING
0086  E9 04CA R         L0_1:   JMP     S8SETJ          ; ENABLE BASE ROM DECODER
0089  E9 04CA R         L0_2:   JMP     S8SETJ          ; SET UP I/O REG DECODER
                        ;-------------------------------------------------------------
                        ; SETUP                                                       :
                        ;       DISABLE NMI, MASKABLE INTS, SOUND CHIP, AND VIDEO.    :
                        ;       TURN DRIVE 0 MOTOR OFF.                               :
                        ;-------------------------------------------------------------
008C  B0 00             L0:     MOV     AL,0            ; DISABLE NMI
008E  E6 A0                     OUT     NMI_PORT,AL     ;
0090  E4 A0                     IN      AL,NMI_PORT     ; RESET NMI F/F
                                                        ; DISABLE ATTENUATION IN SOUND CHIP
0092  B8 207F                   MOV     AX,207FH        ; REG ADDRESS IN AH, ATTENUATOR OFF
                                                        ; IN AL
0095  B9 0004                   MOV     CX,4            ; 4 ATTENABLE
0098  02 C4             L1:     ADD     AL,AH           ; COMBINE REG ADDRESS AND DATA
009A  E6 C0                     OUT     SND_CTL,AL      ;                                       $USEK$LIB
009C  E2 FA                     LOOP    L1

009E  B0 A0                     MOV     AL,WD_ENABLE+FDC_RESET  ; TURN DRIVE 0 MOTOR OFF,
00A0  E6 F2                     OUT     NEC_CTL,AL      ; ENABLE TIMER
00A2  BA 03DA                   MOV     DX,VGA_CTL      ; VIDEO GATE ARRAY CONTROL
00A5  EC                        IN      AL,DX           ; SYNC VGA TO ACCEPT REG
00A6  B0 04                     MOV     AL,4            ; SET VGA RESET REG
00A8  EE                        OUT     DX,AL           ; SELECT IT
00A9  B0 02                     MOV     AL,2            ; SET SYNC RESET
00AB  EE                        OUT     DX,AL           ; RESET VIDEO GATE ARRAY
                        ;-------------------------------------------------------------
                        ; TEST 1                                                      :
                        ;       8088 PROCESSOR TEST                                   :
                        ; DESCRIPTION                                                 :
                        ;       VERIFY 8088 FLAGS, REGISTERS, AND CONDITIONAL JUMPS   :
                        ; MFG ERROR CODE = 0001                                       :
                        ;-------------------------------------------------------------
00AC  B4 D5                     MOV     AH,0D5H         ; SET SF, CF, ZF, AND AF FLAGS ON
00AE  9E                        SAHF
00AF  73 4C                     JNC     L4              ; GO TO ERR ROUTINE IF CF NOT SET
00B1  75 4A                     JNZ     L4              ; GO TO ERR ROUTINE IF ZF NOT SET
00B3  7B 48                     JNP     L4              ; GO TO ERR ROUTINE IF PF NOT SET
00B5  79 46                     JNS     L4              ; GO TO ERR ROUTINE IF SF NOT SET
00B7  9F                        LAHF                    ; LOAD FLAG IMAGE TO AH
00B8  B1 05                     MOV     CL,5            ; LOAD CNT REG WITH SHIFT CNT
00BA  D2 EC                     SHR     AH,CL           ; SHIFT AF INTO CARRY BIT POS

00BC  73 3F                     JNC     L4              ; GO TO ERR ROUTINE IF AF NOT SET
00BE  B0 40                     MOV     AL,40H          ; SET THE OF FLAG ON
00C0  D0 E0                     SHL     AL,1            ; SETUP FOR TESTING
00C2  71 39                     JNO     L4              ; GO TO ERR ROUTINE IF OF NOT SET
00C4  32 E4                     XOR     AH,AH           ; SET AH = 0
00C6  9E                        SAHF                    ; CLEAR SF, CF, ZF, AND PF
00C7  76 34                     JBE     L4              ; GO TO ERR ROUTINE IF CF ON
                                                        ; GO TO ERR ROUTINE IF ZF ON
00C9  78 32                     JS      L4              ; GO TO ERR ROUTINE IF SF ON
00CB  7A 30                     JP      L4              ; GO TO ERR ROUTINE IF PF ON
00CD  9F                        LAHF                    ; LOAD FLAG IMAGE TO AH
00CE  B1 05                     MOV     CL,5            ; LOAD CNT REG WITH SHIFT CNT
00D0  D2 EC                     SHR     AH,CL           ; SHIFT 'AF' INTO CARRY BIT POS
00D2  72 29                     JC      L4              ; GO TO ERR ROUTINE IF ON
00D4  D0 E4                     SHL     AH,1            ; CHECK THAT 'OF' IS CLEAR
00D6  70 25                     JO      L4              ; GO TO ERR ROUTINE IF ON
                        ;----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
                        ;       WITH ALL ONE'S AND ZEROES'S.
00D8  B8 FFFF                   MOV     AX,0FFFFH       ; SETUP ONE'S PATTERN IN AX
00DB  F9                        STC
00DC  8E D8             L2:     MOV     DS,AX           ; WRITE PATTERN TO ALL REGS
00DE  8C DB                     MOV     BX,DS
00E0  8E C3                     MOV     ES,BX
00E2  8C C1                     MOV     CX,ES
00E4  8E D1                     MOV     SS,CX
00E6  8C D2                     MOV     DX,SS
00E8  8B E2                     MOV     SP,DX
00EA  8B EC                     MOV     BP,SP
00EC  8B F5                     MOV     SI,BP
00EE  8B FE                     MOV     DI,SI
00F0  73 07                     JNC     L3
00F2  33 C7                     XOR     AX,DI           ; PATTERN MAKE IT THRU ALL REGS
00F4  75 07                     JNZ     L4              ; NO -GOTO ERR ROUTINE
00F6  F8                        CLC
```

```
00F7  EB E3                     JMP     L2
00F9  0B C7             L3:     OR      AX,DI           ; ZERO PATTERN MAKE IT THRU?
00FB  74 09                     JZ      L5              ; YES -GOTO NEXT TEST

00FD  B0 00             L4:     MOV     AL,0            ; MFG ERROR CODE = 0001
00FF  E6 11                     OUT     MFG_PORT+1,AL   ;
0101  B0 01                     MOV     AL,1            ;
0103  E6 12                     OUT     MFG_PORT+2,AL   ;

0105  F4                        HLT                     ; HALT
0106                    L5:
                        ;--------------------------------------------------------
                        ; TEST 2                                                 :
                        ;       8255 INITIALIZATION AND TEST                     :
                        ; DESCRIPTION                                            :
                        ;       FIRST INITIALIZE 8255 PROG. PERIPHERAL INTERFACE.:
                        ;       PORTS A&B ARE LATCHED OUTPUT BUFFERS. C IS INPUT.:
                        ; MFG ERROR CODE = 0002                                   :
                        ;--------------------------------------------------------

0106  B0 FE                     MOV     AL,0FEH         ; SEND 'FE' TO MFG_TESTER
0108  E6 10                     OUT     MFG_PORT,AL     ;
010A  B0 89                     MOV     AL,MODE_8255
010C  E6 63                     OUT     CMD_PORT,AL     ; CONFIGURES I/O PORTS
010E  2B C0                     SUB     AX,AX           ; TEST PATTERN SEED = 0000
0110  8A C4             L6:     MOV     AL,AH
0112  E6 60                     OUT     PORT_A,AL       ; WRITE PATTERN TO PORT A
0114  E4 60                     IN      AL,PORT_A       ; READ PATTERN FROM PORT A
0116  E6 61                     OUT     PORT_B,AL       ; WRITE PATTERN TO PORT B
0118  E4 61                     IN      AL,PORT_B       ; READ OUTPUT PORT
011A  3A C4                     CMP     AL,AH           ; DATA AS EXPECTED?
011C  75 06                     JNE     L7              ; IF NOT, SOMETHING IS WRONG
011E  FE C4                     INC     AH              ; MAKE NEW DATA PATTERN
0120  75 EE                     JNZ     L6              ; LOOP TILL 255 PATTERNS DONE
0122  EB 05                     JMP     SHORT L8        ; CONTINUE IF DONE
0124  B3 02             L7:     MOV     BL,02H          ; SET ERROR FLAG (BH=00 NOW)
0126  E9 0D32 R                 JMP     E_MSG           ; GO ERROR ROUTINE

0129  BA 03DF           L8:     MOV     DX,PAGREG       ;
012C  B0 1B                     MOV     AL,1BH          ;  PAGE 3
012E  EE                        OUT     DX,AL           ;

012F  B0 0D                     MOV     AL,00001101B    ; INITIALIZE OUTPUT PORTS
0131  E6 61                     OUT     PORT_B,AL
                        ;--------------------------------------------------------
                        ; PART 3                                                 :
                        ;       SET UP 46505 AND VIDEO GATE ARRAY TO GET MEMORY WORKING :
                        ;--------------------------------------------------------

0133  B0 FD                     MOV     AL,0FDH         ; SEND 'FD' TO MFG_TESTER
0135  E6 10                     OUT     MFG_PORT,AL     ;
0137  BB 0000 E                 MOV     BX,OFFSET VIDEO_PARMS ; POINT TO 46505 PARMS
013A  B9 0010                   MOV     CX,16           ; SET PARM LEN
013D  32 E4                     XOR     AH,AH           ; AH IS REG 0
013F  8A C4             L9:     MOV     AL,AH           ; GET 46505 REG 0
0141  BA 03D4                   MOV     DX,CRT_CTL      ; SET ADDRESS OF 46505
0144  EE                        OUT     DX,AL
0145  FE C4                     INC     AH              ; NEXT REG VALUE
0147  42                        INC     DX              ; POINT TO DATA PORT
0148  2E: 8A 07                 MOV     AL,CS:[BX]      ; GET TABLE VALUE
014B  EE                        OUT     DX,AL           ; OUT TO CHIP
014C  43                        INC     BX              ; NEXT IN TABLE
014D  E2 F0                     LOOP    L9              ;
                        ;----- START VGA WITHOUT VIDEO ENABLED
014F  BA 03DA                   MOV     DX,VGA_CTL      ; SET ADDRESS OF VGA
0152  EC                        IN      AL,DX           ; BE SURE ADDR/DATA FLAG IS
                                                        ; IN THE PROPER STATE
0153  B1 05                     MOV     CL,5            ; # OF REGISTERS
0155  32 E4                     XOR     AH,AH           ; AH IS REG COUNTER
0157  8A C4             L10:    MOV     AL,AH           ; GET REG #
0159  EE                        OUT     DX,AL           ; SELECT IT
015A  32 C0                     XOR     AL,AL           ; SET ZERO FOR DATA
015C  EE                        OUT     DX,AL
015D  FE C4                     INC     AH              ; NEXT REG
015F  E2 F6                     LOOP    L10             ;

0161  BA 03DD                   MOV     DX,VGA_CTL_E    ;
0164  EC                        IN      AL,DX           ;
0165  B0 04                     MOV     AL,4            ; ENABLE EXT. VIDEO CLOCK
0167  EE                        OUT     DX,AL           ;
0168  EE                        OUT     DX,AL           ;
0169  B0 01                     MOV     AL,1            ; DISABLE EXT. VIDEO PROCESSOR
016B  EE                        OUT     DX,AL           ;
016C  EE                        OUT     DX,AL           ;

016D  8C C8                     MOV     AX,CS           ; SET UP STACK SEGMENT & POINTER
016F  8E D0                     MOV     SS,AX           ;
0171  BC 002E R                 MOV     SP,OFFSET Z_0   ;
0174  E9 04CA R                 JMP     S8SETJ          ; DISABLE SX-03 DECODER
☎77                     L11:
                        ;--------------------------------------------------------
                        ; TEST 4                                                 :
                        ;       PLANAR BOARD ROS CHECKSUM TEST                   :
                        ; DESCRIPTION                                            :
                        ;       A CHECKSUM TEST IS DONE FOR EACH ROS MODULES     :
                        ;       ON THE PLANAR BOARD.                             :
                        ; MFG ERROR CODE = 0007  MODULE AT ADDRESS E0000H ERROR   :
                        ;                  0008  MODULE AT ADDRESS E8000H ERROR   :
                        ;                  0003  MODULE AT ADDRESS F0000H ERROR   :
                        ;                  0004  MODULE AT ADDRESS F8000H ERROR   :
                        ;--------------------------------------------------------

0177  B0 FC                     MOV     AL,0FCH         ; SEND 'FC' TO MFG_TESTER
0179  E6 10                     OUT     MFG_PORT,AL     ;
017B  BA E000                   MOV     DX,0E000H       ; SET DS TO E000H
017E  8E DA             L12:    MOV     DS,DX           ;
```

```
0180  33 F6                    XOR     SI,SI           ; SET OFFSET
0182  BC 0034 R                MOV     SP,OFFSET Z1    ; SET UP STACK POINTER
0185  B9 8000                  MOV     CX,8000H        ; NUMBER OF BYTES TO BE TESTED, 32K
0188  EB 23                    JMP     SHORT ROS_CHECKSUM ; 32K ROM OK ?
018A  74 1A             L13:   JZ      L14             ; YES -
018C  BB 0003                  MOV     BX,0003H        ; SET ERROR CODE (0003)
018F  80 FE F0                 CMP     DH,0F0H         ;
0192  74 0F                    JE      L_E             ;
0194  43                       INC     BX              ; (0004)
0195  80 FE F8                 CMP     DH,0F8H         ;
0198  74 09                    JE      L_E             ;
019A  43                       INC     BX              ;
019B  43                       INC     BX              ;
019C  43                       INC     BX              ; (0007)
019D  80 FE E0                 CMP     DH,0E0H         ;
01A0  74 01                    JE      L_E             ;
01A2  43                       INC     BX              ; (0008)
01A3  E9 0D32 R         L_E:   JMP     E_MSG           ; INDICATE ERROR
01A6  80 C6 08          L14:   ADD     DH,08H          ; END OF ROM SPACE ?
01A9  75 D3                    JNZ     L12             ; NO -
01AB  EB 0A                    JMP     SHORT L15       ; YES- GOTO NEXT TEST
                        ;--------------------------------------------------------
                        ; SUBROUTINE
                        ;       ARITHMETIC CHECKSUM SUBROUTINE
                        ; ENTRY:
                        ;        DS = DATA SEGMENT OF ROM SPACE TO BE CHECKED
                        ;        SI = INDEX OFFSET INTO DS POINTING TO 1ST BYTE
                        ;        CX = LENGTH OF SPACE TO BE CHECKED
                        ; EXIT: ZERO FLAG OFF=ERROR, ON= SPACE CHECKED OK
                        ;--------------------------------------------------------
01AD                    ROS_CHECKSUM    PROC    NEAR
01AD  B0 00                    MOV     AL,0
01AF  02 04             RC_0:  ADD     AL,DS:[SI]
01B1  46                       INC     SI
01B2  E2 FB                    LOOP    RC_0
01B4  0A C0                    OR      AL,AL
01B6  C3                       RET

01B7                    ROS_CHECKSUM    ENDP

01B7                    L15:
                        ;--------------------------------------------------------
                        ; PART 5                                                 :
                        ;       RAM MAPPING (SAVED IN PORT_A DURING POST)        :
                        ;           BIT  7   : SET IN PART 7                     :
                        ;           BIT  6   : SET IN TEST 13 & 15               :
                        ;           BIT  5-4 : RESERVED                          :
                        ;           BIT  3   : 64K RAM CARD INSTALLED            :
                        ;           BIT  2   : RESERVED                          :
                        ;           BIT  1-0 : NUMBER OF EXP 128K RAM CARD       :
                        ;                      00  NO CARD INSTALLED             :
                        ;                      01  1 CARD  INSTALLED (128K - 20000) :
                        ;                      10  2 CARDS INSTALLED (256K - 40000) :
                        ;                      11  3 CARDS INSTALLED (384K - 60000) :
                        ;--------------------------------------------------------
01B7  B0 FB                    MOV     AL,0FBH         ; SEND 'FB' TO MFG_TESTER
01B9  E6 10                    OUT     MFG_PORT,AL     ;
01BB  33 DB                    XOR     BX,BX           ; INIT 128K CARD COUNTER
01BD  8E DB                    MOV     DS,BX           ; SET FIRST SEGMENT TO BE TESTED
01BF  33 F6                    XOR     SI,SI           ;
01C1  E4 62                    IN      AL,PORT_C       ; GET CONFIG BITS
01C3  24 08                    AND     AL,00001000B    ; 64K CARD IN ?
01C5  75 02                    JNZ     MCONF0          ;
01C7  B3 08                    MOV     BL,00001000B    ; SET 64K CARD FLAG BIT
01C9  B8 5A5A           MCONF0: MOV     AX,5A5AH        ; SET WRITE DATA
01CC  89 04                    MOV     DS:[SI],AX      ; WRITE DATA TO TOP OF 128K BOUNDRY
01CE  EB 00                    JMP     $+2             ;  DELAY
01D0  EB 00                    JMP     $+2             ;
01D2  8B 14                    MOV     DX,DS:[SI]      ; READ BACK
01D4  3B C2                    CMP     AX,DX           ; 5A5A ?
01D6  74 0E                    JE      MCONF1          ; YES-
01D8  F7 D0                    NOT     AX              ;
01DA  89 04                    MOV     DS:[SI],AX      ; WRITE DATA TO TOP OF 128K BOUNDRY
01DC  EB 00                    JMP     $+2             ;  DELAY
01DE  EB 00                    JMP     $+2             ;
01E0  8B 14                    MOV     DX,DS:[SI]      ; READ BACK
01E2  3B C2                    CMP     AX,DX           ; A5A5 ?
01E4  75 10                    JNE     MCONF2          ; NO -GOTO MAIN RAM MAPPING
01E6  43                MCONF1: INC     BX              ; YES-INC 128K CARD COUNT
01E7  83 C4 06                 ADD     SP,6            ; INC TABLE POINTER
01EA  8C D8                    MOV     AX,DS           ;
01EC  80 C4 20                 ADD     AH,20H          ; SET SEGMENT OF NEXT 128K
01EF  8E D8                    MOV     DS,AX           ;
01F1  80 FC 60                 CMP     AH,60H          ; 512K EXCEEDED ?
01F4  72 D3                    JB      MCONF0          ; NO -CHECK NEXT 128K
01F6  8B C3             MCONF2: MOV     AX,BX           ; SAVE # OF 128K CARD IN PORT_A
01F8  E6 60                    OUT     PORT_A,AL       ;

01FA  E9 04CA R                JMP     S8SETJ          ; SET RAM MAPPING
01FD                    MCONF3:
                        ;--------------------------------------------------------
                        ; TEST 6                                                 :
                        ;       BASE 8K READ/WRITE STORAGE TEST                  :     TEDFT122

                        ; DESCRIPTION                                            :
                        ;       WRITE/READ/VERIFY DATA PATTERNS AA, 55, FF AND 00 TO  :
                        ;       1ST 8K OF STORAGE AND THE 2K OF VIDEO RAM (CRT BUFFER). :
                        ;       VERIFY STORAGE ADDRESSABILITY.                   :
                        ;       ON EXIT SET CRT PAGE TO 15. SET TEMPORARY STACK ALSO. :
                        ; MFG ERROR CODE = 03XX FOR 128K RAM CARD                :
                        ;                  04XX FOR BASE 64K RAM                 :
                        ;                  05XX FOR 64K RAM CARD                 :
                        ;                  06XX FOR BOTH 64K RAM                 :
```

```
                        ;                (XX= ERROR BITS - OR'ED HIGH & LOW BYTES)    :
                        ;                0005 FOR BASE 64K RAM (B8000 ADDR PATH)      :
                        ;                0006 FOR BASE 32K VIDEO RAM (B8000 ADDR PATH):
                        ;-------------------------------------------------------------------
01FD  B0 FA                     MOV     AL,0FAH         ; SEND 'FA' TO MFG_TESTER
01FF  E6 10                     OUT     MFG_PORT,AL     ;
0201  BC 004E R                 MOV     SP,OFFSET Z2    ; SET RETURN ADDR
0204  B9 1000                   MOV     CX,1000H        ; SET FOR 4K WORDS (8K BYTES)
0207  33 C0                     XOR     AX,AX
0209  8E C0                     MOV     ES,AX           ; LOAD ES
020B  E9 0DEA R                 JMP     PODSTG          ; TEST 1ST 4K RAM
020E  74 2B             L16:    JZ      L23             ; TEST OK
0210  B7 03                     MOV     BH,03H          ; ERROR 03....
0212  E4 60                     IN      AL,PORT_A       ; LOAD CONFIG FLAG
0214  A8 03                     TEST    AL,00000011B    ; 128K CARD INSTALLED ?
0216  75 08                     JNZ     L20             ; YES-
0218  B7 04             L17:    MOV     BH,04H          ; ERROR 04....
021A  E4 60                     IN      AL,PORT_A       ;
021C  A8 08                     TEST    AL,00001000B    ; 64K CARD INSTALLED ?
021E  75 06                     JNZ     L21             ; YES-WORRY ABOUT ODD/EVEN
0220  8A D9             L20:    MOV     BL,CL           ; NO -COMBINE ERR BITS
0222  0A DD                     OR      BL,CH
0224  EB 12                     JMP     SHORT L22       ;
0226  80 FC 02          L21:    CMP     AH,02           ; EVEN BYTE ERROR? ERR 04XX
0229  8A D9                     MOV     BL,CL
022B  74 0B                     JE      L22
022D  FE C7                     INC     BH              ; MAKE INTO 05XX ERR
022F  0A DD                     OR      BL,CH           ; MOVE AND POSSIBLY COMBINE
                                                        ; ERROR BITS
0231  80 FC 01                  CMP     AH,1            ; ODD BYTE ERROR
0234  74 02                     JE      L22
0236  FE C7                     INC     BH              ; MUST HAVE BEEN BOTH
                                                        ; - MAKE INTO 06XX
0238  E9 0D32 R         L22:    JMP     E_MSG           ; JUMP TO ERROR OUTPUT ROUTINE
                        ;----- SETUP TEMPORARY STACK SEG AND SP
023B  B8 0030           L23:    MOV     AX,0030H        ; GET STACK VALUE
023E  8E D0                     MOV     SS,AX           ; SET THE STACK UP
0240  BC 0100 R                 MOV     SP,OFFSET TOS   ; STACK IS READY TO GO
0243  33 C0                     XOR     AX,AX           ; SET UP DATA SEG
0245  8E D8                     MOV     DS,AX
                        ;----- TEST FIRST 2K OF 64K VRAM
0247  E4 60                     IN      AL,PORT_A       ;
0249  A8 03                     TEST    AL,00000011B    ; 128K RAM CARD ?

024B  74 19                     JZ      L24             ; NO -BYPASS TESTING
024D  B9 4000                   MOV     CX,4000H        ;
0250  A8 01                     TEST    AL,00000001B    ;
0252  74 08                     JZ      L23_1           ;
0254  B5 20                     MOV     CH,20H          ;
0256  A8 02                     TEST    AL,00000010B    ;
0258  74 02                     JZ      L23_1           ;
025A  B5 60                     MOV     CH,60H          ;
025C  8E C1             L23_1:  MOV     ES,CX           ;
025E  B9 0400                   MOV     CX,0400H        ; 2K BYTES
0261  E8 0DEA R                 CALL    PODSTG          ;
0264  75 B2                     JNZ     L17             ;
                        ;----- TEST BOTTOM 2K OF PAGE 3 IN 64K VRAM USING B8000 ADDR PATH
0266  B0 F9             L24:    MOV     AL,0F9H         ; SEND 'F9' TO MFG_TESTER
0268  E6 10                     OUT     MFG_PORT,AL     ;
026A  B9 0400                   MOV     CX,0400H        ; 2K BYTES
026D  B8 B880                   MOV     AX,0B880H       ; POINT TO AREA JUST TESTED WITH
0270  8E C0                     MOV     ES,AX           ; DIRECT ADDRESSING
0272  E8 0DEA R                 CALL    PODSTG          ;
0275  74 06                     JZ      L25             ;
0277  BB 0005                   MOV     BX,0005H        ; ERROR 0005
027A  E9 0D32 R                 JMP     E_MSG           ;
                        ;----- DISABLE VRAM1 DECODER & ENABLE VRAM2 DECODER
027D  E8 04C2 R         L25:    CALL    S8SET           ;
0280  09                        DB      009H            ;  START REG ADDR
0281  60                        DB      060H            ;  MASK DATA
0282  02                        DB      2               ;  PARAMETER LENGTH
0283  00                        DB      000H            ;  DISABLE VRAM1 DECODER
0284  B7                        DB      0B7H            ;  ENABLE VRAM2 DECODER

0285  BA 03D9                   MOV     DX,PAGREG2      ;  SET PAGE REG 8
0288  B0 00                     MOV     AL,0            ;
028A  EE                        OUT     DX,AL           ;
                        ;----- TEST BOTTOM 2K OF PAGE 8 IN 32K VRAM USING B8000 ADDR PATH
028B  B9 0400                   MOV     CX,0400H        ; 2K BYTES
028E  B8 B880                   MOV     AX,0B880H       ; POINT TO AREA JUST TESTED WITH
0291  8E C0                     MOV     ES,AX           ; DIRECT ADDRESSING
0293  E8 0DEA R                 CALL    PODSTG          ;
0296  74 06                     JZ      L26             ;
0298  BB 0006                   MOV     BX,0006H        ; ERROR 0006
029B  E9 0D32 R                 JMP     E_MSG
                        ;-------------------------------------------------------------------
                        ; PART 7                                                          :
                        ;       ROM CARTRIDGE CONFIGURATION CHECK                        :
                        ; DESCRIPTION                                                    :
                        ;       THIS ROUTINE CHECKS ROM CARTRIDGE ADDRESS CONFIGURATIONS. :
                        ;       IF ROM CARTRIDGE OVERLAPS SYSTEM ROM AREA, SYSTEM ROM IS  :
                        ;       DISABLED AND THEN CARTRIDGE ROM IS ENABLED.               :
                        ;                                                                 :
                        ; BASE ROM OVERLAPPED FLAG (SAVED IN PORT A)                      :
                        ;           BIT  7   : SYSTEM MODE FLAG                           :
                        ;                      0   NATIVE MODE                            :
                        ;                      1   PCJR MODE                              :
                        ;                                                                 :
                        ;           BIT  6   : SET IN TEST 13 & 15                        :
                        ;           BIT  5-4 : RESERVED                                   :
                        ;           BIT  3   : SET IN PART 5                              :
                        ;           BIT  2   : RESERVED                                   :
```

```
                        ;          BIT  1-0 : SET IN PART 5                                    :
                        ;--------------------------------------------------------------------
029E  B0 F8             L26:    MOV     AL,0F8H         ; SEND 'F8' TO MFG_TES
02A0  E6 10                     OUT     MFG_PORT,AL     ;
                        ;----- DISABLE VRAM2 DECODER
02A2  E8 04C2 R                 CALL    S8SET           ;
02A5  0A                        DB      00AH            ;  START REG ADDR
02A6  00                        DB      000H            ;  MASK DATA
02A7  01                        DB      1               ;  PARAMETER LENGTH
02A8  00                        DB      000H            ;
                        ;----- ALOCATE CART. ROMS TO ADDR 90000H-BFFFFH TEMPORARILY
02A9  E8 04C2 R                 CALL    S8SET           ;
02AC  01                        DB      001H            ; START REG ADDR
02AD  20                        DB      020H            ; MASK DATA
02AE  06                        DB      6               ; PARAMETER LENGTH
02AF  B2                        DB      0B2H            ; CART. ROM 2 (90000-97FFF)
02B0  B3                        DB      0B3H            ; CART. ROM 3 (98000-9FFFF)
02B1  B4                        DB      0B4H            ; CART. ROM 4 (A0000-A7FFF)
02B2  B5                        DB      0B5H            ; CART. ROM 5 (A8000-AFFFF)
02B3  B6                        DB      0B6H            ; CART. ROM 6 (B0000-B7FFF)
02B4  B7                        DB      0B7H            ; CART. ROM 7 (B8000-BFFFF)

02B5  BA 01FF                   MOV     DX,S8STATUS     ;
02B8  EC                        IN      AL,DX           ; READ STATUS
02B9  A8 20                     TEST    AL,EROM7IN      ; F8000-FFFFF OVERLAPPED BY CART.?
02BB  75 1A                     JNZ     RCONF2          ; NO -
02BD  B9 B000                   MOV     CX,0B000H       ; GET SYSTEM ID FROM FFFFE
02C0  8E D9                     MOV     DS,CX           ;
02C2  BE FFFE                   MOV     SI,0FFFEH       ;
02C5  8A 24                     MOV     AH,DS:[SI]      ;
02C7  80 FC FD                  CMP     AH,PJSYSID      ; PCJR SYSTEM CART.?
02CA  74 03                     JE      RCONF1          ; YES-
02CC  E9 047D R                 JMP     SYSSWP8         ; NO-GO TO SYSTEM SWAP (NATIVE MODE)
02CF  50                RCONF1: PUSH    AX              ;
02D0  E4 60                     IN      AL,PORT_A       ; SET PCJR MODE FLAG
02D2  0C 80                     OR      AL,10000000B    ;
02D4  E6 60                     OUT     PORT_A,AL       ;
02D6  58                        POP     AX              ;
02D7  A8 40             RCONF2: TEST    AL,EROM6IN      ; F0000-F7FFF OVERLAPPED BY CART.?
02D9  74 34                     JZ      RCONF12         ; YES-

02DB  BA 9000                   MOV     DX,9000H        ; SET TEMPORARY CART. ROM TOP ADDR
02DE  33 F6                     XOR     SI,SI       FFSET
02E0  8E DA             RCONF3: MOV     DS,DX           ;
02E2  8B 04                     MOV     AX,DS:[SI]      ; READ ROM CARTRIDGE ID
02E4  50                        PUSH    AX              ; BUS SETTLING
02E5  58                        POP     AX              ;
02E6  3D AA55                   CMP     AX,0AA55H       ; ID FOUND (PCJR MODE) ?
02E9  74 11                     JE      RCONF7          ; YES-

02EB  3D 55AA                   CMP     AX,55AAH        ; (NATIVE MODE)
02EE  74 0C                     JE      RCONF7          ; YES-
02F0  81 C2 0080                ADD     DX,0080H        ; POINT TO NEXT 2K ADDR
02F4  81 FA B000                CMP     DX,0B000H       ; ADDR F0000 ?
02F8  72 E6                     JB      RCONF3          ; NO -GO CHECK NEXT AREA
02FA  EB 37                     JMP     SHORT RCONF14   ; YES-AREA E0000-FFFFF IS NOT
                                                        ;     OVERLAPPED BY CARTRIDGE
02FC  8A 64 02          RCONF7: MOV     AH,DS:[SI+2]    ; SET CART. ROM LENGTH
02FF  32 C0                     XOR     AL,AL           ;
0301  B1 03                     MOV     CL,3            ;
0303  D3 E8                     SHR     AX,CL           ;
0305  03 D0                     ADD     DX,AX           ; CALCULATE CART. ROM END ADDR+1
0307  81 FA A000                CMP     DX,0A000H       ; CART. ROM END ADDR+1 <= E0000 ?
030B  76 D3                     JBE     RCONF3          ; YES-
030D  EB 0F                     JMP     SHORT RCONF13   ; NO -AREA E0000-EFFFF IS
                                                        ;     OVERLAPPED BY CARTRIDGE
                        ;----- BASE ROM ACTIVE (F8000-FFFFF)
030F  E8 04C2 R         RCONF12:CALL    S8SET           ;
0312  00                        DB      000H            ; START REG ADDR
0313  60                        DB      060H            ; MASK DATA
0314  07                        DB      7               ; PARAMETER LENGTH
0315  BF                        DB      0BFH            ; DISABLE BASE ROM    (E0000-F7FFF)
0316  BA                        DB      0BAH            ; ENABLE  CART. ROM 2 (D0000-D7FFF)
0317  BB                        DB      0BBH            ; ENABLE  CART. ROM 3 (D8000-DFFFF)
0318  BC                        DB      0BCH            ; ENABLE  CART. ROM 4 (E0000-E7FFF)
0319  BD                        DB      0BDH            ; ENABLE  CART. ROM 5 (E8000-EFFFF)
031A  BE                        DB      0BEH            ; ENABLE  CART. ROM 6 (F0000-F7FFF)
031B  00                        DB      000H            ; DISABLE CART. ROM 7 (F8000-FFFFF)
031C  EB 21                     JMP     SHORT RCONF15   ;
                        ;----- BASE ROM ACTIVE (F0000-FFFFF)
031E  E8 04C2 R         RCONF13:CALL    S8SET           ;
0321  00                        DB      000H            ; START REG ADDR
0322  61                        DB      061H            ; MASK DATA
0323  01                        DB      1               ; PARAMETER LENGTH
0324  BE                        DB      0BEH            ; BASE ROM (E0000-EFFFF) DISABLE
0325  E8 04C2 R                 CALL    S8SET           ;                                    -
0328  01                        DB      001H            ; START REG ADDR
0329  60                        DB      060H            ; MASK DATA                          FFFFFFFFFFF
032A  06                        DB      6               ; PARAMETER LENGTH
032B  BA                        DB      0BAH            ; ENABLE  CART. ROM 2 (D0000-D7FFF)
032C  BB                        DB      0BBH            ; ENABLE  CART. ROM 3 (D8000-DFFFF)
032D  BC                        DB      0BCH            ; ENABLE  CART. ROM 4 (E0000-E7FFF)
032E  BD                        DB      0BDH            ; ENABLE  CART. ROM 5 (E8000-EFFFF)
032F  00                        DB      000H            ; DISABLE CART. ROM 6 (F0000-F7FFF)
0330  00                        DB      000H            ; DISABLE CART. ROM 7 (F8000-FFFFF)
0331  EB 0C                     JMP     SHORT RCONF15   ;
                        ;----- BASE ROM ACTIVE (E0000-FFFFF)
0333  E8 04C2 R         RCONF14:CALL    S8SET           ;
0336  01                        DB      001H            ; START REG ADDR
0337  60                        DB      060H            ; MASK DATA
0338  06                        DB      6               ; PARAMETER LENGTH
0339  BA                        DB      0BAH            ; ENABLE  CART. ROM 2 (D0000-D7FFF)
033A  BB                        DB      0BBH            ; ENABLE  CART. ROM 3 (D8000-DFFFF)
```

```
033B  00                        DB      000H            ; DISABLE CART. ROM 4 (E0000-E7FFF)
033C  00                        DB      000H            ; DISABLE CART. ROM 5 (E8000-EFFFF)
033D  00                        DB      000H            ; DISABLE CART. ROM 6 (F0000-F7FFF)
033E  00                        DB      000H            ; DISABLE CART. ROM 7 (F8000-FFFFF)
                        ;----- ENABLE VRAM2 DECODER
033F  E8 04C2 R         RCONF15:CALL    S8SET           ;
0342  0A                        DB      00AH            ; START REG ADDR
0343  60                        DB      060H            ; MASK DATA
0344  01                        DB      1               ; PARAMETER LENGTH
0345  B7                        DB      0B7H            ; VRAM2 (B8000-BFFFF)

0346  E9 04F5 R                 JMP     L26_2           ; CONTINUE NEXT TEST

                        ;--------------------------------------------------------------------
                        ;       SYSTEM ROM SWAP TO CARTRIDGE ROM
                        ;--------------------------------------------------------------------
0349  1E                SYSSWAP:PUSH    DS              ; SAVE DS SEG

034A  FA                        CLI                     ; DISABLE EXTERNAL INTERRUPTS
                        ;--------------------------------------------------------------------
                        ; TEST 22                                                          :
                        ;       PCjr CARTRIDGE ROM CHECKSUM TEST                          :
                        ; DESCRIPTION                                                     :
                        ;       CHECK PCjr CART. ROM (F0000-F7FFF) WITH CHECKSUM.          :
                        ; MFG ERROR CODE = 2AF0 (F0=MSB OF SEGMENT THAT HAS CHECKSUM)     :
                        ;--------------------------------------------------------------------
034B  E8 0F7C R                 CALL    MFG_UP          ; SEND 'E9' TO MFG_TESTER

034E  BA 01FF                   MOV     DX,S8STATUS     ;
0351  EC                        IN      AL,DX           ;
0352  A8 40                     TEST    AL,EROM6IN      ; CART. ROM IN F0000-F7FFF ?
0354  75 2D                     JNZ     JROM1           ; NO -BYPASS
0356  B8 F000                   MOV     AX,0F000H       ; SET BASE ROM ADDR
0359  8E D8                     MOV     DS,AX           ;
035B  33 F6                     XOR     SI,SI           ;
035D  8B 04                     MOV     AX,DS:[SI]      ;
035F  3D AA55                   CMP     AX,0AA55H       ; PCJR CART. ROM ID FOUND ?
0362  74 1F                     JE      JROM1           ; YES-BYPASS
0364  B9 8000                   MOV     CX,8000H        ; SET ROM LENGTH 32KB
0367  E8 01AD R                 CALL    ROS_CHECKSUM    ; ARITHMETIC CHECKSUM GOOD ?
036A  74 17                     JZ      JROM1           ; YES-
036C  B4 02                     MOV     AH,2            ;
036E  B7 00                     MOV     BH,0            ;
0370  BA 081C                   MOV     DX,081CH        ; POSITION CURSOR, ROW 8, COL 28
0373  CD 10                     INT     INT_10          ;
0375  B0 F0                     MOV     AL,0F0H         ;
0377  E8 0F90 R                 CALL    XPC_BYTE        ; DISPLAY MSB OF DATA SEG
037A  BB 2AF0                   MOV     BX,2AF0H        ; SET ERROR CODE
037D  BE 0064 R                 MOV     SI,OFFSET CART_ERR ;
0380  E8 0D32 R                 CALL    E_MSG           ; GO ERROR ROUTINE
0383                    JROM1:
                        ;-----
                                ASSUME  DS:XXDATA       ;

0383  E8 0FE2 R                 CALL    DDX             ;
0386  80 3E 0018 R 00           CMP     POST_ERR,00H    ; CHECK FOR "POST_ERR" NON-ZERO
                                ASSUME  DS:DATA         ;
038B  1F                        POP     DS              ; RESTORE DS
038C  74 0D                     JE      JROM2           ; CONTINUE IF NO ERROR
038E  B2 02                     MOV     DL,2            ; 2 SHORT BEEPS (ERROR)
0390  E8 0FAB R                 CALL    ERR_BEEP        ;
0393                    ERR_WAITJ:
0393  B4 00                     MOV     AH,00           ;
0395  CD 16                     INT     INT_16          ; WAIT FOR "ENTER" KEY
0397  3C 0D                     CMP     AL,0DH          ;
0399  75 F8                     JNE     ERR_WAITJ       ;

039B  B0 00             JROM2:  MOV     AL,0            ; DISABLE NMI
039D  E6 A0                     OUT     NMI_PORT,AL     ;
039F  E6 60                     OUT     PORT_A,AL       ; CLEAR KBD PORT
                        ;----- COPY DATA IN FIRST 128K RAM CARD TO 64K VRAM
03A1  C7 06 0072 R 1234         MOV     RESET_FLAG,1234H ; SET WARM START INDICATOR FOR
                                                        ;  RETURNING FROM PCJR MODE
03A7  8B 1E 0015 R              MOV     BX,TRUE_MEM     ;
03AB  81 FB 0080                CMP     BX,128          ; IS MEMORY SIZE =< 128K ?
03AF  76 27                     JBE     SYSSWP2         ; YES-
03B1  83 EB 40                  SUB     BX,64           ;
03B4  E4 62                     IN      AL,PORT_C       ;
03B6  A8 08                     TEST    AL,08H          ;
03B8  75 03                     JNZ     SYSSWP1         ;
03BA  83 EB 40                  SUB     BX,64           ;
03BD  B1 06             SYSSWP1:MOV     CL,6            ;
03BF  D3 E3                     SHL     BX,CL           ;
03C1  8E C3                     MOV     ES,BX           ;
03C3  33 FF                     XOR     DI,DI           ;
03C5  8E DF                     MOV     DS,DI           ;
03C7  33 F6                     XOR     SI,SI           ; COPY 4KB DATA FROM 1ST 128K RAM
03C9  B9 0800                   MOV     CX,0800H        ; CARD TO 64K BASE RAM OR 64K RAM CD
03CC  F3/ A5                    REP     MOVSW           ; (DS:SI=>ES:DI)
                        ;----- SET 128K RAM CARD START ADDR FROM 20000H
03CE  BA 03DA                   MOV     DX,VGA_CTL      ;
03D1  EC                        IN      AL,DX           ;
03D2  B0 03                     MOV     AL,3            ;
03D4  EE                        OUT     DX,AL           ;
03D5  B0 10                     MOV     AL,10H          ;
03D7  EE                        OUT     DX,AL           ;
                        ;----- ALLOCATE PROGRAMMABLE RAM AT 00000-1FFFF
03D8  BA 01FF           SYSSWP2:MOV     DX,S8STATUS     ;
03DB  EC                        IN      AL,DX           ;
03DC  B0 08                     MOV     AL,08H          ;
03DE  EE                        OUT     DX,AL           ;
03DF  B0 A0                     MOV     AL,0A0H         ;
03E1  EE                        OUT     DX,AL           ;
```

```
03E2  B0 63                     MOV     AL,63H          ;
03E4  EE                        OUT     DX,AL           ;
                        ;----- SET UP VIDEO SYSTEM TO PCjr MODE
03E5  BA 03DA                   MOV     DX,VGA_CTL      ;

03E8  EC                        IN      AL,DX           ;
03E9  B0 00                     MOV     AL,0            ; DISABLE NATIVE VIDEO PATH
03EB  EE                        OUT     DX,AL           ;
03EC  EE                        OUT     DX,AL           ;
03ED  B0 01                     MOV     AL,1            ; ENABLE VRAM1 & ALL PALETTE REGS
03EF  EE                        OUT     DX,AL           ;
03F0  B0 1F                     MOV     AL,1FH          ;
03F2  EE                        OUT     DX,AL           ;
                        ;-----
03F3  52                        PUSH    DX              ;
03F4  E8 04C2 R                 CALL    S8SET           ; SET UP VGA
03F7  8C                        DB      08CH            ;  START REG ADDR
03F8  00                        DB      000H            ;  MASK DATA
03F9  02                        DB      2               ;  PARAMETER LENGTH
03FA  80                        DB      080H            ;  ENABLE PCJR VGA
03FB  00                        DB      000H            ;  DISABLE NATIVE VGA
03FC  E8 04C2 R                 CALL    S8SET           ; SET UP VRAM DECODER
03FF  09                        DB      009H            ;  START REG ADDR
0400  60                        DB      060H            ;  MASK DATA
0401  02                        DB      2               ;  PARAMETER LENGTH
0402  B7                        DB      0B7H            ;  ENABLE VRAM1 DECODER (B8000-BFFFF)
0403  00                        DB      000H            ;  DISABLE VRAM2 DECODER
0404  E8 04C2 R                 CALL    S8SET           ; SET UP PAGE REG
0407  90                        DB      090H            ;  START REG ADDR
0408  00                        DB      000H            ;  MASK DATA
0409  01                        DB      1               ;  PARAMETER LENGTH
040A  00                        DB      000H            ;  DISABLE CRT/CPU PAGE REG 2
040B  5A                        POP     DX              ;
                        ;-----
040C  B9 0007                   MOV     CX,7            ; CLEAR VGA REG 0-6 WITH ZERO
040F  EC                        IN      AL,DX           ;
0410  B4 00                     MOV     AH,0            ;
0412  8A C4             SYSSWP3:MOV     AL,AH           ;
0414  EE                        OUT     DX,AL           ;
0415  B0 00                     MOV     AL,0            ;
0417  EE                        OUT     DX,AL           ;
0418  FE C4                     INC     AH              ;
041A  E2 F6                     LOOP    SYSSWP3         ;
                        ;-----
041C  E8 0000 E                 ASSUME  DS:DATA         ;
041F  E4 62                     CALL    DDS             ;
0421  A8 08                     IN      AL,PORT_C       ;
0423  75 38                     TEST    AL,0000I000B    ; 64K RAM CARD INSTALLED ?
0425  BB 0080                   JNZ     SYSSWP6         ; NO -
0428  33 FF                     MOV     BX,128          ; SET INITIAL TRUE MEMORY SIZE
042A  B9 2000                   XOR     DI,DI           ;
042D  8E C1                     MOV     CX,2000H        ;
042F  B8 5A5A           SYSSWP4:MOV     ES,CX           ;
0432  26: 89 05                 MOV     AX,5A5AH        ;
0435  EB 00                     MOV     ES:[DI],AX      ;
0437  EB 00                     JMP     $+2             ;
0439  26: 8B 15                 JMP     $+2             ;
                                MOV     DX,ES:[DI]      ;

043C  33 C2                     XOR     AX,DX           ; 128K RAM CARD INSTALLED ?
043E  75 0F                     JNZ     SYSSWP5         ; NO -
0440  26: 89 05                 MOV     ES:[DI],AX      ; CLEAR MEMORY
0443  81 C3 0080                ADD     BX,128          ;
0447  80 C5 20                  ADD     CH,20H          ;
044A  80 FD 80                  CMP     CH,80H          ;
044D  72 DE                     JB      SYSSWP4         ;
044F  89 1E 0015 R      SYSSWP5:MOV     TRUE_MEM,BX     ; SAVE RAM SIZE
0453  C7 06 0013 R 0070         MOV     MEMORY_SIZE,112 ;
0459  B0 3F                     MOV     AL,3FH          ; CPU/CRT PAGE 7
045B  EB 14                     JMP     SHORT SYSSWP7   ;
045D  C7 06 0015 R 0040 SYSSWP6:MOV     TRUE_MEM,64     ; SAVE RAM SIZE
0463  C7 06 0013 R 0030         MOV     MEMORY_SIZE,48  ;
0469  81 26 0010 R 00FB         AND     EQUIP_FLAG,0FBH ; PLANAR RAM SIZE 48K
046F  B0 1B                     MOV     AL,1BH          ; CPU/CRT PAGE 3
0471  BA 03DF           SYSSWP7:MOV     DX,PAGREG       ; SET PAGE REG
0474  EE                        OUT     DX,AL           ;
0475  A2 008A R                 MOV       ; SAVE PAGE REG DATA

0478  C6 06 0062 R 07           MOV     ACTIVE_PAGE,7   ; SAVE CRT PAGE
                        ;----- COPY SWAP HANDLER IN RAM AREA
047D  B8 F800           SYSSWP8:MOV     AX,0F800H       ; SET BASE ROM
0480  8E D8                     MOV     DS,AX           ;
0482  BE 0495 R                 MOV     SI,OFFSET SYSTRANS  ; SET SYS TRANS RTN START ADDR
0485  33 C0                     XOR     AX,AX           ;
0487  8E C0                     MOV     ES,AX           ;
0489  33 FF                     XOR     DI,DI           ; SET COPY AREA TOP ADDR
048B  B9 0028                   MOV     CX,40           ;
048E  F3/ A5                    REP     MOVSW           ;
                        ;----  JUMP TO SWAP HANDLER TO CHANGE THE SYSTEM
0490  EA                        DB      0EAH            ; JUMP FAR
0491  0000                      DW      0000H           ;
0493  0000                      DW      0000H           ;
```

```
;--------------------------------------------------------------------
; SUBROUTINE
;       SYSTEM SWAP HANDLER
; DESCRIPTION
;    1. IF SYSTEM IS OVERLAPPED BY THE ROM CARTRIDGE, THIS
;       PROGRAM IS COPIED TO TEMPORARY RAM AREA.
;    2. THE CONTROL IS PASSED TO THE ROUTINE IN THE RAM AREA.
;    3. THE ROUTINE IN THE RAM AREA DISABLES SYSTEM ROM
;       AND ENABLES OVERLAPPED ROM CARTRIDGE.
;    4. THE CONTROL IS PASSED TO THE POWER-ON RESET VECTOR
;       IN THE OVERLAPPED ROM CARTRIDGE.
;--------------------------------------------------------------------
```

```
0495  BA 01FF                 SYSTRANS: MOV   DX,S8STATUS     ;
0498  EC                              IN      AL,DX           ; READ AND CLEAR CTRL REG

0499  B9 0015                         MOV     CX,3H7          ; SET COUNTER

049C  33 C0                           XOR     AX,AX           ;
049E  8E D8                           MOV     DS,AX           ;
04A0  BE 0018 90                      MOV     SI,SYSTR5-SYSTRANS ; SET TABLE ADDR

04A4  AC                        SYSTR1: LODSB                   ; LOAD SET DATA
04A5  EE                              OUT     DX,AL           ;
04A6  E2 FC                           LOOP    SYSTR1          ;
                              ;----- JUMP TO POWER ON RESET VECTOR IN ROM CARTRIDGE
04A8  EA                              DB      0EAH            ; JUMP FAR
04A9  FFF0                            DW      0FFF0H          ;
04AB  F000                            DW      0F000H          ;

04AD  00 00 00                SYSTR5: DB      000H,000H,000H  ; DISABLE BASE ROM
04B0  01 BA 60                        DB      001H,0BAH,060H  ; ENABLE CART. ROM 2 (D0000-D7FFF)
04B3  02 BB 60                        DB      002H,0BBH,060H  ; ENABLE CART. ROM 3 (D8000-DFFFF)
04B6  03 BC 60                        DB      003H,0BCH,060H  ; ENABLE CART. ROM 4 (E0000-E7FFF)
04B9  04 BD 60                        DB      004H,0BDH,060H  ; ENABLE CART. ROM 5 (E8000-EFFFF)
04BC  05 BE 60                        DB      005H,0BEH,060H  ; ENABLE CART. ROM 6 (F0000-F7FFF)
04BF  06 BF 60                        DB      006H,0BFH,060H  ; ENABLE CART. ROM 7 (F8000-FFFFF)
                                      ;
                              ;-------------------------------------------------------+------
                              ; SUBROUTINE                                           :
                              ;       SET UP OF PROGRAMMABLE MEMORY & I/O DECODER     :
                              ; DESCRIPTION                                          :
                              ;       THIS SUBROUTINE SET UP PROGRAMMABLE MEMORY AND  :
                              ;       I/O DECODER. THE CALL INSTRUCTION IS FOLLOWED BY THE
                              ;       PARAMETERS AND ITS LENGTH. THE PARAMETERS INCLUDES CONTROL
                              ;       REGISTER (ADDR, CONTROL REG 1 AND 2 ) INFORMATION
                              ;       AND ARE SET SEQUENCIALY.
                              ;       THE CALLER SETS FIRST CONTROL REGISTER ADDRESS, MASK DATA
                              ;       PARAMETER LENGTH, AND VALIABLE REGISTER 1 VALUE.
                              ;
                              ; --------- CALLING SEQUENCE ---------
                              ;
                              ;       CALL  S8SET         ; CALL SET UP SUBROUTINE
                              ;
                              ;       DB    ??H           ; START REG ADDR ( 00H-93H )
                              ;       DB    ??H           ; MASK DATA
                              ;       DB    ??            ; PARAMETER LENGTH
                              ;       DB    ??H           ; FIRST  REG 1 VALUE
                              ;       DB    ??H           ; SECOND REG 1 VALUE
                              ;       :     :             :
                              ;       DB    ??H           ; LAST   REG 1 VALUE
                              ;-------------------------------------------------------------
04C2                          S8SET   PROC    NEAR            ; ENTRY POINT BY "CALL"
04C2  5E                              POP     SI              ; SET TABLE POINTER
04C3  1E                              PUSH    DS              ; SAVE DS
04C4  33 FF                           XOR     DI,DI           ; "CALL" ENTRY FLAG
04C6  8C C8                           MOV     AX,CS           ; CS:IP
04C8  EB 07                           JMP     SHORT S8SETX    ;
04CA                          S8SETJ  LABEL   NEAR            ; ENTRY POINT BY "JMP"
04CA  8B F4                           MOV     SI,SP           ; SET TABLE POINTER
04CC  BF FFFF                         MOV     DI,0FFFFH       ; "JMP" ENTRY FLAG
04CF  8C D0                           MOV     AX,SS           ; SS:SP
04D1  8E D8                   S8SETX: MOV     DS,AX           ;
04D3  AD                              LODSW                   ; GET START REG ADDR (BL) AND
04D4  8B D8                           MOV     BX,AX           ; MASK DATA (BH)

04D6  AC                              LODSB                   ; GET SET DATA LENGTH (CX)
04D7  33 C9                           XOR     CX,CX           ;
04D9  8A C8                           MOV     CL,AL           ;

04DB  BA 01FF                         MOV     DX,S8STATUS     ; READ & CLEAR CTRL REG
04DE  EC                              IN      AL,DX           ;

04DF  8A C3                   S801:   MOV     AL,BL           ; GET CTRL REG ADDR
04E1  EE                              OUT     DX,AL           ; WRITE CTRL REG ADDR
04E2  FE C3                           INC     BL              ; SET NEXT CTRL REG ADDR
04E4  AC                              LODSB                   ; LOAD CTRL REG 2 DATA ( VALIABLE )
04E5  EE                              OUT     DX,AL           ; SET REG 1 DATA
04E6  8A C7                           MOV     AL,BH           ; SET REG 2 DATA
04E8  EE                              OUT     DX,AL           ;
04E9  E2 F4                           LOOP    S801

04EB  0B FF                           OR      DI,DI           ;                                :K
04ED  74 03                           JZ      S802            ;
04EF  8B E6                           MOV     SP,SI           ;
04F1  C3                              RET                     ; RETURN TO CALLER ("JUMP")

04F2  1F                      S802:   POP     DS              ; RESTORE DS
04F3  56                              PUSH    SI              ; ADJUST RETURN POINT
04F4  C3                              RET                     ; RETURN TO CALLER ("CALL")
04F5                          S8SET   ENDP
04F5                          L26_2:                          ;
                              ;-------------------------------------------------------------
                              ; PART 8                                                     :
                              ;       INTERRUPTS                                           :
                              ; DESCRIPTION                                                :
                              ;       32 INTERRUPTS ARE INITIALIZED TO POINT TO A DUMMY    :
                              ;       HANDLER. THE BIOS INTERRUPTS ARE LOADED. DIAGNOSTIC  :
                              ;       INTERRUPTS ARE LOADED SYSTEM CONFIGURATION WORD IS PUT :
                              ;       IN MEMORY. THE DUMMY INTERRUPT HANDLER RESIDES HERE. :
                              ;-------------------------------------------------------------
                                      ASSUME  DS:XXDATA
04F5  E8 0FE2 R                       CALL    DDX             ; SET DS XXDATA SEG
04F8  C6 06 0005 R F7                 MOV     MFG_TST,0F7H    ; SEND 'F7' TO MFG_TESTER
04FD  E8 0F7C R                       CALL    MFG_UP          ; UPDATE MFG CHECKPOINT
0500  C7 06 0022 R 0DA7 R   C         MOV     MFG_RTN,OFFSET MFG_OUT
```

```
0506  8C C8                        MOV     AX,CS
0508  A3 0024 R                     MOV     MFG_RTN+2,AX    ; SET DOUBLEWORD POINTER TO MFG.
                                                             ; ERROR OUTPUT ROUTINE SO DIAGS.
                                                             ; DON'T HAVE TO DUPLICATE CODE
                                    ASSUME  CS:CODE,DS:ABS0
                                    XOR     AX,AX           ;
050B  33 C0                         MOV     DS,AX
050D  8E D8                   ;----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
                                    MOV     CX,255          ; FILL ALL INTERRUPTS
050F  B9 00FF                       SUB     DI,DI           ; FIRST INTERRUPT LOCATION IS 0000
0512  2B FF                         MOV     ES,DI           ; SET ES=0000 ALSO
0514  8E C7                   D3:   MOV     AX,OFFSET D11   ; MOVE ADDR OF INTR PROC TO TBL
0516  B8 0599 R                     STOSW
0519  AB                            MOV     AX,CS           ; GET ADDR OF INTR PROC SEG
051A  8C C8                         STOSW
051C  AB                            LOOP    D3              ; VECTBL0
051D  E2 F7                         MOV     EXST,OFFSET EXTAB ; SET UP EXT. SCAN TABLE
051F  C7 06 0124 R 0000 E   C  ;----- SET UP BIOS INTERRUPTS
                                    MOV     DI,OFFSET VIDEO_INT ; SET UP VIDEO INT
0525  BF 0040 R                     PUSH    CS
0528  0E                            POP     DS              ; PLACE CS IN DS
0529  1F                            MOV     SI,OFFSET VECTOR_TABLE+16
052A  BE 15D0 R                     MOV     CX,16
052D  B9 0010                 D4:   MOVSW                   ; MOVE INTERRUPT VECTOR TO LOW
0530  A5                                                    ; MEMORY
                                    INC     DI
0531  47                           INC     DI              ; POINT TO NEXT VECTOR ENTRY
0532  47                           LOOP    D4              ; REPEAT FOR ALL 16 BIOS INTERRUPTS
0533  E2 FB                 ;----- SET UP INT 78, 79, & 7A
                                    MOV     DI,78H*4        ; INT 78H
0535  BF 01E0                       MOV     AX,OFFSET KKKFDM ;
0538  B8 0000 E                     STOSW                   ;
053B  AB                            INC     DI              ;
053C  47                            INC     DI              ;
053D  47                            MOV     AX,OFFSET BUFFER_QUEING  ; INT 79H
053E  B8 0000 E                     STOSW                   ;
0541  AB                            INC     DI              ;
0542  47                            INC     DI              ;
0543  47
                                    MOV     AX,OFFSET DICT_ADDR ; INT 7AH
0544  B8 3A00                       STOSW                   ;
0547  AB                            MOV     AX,0F000H       ;
0548  B8 F000                      STOSW                   ;
0548  AB                     ;---- SET UP RS232 WRAP TEST INTERRUPTS
                                    MOV     DI,84H*4        ;
054C  BF 0210                       MOV     AX,OFFSET WRAP_TEST ;
054F  B8 1335 R                     STOSW                   ;
0552  AB                      ;----- SET UP DEFAULT EQUIPMENT DETERMINATION WORD
                              ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED
                              ;       BIT 13 = RESERVED
                              ;       BIT 12 = GAME I/O ATTACHED
                              ;       -------
                              ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
                              ;       BIT 8 = DMA (0=DMA PRESENT, 1=NO DMA ON SYSTEM)
                              ;       =======
                              ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES
                              ;               00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
                              ;       BIT 5,4 = INITIAL VIDEO MODE
                              ;                       00 - UNUSED
                              ;                       01 - NATIVE MODE (20 ZENKAKU X 11)
                              ;                       10 - EXTENSION MODE
                              ;                       11 - RESERVED
                              ;       -------
                              ;       BIT 3,2 = PLANAR RAM SIZE (10=48K,11=64K)
                              ;       BIT 1 = RESERVED
                              ;       BIT 0 = 1 (IPL DISKETTE INSTALLED)
                              ;-----------------------------------------------
                                    ASSUME  CS:CODE,DS:ABS0
0553  8E D9                         MOV     DS,CX           ; SET DS TO 0
                                                             ; DEFAULT:1 PARALLEL PRINTER,GAMEIO,
                                                             ;  NO DMA,NATIVE MODE,64K ON PLANAR
0555  C7 06 0410 R 511C             MOV     DATA_WORD[EQUIP_FLAG-RS232_BASE],511CH ;
                                                             ; SET PRINTER ADDR
055B  C7 06 0408 R 0378             MOV     DATA_WORD[PRINTER_BASE-RS232_BASE],PARAL_PORT ;
                              ;--------------------------------------------------------------
                              ; TEST 9                                                      :
                              ;       INITIALIZE AND TEST THE 8259 INTERRUPT CONTROLLER CHIP  :
                              ; MFG ERROR CODE = 07XX (XX=00, DATA PATH OR INERNAL FAILURE,  :
                              ;                       XX=ANY OTHER BITS ON=UNEXPECTED INT'S)  :
                              ;--------------------------------------------------------------
0561  E8 0F7C R                     CALL    MFG_UP          ; SEND 'F6' TO MFG_TESTER
                                    ASSUME  DS:ABS0,CS:CODE
0564  B0 13                         MOV     AL,13H          ; ICW1 - RESET EDGE SENSE CIRCUIT,
                                                             ;  SET SINGLE 8259 CHIP & ICW4 READ
0566  E6 20                         OUT     INTA00,AL
0568  B0 08                         MOV     AL,8            ; ICW2 - SET INTERRUPT TYPE 8 (8-F)
056A  E6 21                         OUT     INTA01,AL
056C  B0 09                         MOV     AL,9            ; ICW4 - SET BUFFERED MODE/SLAVE
                                                             ;  AND 8086 MODE
056E  E6 21                         OUT     INTA01,AL
                              ;--------------------------------------------------------------
                              ;       TEST ABILITY TO WRITE/READ THE MASK REGISTER
                              ;--------------------------------------------------------------
0570  B0 00                         MOV     AL,0            ; WRITE ZEROES TO IMR
0572  8A D8                         MOV     BL,AL           ; PRESET ERROR INDICATOR
0574  E6 21                         OUT     INTA01,AL       ; DEVICE INTERRUPTS ENABLED
0576  E4 21                         IN      AL,INTA01       ; READ IMR
0578  0A C0                         OR      AL,AL           ; IMR = 0?
057A  75 18                         JNZ     GERROR          ; NO - GO TO ERROR ROUTINE
057C  B0 FF                         MOV     AL,0FFH         ; DISABLE DEVICE INTERRUPTS
057E  E6 21                         OUT     INTA01,AL       ; WRITE ONES TO IMR
0580  E4 21                         IN      AL,INTA01       ; READ IMR
0582  04 01                         ADD     AL,1            ; ALL IMR BITS ON?
                                                             ; (ADD SHOULD PRODUCE 0)
```

```
0584  75 0E                     JNZ     GERROR          ; NO - GO TO ERROR ROUTINE
                        ;--------------------------------------------------------
                        ;       CHECK FOR HOT INTERRUPTS
                        ;--------------------------------------------------------
                        ;----- INTERRUPTS ARE MASKED OFF.  NO INTERRUPTS SHOULD OCCUR.
0586  FB                        STI                     ; ENABLE EXTERNAL INTERRUPTS
0587  B9 0050                   MOV     CX,50H
058A  E2 FE                     LOOP    $               ; WAIT FOR ANY INTERRUPTS
058C  8A 1E 0484 R              MOV     BL,DATA_AREA[INTR_FLAG-RS232_BASE] ; ANY INT OCCUR?
0590  0A DB                     OR      BL,BL
0592  74 2D                     JZ      END_TESTG       ; NO - GO TO NEXT TEST
0594  B7 07               GERROR: MOV     BH,07H          ; SET 07 SECTION OF ERROR MSG
0596  E9 0D32 R                 JMP     E_MSG
                        ;--------------------------------------------------------------
                        ; SUBROUTINE
                        ;       TEMPORARY INTERRUPT SERVICE
                        ; DESCRIPTION
                        ;       THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE
                        ;       POWER ON DIAGNOSTICS TO SERVICE UNUSED
                        ;       INTERRUPT VECTORS. LOCATION 'INTR_FLAG' WILL
                        ;       CONTAIN EITHER: 1. LEVEL OF HARDWARE INT. THAT
                        ;       CAUSED CODE TO BE EXEC.

                        ;       2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WERE
                        ;       EXECUTED ACCIDENTLY.
                        ;--------------------------------------------------------------
0599                    D11     PROC    NEAR            ;
                                ASSUME  DS:DATA         ;
0599  1E                        PUSH    DS              ;
059A  50                        PUSH    AX              ; SAVE REG AX CONTENTS
059B  E8 0000 E                 CALL    DDS             ;
059E  B0 0B                     MOV     AL,0BH          ; READ IN-SERVICE REG
05A0  E6 20                     OUT     INTA00,AL       ; (FIND OUT WHAT LEVEL BEING
05A2  90                        NOP                     ; SERVICED)
05A3  E4 20                     IN      AL,INTA00       ; GET LEVEL
05A5  8A E0                     MOV     AH,AL           ; SAVE IT
05A7  0A C4                     OR      AL,AH           ; 00? (NO HARDWARE ISR ACTIVE)
05A9  75 04                     JNZ     HW_INT          ;
05AB  B4 FF                     MOV     AH,0FFH         ;
05AD  EB 0A                     JMP     SHORT SET_INTR_FLAG ; SET FLAG TO FF IF NON-HDWARE
05AF  E4 21             HW_INT: IN      AL,INTA01       ; GET MASK VALUE
05B1  0A C4                     OR      AL,AH           ; MASK OFF LVL BEING SERVICED
05B3  E6 21                     OUT     INTA01,AL       ;
05B5  B0 20                     MOV     AL,EOI          ;
05B7  E6 20                     OUT     INTA00,AL       ;
05B9                    SET_INTR_FLAG:                  ;
05B9  88 26 0084 R              MOV     INTR_FLAG,AH    ; SET FLAG
05BD  58                        POP     AX              ; RESTORE REG AX CONTENTS
05BE  1F                        POP     DS              ;
05BF  FB                        STI                     ; INTERRUPTS BACK ON
05C0                    DUMMY_RETURN:                   ; NEED IRET FOR VECTOR TABLE
05C0  CF                        IRET                    ;
05C1                    D11     ENDP                    ;

05C1                    END_TESTG:
                        ;----- FIRE THE DISKETTE WATCHDOG TIMER
05C1  B0 E0                     MOV     AL,WD_ENABLE+WD_STROBE+FDC_RESET
05C3  E6 F2                     OUT     NEC_CTL,AL      ;
05C5  B0 A0                     MOV     AL,WD_ENABLE+FDC_RESET
05C7  E6 F2                     OUT     NEC_CTL,AL      ;
                                ASSUME  CS:CODE,DS:ABS0
                        ;--------------------------------------------------------------
                        ; TEST 10                                                     :
                        ;       8253 TIMER CHECKOUT                                  :
                        ; DESCRIPTION                                                :
                        ;       VERIFY THAT THE TIMERS (0, 1, AND 2) FUNCTION PROPERLY.  :
                        ;       THIS INCLUDES CHECKING FOR STUCK BITS IN ALL THE TIMERS,  :
                        ;       THAT TIMER 1 RESPONDS TO TIMER 0 OUTPUTS, THAT TIMER 0   :
                        ;       INTERRUPTS WHEN IT SHOULD, AND THAT TIMER 2'S OUTPUT     :
                        ;       WORKS AS IT SHOULD.                                      :
                        ;       THERE ARE 7 POSSIBLE ERRORS DURING THIS CHECKOUT.        :
                        ;       BL VALUES FOR THE CALL TO E_MSG INCLUDE:                 :
                        ;       0: STUCK BITS IN TIMER 0                                 :
                        ;       1: TIMER 1 DOES NOT RESPOND TO TIMER 0 OUTPUT            :
                        ;       2: TIMER 0 INTERRUPT DOES NOT OCCUR                      :
                        ;       3: STUCK BITS IN TIMER 1                                 :
                        ;       4: TIMER 2 OUTPUT INITIAL VALUE IS NOT LOW               :
                        ;       5: STUCK BITS IN TIMER 2                                 :
                        ;       6: TIMER 2 OUTPUT DOES NOT GO HIGH ON TERMINAL COUNT     :
                        ;       7: TIMER 0 OUTPUT OUT OF RANGE                           :
                        ;--------------------------------------------------------------
                        ;       INITIALIZE TIMER 1 AND TIMER 0 FOR TEST
                        ;--------------------------------------------------------------
05C9  E8 0F7C R                 CALL    MFG_UP          ; SEND 'FS' TO MFG_TESTER
05CC  B8 0176                   MOV     AX,0176H        ; SET TIMER 1 TO MODE 3 BINARY
05CF  BB FFFF                   MOV     BX,0FFFFH       ; INITIAL COUNT OF FFFF
05D2  E8 06F2 R                 CALL    INIT_TIMER      ; INITIALIZE TIMER 1
05D5  B8 0036                   MOV     AX,0036H        ; SET TIMER 0 TO MODE 3 BINARY
                                                        ; INITIAL COUNT OF FFFF
05D8  E8 06F2 R                 CALL    INIT_TIMER      ; INITIALIZE TIMER 0
                        ;--------------------------------------------------------------
                        ;       SET BIT 5 OF PORT A0 SO TIMER 1 CLOCK WILL BE PULSED BY THE
                        ;       TIMER 0 OUTPUT RATHER THAN THE SYSTEM CLOCK.
                        ;--------------------------------------------------------------
05DB  B0 20                     MOV     AL,00100000B
05DD  E6 A0                     OUT     NMI_PORT,AL     ;
                        ;--------------------------------------------------------------
                        ;       CHECK IF ALL BITS GO ON AND OFF IN TIMER 0 (CHECK FOR STUCK
                        ;       BITS)
                        ;--------------------------------------------------------------
05DF  B4 00                     MOV     AH,0            ; TIMER 0
05E1  E8 06A9 R                 CALL    BITS_ON_OFF     ; LET SUBROUTINE CHECK IT
05E4  73 05                     JNB     TIMER1_NZ       ; NO STUCK BITS (CARRY FLAG NOT SET)
```

```
05E6  B3 00                     MOV     BL,0                    ; STUCK BITS IN TIMER 0
05E8  E9 069F R                 JMP     TIMER_ERROR
                        ;-----------------------------------------------------------------
                        ;       SINCE TIMER 0 HAS COMPLETED AT LEAST ONE COMPLETE CYCLE,
                        ;       TIMER 1 SHOULD BE NON-ZERO.  CHECK THAT THIS IS THE CASE.
                        ;-----------------------------------------------------------------
05EB                    TIMER1_NZ:
05EB  E4 41                     IN      AL,TIMER+1              ; READ LSB OF TIMER 1
05ED  8A E0                     MOV     AH,AL                   ; SAVE LSB

05EF  E4 41                     IN      AL,TIMER+1              ; READ MSB OF TIMER 1
05F1  3D FFFF                   CMP     AX,0FFFFH               ; STILL FFFF?
05F4  75 05                     JNE     TIMER0_INTR             ; NO - TIMER 1 HAS BEEN BUMPED
05F6  B3 01                     MOV     BL,1                    ; TIMER 1 WAS NOT BUMPED BY TIMER 0
05F8  E9 069F R                 JMP     TIMER_ERROR
                        ;-----------------------------------------------------------------
                        ;       CHECK FOR TIMER 0 INTERRUPT
                        ;-----------------------------------------------------------------
05FB                    TIMER0_INTR:
05FB  FB                        STI                             ; ENABLE MASKABLE EXT INTERRUPTS
05FC  E4 21                     IN      AL,INTA01
05FE  24 FE                     AND     AL,0FEH                 ; MASK ALL INTRS EXCEPT LVL 0
0600  20 06 0484 R              AND     DATA_AREA[INTR_FLAG-RS232_BASE],AL ; CLR INT RCVED
0604  E6 21                     OUT     INTA01,AL               ; WRITE THE 8259 IMR
0606  33 C9                     XOR     CX,CX                   ; SET LOOP COUNT
0608                    WAIT_INTR_LOOP:
0608  F6 06 0484 R 01           TEST    DATA_AREA[INTR_FLAG-RS232_BASE],1 ; TMR0 INT OCCUR?
060D  75 07                     JNZ     RESET_INTRS             ; YES - CONTINUE
060F  E2 F7                     LOOP    WAIT_INTR_LOOP          ; WAIT FOR INTR FOR SPECIFIED TIME
0611  B3 02                     MOV     BL,2                    ; TIMER 0 INTR DIDN'T OCCUR
0613  E9 069F R                 JMP     TIMER_ERROR             ;
                        ;-----------------------------------------------------------------
                        ;       HOUSEKEEPING FOR TIMER 0 INTERRUPTS
                        ;-----------------------------------------------------------------
0616                    RESET_INTRS:
0616  FA                        CLI                             ;
                        ;-----------------------------------------------------------------
                        ;       CHECK COUNT DOWN SPEED OF TIMER 0
                        ;-----------------------------------------------------------------
0617  B0 00                     MOV     AL,0                    ;
0619  E6 43                     OUT     TIM_CTL,AL              ; LATCH TIMER 0
061B  50                        PUSH    AX                      ; PAUSE
061C  58                        POP     AX                      ;
061D  E4 40                     IN      AL,TIMER+0              ; READ TIMER 0 LSB
061F  8A D0                     MOV     DL,AL                   ;
0621  E4 40                     IN      AL,TIMER+0              ;       "       MSB
0623  8A F0                     MOV     DH,AL                   ;

0625  B9 05E2                   MOV     CX,05E2H                ; WAIT 1 MS (CH=5, CL=226)
0628                    TIMER0_W:                               ;
0628  D3 E8                     SHR     AX,CL                   ;
062A  FE CD                     DEC     CH                      ;
062C  75 FA                     JNZ     TIMER0_W                ;

062E  B0 00                     MOV     AL,0                    ;
0630  E6 43                     OUT     TIM_CTL,AL              ; LATCH TIMER 0
0632  50                        PUSH    AX                      ; PAUSE
0633  58                        POP     AX                      ;
0634  E4 40                     IN      AL,TIMER+0              ; READ TIMER 0 LSB
0636  8A C8                     MOV     CL,AL                   ;
0638  E4 40                     IN      AL,TIMER+0              ;       "       MSB
063A  8A E8                     MOV     CH,AL                   ;
063C  B3 07                     MOV     BL,7                    ; TIMER 0 OUTPUT OUT OF RANGE

063E  2B D1                     SUB     DX,CX                   ; (0955H)
0640  80 FE 0A                  CMP     DH,0AH                  ; WITHIN TOLERANCE ?
0643  77 5A                     JA      TIMER_ERROR             ; NO -
0645  80 FE 08                  CMP     DH,08H                  ;
0648  72 55                     JB      TIMER_ERROR             ;
                        ;-----------------------------------------------------------------
                        ;       RESET D5  OF PORT A0 SO THAT THE TIMER 1 CLOCK WILL BE
                        ;       PULSED BY THE SYSTEM CLOCK.
                        ;-----------------------------------------------------------------
064A  B0 00             TIME_1: MOV     AL,0                    ; MAKE AL = 00
064C  E6 A0                     OUT     NMI_PORT,AL             ;
                        ;-----------------------------------------------------------------
                        ;       CHECK FOR STUCK BITS IN TIMER 1
                        ;-----------------------------------------------------------------
064E  B4 01                     MOV     AH,1                    ; TIMER 1
0650  E8 06A9 R                 CALL    BITS_ON_OFF
0653  73 04                     JNB     TIMER2_INIT             ; NO STUCK BITS
0655  B3 03                     MOV     BL,3                    ; STUCK BITS IN TIMER 1
0657  EB 46                     JMP     SHORT TIMER_ERROR
                        ;-----------------------------------------------------------------
                        ;       INITIALIZE TIMER 2
                        ;-----------------------------------------------------------------
0659                    TIMER2_INIT:
0659  B8 02B6                   MOV     AX,02B6H                ; SET TIMER 2 TO MODE 3 BINARY
065C  BB FFFF                   MOV     BX,0FFFFH               ; INITIAL COUNT
065F  E8 06F2 R                 CALL    INIT_TIMER
                        ;-----------------------------------------------------------------
                        ;       SET PB0 OF PORT_B OF 8255 (TIMER 2 GATE)
                        ;-----------------------------------------------------------------
0662  E4 61                     IN      AL,PORT_B               ; CURRENT STATUS
0664  0C 01                     OR      AL,00000001B            ; SET BIT 0 - LEAVE OTHERS ALONE
0666  E6 61                     OUT     PORT_B,AL
                        ;-----------------------------------------------------------------
                        ;       CHECK FOR STUCK BITS IN TIMER 2
                        ;-----------------------------------------------------------------
0668  B4 02                     MOV     AH,2                    ; TIMER 2
066A  E8 06A9 R                 CALL    BITS_ON_OFF
066D  73 04                     JNB     REINIT_T2               ; NO STUCK BITS
```

```
066F  B3 05                   MOV     BL,5            ; STUCK BITS IN TIMER 2
0671  EB 2C                   JMP     SHORT TIMER_ERROR
                      ;--------------------------------------------------------
                      ;       RE_INITIALIZE TIMER 2 WITH MODE 0 AND A SHORT COUNT
                      ;--------------------------------------------------------
0673                  REINIT_T2:
                      ;----- DROP GATE TO TIMER 2
0673  E4 61                   IN      AL,PORT_B       ; CURRENT STATUS
0675  24 FE                   AND     AL,11111110B    ; RESET BIT 0 - LEAVE OTHERS ALONE
0677  E6 61                   OUT     PORT_B,AL
0679  B8 02B0                MOV     AX,02B0H        ; SET TIMER 2 TO MODE 0 BINARY
067C  BB 000A                 MOV     BX,000AH        ; INITIAL COUNT OF 10
067F  E8 06F2 R               CALL    INIT_TIMER
                      ;--------------------------------------------------------
                      ;       CHECK PC5 OF PORT_C OF 8255 TO SEE IF THE OUTPUT OF TIMER 2
                      ;       IS LOW
                      ;--------------------------------------------------------
0682  E4 62                   IN      AL,PORT_C       ; CURRENT STATUS
0684  24 20                   AND     AL,00100000B    ; MASK OFF OTHER BITS
0686  74 04                   JZ      CK2_ON          ; IT'S LOW
0688  B3 04                   MOV     BL,4            ; PC5 OF PORT_C WAS HIGH WHEN IT
068A  EB 13                   JMP     SHORT TIMER_ERROR ; SHOULD HAVE BEEN LOW
                      ;----- TURN GATE BACK ON
068C  E4 61           CK2_ON: IN      AL,PORT_B       ; CURRENT STATUS
068E  0C 01                   OR      AL,00000001B    ; SET BIT 0 - LEAVE OTHERS ALONE
0690  E6 61                   OUT     PORT_B,AL
                      ;--------------------------------------------------------
                      ;       CHECK PC5 OF PORT_C TO SEE IF THE OUTPUT OF TIMER 2 GOES
                      ;       HIGH
                      ;--------------------------------------------------------
0692  B9 000A                 MOV     CX,000AH        ; WAIT FOR OUTPUT GO HIGH, SHOULD
0695  E2 FE           CK2_L0: LOOP    CK2_L0          ; BE LONGER THAN INITIAL COUNT
0697  E4 62                   IN      AL,PORT_C       ; CURRENT STATUS
0699  24 20                   AND     AL,00100000B    ; MASK OFF ALL OTHER BITS
069B  75 65                   JNZ     POD13_END       ; IT'S HIGH - WE'RE DONE!
069D  B3 06                   MOV     BL,6            ; TIMER 2 OUTPUT DID NOT GO HIGH
                      ;--------------------------------------------------------
                      ;       8253 TIMER ERROR OCCURRED.  SET BH WITH MAJOR ERROR
                      ;       INDICATOR AND CALL E_MSG TO INFORM THE SYSTEM OF THE ERROR.
                      ;       (BL ALREADY CONTAINS THE MINOR ERROR INDICATOR TO TELL
                      ;       WHICH PART OF THE TEST FAILED.)
                      ;--------------------------------------------------------
069F                  TIMER_ERROR:
069F  B7 08                   MOV     BH,8            ; TIMER ERROR INDICATOR
06A1  E8 0D32 R               CALL    E_MSG
06A4  EB 5C                   JMP     SHORT POD13_END
                      ;--------------------------------------------------------
                      ; SUBROUTINE
                      ;       BITS ON/OFF
                      ; DESCRIPTION
                      ;       USED FOR DETERMINING IF A  PARTICULAR TIMER'S BITS GO ON
                      ;       AND OFF AS THEY SHOULD. THIS ROUTINE ASSUMES THAT THE
                      ;       TIMER IS USING BOTH THE LSB AND THE MSB.
                      ; CALLING PARAMETER
                      ;       (AH) = TIMER NUMBER (0, 1, OR 2)
                      ; RETURNS
                      ;       (CF) = 1 IF FAILED
                      ;       (CF) = 0 IF PASSED
                      ; REGISTERS AX, BX, CX, DX, DI, AND SI ARE ALTERED.
                      ;--------------------------------------------------------
06A6                  LATCHES LABEL   BYTE
06A6  00                      DB      00H             ; LATCH MASK FOR TIMER 0
06A7  40                      DB      40H             ; LATCH MASK FOR TIMER 1
06A8  80                      DB      80H             ; LATCH MASK FOR TIMER 2

06A9                  BITS_ON_OFF     PROC  NEAR
06A9  33 DB                   XOR     BX,BX           ; INITIALIZE BX REGISTER
06AB  33 F6                   XOR     SI,SI           ; 1ST PASS - SI = 0

06AD  BA 0040                 MOV     DX,TIMER        ; BASE PORT ADDRESS FOR TIMERS
06B0  02 D4                   ADD     DL,AH
06B2  BF 06A6 R               MOV     DI,OFFSET LATCHES ; SELECT LATCH MASK
06B5  32 C0                   XOR     AL,AL           ; CLEAR AL
06B7  86 C4                   XCHG    AL,AH           ; AH -> AL
06B9  03 F8                   ADD     DI,AX           ; TIMER LATCH MASK INDEX
                      ;----- 1ST PASS - CHECKS FOR ALL BITS TO COME ON
                      ;       2ND PASS - CHECKS FOR ALL BITS TO GO OFF
06BB                  OUTER_LOOP:
06BB  B9 0008                 MOV     CX,8            ; OUTER LOOP COUNTER
06BE                  INNER_LOOP:
06BE  51                      PUSH    CX              ; SAVE OUTER LOOP COUNTER
06BF  33 C9                   XOR     CX,CX           ; INNER LOOP COUNTER
06C1                  TST_BITS:
06C1  2E: 8A 05               MOV     AL,CS:[DI]      ; TIMER LATCH MASK
06C4  E6 43                   OUT     TIM_CTL,AL      ; LATCH TIMER
06C6  50                      PUSH    AX              ; PAUSE
06C7  58                      POP     AX
06C8  EC                      IN      AL,DX           ; READ TIMER LSB
06C9  0B F6                   OR      SI,SI
06CB  75 0C                   JNE     SECOND          ; SECOND PASS
06CD  0C 01                   OR      AL,01H          ; TURN LS BIT ON
06CF  0A D8                   OR      BL,AL           ; TURN 'ON' BITS ON
06D1  EC                      IN      AL,DX           ; READ TIMER MSB
06D2  0A F8                   OR      BH,AL           ; TURN 'ON' BITS ON
06D4  83 FB FF                CMP     BX,0FFFFH       ; ARE ALL TIMER BITS ON?
06D7  EB 07                   JMP     SHORT TST_CMP   ; DON'T CHANGE FLAGS
06D9                  SECOND:
06D9  22 D8                   AND     BL,AL           ; CHECK FOR ALL BITS OFF
06DB  EC                      IN      AL,DX           ; READ MSB
06DC  22 F8                   AND     BH,AL           ; TURN OFF BITS
06DE  0B DB                   OR      BX,BX           ; ALL OFF?
06E0                  TST_CMP:
```

```
06E0  74 07                     JE      CHK_END         ; YES - SEE IF DONE
06E2  E2 DD                     LOOP    TST_BITS        ; KEEP TRYING
06E4  59                        POP     CX              ; RESTORE OUTER LOOP COUNTER
06E5  E2 D7                     LOOP    INNER_LOOP      ; TRY AGAIN
06E7  F9                        STC                     ; ALL TRIES EXHAUSTED - FAILED TEST
06E8  C3                        RET
06E9                    CHK_END:
06E9  59                        POP     CX              ; POP FORMER OUTER LOOP COUNTER
06EA  46                        INC     SI
06EB  83 FE 02                  CMP     SI,2
06EE  75 CB                     JNE     OUTER_LOOP      ; CHECK FOR ALL BITS TO GO OFF
06F0  F8                        CLC                     ; TIMER BITS ARE WORKING PROPERLY
06F1  C3                        RET
06F2                    BITS_ON_OFF     ENDP
                        ;--------------------------------------------------------------
                        ; SUBROUTINE
                        ;       INITIALIZE TIMER
                        ; DESCRIPTION
                        ;       ASSUMES BOTH THE LSB AND MSB OF THE TIMER WILL BE USED.
                        ; CALLING PARAMETERS
                        ;
                        ;       (AH) = TIMER #
                        ;       (AL) = BIT PATTERN OF INITIALIZATION WORD
                        ;       (BX) = INITIAL COUNT
                        ;              (BH) = MSB COUNT
                        ;              (BL) = LSB COUNT
                        ; ALTERS REGISTERS DX AND AL.
                        ;--------------------------------------------------------------
06F2                    INIT_TIMER      PROC    NEAR
06F2  E6 43                     OUT     TIM_CTL,AL      ; OUTPUT INITIAL CONTROL WORD
06F4  BA 0040                   MOV     DX,TIMER        ; BASE PORT ADDR FOR TIMERS
06F7  02 D4                     ADD     DL,AH           ; ADD IN THE TIMER #
06F9  8A C3                     MOV     AL,BL           ; LOAD LSB
06FB  EE                        OUT     DX,AL
06FC  52                        PUSH    DX              ; PAUSE
06FD  5A                        POP     DX
06FE  8A C7                     MOV     AL,BH           ; LOAD MSB
0700  EE                        OUT     DX,AL
0701  C3                        RET
0702                    INIT_TIMER      ENDP
0702                    POD13_END:
                        ;--------------------------------------------------------------
                        ; TEST 11                                                     :
                        ;       CRT ATTACHMENT TEST                                   :
                        ; DESCRIPTION                                                 :
                        ;       1. INIT CRT TO 40 X 25 - COLOR                        :
                        ;       2. CHECK FOR VERTICAL AND VIDEO ENABLES, AND CHECK    :
                        ;          TIMING OF SAME                                     :
                        ;       3. CHECK VERTICAL INTERRUPT                           :
                        ;       4. CHECK RED, BLUE, GREEN, AND INTENSIFY DOTS         :
                        ; MFG ERROR CODE = 09XX (XX-SEE COMMENETS IN CODE)             :
                        ;--------------------------------------------------------------
= A04C                  MAVT    EQU     0A04CH          ; MAXIMUM TIME FOR VERT/VERT
                                                        ; (NOMINAL + 23%)
= C418                  MIVT    EQU     0C418H          ; MINIMUM TIME FOR VERT/VERT
                                                        ; (NOMINAL - 23%)
                        ;----- NOMINAL TIME IS B232H FOR 60 HZ.
= 00C8                  EPF     EQU     200             ; NUMBER OF ENABLES PER FRAME
                                                        ; (8 DOTS X 25 LINES)
0702  E8 0F7C R                 CALL    MFG_UP          ; SEND 'F4' TO MFG_TESTER
0705  FA                        CLI
0706  B0 70                     MOV     AL,01110000B    ; SET TIMER 1 TO MODE 0
0708  E6 43                     OUT     TIM_CTL,AL
070A  B9 8000                   MOV     CX,8000H
070D  E2 FE                     LOOP    $               ; WAIT FOR MODE SET TO "TAKE"
070F  B0 00                     MOV     AL,0
0711  E6 41                     OUT     TIMER+1,AL      ; SEND FIRST BYTE TO TIMER
0713  BA 03DA                   MOV     DX,VGA_CTL      ; SET ADDRESSING TO VIDEO ARRAY
0716  2B C9                     SUB     CX,CX           ;
                        ;----- LOOK FOR VERTICAL
0718  EC                Q2:    IN      AL,DX           ; GET STATUS
0719  A8 08                     TEST    AL,00001000B    ; VERTICAL THERE YET?
071B  75 06                     JNE     Q3              ; CONTINUE IF IT IS
071D  E2 F9                     LOOP    Q2              ; KEEP LOOKING TILL COUNT EXHAUSTED
071F  B3 00                     MOV     BL,0            ;
0721  EB 4C                     JMP     SHORT Q115      ; NO VERTICAL = ERROR 0900
                        ;----- GOT VERTICAL - START TIMER
0723  32 C0              Q3:    XOR     AL,AL           ;
0725  E6 41                     OUT     TIMER+1,AL      ; SEND 2ND BYTE TO TIMER TO START
0727  2B DB                     SUB     BX,BX           ; INIT. ENABLE COUNTER
                        ;----- WAIT FOR VERTICAL TO GO AWAY
0729  33 C9                     XOR     CX,CX
072B  EC                 Q4:    IN      AL,DX           ; GET STATUS
072C  A8 08                     TEST    AL,00001000B    ; VERTICAL STILL THERE?
072E  74 06                     JZ      Q5              ; CONTINUE IF IT'S GONE
0730  E2 F9                     LOOP    Q4              ; KEEP LOOKING TILL COUNT EXHAUSTED
0732  B3 01                     MOV     BL,01H          ;
0734  EB 39                     JMP     SHORT Q115      ; VERTICAL STUCK ON = ERROR 0901
                        ;----- NOW START LOOKING FOR ENABLE TRANSITIONS
0736  2B C9              Q5:    SUB     CX,CX
0738  EC                 Q6:    IN      AL,DX           ; GET STATUS
0739  A8 01                     TEST    AL,00000001B    ; ENABLE ON YET?
073B  75 0A                     JNE     Q7              ; GO ON IF IT IS

073D  A8 08                     TEST    AL,00001000B    ; VERTICAL ON AGAIN?
073F  75 22                     JNE     Q11             ; CONTINUE IF IT IS
0741  E2 F5                     LOOP    Q6              ; KEEP LOOKING IF NOT
0743  B3 02                     MOV     BL,02H          ;
0745  EB 28                     JMP     SHORT Q115      ; ENABLE STUCK OFF = ERROR 0902
                        ;----- MAKE SURE VERTICAL WENT OFF WITH ENABLE GOING ON
0747  A8 08              Q7:    TEST    AL,00001000B    ; VERTICAL OFF?
0749  74 04                     JZ      Q8              ; GO ON IF IT IS
074B  B3 03                     MOV     BL,03H          ;
```

```
074D  EB 20                     JMP     SHORT Q115      ; VERTICAL STUCK ON = ERROR 0903
                        ;----- NOW WAIT FOR ENABLE TO GO OFF
                        Q8:     SUB     CX,CX
074F  2B C9             Q9:     IN      AL,DX           ; GET STATUS
0751  EC                        TEST    AL,00000001B    ; ENABLE OFF YET?
0752  A8 01                     JE      Q10             ; PROCEED IF IT IS
0754  74 06                     LOOP    Q9              ; KEEP LOOKING IF NOT YET LOW
0756  E2 F9                     MOV     BL,04H          ;
0758  B3 04                     JMP     SHORT Q115      ; ENABLE STUCK ON = ERROR 0904
075A  EB 13             ;----- ENABLE HAS TOGGLED, BUMP COUNTER AND TEST FOR NEXT VERTICAL
075C  43                Q10:    INC     BX              ; BUMP ENABLE COUNTER
075D  74 04                     JZ      Q11             ; IF COUNTER WRAPS, ERROR
075F  A8 08                     TEST    AL,00001000B    ; DID ENABLE GO LOW BECAUSE OF
                                                        ; VERTICAL?
0761  74 D3                     JZ      Q5              ; IF NOT, LOOK FOR ANOTHER ENABLE
                                                        ;   TOGGLE
                        ;----- HAVE HAD COMPLETE VERTICAL-VERTICAL CYCLE, NOW TEST RESULTS
0763  B0 40             Q11:    MOV     AL,40H          ; LATCH TIMER1
0765  E6 43                     OUT     TIM_CTL,AL      ;
0767  81 FB 00C8                CMP     BX,0FFH         ; NUMBER OF ENABLES BETWEEN
                                                        ; VERTICALS O.K.?
076B  74 05                     JE      Q12             ;
076D  B3 05                     MOV     BL,05H          ;
076F  E9 07FF R         Q115:   JMP     Q22             ; WRONG # ENABLES = ERROR 0905
0772  E4 41             Q12:    IN      AL,TIMER+1      ; GET TIMER VALUE LOW
0774  8A E0                     MOV     AH,AL           ; SAVE IT
0776  90                        NOP                     ;
0777  E4 41                     IN      AL,TIMER+1      ; GET TIMER HIGH
0779  86 E0                     XCHG    AH,AL           ;
077B  FB                        STI                     ; INTERRUPTS BACK ON
077C  90                        NOP                     ;
077D  3D A04C                   CMP     AX,MAVT         ;
0780  73 04                     JAE     Q13             ;
0782  B3 06                     MOV     BL,06H          ;
0784  EB 79                     JMP     SHORT Q22       ; VERTICALS TOO FAR APART
                                                        ; = ERROR 0906
0786  3D C418           Q13:    CMP     AX,MIVT         ;
0789  76 04                     JBE     Q14             ;
078B  B3 07                     MOV     BL,07H          ;
078D  EB 70                     JMP     SHORT Q22       ; VERTICALS TOO CLOSE TOGETHER
                                                        ;   = ERROR 0907
                        ;----- TIMINGS SEEM O.K., NOW CHECK VERTICAL INTERRUPT (LEVEL 5)
078F  2B C9             Q14:    SUB     CX,CX           ; SET TIMEOUT REG
0791  E4 21                     IN      AL,INTA01       ;
0793  24 DF                     AND     AL,11011111B    ; UNMASK INT. LEVEL 5
0795  E6 21                     OUT     INTA01,AL       ;
0797  20 06 0484 R              AND     DATA_AREA[INTR_FLAG-RS232_BASE],AL
079B  FB                        STI                     ; ENABLE INTS.
079C  F6 06 0484 R 20   Q15:    TEST    DATA_AREA[INTR_FLAG-RS232_BASE],00100000B ; SEE IF
                                                        ;    INT 5 HAPPENED YET
07A1  75 06                     JNZ     Q16             ; GO ON IF IT DID
07A3  E2 F7                     LOOP    Q15             ; KEEP LOOKING IF IT DIDN'T
07A5  B3 08                     MOV     BL,08H          ;
07A7  EB 56                     JMP     SHORT Q22       ; NO VERTICAL INTERRUPT
                                                        ; = ERROR 0908
07A9  E4 21             Q16:    IN      AL,INTA01       ; DISABLE INTERRUPTS FOR LEVEL 5
07AB  0C 20                     OR      AL,00100000B    ;
07AD  E6 21                     OUT     INTA01,AL       ;

                        ;----- SEE IF RED, GREEN, BLUE AND INTENSIFY DOTS WORK
                        ;       FIRST, SET A LINE OF REVERSE VIDEO, INTENSIFIED BLANKS
                        ;       INTO VIDEO BUFFER
07AF  52                        PUSH    DX              ;
07B0  33 C0                     XOR     AX,AX           ; ENABLE DISPLAY & CLEAR SCREEN
07B2  CD 10                     INT     INT_10          ; (MODE: 40 X 25 - COLOR)
07B4  B8 0507                   MOV     AX,0507H        ; SET TO VIDEO PAGE 7
07B7  CD 10                     INT     INT_10          ;
07B9  B8 095F                   MOV     AX,095FH        ; WRITE CHARS, UNDERSCORE
07BC  BB 077F                   MOV     BX,077FH        ; PAGE 7, REVERSE VIDEO, HIGH INT
07BF  B9 0028                   MOV     CX,40           ; 40 CHARACTERS
07C2  CD 10                     INT     INT_10          ;

07C4  E8 04C2 R                 CALL    S8SET           ; DISABLE NATIVE VGA
07C7  8D                        DB      08DH            ;  START REG ADDR
07C8  00                        DB      000H            ;  MASK DATA
07C9  01                        DB      1               ;  PARAMETER LENGTH
07CA  00                        DB      000H            ;
07CB  E8 04C2 R                 CALL    S8SET           ; ENABLE PCJR VGA
07CE  8C                        DB      08CH            ;  START REG ADDR
07CF  00                        DB      000H            ;  MASK DATA
07D0  01                        DB      1               ;  PARAMETER LENGTH
07D1  80                        DB      080H            ;
07D2  5A                        POP     DX              ;

07D3  33 C0                     XOR     AX,AX           ; START WITH BLUE DOTS
07D5  2B C9             Q17:    SUB     CX,CX           ;
07D7  EE                        OUT     DX,AL           ; SET VIDEO ARRAY ADDRESS FOR DOTS
                        ;----- SEE IF DOT COMES ON
07D8  EC                Q18:    IN      AL,DX           ; GET STATUS
07D9  A8 10                     TEST    AL,00010000B    ; DOT THERE?
07DB  75 08                     JNZ     Q19             ; GO LOOK FOR DOT TO TURN OFF
07DD  E2 F9                     LOOP    Q18             ; CONTINUE TESTING FOR DOT ON
07DF  B3 10                     MOV     BL,10H          ;
07E1  0A DC                     OR      BL,AH           ; OR IN DOT BEING TESTED
07E3  EB 1A                     JMP     SHORT Q22       ; DOT NOT COMING ON = ERROR 091X
                                                        ; ( X=0, BLUE; X=1, GREEN;
                                                        ; X=2, RED; X=3, INTENSITY)

                        ;----- SEE IF DOT GOES OFF
07E5  2B C9             Q19:    SUB     CX,CX
07E7  EC                Q20:    IN      AL,DX           ; GET STATUS
07E8  A8 10                     TEST    AL,00010000B    ; IS DOT STILL ON?
```

```
07EA  74 08                     JE      Q21             ; GO ON IF DOT OFF
07EC  E2 F9                     LOOP    Q20             ; ELSE, KEEP WAITING FOR DOT
                                                        ; TO GO OFF
07EE  B3 20                     MOV     BL,20H          ;
07F0  0A DC                     OR      BL,AH           ; OR IN DOT BEING TESTED
07F2  EB 0B                     JMP     SHORT Q22       ; DOT STUCK ON = ERROR 092X
                                                        ; (X=0, BLUE; X=1, GREEN;
                                                        ;  X=2, RED; X=3, INTENSITY)
                        ;----- ADJUST TO POINT TO NEXT DOT
07F4  FE C4             Q21:    INC     AH              ;
07F6  80 FC 04                  CMP     AH,4            ; ALL 4 DOTS DONE?
07F9  74 09                     JE      Q23             ; GO END
07FB  8A C4                     MOV     AL,AH
07FD  EB D6                     JMP     Q17             ; GO LOOK FOR ANOTHER DOT
07FF  B7 09             Q22:    MOV     BH,09H          ; SET MSB OF ERROR CODE
0801  E9 0D32 R                 JMP     E_MSG
                        ;-----
0804  B0 76             Q23:    MOV     AL,01110110B    ; RE-INIT TIMER 1
0806  E6 43                    OUT     TIM_CTL,AL      ;
0808  B0 00                     MOV     AL,00H
080A  E6 41                     OUT     TIMER+1,AL
080C  EB 00                     JMP     $+2             ;
080E  E6 41                     OUT     TIMER+1,AL
                        ;--------------------------------------------------------------------
                        ; PART 12                                                          :
                        ;       SET UP KEYBOARD PARAMETERS                                 :
                        ;--------------------------------------------------------------------
                                ASSUME  DS:ABS0
0810  E8 0F7C R                 CALL    MFG_UP          ; SEND 'F3' TO MFG_TESTER
0813  33 C0                     XOR     AX,AX
0815  8E D8                     MOV     DS,AX
0817  C7 06 0008 R 0000 E   C   MOV     NMI_PTR,OFFSET KBDNMI ; SET INTERRUPT VECTOR
081D  C7 06 0120 R 0FEA R   C   MOV     KEY62_PTR,OFFSET KEY_SCAN_SAVE  ; SET VECTOR FOR
                                                        ; POD INT HANDLER
0823  0E                        PUSH    CS
0824  58                       POP     AX
0825  A3 0122 R                 MOV     KEY62_PTR+2,AX
                                ASSUME  DS:DATA
0828  E8 0000 E                 CALL    DDS             ; SET DATA SEGMENT
082B  BE 0304 R                 MOV     SI,OFFSET KB_BUFFER_J ; SET KEYBOARD PARMS
082E  89 36 001A R              MOV     BUFFER_HEAD,SI
0832  89 36 001C R              MOV     BUFFER_TAIL,SI
0836  89 36 0080 R              MOV     BUFFER_START,SI
083A  83 C6 32                  ADD     SI,50           ; SET DEFAULT BUFFER OF 50 BYTES
083D  89 36 0082 R              MOV     BUFFER_END,SI
0841  E4 A0                     IN      AL,NMI_PORT     ; CLEAR NMI F/F
0843  B0 80                     MOV     AL,80H          ; ENABLE NMI
0845  E6 A0                     OUT     NMI_PORT,AL     ;
                        ;----- IF A KEY IS STUCK, THE BUFFER SHOULD FILL WITH THAT KEY'S
                        ;      CODE. THIS WILL BE CHECKED LATER.
                        ;--------------------------------------------------------------------
                        ; TEST 13                                                          :
                        ;       32K VRAM (VRAM2) TEST                                      :
                        ; MFG ERROR CODE = 1EXX  BASE 32K VRAM ERROR                       :
                        ;                  1FXX  EXP 32K VRAM ERROR                       :
                        ;--------------------------------------------------------------------
0847  E8 0F7C R                 CALL    MFG_UP          ; SEND 'F2' TO MFG_TESTER
                        ;----- ENABLE NATIVE AND PCJR VGA FOR BLANKING SCREEN
084A  E8 04C2 R                 CALL    S8SET           ;
084D  8C                        DB      08CH            ; START REG ADDR
084E  00                        DB      000H            ; MASK DATA
084F  02                        DB      2               ; PARAMETER LENGTH
0850  80                        DB      080H            ; ENABLE PCJR VGA
0851  80                        DB      080H            ; ENABLE NATIVE VGA
0852  BA 03DA                  MOV     DX,VGA_CTL      ; DISABLE VIDEO
0855  EC                       IN      AL,DX           ;
0856  33 C0                     XOR     AX,AX           ;
0858  EE                       OUT     DX,AL           ;
0859  EE                       OUT     DX,AL           ;
                        ;----- BASE 32K VRAM AND EXP 32K VRAM TEST (EXCEPT REGEN BUFF)
085A  B8 B800                   MOV     AX,0B800H       ; SET START ADDR (B8000)
085D  8E C0                     MOV     ES,AX           ;
085F  B9 1E00                   MOV     CX,1E00H        ; SET TEST MEM SIZE 15K
0862  BB 0010                   MOV     BX,0010H        ;
0865  BA 01FF                   MOV     DX,S8STATUS     ;
0868  EC                        IN      AL,DX           ;
0869  A8 80                     TEST    AL,VRAM2IN      ;
086B  75 02                     JNZ     VRM1            ;
086D  B3 20                     MOV     BL,20H          ;
086F  53                VRM1:   PUSH    BX              ;
0870  F6 C7 08                  TEST    BH,00001000B    ; 1ST 16KB ?
0873  75 06                     JNZ     VRM2            ; NO -
0875  E4 60                     IN      AL,PORT_A       ;
0877  0C 40                     OR      AL,01000000B    ; SET RETENTION TEST REQ'ED FLAG
0879  E6 60                     OUT     PORT_A,AL       ;                                  ALL     PODSTG
087B  E8 0DEA R         VRM2:   CALL    PODSTG          ; CALL MEM TEST
087E  5B                        POP     BX              ;
087F  74 06                     JZ      VRM3            ;

0881  E8 1007 R                 CALL    PUT_LOGO        ; PUT LOGO ON SCREEN

0884  E9 097F R                 JMP     Q29             ; ERROR

0887  E4 60             VRM3:   IN      AL,PORT_A       ;
0889  24 BF                     AND     AL,10111111B    ; RESET RETENTION TEST REQ'ED FLAG
088B  E6 60                     OUT     PORT_A,AL       ;

088D  B9 2000                   MOV     CX,2000H        ; SET TEST MEM SIZE 16K
0890  BA 03D9                   MOV     DX,PAGREG2      ;
0893  80 C7 08                  ADD     BH,00001000B    ; UPDATE CPU/CRT PAGE
0896  8A C7                     MOV     AL,BH           ;
0898  EE                        OUT     DX,AL           ; SET PAGE REG 2
```

```
0899  3A C3                     CMP     AL,BL           ; END ?
089B  75 D2                     JNE     VRM1            ; NO -

                        ;-----------------------------------------------------------------
                        ; TEST 14                                                       :
                        ;       KJ-ROM AND GAIJI SRAM TEST                              :
                        ; MFG ERROR CODE = 2700  MODULE AT ADDRESS 80000H ERROR          :
                        ;                  2701  MODULE AT ADDRESS 90800H ERROR         :
                        ;                  2702  MODULE AT ADDRESS 91000H ERROR         :
                        ;                  2703  MODULE AT ADDRESS 91800H ERROR         :
                        ;                  27FF  GAIJI RAM R/W ERROR                    :
                        ;-----------------------------------------------------------------

089D  E8 0F7C R                 CALL    MFG_UP          ; SEND 'F1' TO MFG_TESTER
                        ;----- DISABLE VRAM1 & VRAM2 DECODERS AND THEN ENABLE KJ-ROM
08A0  E8 04C2 R                 CALL    S8SET           ;
08A3  09                        DB      009H            ; START REG ADDR
08A4  60                        DB      060H            ; MASK DATA
08A5  02                        DB      2               ; PARAMETER LENGTH
08A6  00                        DB      000H            ; DISABLE VRAM1 DECODER
08A7  00                        DB      000H            ; DISABLE VRAM2 DECODER
08A8  E8 04C2 R                 CALL    S8SET           ;
08AB  07                        DB      007H            ; START REG ADDR
08AC  67                        DB      067H            ; MASK DATA
08AD  01                        DB      1               ; PARAMETER LENGTH
08AE  B0                        DB      0B0H            ; ENABLE KJ-ROM DECODER (80000-BFFFF)
                        ;-------------------------------;
                        ;       TEST KJ-ROM             ;
                        ;-------------------------------;
08AF  B0 00                     MOV     AL,0            ; CLEAR CHECHSUM COUNTER
08B1  33 DB                     XOR     BX,BX           ; BH = 0 : GOOD COMP FLAG
                                                        ; BL = 0 : INITIAL KJ ROM ERR CODE
08B3  BA 8000                   MOV     DX,8000H        ; SET KJ-ROM TOP ADDR (80000H)
08B6  8E DA                     MOV     DS,DX           ;
08B8  33 F6                     XOR     SI,SI           ;
08BA  B9 8000                   MOV     CX,8000H        ; TEST MODULE AT ADDR 80000H
08BD  02 04             KJ2:    ADD     AL,DS:[SI]      ; (CALCULATE CHECKSUM)
08BF  46                        INC     SI              ;
08C0  E2 FB                     LOOP    KJ2             ;
08C2  0A C0                     OR      AL,AL           ; CHECKSUM GOOD ?
08C4  75 38                     JNE     KJERR           ; NO -

08C6  BA 9080                   MOV     DX,9080H        ; TEST MODULE AT ADDR 90800H,
08C9  52                KJ3:    PUSH    DX              ; 91000H, AND 91800H
08CA  B1 10                     MOV     CL,16           ;
08CC  8E DA             KJ4:    MOV     DS,DX           ;-----------------:
08CE  33 F6                     XOR     SI,SI           ;                 :
08D0  51                        PUSH    CX              ;                 :
08D1  B9 0800                   MOV     CX,0800H        ;          16 TIMES LOOP
08D4  02 04             KJ5:    ADD     AL,DS:[SI]      ;------:          :
08D6  46                        INC     SI              ; 2K TIMES LOOP   :
08D7  E2 FB                     LOOP    KJ5             ;------:          :
08D9  59                        POP     CX              ;                 :
08DA  80 C6 02                  ADD     DH,2            ;                 :
08DD  E2 ED                     LOOP    KJ4             ;---:
08DF  5A                        POP     DX              ;

08E0  FE C3                     INC     BL              ; INCREMENT ERROR CODE
08E2  0A C0                     OR      AL,AL           ; CHECKSUM GOOD ?
08E4  75 18                     JNE     KJERR           ; NO -
08E6  81 C2 0080                ADD     DX,0080H        ;
08EA  80 FB 03                  CMP     BL,3            ;
08ED  72 DA                     JB      KJ3             ;
                        ;-------------------------------;
                        ;       TEST GAIJI SRAM         ;
                        ;-------------------------------;
08EF  B5 04             KJ6:    MOV     CH,04H          ; SET TEST ADDR LENGTH (2K)
08F1  B6 88                     MOV     DH,88H          ; SET START ADDR (8800H)
08F3  8E C2                     MOV     ES,DX           ;
08F5  E8 0DEA R                 CALL    PODSTG          ; GAIJI SRAM GOOD ?
08F8  B7 00                     MOV     BH,0            ;   SET GOOD COMP FLAG
08FA  74 04                     JZ      KJ7             ; YES-
08FC  B3 FF                     MOV     BL,0FFH         ;   SET GAIJI SRAM ERROR CODE
08FE  B7 27             KJERR:  MOV     BH,27H          ;
0900  53                KJ7:    PUSH    BX              ;
                        ;----- DISABLE KJ-ROM DECODER AND THEN ENABLE VRAM2 DECODER
0901  E8 04C2 R                 CALL    S8SET           ;
0904  07                        DB      007H            ; START REG ADDR
0905  67                        DB      067H            ; MASK DATA
0906  01                        DB      1               ; PARAMETER LENGTH
0907  00                        DB      000H            ; DIABLE KJ-ROM DECODER
0908  E8 04C2 R                 CALL    S8SET           ;
090B  0A                        DB      00AH            ; START REG ADDR
090C  60                        DB      060H            ; MASK DATA
090D  01                        DB      1               ; PARAMETER LENGTH
090E  B7                        DB      0B7H            ; ENABLE VRAM2 DECODER (B8000-BFFFF)
                        ;-----
090F  E8 1007 R                 CALL    PUT_LOGO        ; PUT LOGO ON SCREEN
                        ;-----
0912  5B                        POP     BX              ;
0913  0A FF                     OR      BH,BH           ; ANY ERROR OCCURRED ?
0915  74 06                     JZ      KJ9             ; NO -

0917  BE 0067 R                 MOV     SI,OFFSET KFNT_ERR ; KANJI ROM OR GAIJI RAM ERROR
091A  E8 0D32 R                 CALL    E_MSG           ;
091D                    KJ9:
                        ;-----------------------------------------------------------------
                        ; TEST 15                                                       :
                        ;       MEMORY SIZE DETERMINE AND TEST                          :
                        ; DESCRIPTION                                                   :
                        ;       THIS ROUTINE WILL DETERMINE HOW MUCH MEM                :
                        ;       IS ATTACHED TO THE SYSTEM (UP TO 512KB)                 :
                        ;       AND SET "MEMORY_SIZE" AND "REAL_MEMORY"                 :
                        ;       WORDS IN THE DATA AREA.                                 :
```

```
;       AFTER THIS, MEMORY WILL BE EITHER TESTED                        :
;       OR CLEARED, DEPENDING ON THE CONTENTS OF                        :
;       "RESET_FLAG".                                                   :
; MFG ERROR CODES = 0AXX PLANAR BOARD ERROR                             :
;                   0BXX 64K RAM CARD ERROR                             :
;                   0CXX ERRORS IN BOTH ODD & EVEN BYTES                :
;                        IN A 128K SYS                                  :
;                   1YXX 128K RAM CARD ERROR                            :
;                    Y=SEGMENT HAVING TROUBLE                           :
;                   1EXX BASE 32K VRAM ERROR                            :
;                   1FXX 32K VRAM CARD ERROR                            :
;                     XX= ERROR BITS                                    :
;-------------------------------------------------------------------------
        ASSUME  DS:DATA
        CALL    DDS                     ;
        CALL    MFG_UP                  ; SEND 'F0' TO MFG_TESTER

        IN      AL,PORT_A               ; RESTORE RAM MAP INFOMATION
        MOV     BX,64                   ;
        TEST    AL,00001000B            ; 64K RAM CARD?
        JZ      Q25_1                   ; NO -
        ADD     BX,64                   ;
Q25_1:  TEST    AL,00000001B            ; 1ST 128K RAM CARD IN?
        JZ      Q25_2                   ; YES-
        ADD     BX,128                  ;
Q25_2:  TEST    AL,00000010B            ; 2ND 128K RAM CARD IN?
        JZ      Q25_3                   ; YES-
        ADD     BX,256                  ;
Q25_3:  MOV     [MEMORY_SIZE],BX        ; SET CONTINUOUSE MEMORY SIZE
        MOV     [TRUE_MEM],BX           ; SET TOTAL MEMORY WORD
;----- SIZE HAS BEEN DETERMINED, NOW TEST OR CLEAR ALL OF MEMORY
        MOV     AX,8                    ; DISPLAY GOOD MEMORY SIZE
        CALL    MSIZE                   ;
        XOR     AX,AX                   ;
        MOV     DX,0200H                ; SET START ADDR (8K BYTE)
        MOV     ES,DX                   ;
        MOV     CX,7000H                ; TEST 56K BYTES
        XOR     DX,DX                   ;
Q26:    PUSH    DX                      ;
        PUSH    BX                      ;
        PUSH    AX                      ;
        IN      AL,PORT_A               ;
        OR      AL,01000000B            ; SET RETENTION TEST REQ'ED FLAG

        OUT     PORT_A,AL               ;
        CALL    PODSTG                  ; TEST OR FILL MEM
        POP     AX                      ;
        POP     BX                      ;
        POP     DX                      ;
        JNZ     Q31                     ; ERROR
Q27:    ADD     AX,64                   ; DISPLAY GOOD MEMORY SIZE
        CALL    MSIZE                   ;
        ADD     DH,10H                  ;
        MOV     ES,DX                   ;
        MOV     CX,8000H                ; TEST SIZE 64K
        CMP     AX,BX                   ; END OF MEMORY ?
        JNE     Q26                     ; NO -
        JMP     Q43                     ; GOTO NEXT TEST
;----- ON ENTRY TO MEMORY ERROR ROUTINE, CX HAS ERROR BITS
;       AH HAS ODD/EVEN INFO, OTHER USEFUL INFO ON THE STACK
Q29:    CMP     BL,10H                  ; BASE 32K VRAM ?
        JNE     Q30                     ; NO -
Q29_1:  MOV     BH,1EH                  ; ERROR CODE - BASE 32K VRAM
        JMP     SHORT Q33               ;
Q30:    CMP     BH,10H                  ; BASE 32K VRAM ?
        JB      Q29_1                   ; YES-
        MOV     BH,1FH                  ; ERROR CODE - 32K VRAM CARD
        JMP     SHORT Q33               ;

Q31:    IN      AL,PORT_A               ;
        TEST    AL,00000011B            ; 128K RAM CARD IN ?
        JZ      Q32                     ; NO -
        PUSH    AX                      ; SAVE
        PUSH    CX                      ;
        MOV     CL,13                   ; FIND END ADDR OF 128K RAM CARDS
        SHL     AX,CL                   ;
        CMP     DX,AX                   ; 128K RAM CARD AREA ?
        POP     CX                      ; RESTORE
        POP     AX                      ;
        JAE     Q32                     ;
        MOV     BL,CL                   ; YES-FORM ERROR BITS ("XX")
        OR      BL,CH
        MOV     CL,4                    ; ROTATE MOST SIGNIFIGANT
                                        ; NIBBLE OF SEGMET
        SHR     DH,CL                   ; TO LOW NIBBLE OF DH
        MOV     BH,10H                  ;
        OR      BH,DH                   ; FORM "1Y" VALUE
        JMP     SHORT Q35               ;
Q32:    MOV     BH,0AH                  ; ERROR 0A....
        TEST    AL,00001000B            ; TEST FOR 64K RAM CARD PRESENT
        JNZ     Q34                     ; WORRY ABOUT ODD/EVEN IF IT IS
Q33:    MOV     BL,CL                   ;
        OR      BL,CH                   ; COMBINE ERROR BITS IF IT ISN'T
        JMP     SHORT Q35               ;
Q34:    CMP     AH,02                   ; EVEN BYTE ERROR? ERR 0AXX
        MOV     BL,CL
        JE      Q35                     ;

        INC     BH                      ; MAKE INTO 0BXX ERR
        OR      BL,CH                   ; MOVE AND COMBINE ERROR BITS
        CMP     AH,1                    ; ODD BYTE ERROR
        JE      Q35                     ;
```

```
091D  E8 0000 E
0920  E8 0F7C R

0923  E4 60
0925  BB 0040
0928  A8 08
092A  74 03
092C  83 C3 40
092F  A8 01
0931  74 04
0933  81 C3 0080
0937  A8 02
0939  74 04
093B  81 C3 0100
093F  89 1E 0013 R
0943  89 1E 0015 R

0947  B8 0008
094A  E8 09D5 R
094D  33 C0
094F  BA 0200
0952  8E C2
0954  B9 7000
0957  33 D2
0959  52
095A  53
095B  50
095C  E4 60
095E  0C 40

0960  E6 60
0962  E8 0DEA R
0965  58
0966  5B
0967  5A
0968  75 27
096A  05 0040
096D  E8 09D5 R
0970  80 C6 10
0973  8E C2
0975  B9 8000
0978  3B C3
097A  75 DD
097C  E9 0A11 R

097F  80 FB 10
0982  75 04
0984  B7 1E
0986  EB 2F
0988  80 FF 10
098B  72 F7
098D  B7 1F
098F  EB 26

0991  E4 60
0993  A8 03
0995  74 1A
0997  50
0998  51
0999  B1 0D
099B  D3 E0
099D  3B D0
099F  59
09A0  58
09A1  73 0E
09A3  8A D9
09A5  0A DD
09A7  B1 04

09A9  D2 EE
09AB  B7 10
09AD  0A FE
09AF  EB 1E
09B1  B7 0A
09B3  A8 08
09B5  75 06
09B7  8A D9
09B9  0A DD
09BB  EB 12
09BD  80 FC 02
09C0  8A D9
09C2  74 0B

09C4  FE C7
09C6  0A DD
09C8  80 FC 01
09CB  74 02
```

```
09CD  FE C7                     INC     BH              ; MUST HAVE BEEN BOTH
                                                        ; - MAKE INTO 0CXX
09CF  BE 005E R         Q35:    MOV     SI,OFFSET MEM_ERR ;
09D2  E8 0D32 R                 CALL    E_MSG           ; LET ERROR ROUTINE FIGURE OUT
                                                        ; WHAT TO DO
                        ;--------------------------------------------------------
                        ; SUBROUTINE
                        ;       MEMORY SIZE PRINT ON CRT
                        ; DESCRIPTION
                        ;       PRINT TESTED MEMORY OK MSG ON THE CRT
                        ; CALL PARMS: AX = K OF GOOD MEMORY (IN HEX)
                        ;--------------------------------------------------------
09D5                    MSIZE   PROC    NEAR
09D5  53                        PUSH    BX
09D6  51                        PUSH    CX
09D7  52                        PUSH    DX
09D8  50                        PUSH    AX              ; SAVE WORK REGS
09D9  B4 02                     MOV     AH,2            ; SET CURSOR TOWARD THE END OF
09DB  BA 0921                   MOV     DX,0921H        ; (ROW 9, COL. 33)
09DE  B7 00                     MOV     BH,0            ; PAGE 0
09E0  CD 10                     INT     INT_10          ;
09E2  58                        POP     AX              ;
09E3  50                        PUSH    AX
09E4  BB 000A                   MOV     BX,10           ; SET UP FOR DECIMAL CONVERT
09E7  B9 0003                   MOV     CX,3            ; OF 3 NIBBLES
09EA  33 D2             Q36:    XOR     DX,DX           ;
09EC  F7 F3                     DIV     BX              ; DEVIDE BY 10
09EE  80 CA 30                  OR      DL,30H          ; MAKE INTO ASCII
09F1  52                        PUSH    DX              ; SAVE
09F2  E2 F6                     LOOP    Q36             ;
09F4  B9 0003                   MOV     CX,3            ;
09F7  58                Q37:    POP     AX              ; RECOVER A NUMBER
09F8  E8 0FA1 R                 CALL    PRT_HEX
09FB  E2 FA                     LOOP    Q37
09FD  B9 0003                   MOV     CX,3
0A00  BE 0056 R                 MOV     SI,OFFSET F3B   ; PRINT " KB"
0A03  2E: 8A 04         Q38:    MOV     AL,CS:[SI]
0A06  46                        INC     SI
0A07  E8 0FA1 R                 CALL    PRT_HEX
0A0A  E2 F7                     LOOP    Q38
0A0C  58                        POP     AX
0A0D  5A                        POP     DX
0A0E  59                        POP     CX
0A0F  5B                        POP     BX
0A10  C3                Q35E:   RET
0A11                    MSIZE   ENDP

0A11                    Q43:
                        ;--------------------------------------------------------
                        ; TEST 16                                               :
                        ;       KEYBOARD TEST                                   :
                        ; DESCRIPTION                                           :
                        ;       NMI HAS BEEN ENABLED FOR QUITE A FEW            :
                        ;       SECONDS NOW. CHECK THAT NO SCAN CODES           :
                        ;       HAVE SHOWN UP IN THE BUFFER. (STUCK             :
                        ;       KEY) IF THEY HAVE, DISPLAY THEM AND             :
                        ;       POST ERROR.                                     :
                        ; MFG ERROR CODE = 2000 STRAY NMI INTERRUPTS OR KBD RCV ERRORS :
                        ;                 21XX CARD FAILURE                     :
                        ;                   XX=01, KB DATA STUCK HIGH           :
                        ;                   XX=02, KB DATA STUCK LOW            :
                        ;                   XX=03, NO NMI INTERRUPT             :
                        ;--------------------------------------------------------
= 0070                  INT_70  EQU     70H
                                ASSUME  DS:DATA
0A11  E8 0F7C R                 CALL    MFG_UP          ; SEND 'EF' TO MFG_TESTER
0A14  E8 0000 E                 CALL    DDS             ; ESTABLISH ADDRESSING
                        ;----- CHECK LINK CARD, IF PRESENT
0A17  BA 0201                   MOV     DX,JOY_PORT     ;
0A1A  EC                        IN      AL,DX           ; BURN-IN MODE ?
0A1B  24 F0                     AND     AL,0F0H         ;
0A1D  74 06                     JZ      F6              ; YES-BYPASS CHECK
0A1F  E4 62                     IN      AL,PORT_C       ; KEYBOARD CABLE ATTACHED ?
0A21  24 80                     AND     AL,10000000B    ;
0A23  75 03                     JNZ     F7              ; NO -
0A25  EB 7D 90          F6:     JMP     F6_X            ; YES-BYPASS CHECK

0A28  E4 61             F7:     IN      AL,PORT_B       ;
0A2A  24 FC                     AND     AL,11111100B    ; DROP SPEAKER DATA
0A2C  E6 61                     OUT     PORT_B,AL       ;
0A2E  B0 B6                     MOV     AL,0B6H         ; MODE SET TIMER 2
0A30  E6 43                     OUT     TIM_CTL,AL      ;
0A32  B0 40                     MOV     AL,040H         ; DISABLE NMI
0A34  E6 A0                     OUT     NMI_PORT,AL     ;
0A36  B0 20                     MOV     AL,32           ; LSB TO TIMER 2
                                                        ; (APPROX. 40Khz VALUE)
0A38  BA 0042                   MOV     DX,TIMER+2
0A3B  EE                        OUT     DX,AL
0A3C  2B C0                     SUB     AX,AX
0A3E  8B C8                     MOV     CX,AX
0A40  EE                        OUT     DX,AL           ; MSB TO TIMER 2 (START TIMER)
0A41  E4 61                     IN      AL,PORT_B       ;
0A43  0C 01                     OR      AL,1            ;
0A45  E6 61                     OUT     PORT_B,AL       ; ENABLE TIMER 2
0A47  E4 62             F7_0:   IN      AL,PORT_C       ; SEE IF KEYBOARD DATA ACTIVE
0A49  24 40                     AND     AL,01000000B    ;
0A4B  75 06                     JNZ     F7_1            ; EXIT LOOP IF DATA SHOWED UP
0A4D  E2 F8                     LOOP    F7_0            ;
0A4F  B3 02                     MOV     BL,02H          ; SET NO KEYBOARD DATA ERROR
0A51  EB 49                     JMP     SHORT F6_1      ; (2102)

0A53  06                F7_1:   PUSH    ES              ; SAVE ES
```

```
0A54  2B C0                    SUB     AX,AX           ; SET UP SEGMENT REG
0A56  8E C0                    MOV     ES,AX           ; M
0A58  26: C7 06 0008 R 0599 C  MOV     ES:[NMI_PTR],OFFSET D11 ; SET UP NEW NMI VECTOR
0A5F  A2 0084 R                MOV     INTR_FLAG,AL    ; RESET INTR FLAG
0A62  E4 61                    IN      AL,PORT_B       ; DISABLE INTERNAL BEEPER TO
0A64  0C 30                    OR      AL,00110000B    ; PREVENT ERROR BEEP
0A66  E6 61                    OUT     PORT_B,AL
0A68  B0 C0                    MOV     AL,0C0H
0A6A  E6 A0                    OUT     NMI_PORT,AL     ; ENABLE NMI
0A6C  B9 0100                  MOV     CX,0100H        ;
0A6F  E2 FE                    LOOP    $               ; WAIT A BIT
0A71  E4 61                    IN      AL,PORT_B       ; RE-ENABLE BEEPER
0A73  24 CF                    AND     AL,11001111B
0A75  E6 61                    OUT     PORT_B,AL
0A77  A0 0084 R                MOV     AL,INTR_FLAG    ; GET INTR FLAG
0A7A  0A C0                    OR      AL,AL           ; WILL BE NON-ZERO IF NMI HAPPENED
0A7C  B3 03                    MOV     BL,03H          ; SET POSSIBLE ERROR CODE
0A7E  26: C7 06 0008 R 0000 C  MOV     ES:[NMI_PTR],OFFSET KBDNMI ; RESET NMI VECTOR
0A85  07                       POP     ES              ; RESTORE ES
0A86  74 14                    JZ      F6_1            ; JUMP IF NO NMI (2103)
0A88  B0 00                    MOV     AL,00H          ; DISABLE FEEDBACK CKT
0A8A  E6 A0                    OUT     NMI_PORT,AL     ;
0A8C  E4 61                    IN      AL,PORT_B       ;
0A8E  24 FE                    AND     AL,11111110B    ; DROP GATE TO TIMER 2
0A90  E6 61                    OUT     PORT_B,AL       ;
0A92  E4 62                F6_2: IN      AL,PORT_C       ; SEE IF KEYBOARD DATA ACTIVE
0A94  24 40                    AND     AL,01000000B    ;
0A96  74 0C                    JZ      F6_X            ; EXIT LOOP IF DATA WENT LOW
0A98  E2 F8                    LOOP    F6_2
0A9A  B3 01                    MOV     BL,01H          ; SET KEYBOARD DATA STUCK HIGH ERR
                                                       ; (2101)
0A9C  B7 21                F6_1: MOV     BH,21H          ; POST ERROR  "21XX"
0A9E  BE 005F R                MOV     SI,OFFSET KEY_ERR ; GET MSG ADDR
0AA1  E8 0D32 R                CALL    E_MSG           ; PRINT MSG ON SCREEN

0AA4  B0 00                F6_X: MOV     AL,00H          ; DISABLE FEEDBACK CKT
0AA6  E6 A0                    OUT     NMI_PORT,AL     ;

0AA8  B8 FF20                  MOV     AX,0FF20H       ; INITIALIZE KANA-KANJI FUNCTION
0AAB  CD 70                    INT     INT_70          ;

0AAD  B8 0703                  MOV     AX,0703H        ; DISABLE INDICATOR ROW DISPLAY
0AB0  CD 16                    INT     INT_16          ;
                       ;-----------------------------------------------------------------
                       ; TEST 17                                                       :
                       ;       CASSETTE INTERFACE TEST                                 :
                       ; DESCRIPTION                                                   :
                       ;       TURN CASSETTE MOTOR OFF. WRITE A BIT OUT TO THE         :
                       ;       CASSETTE DATA BUS. VERIFY THAT CASSETTE DATA            :
                       ;       READ IS WITHIN A VALID RANGE.                           :
                       ; MFG ERROR CODE = 2300 (DATA PATH ERROR)                       :
                       ;                  23FF (RELAY FAILED TO PICK)                  :
                       ;-----------------------------------------------------------------
= 0A9A                 MAX_PERIOD      EQU     0A9AH   ;NOM.+10%
= 08AD                 MIN_PERIOD      EQU     08ADH   ;NOM -10%
                       ;----- TURN THE CASSETTE MOTOR OFF
0AB2  E8 0F7C R                CALL    MFG_UP          ; SEND 'EE' TO MFG_TESTER
0AB5  E4 61                    IN      AL,PORT_B
0AB7  0C 09                    OR      AL,00001001B    ; SET TIMER 2 SPK OUT, AND CASSETTE
0AB9  E6 61                    OUT     PORT_B,AL       ; OUT BITS ON, CASSETTE MOT OFF
                       ;----- WRITE A BIT
0ABB  E4 21                    IN      AL,INTA01
0ABD  0C 01                    OR      AL,01H          ; DISABLE TIMER INTERRUPTS
0ABF  E6 21                    OUT     INTA01,AL
0AC1  B0 B6                    MOV     AL,0B6H         ; SEL TIM 2, LSB, MSB, MD 3
0AC3  E6 43                    OUT     TIMER+3,AL      ; WRITE 8253 CMD/MODE REG
0AC5  B8 04D2                  MOV     AX,1234         ; SET TIMER 2 CNT FOR 1000 USEC
0AC8  E6 42                    OUT     TIMER+2,AL      ; WRITE TIMER 2 COUNTER REG
0ACA  8A C4                    MOV     AL,AH           ; WRITE MSB
0ACC  E6 42                    OUT     TIMER+2,AL
0ACE  2B C9                    SUB     CX,CX           ; CLEAR COUNTER FOR LONG DELAY
0AD0  E2 FE                    LOOP    $               ; WAIT FOR COUNTER TO INIT
                       ;----- READ CASSETTE INPUT
0AD2  E4 62                    IN      AL,PORT_C       ; READ VALUE OF CASS IN BIT
0AD4  24 10                    AND     AL,10H          ; ISOLATE FROM OTHER BITS
0AD6  A2 006B R                MOV     LAST_VAL,AL
0AD9  E8 0000 E                CALL    READ_HALF_BIT   ; TO SET UP CONDITIONS FOR CHECK
0ADC  E8 0000 E                CALL    READ_HALF_BIT
0ADF  E3 3E                    JCXZ    F8              ; CAS_ERR
0AE1  53                       PUSH    BX              ; SAVE HALF BIT TIME VALUE
0AE2  E8 0000 E                CALL    READ_HALF_BIT
0AE5  58                       POP     AX              ; GET TOTAL TIME
0AE6  E3 37                    JCXZ    F8              ; CAS_ERR
0AE8  03 C3                    ADD     AX,BX
0AEA  3D 0A9A                  CMP     AX,MAX_PERIOD
0AED  73 30                    JAE     F8              ; CAS_ERR
0AEF  3D 08AD                  CMP     AX,MIN_PERIOD
0AF2  72 2B                    JB      F8              ;
0AF4  BA 0201                  MOV     DX,JOY_PORT     ;
0AF7  EC                       IN      AL,DX
0AF8  24 F0                    AND     AL,0F0H         ; DETERMINE MODE
0AFA  3C 10                    CMP     AL,00010000B    ; MFG?
0AFC  74 04                    JE      F9
0AFE  3C 40                    CMP     AL,01000000B    ; SERVICE?
0B00  75 26                    JNE     T16_END         ; GO TO NEXT TEST IF NOT

                       ;----- CHECK THAT CASSETTE RELAY IS PICKING (CAN'T DO TEST IN NORMAL
                       ;      MODE BECAUSE OF POSSIBILITY OF WRITING ON CASSETTE IF "RECORD"
                       ;      BUTTON IS DEPRESSED.)
0B02  E4 61                F9:     IN      AL,PORT_B
0B04  8A D0                    MOV     DL,AL           ; SAVE PORT B CONTENTS
0B06  24 E5                    AND     AL,11100101B    ; SET CASSETTE MOTOR ON
0B08  E6 61                    OUT     PORT_B,AL       ;
```

```
0B0A  33 C9                     XOR     CX,CX
0B0C  E2 FE                     LOOP    $               ; WAIT FOR RELAY TO SETTLE
0B0E  E8 0000 E                 CALL    READ_HALF_BIT
0B11  E8 0000 E                 CALL    READ_HALF_BIT
0B14  8A C2                     MOV     AL,DL           ; DROP RELAY
0B16  E6 61                     OUT     PORT_B,AL
0B18  E3 0E                     JCXZ    T16_END         ; READ_HALF_BIT SHOULD TIME OUT IN
                                                        ; THIS SITUATION
0B1A  BB 23FF                   MOV     BX,23FFH        ; ERROR 23FF
0B1D  EB 03                     JMP     SHORT F81
0B1F                    F8:                             ; CAS_ERR
0B1F  BB 2300                   MOV     BX,2300H        ; ERR. CODE 2300H
0B22  BE 0060 R         F81:    MOV     SI,OFFSET CASS_ERR ; CASSETTE WRAP FAILED
0B25  E8 0D32 R                 CALL    E_MSG           ; GO PRINT ERROR MSG
0B28  E4 21             T16_END:IN      AL,INTA01       ;
0B2A  24 FE                     AND     AL,0FEH         ; ENABLE TIMER INTS
0B2C  E6 21                     OUT     INTA01,AL
0B2E  E4 A0                     IN      AL,NMI_PORT     ; CLEAR NMI FLIP/FLOP
0B30  B0 80                     MOV     AL,80H          ; ENABLE NMI INTERUPTS
0B32  E6 A0                     OUT     NMI_PORT,AL
                        ;--------------------------------------------------------------------
                        ; TEST 18                                                          :
                        ;       SERIAL PORT (2FX) TEST                                     :
                        ; DESCRIPTION:                                                     :
                        ;       CHECKS IF THE SERIAL CARD IS ATTACHED. IF NOT, EXITS.      :
                        ;       VERIFIES THAT THE SERIAL CARD UART FUNCTIONS PROPERLY.      :
                        ;       ERROR CODES RETURNED BY 'UART' RANGE FROM 1 TO 1FH AND     :
                        ;       ARE REPORTED VIA REGISTER BL.  SEE LISTING OF 'UART'       :
                        ;       FOR POSSIBLE ERRORS.                                       :
                        ; MFG. ERROR CODES = 23XX FOR SERIAL PORT 2FX                       :
                        ;--------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
                        ;--------------------------------------------------------------------
                        ;       TEST SERIAL CARD INS8250 UART (2FX)
                        ;--------------------------------------------------------------------
0B34  E8 0F7C R                 CALL    MFG_UP          ; SEND 'ED' TO MFG_TESTER
0B37  E8 0000 E                 CALL    DDS             ; POINT TO DATA AREA
0B3A  E4 62                     IN      AL,PORT_C       ; TEST FOR SERIAL CARD PRESENT
0B3C  24 02                     AND     AL,00000010B    ; ONLY CONCERNED WITH BIT 1
0B3E  75 17                     JNZ     TM              ; IT'S NOT THERE
0B40  BA 02F8                   MOV     DX,RS232_2_PORT ; ADDRESS OF SERIAL CARD
0B43  89 16 0000 R              MOV     RS232_BASE,DX   ; SAVE RS232 CARD ADDR
0B47  80 0E 0011 R 02           OR      BYTE PTR EQUIP_FLAG+1,02H  ; SET EQUIPMENT FLAG
0B4C  E8 111D R                 CALL    UART            ; ASYNCH. COMM. ADAPTER TEST
0B4F  73 06                     JNC     TM              ; PASSED
0B51  BE 0061 R                 MOV     SI,OFFSET COM1_ERR ; CODE FOR DISPLAY
0B54  E8 0D32 R                 CALL    E_MSG           ; REPORT ERROR
0B57                    TM:
                        ;--------------------------------------------------------------------
                        ; TEST 19                                                          :
                        ;       PARALLEL PORT TEST                                         :
                        ; MFG ERROR CODE = 28XX  DATA PORT WRAP ERROR                       :
                        ;                  29XX  CONTROL PORT WRAP ERROR                   :
                        ;                   (XX  ERROR BITS)                               :
                        ;--------------------------------------------------------------------
0B57  E8 0F7C R                 ASSUME  CS:CODE,DS:DATA ;
0B5A  E8 0000 E                 CALL    MFG_UP          ; SEND 'EC' TO MFG_TESTER
                                CALL    DDS             ; POINT TO DATA AREA

                        ;----- CHECK DATA PORT
0B5D  0E                        PUSH    CS              ; SET DS
0B5E  07                        POP     ES              ;
0B5F  BF 0BAC R                 MOV     DI,OFFSET PPD   ; SET TEST PATARN ADDR (DATA)
0B62  BA 037A                   MOV     DX,PARAL_PORT+2 ; SET CONTROL PORT
0B65  B0 08                     MOV     AL,08H          ; SET SELECT IN SIGNAL
0B67  EE                        OUT     DX,AL           ;
0B68  B2 78                     MOV     DL,LOW PARAL_PORT ; SET DATA PORT
0B6A  B7 28                     MOV     BH,28H          ; SET ERROR CODE 28XX
0B6C  B9 0004                   MOV     CX,4            ; SET LOOP COUNTER
0B6F  26: 8A 05         PP1:    MOV     AL,ES:[DI]      ; LOAD TEST PATRAN
0B72  47                        INC     DI              ;
0B73  8A E0                     MOV     AH,AL           ;
0B75  EE                        OUT     DX,AL           ; OUT DATA
0B76  50                        PUSH    AX              ;  DELAY
0B77  58                        POP     AX              ;
0B78  EC                        IN      AL,DX           ; INPUT DATA
0D79  32 E0                     XOR     AH,AL           ; COMPARE DATA

0B7B  75 37                     JNZ     PERR            ; OK ?
0B7D  E2 F0                     LOOP    PP1             ;

                        ;----- CHECK CONTROL PORT
0B7F  BA 0201                   MOV     DX,JOY_PORT     ; SERVICE MODE LOOP ?
0B82  EC                        IN      AL,DX           ;
0B83  24 F0                     AND     AL,0F0H         ;
0B85  3C 20                     CMP     AL,00100000B    ;
0B87  74 08                     JE      PP2             ; YES-
0B89  81 3E 0072 R 3412         CMP     RESET_FLAG,3412H ; WARM START WITH "CTRL+ALT+INS"?
0B8F  75 28                     JNE     PP4             ;  NO -GOTO NEXT TEST

0B91  BA 037A           PP2:    MOV     DX,PARAL_PORT+2 ; SET CONTROL PORT WRITE
0B94  B7 29                     MOV     BH,29H          ; SET ERROR CODE 29XX
0B96  B1 04                     MOV     CL,4            ; SET LOOP COUNTER
0B98  26: 8A 05         PP3:    MOV     AL,ES:[DI]      ; LOAD TEST PATRAN
0B9B  47                        INC     DI              ;
0B9C  8A E0                     MOV     AH,AL           ; DATA SAVE
0B9E  EE                        OUT     DX,AL           ; OUT DATA
0B9F  50                        PUSH    AX              ;  DELAY
0BA0  58                        POP     AX              ;
0BA1  EC                        IN      AL,DX           ; INPUT DATA
0BA2  24 1F                     AND     AL,1FH          ; DROP BIT 7-5

0BA4  32 E0                     XOR     AH,AL           ; COMPARE DATA
```

```
0BA6  75 0C                     JNZ     PERR            ; OK ?
0BA8  E2 EE                     LOOP    PP3             ;
0BAA  EB 10                     JMP     SHORT PP4       ;

0BAC  FF AA 55 00       PPD:    DB      0FFH,0AAH,055H,000H  ; DATA PORT TEST PATARN
0BB0  1B 00 1C 0C               DB      01BH,000H,01CH,00CH  ; CONTROL PORT TEST PATARN

0BB4  8A DC             PERR:   MOV     BL,AH           ; SET ERROR BITS
0BB6  BE 0066 R                 MOV     SI,OFFSET PRT_ERR ; PARALLEL PRINTER ERROR
0BB9  E8 0D32 R                 CALL    E_MSG           ;
0BBC                    PP4:
                        ;--------------------------------------------
                        ;       SETUP HARDWARE INT. VECTOR TABLE
                        ;--------------------------------------------
                                ASSUME  CS:CODE,DS:ABS0
0BBC  2B C0                     SUB     AX,AX
0BBE  8E C0                     MOV     ES,AX
0BC0  B9 0008                   MOV     CX,08           ; GET VECTOR CNT
0BC3  0E                        PUSH    CS              ; SETUP DS REG
0BC4  1F                        POP     DS
0BC5  BE 15C0 R                 MOV     SI,OFFSET VECTOR_TABLE
0BC8  BF 0020 R                 MOV     DI,OFFSET INT_PTR
0BCB  A5                F7A:    MOVSW
0BCC  47                        INC     DI              ; SKIP OVER SEGMENT
0BCD  47                        INC     DI
0BCE  E2 FB                     LOOP    F7A
                        ;==============================
0BD0  E4 60             F7B:    IN      AL,PORT_A       ; SAVE CONTENTS OF CONF. FLAG
0BD2  50                        PUSH    AX              ; (RESTORE AT LABEL F17)
0BD3  8B E8                     MOV     BP,AX           ;
                        ;----- SET UP OTHER INTERRUPTS AS NECESSARY
                                ASSUME  DS:ABS0
0BD5  8E D9                     MOV     DS,CX
0BD7  C7 06 0014 R 0000 E   C   MOV     INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
0BDD  C7 06 0120 R 0000 E   C   MOV     KEY62_PTR,OFFSET KEY62_INT ; 62 KEY CONVERSION
                                                                   ; ROUTINE
0BE3  C7 06 0060 R 0000 E   C   MOV     BASIC_PTR,OFFSET BAS_ENT ; CASSETTE BASIC ENTRY
0BE9  0E                        PUSH    CS
0BEA  58                        POP     AX
0BEB  A3 0062 R                 MOV     WORD PTR BASIC_PTR+2,AX ; CODE SEGMENT FOR CASSETTE
                        ;--------------------------------------------------------------------
                        ; TEST 20                                                          :
                        ;       OPTIONAL ROM TEST                                          :
                        ; DESCRIPTION                                                      :
                        ;       CHECK FOR OPTIONAL ROM FROM C0000 TO F8000 IN 2K BLOCKS    :
                        ;       (A VALID MODULE HAS 'AA55' OR '55AA' IN THE 1ST 2 LOC'S,   :
                        ;       LENGTH INDICATOR (LENGTH/512) IN THE 3RD LOCATION AND      :
                        ;       TEST/INIT. CODE STARTING IN THE 4TH LOCATION.)             :
                        ; MFG ERROR CODE = 25XX (XX=MSB OF SEGMENT THAT HAS CRC CHECK)     :
                        ;--------------------------------------------------------------------
0BEE  B0 01                     MOV     AL,01H          ; MFG HART BEAT
0BF0  E6 17                     OUT     MFG_PORT+7,AL   ;
0BF2  E8 0F7C R                 CALL    MFG_UP          ; SEND 'EB' TO MFG_TESTER

0BF5  BA C000                   MOV     DX,0C000H       ; SET BEGINNING ADDRESS
0BF8                    ROM_SCAN_1:
0BF8  8E DA                     MOV     DS,DX
0BFA  2B DB                     SUB     BX,BX           ; SET BX=0000
0BFC  8B 0F                     MOV     CX,[BX]         ; GET 1ST WORD FROM MODULE
0BFE  53                        PUSH    BX              ; BUS SETTLING
0BFF  5B                        POP     BX
0C00  81 F9 AA55                CMP     CX,0AA55H       ; ID FOUND (PCJR MODE) ?
0C04  74 06                     JE      OROM1           ;

0C06  81 F9 55AA                CMP     CX,55AAH        ; (NATIVE MODE)
0C0A  75 05                     JNE     NEXT_ROM        ; NO -
0C0C  E8 0EC4 R         OROM1:  CALL    ROM_CHECK       ; GO CHECK OUT MODULE
0C0F  EB 04                     JMP     SHORT ARE_WE_DONE ; CHECK FOR END OF ROM SPACE
0C11                    NEXT_ROM:
0C11  81 C2 0080                ADD     DX,0080H        ; POINT TO NEXT 2K ADDRESS
0C15                    ARE_WE_DONE:
0C15  81 FA F800                CMP     DX,0F800H       ; CARTRIDGE AREA END ?
0C19  72 DD                     JB      ROM_SCAN_1      ; NO -GO CHECK ANOTHER ADDR
                        ;--------------------------------------------------------------------
                        ; TEST 21                                                          :
                        ;       DISKETTE ATTACHMENT TEST                                   :
                        ; DESCRIPTION                                                      :
                        ;       1. VERIFY WATCHDOG TIMER IN CASE OF POWER ON ENTRY.        :
                        ;       2. VERIFY STATUS OF FDC AFTER A RESET.                     :
                        ;       3. DETERMINE NUMBER OF DISKETTE DRIVES.                    :
                        ;       4. COMPLETE SYSTEM INITIALIZATION THEN PASS CONTROL        :
                        ;          TO THE BOOT LOADER PROGRAM                              :
                        ; MFG ERROR CODES = 2601 RESET TO DISKETTE CONTROLLER CD. FAILED    :
                        ;                  2603 WATCHDOG TIMER FAILED                      :
                        ;--------------------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
0C1B  E8 0F7C R                 CALL    MFG_UP          ; SEND 'EA' TO MFG_TESTER
0C1E  E8 0000 E                 CALL    DDS             ; POINT TO DATA AREA
0C21  1E                        PUSH    DS              ; INIT DISKETTE SCRATCHPADS
0C22  07                        POP     ES              ;
0C23  BF 0074 R                 MOV     DI,OFFSET TRACK0;
0C26  B8 FFFF                   MOV     AX,0FFFFH       ;
0C29  AB                        STOSW                   ;
0C2A  AB                        STOSW                   ;
0C2B  E4 62                     IN      AL,PORT_C       ; DISKETTE ADAPTER PRESENT?
0C2D  24 04                     AND     AL,00000100B
0C2F  74 03                     JZ      F10             ;
0C31  EB 7B 90                  JMP     F15             ; NO - BYPASS DISKETTE TEST
0C34  83 3E 0072 R 00   F10:    CMP     RESET_FLAG,0    ; RUNNING FROM POWER-ON STATE?
0C39  75 0E                     JNE     F11             ; BYPASS WATCHDOG TEST
0C3B  B0 0A                     MOV     AL,00001010B    ; READ INT. REQUEST REGISTER CMD
0C3D  E6 20                     OUT     INTA00,AL       ;
0C3F  E4 20                     IN      AL,INTA00
```

```
0C41  24 40                     AND     AL,01000000B    ; HAS WATCHDOG GONE OFF?
0C43  75 04                     JNZ     F11             ; PROCEED IF IT HAS
0C45  B3 03                     MOV     BL,03H          ; SET ERROR CODE
0C47  EB 11                     JMP     SHORT F13
0C49  B0 80             F11:    MOV     AL,FDC_RESET    ;
0C4B  E6 F2                     OUT     NEC_CTL,AL      ; DISABLE WATCHDOG TIMER
0C4D  B4 00                     MOV     AH,0            ; RESET NEC FDC
0C4F  8A D4                     MOV     DL,AH           ; SET FOR DRIVE 0
0C51  CD 13                     INT     INT_13          ; VERIFY STATUS AFTER RESET
0C53  F6 C4 FF                  TEST    AH,0FFH         ; STATUS OK?
0C56  B3 01                     MOV     BL,01H          ;  SET ERROR CODE
0C58  74 08                     JZ      F14             ; YES-
0C5A  B7 26             F13:    MOV     BH,26H          ; DSK_ERR:(26XX)
0C5C  BE 0065 R                 MOV     SI,OFFSET DISK_ERR ; GET ADDR OF MSG
0C5F  E8 0D32 R                 CALL    E_MSG           ; GO PRINT ERROR MSG
                        ;----- DETERMINE NUMBER OF DISKETTE DRIVES
0C62  B8 0181           F14:    MOV     AX,0000000110000001B ; AH-DRIVE CONFIG. FLAG
                                                        ; AL-TURN MOTOR ON
0C65  B2 00                     MOV     DL,0            ; SELECT DRIVE 0
0C67  E6 F2             F14_1:  OUT     NEC_CTL,AL      ; WRITE FDC CONTROL REG
0C69  50                        PUSH    AX              ;
0C6A  52                        PUSH    DX              ;

0C6B  2B C9                     SUB     CX,CX           ; WAIT FOR 0.5 SECOND
0C6D  E2 FE                     LOOP    $               ;
0C6F  E2 FE                     LOOP    $               ;
0C71  B5 01                     MOV     CH,1            ; SELECT TRACK 1
0C73  88 16 003E R              MOV     SEEK_STATUS,DL  ;
0C77  E8 0000 E                 CALL    SEEK            ; RECALIBRATE DISKETTE
0C7A  5A                        POP     DX              ;
0C7B  58                        POP     AX              ;
0C7C  72 13                     JC      F14_2           ;
0C7E  08 26 0302 R              OR      BYTE PTR JEQUIP_FLAG,AH ; SET DISKETTE CONFIG.
0C82  B1 06                     MOV     CL,6            ; SET 0 OF DISKETTE
0C84  8A EA                     MOV     CH,DL           ;
0C86  D2 E5                     SHL     CH,CL           ;
0C88  80 26 0010 R 3F           AND     BYTE PTR EQUIP_FLAG,3FH ;
0C8D  08 2E 0010 R              OR      BYTE PTR EQUIP_FLAG,CH  ;
0C91  D1 E0             F14_2:  SHL     AX,1            ;
0C93  80 E4 FE                  AND     AH,11111110B    ;
0C96  0C 80                     OR      AL,FDC_RESET    ;
0C98  42                        INC     DX              ;
0C99  80 FA 04                  CMP     DL,4            ;
0C9C  72 C9                     JB      F14_1           ;
0C9E  F6 06 0302 R 0F           TEST    BYTE PTR JEQUIP_FLAG,0FH ; DISKETTE DRIVE ATTACHED ?
0CA3  74 05                     JZ      F14_3           ; NO -
0CA5  80 0E 0010 R 01           OR      BYTE PTR EQUIP_FLAG,01H ; SET IPL DISKETTE INDICATOR
                        ;----- TURN MOTOR OFF
0CAA  B0 80             F14_3:  MOV     AL,FDC_RESET    ; TURN DRIVE 0 MOTOR OFF
0CAC  E6 F2                     OUT     NEC_CTL,AL      ;
0CAE  C6 06 0084 R 00   F15:    MOV     INTR_FLAG,00H   ; SET STRAY INTERRUPT FLAG = 00
0CB3  BF 0078 R                 MOV     DI,OFFSET PRINT_TIM_OUT ;SET DEFAULT PRT TIMEOUT
0CB6  1E                        PUSH    DS
0CB7  07                        POP     ES
0CB8  B8 1414                   MOV     AX,1414H        ; DEFAULT=20
0CBB  AB                        STOSW
0CBC  AB                        STOSW
0CBD  B8 0101                   MOV     AX,0101H        ;RS232 DEFAULT=01
0CC0  AB                        STOSW
0CC1  AB                        STOSW

0CC2  E4 21                     IN      AL,INTA01
0CC4  24 FE                     AND     AL,0FEH         ; ENABLE TIMER INT. (LVL 0)
0CC6  E6 21                     OUT     INTA01,AL

0CC8  58                        POP     AX              ; RESTORE & SAVE CONF. FLAG
0CC9  50                        PUSH    AX              ;
0CCA  A8 80                     TEST    AL,10000000B    ; PCJR MODE ?
0CCC  75 20                     JNZ     F17             ; YES-BYPASS BEEP
                                ASSUME  DS:XXDATA
0CCE  1E                        PUSH    DS
0CCF  E8 0FE2 R                 CALL    DDX             ;
0CD2  80 3E 0018 R 00           CMP     POST_ERR,00H    ; CHECK FOR "POST_ERR" NON-ZERO
                                ASSUME  DS:DATA
0CD7  1F                        POP     DS              ; RESTORE DS
0CD8  74 0F                     JE      F16             ; CONTINUE IF NO ERROR

0CDA  B2 02                     MOV     DL,2            ; 2 SHORT BEEPS (ERROR)
0CDC  E8 0FAB R                 CALL    ERR_BEEP
0CDF                    ERR_WAIT:
0CDF  B4 00                     MOV     AH,00
0CE1  CD 16                     INT     INT_16          ; WAIT FOR "ENTER" KEY
0CE3  3C 0D                     CMP     AL,0DH          ;
0CE5  75 F8                     JNE     ERR_WAIT
0CE7  EB 05                     JMP     SHORT F17       ;
                        ;-----
0CE9  B2 01             F16:    MOV     DL,1            ; 1 SHORT BEEP (NO ERRORS)
0CEB  E8 0FAB R                 CALL    ERR_BEEP
                        ;================================
0CEE  58                F17:    POP     AX              ; RESTORE CONTENTS OF CONF. FLAG
0CEF  86 E0                     XCHG    AH,AL           ; (SAVED AT LABEL F7B)
                        ;================================
0CF1  B0 00                     MOV     AL,0            ; CLEAR KBD PORT
0CF3  E6 60                     OUT     PORT_A,AL       ;

0CF5  BA 0201                   MOV     DX,JOY_PORT     ; GET MFG./ SERVICE  MODE INFO
0CF8  EC                        IN      AL,DX           ;
0CF9  24 F0                     AND     AL,0F0H         ; IS HIGH ORDER NIBBLE = 0?
0CFB  75 03                     JNZ     F19             ; (BURN-IN MODE)
0CFD  E9 006B R         F18:    JMP     START           ; YES-GOTO BEGINNING OF POST
0D00  3C 20             F19:    CMP     AL,00100000B    ; SERVICE MODE LOOP?
0D02  74 F9                     JE      F18             ; YES-BRANCH TO START
0D04  81 3E 0072 R 4321         CMP     RESET_FLAG,4321H ; DIAG. CONTROL PROGRAM START?
```

```
0D0A  74 19                     JE      F21             ; YES-
0D0C  3C 10                     CMP     AL,00010000B    ; MFG DCP RUN REQUEST?
0D0E  74 15                     JE      F21             ; YES-

                        ;----- GOTO PCjr SYSTEM CARTRIDGE
0D10  F6 C4 80                  TEST    AH,10000000B    ; PCJR MODE ?
0D13  74 03                     JZ      F20             ; NO -
0D15  E9 0349 R                 JMP     SYSSWAP         ; YES-GO SYSTEM SWAP RTN.
                        ;----- GOTO BOOT LOADER
0D18  B8 0019           F20:    MOV     AX,0019H        ; CLEAR SCREEN (GRAPHIC MODE)
0D1B  CD 10                     INT     INT_10          ; (MODE: 320 X 200 - COLOR)

0D1D  C7 06 0072 R 1234         MOV     RESET_FLAG,1234H ; SET WARM START INDICATOR IN CASE
                                                        ; OF CARTRIDGE RESET
0D23  CD 19                     INT     INT_19          ; GOTO THE BOOT LOADER

                        ;----- GOTO DIAGNOSTIC TESTS OR MFG TESTS
                                ASSUME  DS:ABS0
0D25  FA                F21:    CLI                     ; DISABLE EXTERNAL INTERRUPTS
0D26  2B C0                     SUB     AX,AX
0D28  8E D8                     MOV     DS,AX           ; RESET TIMER INT.
0D2A  C7 06 0020 R 0000 E   C   MOV     WORD PTR INT_PTR,OFFSET TIMER_INT ;

0D30  CD 80                     INT     INT_80          ; ENTER DCP THROUGH INT. 80H
                        ;--------------------------------------------------------------------
                        ; SUBROUTINE
                        ;       GENERAL ERROR HANDLER FOR THE POST
                        ; ENTRY REQUIREMENTS:
                        ;       SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
                        ;       BX= ERROR CODE FOR MANUFACTURING OR SERVICE MODE
                        ;       REGISTERS ARE NOT PRESERVED
                        ;       LOCATION "POST_ERR" IS SET NON-ZERO IF AN ERROR OCCURS IN
                        ;       CUSTOMER MODE
                        ; SERVICE/MANUFACTURING FLAGS AS FOLLOWS: (HIGH NIBBLE OF PORT 201)
                        ;       0000 = MANUFACTURING MODE (BURN-IN)
                        ;       0001 = MANUFACTURING MODE (SYSTEM TEST)
                        ;       0010 = SERVICE MODE (LOOP POST)
                        ;       0100 = SERVICE MODE (SYSTEM TEST)
                        ;--------------------------------------------------------------------
0D32                    E_MSG   PROC    NEAR
0D32  8A C7                     MOV     AL,BH           ; SEND DATA TO ADDR 11 & 12
0D34  E6 11                     OUT     MFG_PORT+1,AL   ; SEND HIGH BYTE
0D36  8A C3                     MOV     AL,BL           ;
0D38  E6 12                     OUT     MFG_PORT+2,AL   ; SEND LOW BYTE

0D3A  BA 0201                   MOV     DX,JOY_PORT     ;
0D3D  EC                        IN      AL,DX           ; GET MODE BITS
0D3E  24 F0                     AND     AL,0F0H         ; ISOLATE BITS OF INTEREST
0D40  74 65                     JZ      MFG_OUT         ; MANUFACTURING MODE (BURN-IN)
0D42  3C 10                     CMP     AL,00010000B    ;
0D44  74 61                     JE      MFG_OUT         ; MFG. MODE (SYSTEM TEST)
0D46  8A F0                     MOV     DH,AL           ; SAVE MODE
0D48  80 FF 0A                  CMP     BH,0AH          ; PRE-CRT ATTACHMENT TEST ERROR ?
0D4B  72 72                     JB      BEEPS           ; YES-DO BEEP OUTPUT
0D4D  53                        PUSH    BX              ; SAVE ERROR AND MODE FLAGS
0D4E  56                        PUSH    SI
0D4F  52                        PUSH    DX
0D50  B4 02                     MOV     AH,2            ; SET CURSOR
0D52  BA 0801                   MOV     DX,0801H        ; ROW 8, COL. 1
0D55  B7 00                     MOV     BH,0            ; PAGE 0
0D57  CD 10                     INT     INT_10          ;
0D59  BE 0059 R                 MOV     SI,OFFSET ERROR_ERR
0D5C  B9 0005                   MOV     CX,5            ; PRINT WORD "ERROR"
0D5F  2E: 8A 04         EM_0:   MOV     AL,CS:[SI]
0D62  46                        INC     SI
0D63  E8 0FA1 R                 CALL    PRT_HEX
0D66  E2 F7                     LOOP    EM_0
                        ;----- LOOK FOR A BLANK SPACE TO POSSIBLY PUT CUSTOMER LEVEL ERRORS
                        ;       IN CASE OF MULTI ERROR)
0D68  B2 07                     MOV     DL,7            ; SET CURSOR
0D6A  B4 02             EM_1:   MOV     AH,2            ; ROW 8, COL 7 (OR ABOVE, IF
0D6C  B7 00                     MOV     BH,0            ; MULTIPLE ERRORS)
0D6E  CD 10                     INT     INT_10          ;
0D70  B4 08                     MOV     AH,8            ; READ CHARACTER THIS POSITION
0D72  CD 10                     INT     INT_10          ;
0D74  3C 47                     CMP     AL,'G'          ; ERROR "G" ?

0D76  74 06                     JE      EM_1_0          ; YES-
0D78  42                        INC     DX              ; POINT TO NEXT POSTION
0D79  42                        INC     DX              ;
0D7A  3C 20                     CMP     AL,' '          ; BLANK?
0D7C  75 EC                     JNE     EM_1            ; GO CHECK NEXT POSITION, IF NOT
0D7E  5A                EM_1_0: POP     DX              ; RECOVER ERROR POINTERS
0D7F  5E                        POP     SI
0D80  5B                        POP     BX
0D81  80 FE 20                  CMP     DH,00100000B    ; SERVICE MODE?
0D84  74 12                     JE      SVC_OUT         ;
0D86  80 FE 40                  CMP     DH,01000000B    ;
0D89  74 0D                     JE      SVC_OUT         ;
0D8B  2E: 8A 04                 MOV     AL,CS:[SI]      ; GET ERROR CHARACTER
0D8E  E8 0FA1 R                 CALL    PRT_HEX         ; DISPLAY IT
0D91  80 FF 20                  CMP     BH,20H          ; ERROR BELOW 20? (MEM TROUBLE?)
0D94  73 1F                     JAE     EM_2            ; NO -
0D96  EB 0A                     JMP     SHORT BP2       ; YES-HALT SYSTEM IF SO.

0D98  8A C7             SVC_OUT:MOV     AL,BH           ; PRINT MSB
0D9A  E8 0F90 R                 CALL    XPC_BYTE        ; DISPLAY IT
0D9D  8A C3                     MOV     AL,BL           ; PRINT LSB
0D9F  E8 0F90 R                 CALL    XPC_BYTE

0DA2  B2 02             BP2:    MOV     DL,2            ; 2 SHORT BEEPS
0DA4  E8 0FAB R                 CALL    ERR_BEEP        ;
0DA7                    MFG_OUT:
```

```
0DA7  FA
0DA8  E4 61
0DAA  24 FC
0DAC  E6 61
0DAE  2A C0
0DB0  E6 F2
0DB2  E6 A0
0DB4  F4

0DB5  1E
0DB6  E8 0FE2 R
0DB9  88 3E 0018 R
0DBD  1F

0DBE  C3

0DBF  FA
0DC0  8C C8
0DC2  8E D0

0DC4  B2 02
0DC6  BC 0050 R
0DC9  B3 01
0DCB  E9 0FC0 R
0DCE  33 C9
0DD0  E2 FE

0DD2  FE CA
0DD4  75 F3
0DD6  80 FF 05
0DD9  75 CC
0DDB  80 FE 20
0DDE  74 05
0DE0  80 FE 40
0DE3  75 C2
0DE5  B3 01
0DE7  E9 0FC0 R
0DEA
```

```
TOTLTPO:CLI                     ; DISABLE INTS.
        IN      AL,PORT_B       ; DISABLE SOUND
        AND     AL,0FCH         ;
        OUT     PORT_B,AL       ;
        SUB     AL,AL           ; STOP DISKETTE MOTOR
        OUT     NEC_CTL,AL      ;
        OUT     NMI_PORT,AL     ; DISABLE NMI
        HLT                     ; HALT

        ASSUME  DS:XXDATA       ;
EM_2:   PUSH    DS
        CALL    DDX             ;
        MOV     POST_ERR,BH     ; SET ERROR FLAG NON-ZERO
        POP     DS
        ASSUME  DS:NOTHING
        RET                     ; RETURN TO CALLER

BEEPS:  CLI
        MOV     AX,CS           ; SET CODE SEG= STACK SEG
        MOV     SS,AX           ; (STACK IS LOST, BUT THINGS ARE
                                ; OVER, ANYWAY)
        MOV     DL,2            ; 2 BEEPS
        MOV     SP,OFFSET EX_0  ; SET DUMMY RETURN
EB:     MOV     BL,1            ; SHORT BEEP
        JMP     BEEP            ;
EB0:    XOR     CX,CX           ; WAIT (BEEP OFF)
        LOOP    $               ;

        DEC     DL              ; DONE YET?
        JNZ     EB              ; LOOP IF NOT
        CMP     BH,05H          ; 64K CARD ERROR?
        JNE     TOTLTPO         ; END IF NOT
        CMP     DH,00100000B    ; SERVICE MODE?
        JE      EB1
        CMP     DH,01000000B    ;
        JNE     TOTLTPO         ; END IF NOT
EB1:    MOV     BL,1            ; ONE MORE BEEP FOR 64K ERROR IF IN
        JMP     BEEP            ; SERVICE MODE (JUMP TO TOTLTPO)
E_MSG   ENDP
;--------------------------------------------------------------------
; SUBROUTINE
;       THIS ROUTINE PERFORMS A READ/WRITE TEST ON A BLOCK OF
;       STORAGE (MAX. SIZE = 64KB). IF "WARM START", FILL
;       BLOCK WITH 0000 AND RETURN.
; DATA PATTERNS USED:
;       0->FF ON ONE BYTE TO TEST DATA BUS
;       AAAA,5555,00FF,FF00 FOR ALL WORDS
;       FILL WITH 0000 BEFORE EXIT
; ON ENTRY:
;       ES = ADDRESS OF STORAGE TO BE TESTED
;       DS = ADDRESS OF STORAGE TO BE TESTED
;       CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED
;            (MAX. = 8000H (32K WORDS))
; ON EXIT:
;       ZERO FLAG = OFF IF STORAGE ERROR
;       IF ZERO FLAG = OFF, THEN CX = XOR'ED BIT PATTERN
;       OF THE EXPECTED DATA PATTERN VS. THE ACTUAL DATA
;       READ. (I.E., A BIT "ON" IN AL IS THE bIT IN ERROR)
;       AH=03 IF BOTH BYTES OF WORD HAVE ERRORS
;       AH=02 IF LOW (EVEN) BYTE HAS ERROR
;       AH=01 IF HI (ODD) BYTE HAS ERROR
; AX,BX,CX,DX,DI,SI,BP ARE ALL DESTROYED.
;--------------------------------------------------------------------
```

```
0DEA
```

```
PODSTG  PROC    NEAR
        ASSUME  DS:ABS0
        CLD                     ; SET DIRECTION TO INCREMENT
        SUB     DI,DI           ; SET DI=0000 REL. TO START OF SEG
        SUB     AX,AX           ; INITIAL DATA PATTERN FOR 00-FF
                                ; TEST
        MOV     DS,AX           ; SET DS TO ABS0
        MOV     BX,DATA_WORD[RESET_FLAG-RS232_BASE] ; WARM START?
        CMP     BX,1234H
        MOV     DX,ES
        MOV     DS,DX           ; RESTORE DS
        JNE     P1
P12:    REP     STOSW           ; SIMPLE FILL WITH 0 ON WARM-START
        MOV     DS,AX
        MOV     DATA_WORD[RESET_FLAG-RS232_BASE],BX
        MOV     DS,DX           ; RESTORE DS
        RET                     ; AND EXIT
;-----
P1:     CMP     BX,4321H        ; DIAG. RESTART?
        JE      P12             ; DO FILL WITH ZEROS
        CMP     BX,3412H        ; WARM START WITH RS232C EXT WRAP?
        JE      P12             ; DO FILL WITH ZEROS
;-----
P2:     MOV     [DI],AL         ; WRITE TEST DATA
        MOV     AL,[DI]         ; GET IT BACK
        XOR     AL,AH           ; COMPARE TO EXPECTED
        JZ      PY
        MOV     AH,0            ;

        JMP     P8              ; ERROR EXIT IF MISCOMPARE
PY:     INC     AH              ; FORM NEW DATA PATTERN
        MOV     AL,AH           ;
        JNZ     P2              ; LOOP TILL ALL 256 DATA PATTERNS
;-----                           ;  DONE
        MOV     BP,CX           ; SAVE WORD COUNT
        MOV     AX,0AAAAH       ; LOAD DATA PATTERN
        MOV     BX,AX
        MOV     DX,05555H       ; LOAD OTHER DATA PATTERN
        REP     STOSW           ; FILL WORDS FROM LOW TO HIGH
                                ; WITH AAAA
```

```
0DEA  FC
0DEB  2B FF
0DED  2B C0

0DEF  8E D8
0DF1  8B 1E 0472 R
0DF5  81 FB 1234
0DF9  8C C2
0DFB  8E DA
0DFD  75 0B
0DFF  F3/ AB
0E01  8E D8
0E03  89 1E 0472 R
0E07  8E DA
0E09  C3

0E0A  81 FB 4321
0E0E  74 EF
0E10  81 FB 3412
0E14  74 E9

0E16  88 05
0E18  8A 05
0E1A  32 C4
0E1C  74 05
0E1E  B4 00

0E20  E9 0EAF R
0E23  FE C4
0E25  8A C4
0E27  75 ED

0E29  8B E9
0E2B  B8 AAAA
0E2E  8B D8
0E30  BA 5555
0E33  F3/ AB
```

```
0E35  4F                DEC     DI                      ; POINT TO LAST WORD WRITTEN
0E36  4F                DEC     DI
0E37  FD                STD                             ; SET DIRECTION FLAG TO GO DOWN
0E38  8B F7             MOV     SI,DI                   ; SET INDEX REGS. EQUAL
0E3A  8B CD             MOV     CX,BP                   ; RECOVER WORD COUNT
0E3C                                                    ; GO FROM HIGH TO LOW
0E3C  AD                P3:     LODSW                   ; GET WORD FROM MEMORY
0E3D  33 C3             XOR     AX,BX                   ; EQUAL WHAT S/B THERE?
0E3F  75 6E             JNZ     P8                      ; GO ERROR EXIT IF NOT
0E41  8B C2             MOV     AX,DX                   ; GET 55 DATA PATTERN
0E43  AB                STOSW                           ;  STORE IT IN LOCATION JUST READ
0E44  E2 F6             LOOP    P3                      ; LOOP TILL ALL BYTES DONE
0E46  8B CD             MOV     CX,BP                   ; RECOVER WORD COUNT
0E48  FC                CLD                             ; BACK TO INCREMENT
0E49  46                INC     SI                      ; ADJUST PTRS
0E4A  46                INC     SI
0E4B  8B FE             MOV     DI,SI
0E4D  8B DA             MOV     BX,DX                   ; S/B DATA PATTERN TO BX
                        ;-----
0E4F  BA 0201           MOV     DX,JOY_PORT             ; NORMAL MODE ?
0E52  EC                IN      AL,DX                   ;
0E53  8B D3             MOV     DX,BX                   ;
0E55  24 F0             AND     AL,0F0H                 ;
0E57  3C F0             CMP     AL,0F0H                 ;
0E59  74 26             JE      P4                      ; YES-BYPASS 00FF & FF00 PATTERN TEST
                        ;-----
0E5B  BA 00FF           MOV     DX,00FFH                ; DATA FOR CHECKERBOARD PATTERN
0E5E  AD                PX:     LODSW                   ; GET WORD FROM MEMORY
0E5F  33 C3             XOR     AX,BX                   ; EQUAL WHAT S/B THERE?
0E61  75 4C             JNZ     P8                      ; GO ERROR EXIT IF NOT
0E63  8B C2             MOV     AX,DX                   ; GET OTHER PATTERN
0E65  AB                STOSW                           ; STORE IT IN LOCATION JUST READ
0E66  E2 F6             LOOP    PX                      ; LOOP TILL ALL BYTES DONE
0E68  8B CD             MOV     CX,BP                   ; RECOVER WORD COUNT
0E6A  FD                STD                             ; DECREMENT
0E6B  4E                DEC     SI                      ; ADJUST PTRS
0E6C  4E                DEC     SI
0E6D  8B FE             MOV     DI,SI
0E6F  8B DA             MOV     BX,DX                   ; S/B DATA PATTERN TO BX
0E71  F7 D2             NOT     DX                      ; MAKE PATTERN FF00
0E73  0A D2             OR      DL,DL                   ; FIRST PASS?
0E75  74 E7             JZ      PX
0E77  FC                CLD                             ; INCREMENT
0E78  83 C6 04          ADD     SI,4
0E7B  F7 D2             NOT     DX
0E7D  8B FE             MOV     DI,SI
0E7F  8B CD             MOV     CX,BP
0E81                                                    ; LOW TO HIGH
0E81  AD                P4:     LODSW                   ; GET A WORD
0E82  33 C2             XOR     AX,DX                   ; SHOULD COMPARE TO DX
0E84  75 29             JNZ     P8                      ; GO ERROR IF NOT
0E86  AB                STOSW                           ; WRITE 0000 BACK TO LOCATION
                                                        ; JUST READ
0E87  E2 F8             LOOP    P4                      ; LOOP TILL DONE
0E89  FD                STD                             ; BACK TO DECREMENT
0E8A  4E                DEC     SI                      ; ADJUST POINTER DOWN TO LAST WORD
0E8B  4E                DEC     SI                      ; WRITTEN
                        ;----- CHECK IF IN SERVICE/MFG MODES, IF SO, PERFORM REFRESH CHECK
0E8C  BA 0201           MOV     DX,JOY_PORT             ;
0E8F  EC                IN      AL,DX                   ; GET OPTION BITS
0E90  24 F0             AND     AL,0F0H                 ;
0E92  3C F0             CMP     AL,0F0H                 ; ALL BITS HIGH=NORMAL MODE
0E94  74 0E             JE      P6
0E96  E4 60             IN      AL,PORT_A               ; RETENTION TEST REQ'ED ?
0E98  A8 40             TEST    AL,01000000B            ;
0E9A  74 08             JZ      P6                      ; NO -
                        ;----- WAIT ABOUT 12 SECONDS WITHOUT ACCESSING MEMORY
                        ;       IF REFRESH IS NOT WORKING PROPERLY, THIS SHOULD
                        ;       BE ENOUGH TIME FOR SOME DATA TO GO SOUR.
0E9C  B0 31             MOV     AL,49                   ; SET OUTER LOOP COUNT
0E9E  E2 FE             P5:     LOOP    $
0EA0  FE C8             DEC     AL
0EA2  75 FA             JNZ     P5

0EA4  8B CD             P6:     MOV     CX,BP           ; RECOVER WORD COUNT
0EA6  AD                P7:     LODSW                   ; GET WORD
0EA7  0B C0             OR      AX,AX                   ; = TO 0000
0EA9  75 04             JNZ     P8                      ; ERROR IF NOT
0EAB  E2 F9             LOOP    P7                      ; LOOP TILL DONE
0EAD  EB 13             JMP     SHORT P11               ; THEN EXIT

0EAF  8B C8             P8:     MOV     CX,AX           ; SAVE BITS IN ERROR
0EB1  32 E4             XOR     AH,AH
0EB3  0A ED             OR      CH,CH                   ; HIGH BYTE ERROR?
0EB5  74 02             JZ      P9
0EB7  FE C4             INC     AH                      ; SET HIGH BYTE ERROR
0EB9  0A C9             P9:     OR      CL,CL           ; LOW BYTE ERROR?
0EBB  74 03             JZ      P10
0EBD  80 C4 02          ADD     AH,2
0EC0  0A E4             P10:    OR      AH,AH           ; SET ZERO FLAG=0 (ERROR INDICATION
0EC2  FC                P11:    CLD                     ; SET DIR FLAG BACK TO INCREMENT
0EC3  C3                RET                             ; RETURN TO CALLER
0EC4                    PODSTG  ENDP
                        ;--------------------------------------------------------------------
                        ; SUBROUTINE
                        ;       OPTIONAL ROM CHECK
                        ; DESCRIPTION
                        ;       THIS ROUTINE  CHECKS OPTIONAL ROM MODULES (CHECKSUM
                        ;       FOR MODULES FROM C0000->CFFFF, CRC CHECK FOR CARTRIDGES
                        ;       FROM D0000->F7FFF).
                        ;       IF CHECK IS OK, CALLS INIT/TEST CODE IN MODULE.
                        ; MFG ERROR CODE = 25XX (XX=MSB OF SEGMENT IN ERROR)
                        ;--------------------------------------------------------------------
```

```
0EC4
0EC4  8B C5                   ROM_CHECK       PROC    NEAR
0EC6  A8 80                           MOV     AX,BP           ;
0EC8  74 08                           TEST    AL,10000000B    ; PCJR MODE ?
0ECA  81 F9 AA55                      JZ      ROMC1           ; NO -
0ECE  75 08                           CMP     CX,0AA55H       ; ROM ID = AA55H(WORD) ?
0ED0  EB 19                           JNE     ROMC2           ; NO -
0ED2  81 F9 55AA                      JMP     SHORT ROMC3     ; YES-
0ED6  74 13                   ROMC1:  CMP     CX,55AAH        ; ROM ID = 55AAH(WORD) ?
0ED8  81 FA D000                      JE      ROMC3           ; YES-
0EDC  72 0D                   ROMC2:  CMP     DX,0D000H       ; OPTIONAL ROM AREA ?
                                      JB      ROMC3           ; YES-

0EDE  8A DE                           MOV     BL,DH           ;
0EE0  B7 25                           MOV     BH,25H          ;
0EE2  BE 0068 R                       MOV     SI,OFFSET INVC_ERR ; INVALID CART. ERROR
0EE5  BA F800                         MOV     DX,0F800H       ;
0EE8  52                              PUSH    DX              ;
0EE9  EB 45                           JMP     SHORT ROM_CHECK_0  ; GOTO ERROR HANDLER
                              ;-----
0EEB  2B F6                   ROMC3:  SUB     SI,SI           ; SET SI TO POINT TO BEGINNING
                                                              ; (REL. TO DS)
0EED  2A C0                           SUB     AL,AL           ; ZERO OUT AL
0EEF  8A 67 02                        MOV     AH,[BX+2]       ; GET LENGTH INDICATOR
0EF2  D1 E0                           SHL     AX,1            ; FORM COUNT
0EF4  50                              PUSH    AX              ; SAVE COUNT
0EF5  81 FA D000                      CMP     DX,0D000H       ; SEE IF POINTER IS BELOW D000
0EF9  9C                              PUSHF                   ; SAVE RESULTS
0EFA  B1 04                           MOV     CL,4            ; ADJUST
0EFC  D3 E8                           SHR     AX,CL           ;
0EFE  03 D0                           ADD     DX,AX           ; SET POINTER TO NEXT MODULE
0F00  9D                              POPF                    ; RECOVER FLAGS FROM POINTER RANGE
                                                              ; CHECK
0F01  59                              POP     CX              ; RECOVER COUNT IN CX REGISTER
0F02  52                              PUSH    DX              ; SAVE POINTER
0F03  72 07                           JB      ROM_1           ; DO ARITHMETIC CHECKSUM IF BELOW
                                                              ; D0000
0F05  E8 0F53 R                       CALL    CRC_CHECK       ; DO CRC CHECK
0F08  74 2B                           JZ      ROM_CHECK_1     ; PROCEED IF OK
0F0A  EB 05                           JMP     SHORT ROM_2     ; ELSE POST ERROR
0F0C  E8 01AD R               ROM_1:  CALL    ROS_CHECKSUM    ; DO ARITHMETIC CHECKSUM
0F0F  74 24                           JZ      ROM_CHECK_1     ; PROCEED IF OK
0F11  B4 02                   ROM_2:  MOV     AH,2

0F13  B7 00                           MOV     BH,0            ; PAGE 0
0F15  BA 081C                         MOV     DX,081CH        ; POSITION CURSOR, ROW 8, COL 28
0F18  CD 10                           INT     INT_10          ;
0F1A  8C DA                           MOV     DX,DS           ; RECOVER DATA SEG
0F1C  8A C6                           MOV     AL,DH           ;
0F1E  E8 0F90 R                       CALL    XPC_BYTE        ; DISPLAY MSB OF DATA SEG
0F21  8A DE                           MOV     BL,DH           ; FORM XX VALUE OF ERROR CODE
0F23  B7 25                           MOV     BH,25H          ; FORM 25 PORTION
0F25  80 FE D0                        CMP     DH,0D0H         ; IN CARTRIDGE SPACE?
0F28  BE 0064 R                       MOV     SI,OFFSET CART_ERR
0F2B  73 03                           JAE     ROM_CHECK_0     ;
0F2D  BE 0063 R                       MOV     SI,OFFSET ROM_ERR
0F30                          ROM_CHECK_0:
0F30  E8 0D32 R                       CALL    E_MSG           ; GO ERROR ROUTINE
0F33  EB 1C                           JMP     SHORT ROM_CHECK_END   ; AND EXIT
0F35                          ROM_CHECK_1:
0F35  B8 ---- R                       MOV     AX,XXDATA       ; SET ES TO POINT TO XXDATA AREA
0F38  8E C0                           MOV     ES,AX           ;
0F3A  8B C5                           MOV     AX,BP           ; PCJR MODE ?
0F3C  A8 80                           TEST    AL,10000000B    ;
0F3E  75 11                           JNZ     ROM_CHECK_END   ; YES-
0F40  26: C7 06 0014 R 0003 C         MOV     ES:IO_ROM_INIT,0003H  ; LOAD OFFSET
0F47  26: 8C 1E 0016 R                MOV     ES:IO_ROM_SEG,DS      ; LOAD SEGMENT
0F4C  26: FF 1E 0014 R                CALL    DWORD PTR ES:IO_ROM_INIT ; CALL INIT./TEST ROUTINE
0F51                          ROM_CHECK_END:
0F51  5A                              POP     DX              ; RECOVER POINTER
0F52  C3                              RET                     ; RETURN TO CALLER
0F53                          ROM_CHECK       ENDP
                              ;--------------------------------------------------------------
                              ; SUBROUTINE
                              ;       CRC CHECK/GENERATION
                              ; DESCRIPTION
                              ;       CRC CHECK/GENERATION ROUTINE ROUTINE
                              ;       TO CHECK A ROM MODULE USING THE POLYNOMIMAL:
                              ;                X16 + X12 + X5 + 1
                              ; CALLING PARAMETERS:
                              ;       DS = DATA SEGMENT OF ROM SPACE TO BE CHECKED
                              ;       SI = INDEX OFFSET INTO DS POINTING TO 1ST BYTE
                              ;       CX = LENGTH OF SPACE TO BE CHECKED (INCLUDING CRC BYTES)
                              ; ON EXIT:
                              ;       ZERO FLAG = SET = CRC CHECKED OK
                              ;       AH = 00
                              ;       AL = ??
                              ;       BX = 0000
                              ;       CL = 04
                              ;       DX = 0000 IF CRC CHECKED OK, ELSE, ACCUMULATED CRC
                              ;       SI = (SI(ENTRY)+BX(ENTRY)
                              ;--------------------------------------------------------------
0F53                          CRC_CHECK       PROC    NEAR
                                      ASSUME  DS:NOTHING
0F53  8B D9                           MOV     BX,CX           ; SAVE COUNT
0F55  BA FFFF                         MOV     DX,0FFFFH       ; INIT. ENCODE REGISTER
0F58  FC                              CLD                     ; SET DIR FLAG TO INCREMENT

0F59  32 E4                           XOR     AH,AH           ; INIT. WORK REG HIGH
0F5B  B1 04                           MOV     CL,4            ; SET ROTATE COUNT
0F5D  AC                      CRC_1:  LODSB                   ; GET A BYTE
0F5E  32 F0                           XOR     DH,AL           ; FORM Aj + Cj + 1
0F60  8A C6                           MOV     AL,DH
0F62  D3 C0                           ROL     AX,CL           ; SHIFT WORK REG BACK 4
```

```
0F64  33 D0                     XOR     DX,AX           ; ADD INTO RESULT REG
0F66  D1 C0                     ROL     AX,1            ; SHIFT WORK REG BACK 1
0F68  86 F2                     XCHG    DH,DL           ; SWAP PARTIAL SUM INTO RESULT REG
0F6A  33 D0                     XOR     DX,AX           ; ADD WORK REG INTO RESULTS
0F6C  D3 C8                     ROR     AX,CL           ; SHIFT WORK REG OVER 4
0F6E  24 E0                     AND     AL,11100000B    ; CLEAR OFF (EFGH)
0F70  33 D0                     XOR     DX,AX           ; ADD (ABCD) INTO RESULTS
0F72  D1 C8                     ROR     AX,1            ; SHIFT WORK REG ON OVER (AH=0 FOR
                                                        ; NEXT PASS)
0F74  32 F0                     XOR     DH,AL           ; ADD (ABCD INTO RESULTS LOW)
0F76  4B                        DEC     BX              ; DECREMENT COUNT
0F77  75 E4                     JNZ     CRC_1           ; LOOP TILL COUNT = 0000
0F79  0B D2                     OR      DX,DX           ; DX S/B = 0000 IF O.K.
0F7B  C3                        RET                     ; RETURN TO CALLER
0F7C                    CRC_CHECK       ENDP
                        ;-----------------------------------------------------------------
                        ; SUBROUTINE
                        ;       MFG CHECKPOINT HANDLER
                        ; DESCRIPTION
                        ;       MFG CHECHPOINT DATA IS LOADED FROM SAVE AREA, DATA IS
                        ;       DECREMENTED BY ONE, AND THEN SAVED.
                        ;-----------------------------------------------------------------
0F7C                    MFG_UP  PROC    NEAR
0F7C  50                        PUSH    AX
0F7D  1E                        PUSH    DS
                                ASSUME  DS:XXDATA
0F7E  B8 ---- R                 MOV     AX,XXDATA
0F81  8E D8                     MOV     DS,AX
0F83  A0 0005 R                 MOV     AL,MFG_TST      ; GET MFG CHECKPOINT
0F86  E6 10                     OUT     MFG_PORT,AL     ; OUTPUT IT TO TESTER
0F88  FE C8                     DEC     AL              ; DROP IT BY 1 FOR THE NEXT TEST
0F8A  A2 0005 R                 MOV     MFG_TST,AL
                                ASSUME  DS:ABS0
0F8D  1F                        POP     DS
0F8E  58                        POP     AX
0F8F  C3                        RET
0F90                    MFG_UP  ENDP
                                ASSUME  CS:CODE,DS:DATA
                        ;-----------------------------------------------------------------
                        ; SUBROUTINE
                        ;       CONVERT AND PRINT ASCII CODE
                        ;
                        ; NOTE: AL MUST CONTAIN NUMBER TO BE CONVERTED.
                        ;       AX AND BX DESTROYED.
                        ;-----------------------------------------------------------------
0F90                    XPC_BYTE        PROC    NEAR
0F90  50                        PUSH    AX              ; SAVE FOR LOW NIBBLE DISPLAY
0F91  B1 04                     MOV     CL,4            ; SHIFT COUNT
0F93  D2 E8                     SHR     AL,CL           ; NIBBLE SWAP
0F95  E8 0F9B R                 CALL    XLAT_PR         ; DO THE HIGH NIBBLE DISPLAY
0F98  58                        POP     AX              ; RECOVER THE NIBBLE
0F99  24 0F                     AND     AL,0FH          ; ISOLATE TO LOW NIBBLE
                                                        ; FALL INTO LOW NIBBLE CONVERSION
0F9B                    XLAT_PR PROC    NEAR            ; CONVERT 00-0F TO ASCII CHARACTER
0F9B  04 90                     ADD     AL,090H         ; ADD FIRST CONVERSION FACTOR
0F9D  27                        DAA                     ; ADJUST FOR NUMERIC AND ALPHA
                                                        ; RANGE
0F9E  14 40                     ADC     AL,040H         ; ADD CONVERSION AND ADJUST LOW
                                                        ; NIBBLE
0FA0  27                        DAA                     ; ADJUST HIGH NIBBLE TO ASCII RANGE
0FA1                    PRT_HEX PROC    NEAR
0FA1  53                        PUSH    BX
0FA2  B4 0E                     MOV     AH,14           ; DISPLAY CHARACTER IN AL
0FA4  BB 000F                   MOV     BX,15           ; (WHITE)
0FA7  CD 10                     INT     INT_10          ; CALL VIDEO_IO
0FA9  5B                        POP     BX
0FAA  C3                        RET

0FAB                    PRT_HEX ENDP
0FAB                    XLAT_PR ENDP
0FAB                    XPC_BYTE        ENDP
                        ;-----------------------------------------------------------------
                        ; SUBROUTINE
                        ;       BEEP ON ERROR
                        ; DESCRIPTION
                        ;       THIS ROUTINE ISSUES SHORT TONES TO INDICATE FAILURES THAT
                        ;       1: OCCUR BEFORE THE CRT IS STARTED,
                        ;       2: TO CALL THE OPERATORS ATTENTION TO AN ERROR
                        ;          AT THE END OF POST, OR
                        ;       3: TO SIGNAL THE SUCCESSFUL COMPLETION OF POST
                        ; ENTRY PARAMETERS:
                        ;       DL = NUMBER OF APPROX. 1/2 SEC TONES TO SOUND
                        ;-----------------------------------------------------------------
0FAB                    ERR_BEEP  PROC  NEAR
0FAB  9C                        PUSHF                   ; SAVE FLAGS
0FAC  53                        PUSH    BX
0FAD  FA                        CLI                     ; DISABLE SYSTEM INTERRUPTS
0FAE                    G3:                             ; SHORT_BEEP:
0FAE  B3 01                     MOV     BL,1            ; COUNTER FOR A SHORT BEEP
0FB0  E8 0FC0 R                 CALL    BEEP            ; DO THE SOUND
0FB3  E2 FE                     LOOP    $               ; DELAY BETWEEN BEEPS
0FB5  FE CA                     DEC     DL              ; DONE WITH SHORTS
0FB7  75 F5                     JNZ     G3              ; DO SOME MORE
0FB9  E2 FE                     LOOP    $               ; LONG DELAY BEFORE RETURN
0FBB  E2 FE                     LOOP    $               ;
0FBD  5B                        POP     BX              ; RESTORE ORIG CONTENTS OF BX
0FBE  9D                        POPF                    ; RESTORE FLAGS TO ORIG SETTINGS
0FBF  C3                        RET                     ; RETURN TO CALLER
0FC0                    ERR_BEEP        ENDP
                        ;-----------------------------------------------------------------
                        ; SUBROUTINE
                        ;       SOUND BEEP
                        ;-----------------------------------------------------------------
```

```
0FC0
0FC0  B0 B6           BEEP    PROC    NEAR
0FC2  E6 43                   MOV     AL,10110110B    ; SEL TIM 2,LSB,MSB,BINARY
0FC4  B8 0533                 OUT     TIMER+3,AL      ; WRITE THE TIMER MODE REG
0FC7  E6 42                   MOV     AX,533H         ; DIVISOR FOR 1000 HZ
0FC9  8A C4                   OUT     TIMER+2,AL      ; WRITE TIMER 2 CNT - LSB
0FCB  E6 42                   MOV     AL,AH
0FCD  E4 61                   OUT     TIMER+2,AL      ; WRITE TIMER 2 CNT - MSB
0FCF  8A E0                   IN      AL,PORT_B       ; GET CURRENT SETTING OF PORT
0FD1  0C 03                   MOV     AH,AL           ; SAVE THAT SETTING
0FD3  E6 61                   OR      AL,03           ; TURN SPEAKER ON
0FD5  2B C9                   OUT     PORT_B,AL
0FD7  E2 FE                   SUB     CX,CX           ; SET CNT TO WAIT 500 MS
0FD9  FE CB           G7:     LOOP    $               ; DELAY BEFORE TURNING OFF
0FDB  75 FA                   DEC     BL              ; DELAY CNT EXPIRED?
0FDD  8A C4                   JNZ     G7              ; NO - CONTINUE BEEPING SPK
0FDF  E6 61                   MOV     AL,AH           ; RECOVER VALUE OF PORT
0FE1  C3                      OUT     PORT_B,AL
                              RET                     ; RETURN TO CALLER

0FE2                  BEEP    ENDP
                      ;--------------------------------------------------------
                      ; SUBROUTINE
                      ;       SET DS TO POINT TO XXDATA SEGMENT
                      ;--------------------------------------------------------
0FE2                  DDX     PROC    NEAR                    ;
0FE2  50                      PUSH    AX                      ;
0FE3  B8 ---- R               MOV     AX,XXDATA               ;
0FE6  8E D8                   MOV     DS,AX                   ;
0FE8  58                      POP     AX                      ;
0FE9  C3                      RET                             ;
0FEA                  DDX     ENDP                            ;
                      ;--------------------------------------------------------
                      ; SUBROUTINE                                           ;
                      ;       SAVE KBD SCAN CODE                             ;
                      ; DESCRIPTION                                          ;
                      ;       TO SAVE ANY SCAN CODE RECEIVED BY THE NMI ROUTINE ;
                      ;       (PASSED IN AL) DURING POST IN THE KEYBOARD BUFFER ;
                      ;       CALLED THROUGH INT. 48H                        ;
                      ; MFG ERROR CODE = 2000H                               ;
                      ;--------------------------------------------------------
0FEA                  KEY_SCAN_SAVE   PROC    FAR
                              ASSUME  DS:DATA
0FEA  E8 0000 E               CALL    DDS             ; POINT DS TO DATA AREA
0FED  BE 0304 R               MOV     SI,OFFSET KB_BUFFER_J ; POINT TO FIRST LOC. IN BUFR
0FF0  88 04                   MOV     [SI],AL         ; SAVE SCAN CODE
0FF2  8B C4                   MOV     AX,SP           ; CHECK FOR STACK UNDERFLOW
0FF4  80 E4 E0                AND     AH,11100000B    ; (THESE BITS WILL BE 111 IF
                                                      ; UNDERFLOW HAPPEND)
0FF7  74 0D                   JZ      KS_1
0FF9  32 C0                   XOR     AL,AL
0FFB  E6 A0                   OUT     NMI_PORT,AL     ; SHUT OFF NMI
0FFD  BB 2000                 MOV     BX,2000H        ; ERROR CODE 2000H
1000  8E 005F R               MOV     SI,OFFSET KEY_ERR ; POST MESSAGE
1003  E8 0D32 R               CALL    E_MSG           ; AND HALT SYSTEM
1006  CF              KS_1:   IRET                    ; RETURN TO CALLER
1007                  KEY_SCAN_SAVE   ENDP
                      SUBTTL  LOGO --- LOGO DISPLAY SUBROUTINE
                      ;--------------------------------------------------------
                      ; SUBROUTINE                                           ;
                      ;       PUT LOGO ON SCREEN                             ;
                      ; DESCRIPTION                                          ;
                      ;       THIS PROC DISPLAYS IBM LOGO, A MESSAGE, AND A COLOR BAR ;
                      ;       ON THE SCREEN                                  ;
                      ;--------------------------------------------------------
= 0002                LINENO          EQU     2       ; LINE NUMBER OF ONE ROW

= 008C                BACKROW         EQU     140     ; UPPER LIMIT ROW OF BLUE BACK SCREEN

= 006A                LOGOSROW        EQU     106     ; LOGO START ROW POSITION
= 00D6                LOGOSCOL        EQU     214     ; LOGO START COLUMN POSITION

= 10FB                START_ADDR      EQU     LOGOSROW*160/4+LOGOSCOL/2 ; REGEN OFFSET ADDR

= 9874                WHITE_BOX1      EQU     9874H   ; 2 BYTE CODE OF WHITE BOX

= 0001                BLUE            EQU     1       ; COLOR CODE OF BLUE
= 000F                WHITE           EQU     15      ; COLOR CODE OF WHITE

= 00FF                EOL             EQU     0FFH    ; END OF LINE
= 00FF                EOF             EQU     0FFH    ; END OF FILE

= 0002                VSETCSR         EQU     02H     ; REQUEST SET CURSOR POSITION
= 000C                VWRTDOT         EQU     0CH     ; REQUEST WRITE DOT
= 000E                VWRTTTY         EQU     0EH     ; REQUEST WRITE TTY

1007                  PUT_LOGO PROC   NEAR
1007  50                      PUSH    AX              ; SAVE AX, BX & CX
1008  53                      PUSH    BX              ;
1009  51                      PUSH    CX              ;

100A  B8 0019                 MOV     AX,0019H        ; CLEAR SCREEN (GRAPHIC MODE)
100D  CD 10                   INT     INT_10          ; (MODE: 320 X 200 - COLOR)
100F  B4 01                   MOV     AH,1            ; ERASE CURSOR
1011  B9 2000                 MOV     CX,2000H        ;
1014  CD 10                   INT     INT_10          ;

1016  33 DB                   XOR     BX,BX           ; DISABLE SCREEN
1018  B8 1000           LOGO:  MOV     AX,1000H        ;
101B  CD 10                   INT     INT_10          ;
101D  43                      INC     BX              ;
101E  80 FB 10                CMP     BL,16           ;
1021  72 F5                   JB      LOGO            ;
```

```
                        ;----- PAINT BACKGROUND SCREEN
                                MOV     SI,REGEN_START  ; SET REGEN BUFFER ADDRESS
1023  BE B800                  MOV     ES,SI           ;   TO ES
1026  8E C6
                                MOV     AX,BLUE*1000H OR BLUE*100H OR BLUE*10H OR BLUE
1028  B8 1111
                                XOR     DI,DI           ; START ROW/COLUMN IS 0/0
102B  33 FF                     MOV     CX,BACKROW*320/2/2/4; 2 PEL/BYTE,2 WORD/BYTE,4 SCAN
102D  B9 0AF0
                                MOV     DH,4            ; SCAN COUNT IN REGEN
1030  B6 04              LOGO1:  PUSH    DI              ;
1032  57                       PUSH    CX              ;
1033  51                       REP     STOSW           ; WRITE BLUE CX WORD
1034  F3/ AB                   POP     CX              ;
1036  59                       POP     DI              ;
1037  5F                       ADD     DI,2000H        ;
1038  81 C7 2000               DEC     DH              ; ALL DONE ?
103C  FE CE                    JNZ     LOGO1           ;
103E  75 F2
                        ;----- WRITE COLOR BAR
                                XOR     BH,BH           ;
1040  32 FF                     MOV     DX,0900H        ; SET ROW/COLUMN POSITION TO 9-0
1042  BA 0900            LOGO2:  MOV     AH,VSETCSR      ; SET CURSOR POSITION
1045  B4 02                     INT     INT_10          ;
1047  CD 10                     MOV     BL,BLUE         ; SET START COLOR AS BLUE
1049  B3 01              LOGO3:  MOV     AH,VWRTTTY      ;
104B  B4 0E                     MOV     AL,HIGH WHITE_BOX1 ;
104D  B0 98                     INT     INT_10          ;
104F  CD 10                     MOV     AH,VWRTTTY      ;
1051  B4 0E                     MOV     AL,LOW WHITE_BOX1 ;
1053  B0 74                     INT     INT_10          ;
1055  CD 10                     INC     BL              ; SET NEXT COLOR
1057  FE C3                     CMP     BL,16           ; ALL COLOR WRITEN ?
1059  80 FB 10                  JB      LOGO3           ; NO
105C  72 ED                     INC     DH              ; NEXT ROW
105E  FE C6                     CMP     DH,0AH          ;
1060  80 FE 0A                  JBE     LOGO2           ;
1063  76 E0
                        ;----- WRITE IBM PATTERN
                                MOV     BX,OFFSET IBM   ; POINT DATA AREA
1065  BB 10C8 R                 MOV     DX,LOGOSROW     ; GET START ROW OF LOGO
1068  8A 006A
                        LOGO4:  MOV     CX,LOGOSCOL     ; GET START COLUMN OF LOGO
106B  B9 00D6           LOGO5:  MOV     AL,CS:[BX]      ; GET DATA FOR WHITE PART COLUMN NBR
106E  2E: 8A 07                 CMP     AL,0            ; IF LENGTH IS 0
1071  3C 00                     JE      LOGO8           ; THEN SKIP
1073  74 16              LOGO6:  PUSH    DX              ; SAVE CURRENT ROW POSITION
1075  52                       MOV     AH,LINENO       ; SET LINE NUMBER
1076  B4 02              LOGO7:  PUSH    AX              ; SAVE FOR COUNT
1078  50                       MOV     AH,VWRTDOT      ; WRITE DOT
1079  B4 0C                    MOV     AL,WHITE        ; DOT COLOR IS BLUE
107B  B0 0F                    INT     INT_10          ;
107D  CD 10
                                INC     DX              ; NEXT ROW
107F  42                        POP     AX              ;
1080  58                        DEC     AH              ;
1081  FE CC                     JNZ     LOGO7           ; REPEAT FOR ALL LINE IN ROW
1083  75 F3

                                POP     DX              ; RETORE CURRENT ROW POSITION
1085  5A
                                INC     CX              ; NEXT COLUMN
1086  41                        DEC     AL              ; ALL DOT WRITE ?
1087  FE C8                     JNZ     LOGO6           ; NO
1089  75 EA
                        LOGO8:  INC     BX              ;
108B  43                        MOV     AL,CS:[BX]      ; GET DATA FOR BLUE PART COLUMN NBR
108C  2E: 8A 07                 CMP     AL,EOL          ; END OF COLUMN ?
108F  3C FF                     JNE     LOGO9           ; NO, CONTINUE
1091  75 0C
                                INC     BX              ; GET NEXT DATA
1093  43                       CMP     BYTE PTR CS:[BX],EOF ; ALL END ?
1094  2E: 80 3F FF             JE      LOGO10                ; YES, EXIT
1098  74 0B
                                ADD     DX, LINENOM2    ; SET NEXT ROW, TIMES 2
                                                        ; BECAUSE SKIP BLUE LOW
109A  83 C2 04                  JMP     SHORT LOGO4     ; REPEAT FROM START
109D  EB CC
                        LOGO9:  CBW                     ; CONVERT TO WORD
109F  98                        ADD     CX,AX           ; ADD TO SKIP BLUE PART
10A0  03 C8                     INC     BX              ; NEXT DATA
10A2  43                        JMP     SHORT LOGO5     ;
10A3  EB C9
                        ;----- ROUND 'B'
                        LOGO10: PUSH    DS              ; SAVE DS
10A5  1E                        PUSH    ES              ;
10A6  06                        POP     DS              ; SET REGEN ADDR TO DS FOR QUICK ADDR
10A7  1F                        MOV     AL,BLUE*10H OR BLUE     ; SET COLOR
10A8  B0 11
                                DB      0A2H            ; MOV [50C4],AL
10AA  A2                        DW      50C4H           ;
10AB  50C4
                                DB      0A2H            ; MOV [5166],AL
10AD  A2                        DW      5166H           ;
10AE  5166
                                DB      0A2H            ; MOV [5345],AL
10B0  A2                        DW      5345H           ;
10B1  5345
                                DB      0A2H            ; MOV [72A5],AL
10B3  A2                        DW      72A5H           ;
10B4  72A5
                                DB      0A2H            ; MOV [7486],AL
10B6  A2                        DW      7486H           ;
10B7  7486
```

```
10B9  A2                              DB      0A2H            ; MOV [7524],AL
10BA  7524                            DW      7524H           ;
10BC  1F                              POP     DS              ; RESTORE DS
                                ;-----

10BD  B8 0B00                         MOV     AX,0B00H        ; ENABLE SCREEN
10C0  B7 01                           MOV     BH,1            ;
10C2  CD 10                           INT     INT_10          ;


10C4  59                              POP     CX              ; RESTORE AX, BX & CX
10C5  5B                              POP     BX              ;
10C6  58                              POP     AX

10C7  C3                              RET                     ; RETURN TO CALLER

                                ;----- DATA STRUCTURE
                                ;       DATA HAVE NUMBER OF PIXEL, FITST BYTE HAS LENGTH OF WHITE
                                ;       PART, SECOND BYTE HAS LENGTH OF BLUE PART
10C8                            IBM     LABEL   BYTE
                                ;------------------ I ------------ B --------------- M ------------
                                ;                  W  B  W  B     W  B  W  B     W  B  W  B  W
10C8  14 05 1B 07 0E 0D         DB                    20, 5,   27,        7,   14,13,       14,   EOL
      0E FF
10D0  14 05 1E 04 10 09         DB                    20, 5,   30,        4,   16, 9,       16,   EOL
      10 FF
10D8  00 06 08 10 08 08         DB              0, 6, 8,16,    8, 8, 9, 9,    13, 5,13,          EOL
      09 09 0D 05 0D FF
10E4  00 06 08 10 17 0B         DB              0, 6, 8,16,    23,      11,   15, 1,15,          EOL
      0F 01 0F FF
10EE  00 06 08 10 17 0B         DB              0, 6, 8,16,    23,      11,    8, 1,13, 1, 8,    EOL
      08 01 0D 01 08 FF
10FA  00 06 08 10 08 08         DB              0, 6, 8,16,    8, 8, 9, 9,    8, 3, 9, 3, 8,    EOL
      09 09 08 03 09 03
      08 FF
1108  14 05 1E 04 0D 05         DB                    20, 5,   30,        4,   13, 5, 5, 5,13,   EOL
      05 05 0D FF
1112  14 05 1B 07 0D 07         DB                    20, 5,   27,        7,   13, 7, 1, 7,13,   EOL
      01 07 0D FF
111C  FF                        DB      EOF
111D                            PUT_LOGO ENDP
                                ;--------------------------------------------------------------------------
                                ; SUBROUTINE
                                ;  ASYNCHRONOUS COMMUNICATIONS ADAPTER POWER ON DIAGNOSTIC TEST
                                ; DESCRIPTION:
                                ;  THIS SUBROUTINE PERFORMS A THOROUGH CHECK OUT OF AN INS8250
                                ;  LSI CHIP
                                ;  THE TEST INCLUDES:
                                ;  1)  INITIALIZATION OF THE CHIP TO ASSUME ITS MASTER RESET STATE.
                                ;  2)  READING REGISTERS FOR KNOWN PERMANENT ZERO BITS.
                                ;  3)  TESTING THE INS8250 INTERRUPT SYSTEM AND THAT THE 8250
                                ;      INTERRUPTS TRIGGER AN 8259 (INTERRUPT CONTROLLER) INTERRUPT.
                                ;  4)  PERFORMING THE LOOP BACK TEST:
                                ;      A)  TESTING WHAT WAS WRITTEN/READ AND THAT THE TRANSMITTER
                                ;          HOLDING REG EMPTY BIT AND THE RECEIVER INTERRUPT WORK
                                ;          PROPERLY.
                                ;      B)  TESTING IF CERTAIN BITS OF THE DATA SET CONTROL REGISTER
                                ;          ARE 'LOOPED BACK' TO THOSE IN THE DATA SET STATUS
                                ;          REGISTER.
                                ;      C)  TESTING THAT THE TRANSMITTER IS IDLE WHEN TRANSMISSION
                                ;          TEST IS FINISHED.
                                ; THIS SUBROUTINE EXPECTS TO HAVE THE FOLLOWING PARAMETER PASSED:
                                ;   (DX)= ADDRESS OF THE INS8250 CARD TO TEST.
                                ;   NOTE: THE ASSUMPTION HAS BEEN MADE THAT THE MODEM ADAPTER IS
                                ;         ----  LOCATED AT 03F8H; THE SERIAL PRINTER AT 02F8H.
                                ;
                                ; IT RETURNS:
                                ;  (CF) = 1 IF ANY PORTION OF THE TEST FAILED
                                ;       = 0 IF TEST PASSED
                                ;  (BX) = FAILURE KEY FOR ERROR MESSAGE (ONLY VALID IF TEST FAILED)
                                ;     (BH) = 23H  SERIAL PRINTER ADAPTER TEST FAILURE
                                ;          = 24H  MODEM ADAPTER TEST FAILURE
                                ;     (BL) = 02H  PERMANENT ZERO BITS IN INTERRUPT ENABLE REGISTER
                                ;                 WERE INCORRECT
                                ;            03H  PERMANENT ZERO BITS IN INTERRUPT IDENTIFICATION
                                ;                 REGISTER WERE INCORRECT
                                ;            04H  PERMANENT ZERO BITS IN DATA SET CONTROL REGISTER
                                ;                 WERE INCORRECT
                                ;            05H  PERMANENT ZERO BITS IN THE LINE STATUS REGISTER
                                ;                 WERE INCORRECT
                                ;            06H  RECEIVED DATA AVAILABLE INTERRUPT TEST FAILED
                                ;                 (THE INTERRUPT WAS NOT GENERATED)
                                ;            07H  RESERVED FOR REPORTING THE TRANSMITTER HOLDING
                                ;                 REGISTER EMPTY INTERRUPT TEST FAILED
                                ;                 (NOT USED AT THIS TIME BECAUSE OF THE DIFFERENCES
                                ;                 BETWEEN THE 8250'S WHICH WILL BE USED)
                                ;          08H-0BH  RECEIVER LINE STATUS INTERRUPT TEST FAILED
                                ;                 (THE INTERRUPT WAS NOT GENERATED)
                                ;                 08H - OVERRUN ERROR
                                ;                 09H - PARITY ERROR
                                ;                 0AH - FRAMING ERROR
                                ;                 0BH - BREAK INTERRUPT ERROR
                                ;          0CH-0FH  MODEM STATUS INTERRUPT TEST FAILED
                                ;                 (THE INTERRUPT WAS NOT GENERATED)
                                ;
                                ;                 0CH - DELTA CLEAR TO SEND ERROR
                                ;                 0DH - DELTA DATA SET READY ERROR
                                ;                 0EH - TRAILING EDGE RING INDICATOR ERROR
                                ;                 0FH - DELTA RECEIVE LINE SIGNAL DETECT ERROR
                                ;              10H  AN 8250 INTERRUPT OCCURRED AS EXPECTED, BUT NO
                                ;                   8259 (INTR CONTROLLER) INTERRUPT WAS GENERATED
                                ;              11H  DURING THE TRANSMISSION TEST, THE TRANSMITTER
                                ;                   HOLDING REGISTER WAS NOT EMPTY WHEN IT SHOULD
                                ;                   HAVE BEEN.
                                ;
```

```
;               12H  DURING THE TRANSMISSION TEST, THE RECEIVED DATA
;                    AVAILABLE INTERRUPT DIDN'T OCCUR.
;               13H  TRANSMISSION ERROR - THE CHARACTER RECEIVED
;                    DURING LOOP MODE WAS NOT THE SAME AS THE ONE
;                    TRANSMITTED
;               14H  DURING TRANSMISSION TEST, THE 4 DATA SET CONTROL
;                    OUTPUTS WERE NOT THE SAME AS THE 4 DATA SET
;                    CONTROL INPUTS.
;               15H  THE TRANSMITTER WAS NOT IDLE AFTER THE TRANS-
;                    MISSION TEST COMPLETED.
;               16H  RECEIVED DATA AVAILABLE INTERRUPT FAILED TO CLEAR
;               17H  TRANSMITTER HOLDING REG EMPTY INTR FAILED TO CLEAR
;            18H-1BH  RECEIVER LINE STATUS INTERRUPT FAILED TO CLEAR
;            1CH-1FH  MODEM STATUS INTERRUPT FAILED TO CLEAR
;    ON EXIT:
;        - THE MODEM OR SERIAL PRINTER'S 8259 INTERRUPT (WHICHEVER
;          DEVICE WAS TESTED) IS DISABLED.
;        - THE 8250 IS IN THE MASTER RESET STATE.
;    ONLY THE DS REGISTER IS PRESERVED - ALL OTHERS ARE ALTERED.
;-----------------------------------------------------------------------
= 0084          WRAP    EQU     84H             ; LOOP BACK TRANSMISSION TEST
                                                ; INTERRUPT VECTOR ADDRESS

                        ASSUME  CS:CODE,DS:DATA
111D            UART    PROC    NEAR
111D  1E                PUSH    DS
111E  E4 21             IN      AL,INTA01       ; CURRENT ENABLED INTERRUPTS
1120  50                PUSH    AX              ; SAVE FOR EXIT
1121  0C 01             OR      AL,00000001B    ; DISABLE TIMER INTR DURING THIS
                                                ; TEST
1123  E6 21             OUT     INTA01,AL
1125  9C                PUSHF                   ; SAVE CALLER'S FLAGS (SAVE INTR
                                                ; FLAG)
1126  52                PUSH    DX              ; SAVE BASE ADDRESS OF ADAPTER CARD
1127  E8 0000 E         CALL    DDS             ; SET UP 'DATA' AS DATA SEGMENT
                                                ; ADDRESS
;-----------------------------------------------------------------------
;       INITIALIZE PORTS FOR MASTER RESET STATES AND TEST PERMANENT
;       ZERO DATA BITS FOR CERTAIN PORTS.
;-----------------------------------------------------------------------
112A  E8 1280 R         CALL    I8250
112D  73 03             JNC     AT1             ; ALL OK
112F  E9 1251 R         JMP     AT14            ; A PORT'S ZERO BITS WERE NOT ZERO!
;-----------------------------------------------------------------------
;       INS8250 INTERRUPT SYSTEM TEST

;       ONLY THE INTERRUPT BEING TESTED WILL BE ENABLED.
;-----------------------------------------------------------------------
;       SET DI AND SI FOR CALLS TO 'SUI'
1132  BF 0069 R AT1:    MOV     DI,OFFSET IMASKS ; BASE ADDRESS OF INTERRUPT MASKS
1135  33 F6             XOR     SI,SI           ; MODEM INDEX
1137  80 FE 02          CMP     DH,2            ; OR SERIAL?
113A  75 02             JNE     AT2             ; NO - IT'S MODEM
113C  46                INC     SI              ; IT'S SERIAL PRINTER
113D  47                INC     DI              ; SERIAL PRINTER 8259 MASK ADDRESS
;       RECEIVED DATA AVAILABLE INTERRUPT TEST
113E  E8 12D6 R AT2:    CALL    SUI             ; SET UP FOR INTERRUPTS
1141  FE C3             INC     BL              ; ERROR REPORTER (INIT. IN I8250)
1143  42                INC     DX              ; POINT TO INTERRUPT ENABLE
                                                ; REGISTER
1144  B0 01             MOV     AL,1            ; ENABLE RECEIVED DATA AVAILABLE
                                                ; INTR
1146  EE                OUT     DX,AL
1147  53                PUSH    BX              ; SAVE ERROR REPORTER
1148  83 C2 04          ADD     DX,4            ; POINT TO LINE STATUS REGISTER
114B  B4 01             MOV     AH,1            ; SET RECEIVER DATA READY BIT
114D  BB 0400           MOV     BX,0400H        ; INTR TO CHECK, INTR IDENTIFIER
1150  B9 0003           MOV     CX,3            ; INTERRUPT ID REG 'INDEX'
1153  E8 12B4 R         CALL    ICT             ; PERFORM TEST FOR INTERRUPT
1156  5B                POP     BX              ; RESTORE ERROR INDICATOR
1157  3C FF             CMP     AL,0FFH         ; INTERRUPT ERROR OCCUR?
1159  74 36             JE      AT4             ; YES
115B  E8 12E7 R         CALL    C5059           ; GENERATE 8259 INTERRUPT?
115E  72 34             JC      AT5             ; NO
1160  4A                DEC     DX
1161  4A                DEC     DX              ; RESET INTR BY READING RECR BUFR
1162  EC                IN      AL,DX           ; DON'T CARE ABOUT THE CONTENTS!
1163  42                INC     DX
1164  42                INC     DX              ; INTR ID REG
1165  E8 12FA R         CALL    W8250C          ; WAIT FOR INTR TO CLEAR
1168  73 03             JNC     AT3             ; OK
116A  E9 124E R         JMP     AT13            ; DIDN'T CLEAR
;-----------------------------------------------------------------------
; TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT TEST
;       THIS TEST HAS BEEN MODIFIED BECAUSE THE DIFFERENT 8250'S
;       THAT MAY BE USED IN PRODUCING THIS PRODUCT DO NOT FUNCTION
;       THE SAME DURING THE STANDARD TEST OF THIS INTERRUPT
;       (STANDARD BEING THE SAME METHOD FOR TESTING THE OTHER
;       POSSIBLE 8250 INTERRUPTS).  IT IS STILL VALID FOR TESTING
;       IF AN 8259 INTERRUPT IS GENERATED IN RESPONSE TO THE 8250
;       INTERRUPT AND THAT THE 8250 INTERRUPT CLEARS AS IT SHOULD.
;
;       IF THE TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT IS NOT
;       GENERATED WHEN THAT INTERRUPT IS ENABLED, IT IS NOT TREATED
;       AS AN ERROR.  HOWEVER, IF THE INTERRUPT IS GENERATED, IT
;       MUST GENERATE AN 8259 INTERRUPT AND CLEAR PROPERLY TO PASS
;       THIS TEST.
;-----------------------------------------------------------------------
116D  E8 12D6 R AT3:    CALL    SUI             ; SET UP FOR INTERRUPTS

1170  FE C3             INC     BL              ; BUMP ERROR REPORTER
1172  4A                DEC     DX              ; POINT TO INTERRUPT ENABLE
                                                ; REGISTER
```

```
1173  B0 02                     MOV     AL,2            ; ENABLE XMITTER HOLDING REG EMPTY
                                                        ; INTR
1175  EE                        OUT     DX,AL
1176  EB 00                     JMP     $+2             ; I/O DELAY
1178  42                        INC     DX              ; INTR IDENTIFICATION REG
1179  2B C9                     SUB     CX,CX
117B  EC                  AT31: IN      AL,DX           ; READ IT
117C  3C 02                     CMP     AL,2            ; XMITTER HOLDING REG EMPTY INTR?
117E  74 04                     JE      AT32            ; YES
1180  E2 F9                     LOOP    AT31
1182  EB 12                     JMP     SHORT AT6       ; THE INTR DIDN'T OCCUR - TRY NEXT
                                                        ; TEST
1184                      AT32:                         ; THE INTR DID OCCUR
1184  E8 12E7 R                 CALL    C5059           ; GENERATE 8259 INTERRUPT?
1187  72 0B                     JC      AT5             ; NO
1189  E8 12FA R                 CALL    W8250C          ; WAIT FOR THE INTERRUPT TO CLEAR
                                                        ; (IT SHOULD ALREADY BE CLEAR
                                                        ; BECAUSE 'ICT' READ THE INTR ID
                                                        ; REG)
118C  73 08                     JNC     AT6             ; IT CLEARED
118E  E9 124E R                 JMP     AT13            ; ERROR
1191  E9 1215 R           AT4:  JMP     AT11            ; AVOID OUT OF RANGE JUMPS
1194  EB 7D                AT5:  JMP     SHORT AT10
                          ;-----------------------------------------------------------------
                          ; RECEIVER LINE STATUS INTERRUPT TEST
                          ;       THERE ARE 4 BITS WHICH COULD GENERATE THIS INTERRUPT.
                          ;       EACH ONE IS TESTED INDIVIDUALLY.
                          ;       WHEN:   AH      TESTING
                          ;               --      -------
                          ;                2      OVERRUN
                          ;                4      PARITY
                          ;                8      FRAMING
                          ;               10H     BREAK INTR
                          ;-----------------------------------------------------------------
1196  4A                  AT6:  DEC     DX              ; POINT TO INTERRUPT ENABLE
                                                        ; REGISTER
1197  B0 04                     MOV     AL,4            ; ENABLE RECEIVER LINE STATUS INTR
1199  EE                        OUT     DX,AL
119A  83 C2 04                  ADD     DX,4            ; POINT TO LINE STATUS REGISTER
119D  B9 0003                   MOV     CX,3            ; INTR ID REG 'INDEX'
11A0  BD 0004                   MOV     BP,4            ; LOOP COUNTER
11A3  B4 02                     MOV     AH,2            ; INITIAL BIT TO BE TESTED
11A5  E8 12D6 R           AT7:  CALL    SUI             ; SET UP FOR INTERRUPTS
11A8  FE C3                     INC     BL              ; BUMP ERROR REPORTER
11AA  53                        PUSH    BX              ; SAVE IT
11AB  BB 0601                   MOV     BX,0601H        ; INTR TO CHECK, INTR IDENTIFIER
11AE  E8 12B4 R                 CALL    ICT             ; PERFORM TEST FOR INTERRUPT
11B1  5B                        POP     BX
11B2  24 1E                     AND     AL,00011110B    ; MASK OUT BITS THAT DON'T MATTER
11B4  3A C4                     CMP     AL,AH           ; TEST BIT ON?

11B6  75 5D                     JNE     AT11            ; NO
11B8  E8 12E7 R                 CALL    C5059           ; GENERATE 8259 INTERRUPT?
11BB  72 56                     JC      AT10            ; NO
11BD  83 EA 03                  SUB     DX,3            ; INTR ID REG
11C0  E8 12FA R                 CALL    W8250C          ; WAIT FOR THE INTR TO CLEAR
11C3  73 03                     JNC     AT7_0           ; OK
11C5  E9 124E R                 JMP     AT13            ; IT DIDN'T
11C8  4D                  AT7_0: DEC    BP              ; ALL FOUR BITS TESTED?
11C9  74 07                     JE      AT8             ; YES - GO ON TO NEXT TEST
11CB  D0 E4                     SHL     AH,1            ; GET READY FOR NEXT BIT
11CD  83 C2 03                  ADD     DX,3            ; LINE STATUS REGISTER
11D0  EB D3                     JMP     AT7             ; TEST NEXT BIT
                          ;-----------------------------------------------------------------
                          ; MODEM STATUS INTERRUPT TEST
                          ;       THERE ARE 4 BITS WHICH COULD GENERATE THIS INTERRUPT.
                          ;       THEY ARE TESTED INDIVIDUALLY.
                          ;       WHEN:   AH      TESTING
                          ;               --      -------
                          ;                1      DELTA CLEAR TO SEND
                          ;                2      DELTA DATA SET READY
                          ;                4      TRAILING EDGE RING INDICATOR
                          ;                8      DELTA RECEIVE LINE SIGNAL DETECT
                          ;-----------------------------------------------------------------
11D2  83 C2 04            AT8:  ADD     DX,4            ; MODEM STATUS REGISTER
11D5  EC                        IN      AL,DX           ; CLEAR DELTA BITS THAT MAY BE ON
                                                        ; BECAUSE OF DIFFERENCES AMONG
                                                        ; 8250'S.
11D6  EB 00                     JMP     $+2             ; I/O DELAY
11D8  83 EA 05                  SUB     DX,5            ; INTERRUPT ENABLE REGISTER
11DB  B0 08                     MOV     AL,8            ; ENABLE MODEM STATUS INTERRUPT
11DD  EE                        OUT     DX,AL
11DE  83 C2 05                  ADD     DX,5            ; POINT TO MODEM STATUS REGISTER
11E1  B9 0004                   MOV     CX,4            ; INTR ID REG 'INDEX'
11E4  BD 0004                   MOV     BP,4            ; LOOP COUNTER
11E7  B4 01                     MOV     AH,1            ; INITIAL BIT TO BE TESTED
11E9  E8 12D6 R           AT9:  CALL    SUI             ; SET UP FOR INTERRUPTS
11EC  FE C3                     INC     BL              ; BUMP ERROR INDICATOR
11EE  53                        PUSH    BX              ; SAVE IT
11EF  BB 0001                   MOV     BX,0001H        ; INTR TO CHECK, INTR IDENTIFIER
11F2  E8 12B4 R                 CALL    ICT             ; PERFORM TEST FOR INTERRUPT
11F5  5B                        POP     BX
11F6  24 0F                     AND     AL,00001111B    ; MASK OUT BITS THAT DON'T MATTER
11F8  3A C4                     CMP     AL,AH           ; TEST BIT ON?
11FA  75 19                     JNE     AT11            ; NO
11FC  E8 12E7 R                 CALL    C5059           ; GENERATE 8259 INTERRUPT?
11FF  72 12                     JC      AT10            ; NO
1201  83 EA 04                  SUB     DX,4            ; INTR ID REG
1204  E8 12FA R                 CALL    W8250C          ; WAIT FOR INTERRUPT TO CLEAR
1207  72 45                     JC      AT13            ; IT DIDN'T
1209  4D                        DEC     BP
120A  74 0B                     JE      AT12            ; ALL FOUR BITS TESTED - GO ON
120C  D0 E4                     SHL     AH,1            ; GET READY FOR NEXT BIT
```

```
120E  83 C2 04          ADD     DX,4                    ; MODEM STATUS REGISTER
1211  EB D6              JMP     AT9                     ; TEST NEXT BIT
                ;-------------------------------------------------------------
                ;       POSSIBLE 8259 INTERRUPT CONTROLLER PROBLEM
                ;-------------------------------------------------------------
1213  B3 10     AT10:   MOV     BL,10H                  ; SET ERROR REPORTER
1215  EB 3A     AT11:   JMP     SHORT AT14
                ;-------------------------------------------------------------
                ;       SET 9600 BAUD RATE AND DEFINE DATA WORD AS HAVING 8
                ;       BITS/WORD, 2 STOP BITS, AND ODD PARITY.
                ;-------------------------------------------------------------
1217  42        AT12:   INC     DX                      ; LINE CONTROL REGISTER
1218  E8 130A R         CALL    S8250
                ;-------------------------------------------------------------
                ;       SET DATA SET CONTROL WORD TO BE IN LOOP MODE
                ;-------------------------------------------------------------
1218  83 C2 04          ADD     DX,4
121E  EC                IN      AL,DX                   ; CURRENT STATE
121F  EB 00             JMP     $+2                     ; I/O DELAY
1221  0C 10             OR      AL,00010000B            ; SET BIT 4 OF DATA SET CONTROL REG
1223  EE                OUT     DX,AL
1224  EB 00             JMP     $+2                     ; I/O DELAY
1226  42                INC     DX
1227  42                INC     DX                      ; MODEM STATUS REG
1228  EC                IN      AL,DX                   ; CLEAR POSSIBLE MODEM STATUS
                                                        ; INTERRUPT WHICH COULD BE CAUSED
                                                        ; BY THE OUTPUT BITS BEING LOOPED
                                                        ; TO THE INPUT BITS
1229  EB 00             JMP     $+2                     ; I/O DELAY
122B  83 EA 06          SUB     DX,6                    ; RECEIVER BUFFER
122E  EC                IN      AL,DX                   ; DUMMY READ TO CLEAR DATA READY
                                                        ; BIT IF IT WENT HIGH ON WRITE TO
                                                        ; MCR
                ;-------------------------------------------------------------
                ;       PERFORM THE LOOP BACK TEST
                ;-------------------------------------------------------------
122F  42                INC     DX                      ; INTR ENBL REG
1230  52                PUSH    DX                      ;
1231  BA 0201           MOV     DX,JOY_PORT             ;
1234  EC                IN      AL,DX                   ;
1235  24 F0             AND     AL,0F0H                 ;
1237  3C 20             CMP     AL,00100000B            ; SERVICE MODE LOOP?
1239  5A                POP     DX                      ;
123A  74 0A             JE      AT13_0                  ; YES-
123C  A1 0072 R         MOV     AX,RESET_FLAG           ;
123F  3D 3412           CMP     AX,3412H                ; WARM START WITH "CTRL+ALT+INS"?
1242  B0 00             MOV     AL,0                    ;  SET FOR INTERNAL WRAP TEST
1244  75 02             JNE     AT13_1                  ; YES-
1246  B0 FF     AT13_0: MOV     AL,0FFH                 ;  SET FOR EXTERNAL WRAP TEST
1248  CD 84     AT13_1: INT     WRAP                    ; DO LOOP BACK TRANSMISSION TEST
124A  B1 00             MOV     CL,0                    ; ASSUME NO ERRORS
124C  73 05             JNC     AT15                    ; WRAP TEST PASSED
124E  80 C3 10  AT13:   ADD     BL,10H                  ; ERROR INDICATOR
                ;-------------------------------------------------------------

                ;       AN ERROR WAS ENCOUNTERED SOMEWHERE DURING THE TEST

                ;-------------------------------------------------------------
1251  B1 01     AT14:   MOV     CL,1                    ; SET FAIL INDICATOR
                ;-------------------------------------------------------------
                ;       HOUSEKEEPING:  RE-INITIALIZE THE 8250 PORTS (THE LOOP BIT
                ;       WILL BE RESET), DISABLE THIS DEVICE INTERRUPT, SET UP
                ;       REGISTER BH IF AN ERROR OCCURRED, AND SET OR RESET THE
                ;       CARRY FLAG.
                ;-------------------------------------------------------------
1253  5A        AT15:   POP     DX                      ; GET BASE ADDRESS OF 8250 ADAPTER
1254  53                PUSH    BX                      ; SAVE ERROR CODE
1255  E8 1280 R         CALL    I8250                   ; RE-INITIALIZE 8250 PORTS
1258  5B                POP     BX
1259  2E: 8A 25         MOV     AH,CS:[DI]              ; GET DEVICE INTERRUPT MASK
125C  20 26 0084 R      AND     INTR_FLAG,AH            ; CLEAR DEVICE'S INTERRUPT FLAG BIT
1260  80 F4 FF          XOR     AH,0FFH                 ; FLIP BITS
1263  E4 21             IN      AL,INTA01               ; GET CURRENT INTERRUPT PORT
1265  0A C4             OR      AL,AH                   ; DISABLE THIS DEVICE INTERRUPT
1267  E6 21             OUT     INTA01,AL
1269  9D                POPF                            ; RE-ESTABLISH CALLER'S INTERRUPT
                                                        ; FLAG
126A  0A C9             OR      CL,CL                   ; ANY ERRORS?
126C  74 0C             JE      AT17                    ; NO
126E  B7 24             MOV     BH,24H                  ; ASSUME MODEM ERROR
1270  80 FE 02          CMP     DH,2                    ; OR IS IT SERIAL?
1273  75 02             JNE     AT16                    ; IT'S MODEM
1275  B7 23             MOV     BH,23H                  ; IT'S SERIAL PRINTER
1277  F9        AT16:   STC                             ; SET CARRY FLAG TO INDICATE ERROR
1278  EB 01             JMP     SHORT AT18
127A  F8        AT17:   CLC                             ; RESET CARRY FLAG - NO ERRORS
127B  58        AT18:   POP     AX                      ; RESTORE ENTRY ENABLED INTERRUPTS
127C  E6 21             OUT     INTA01,AL               ; DEVICE INTRS RE-ESTABLISHED
127E  1F                POP     DS                      ; RESTORE REGISTER
127F  C3                RET
1280            UART    ENDP
                ;-------------------------------------------------------------
                ; SUBROUTINE
                ;       INITIALIZE INS8250 PORTS TO THE MASTER RESET STATUS.
                ;       THIS ROUTINE ALSO TESTS THE PORTS' PERMANENT ZERO BITS.
                ; EXPECTS TO BE PASSED:
                ;       (DX) = ADDRESS OF THE 8250 TRANSMIT/RECEIVE BUFFER
                ; UPON RETURN:
                ;       (CF) = 1  IF ONE OF THE PORTS' PERMANENT ZERO BITS WAS NOT
                ;                 ZERO (ERR)
                ;                 (DX) = PORT ADDRESS THAT FAILED TEST
                ;                 (AL) = MEANINGLESS
                ;                 (BL) = 2  INTR ENBL REG  BITS NOT 0
```

```
;                       3  INTR ID REG BITS NOT 0
;                       4  MODEM CTRL REG  BITS NOT 0
;                       5  LINE STAT REG BITS NOT 0
;               0  IF ALL PORTS' PERMANENT ZERO BITS WERE ZERO
;                  (DX) = TRANSMIT/RECEIVE BUFFER ADDRESS
;                  (AL) = LAST VALUE READ FROM RECEIVER BUFFER

;                  (BL) = 5 (MEANINGLESS)
;         PORTS SET UP AS FOLLOWS ON ERROR-FREE RETURN:
;          XF9 - INTR ENBL REG  = 0          ALL INTERRUPTS DISABLED
;          XFA - INTR ID REG    = 00000001B  NO INTERRUPTS PENDING
;          XFB - LINE CTRL REG  = 0          ALL BITS LOW
;          XFC - MODEM CTRL REG = 0          ALL BITS LOW
;          XFD - LINE STAT REG  = 01100000B  TRANSMITTER HOLDING
;                                 REGISTER AND TRANSMITTER EMPTY ON
;          XFE - MODEM STAT REG = XXXX0000B  WHERE X 'S REPRESENT
;                                 INPUT SIGNALS
; REGISTERS DX, AL, AND BL ARE ALTERED.  NO OTHER REGISTERS USED.
;-----------------------------------------------------------------------
1280                    I8250   PROC    NEAR
1280  EC                        IN      AL,DX           ; READ RECVR BUFFER BUT IGNORE
                                                        ; CONTENTS
1281  B3 02                     MOV     BL,2            ; ERROR INDICATOR
1283  E8 132E R                 CALL    RR2             ; READ INTR ENBL REG
1286  24 F0                     AND     AL,11110000B    ; BITS 4-7 OFF?
1288  75 28                     JNE     AT20            ; NO - ERROR
128A  E8 1329 R                 CALL    RR1             ; READ INTR ID REG
128D  24 F8                     AND     AL,11111000B    ; BITS 3-7 OFF?
128F  75 21                     JNE     AT20            ; NO
1291  42                        INC     DX              ; LINE CTRL REG
1292  E8 1329 R                 CALL    RR1             ; READ MODEM CTRL REG
1295  24 E0                     AND     AL,11100000B    ; BITS 5-7 OFF?
1297  75 19                     JNE     AT20            ; NO
1299  E8 1329 R                 CALL    RR1             ; READ LINE STAT REG
129C  24 80                     AND     AL,10000000B    ; BIT 7 OFF?
129E  75 12                     JNE     AT20            ; NO
12A0  B0 60                     MOV     AL,60H
12A2  EE                        OUT     DX,AL
12A3  EB 00                     JMP     $+2             ; I/O DELAY
12A5  42                        INC     DX              ; MODEM STAT REG
12A6  32 C0                     XOR     AL,AL
12A8  EE                        OUT     DX,AL           ; WIRED BITS WILL BE HIGH
12A9  E8 132F R                 CALL    RR3             ; CLEAR BITS 0-3 IN CASE THEY'RE ON
                                                        ; AFTER WRITING TO STATUS REG
12AC  83 EA 06                  SUB     DX,6            ; RECEIVER BUFFER
12AF  EC                        IN      AL,DX           ; IN CASE WRITING TO PORTS CAUSED
                                                        ; DATA READY TO GO HIGH!
12B0  F8                        CLC                     ; NORMAL RETURN
12B1  C3                        RET
12B2  F9                AT20:   STC                     ; ERROR RETURN
12B3  C3                        RET
12B4                    I8250   ENDP
;-----------------------------------------------------------------------
; SUBROUTINE
;       TEST A PARTICULAR 8250 INTERRUPT.  PASS IT THE
;       (BIT 0 + 1) OF THE STATUS REGISTER THAT IS TO BE TESTED.
;       THIS ROUTINE SETS THAT BIT AND CHECKS TO SEE IF THE CORRECT
;       8250 INTERRUPT IS GENERATED.
; IT EXPECTS TO BE PASSED:
;       (AH) = BIT 0 TO BE TESTED

;       (BL) = INTERRUPT IDENTIFIER
;           (0) = RECEIVED DATA AVAILABLE OR TRANSMITTER HOLDING
;                 REGISTER EMPTY INTERRUPT TEST
;           (1) = RECEIVER LINE STATUS OR MODEM STATUS INTERRUPT
;                 TEST
;       (BH) = BITS WHICH DETERMINE WHICH INTERRUPT IS TO BE
;              CHECKED
;           (0) = MODEM STATUS
;           (2) = TRANSMITTER HOLDING REGISTER EMPTY
;           (4) = RECEIVED DATA AVAILABLE
;           (6) = RECEIVER LINE STATUS
;       (CX) = VALUE TO SUBTRACT AND ADD IN ORDER TO REFERENCE THE
;              INTERRUPT IDENTIFICATION REGISTER
;           (3) = RECEIVED DATA AVAILABLE, TRANSMITTER HOLDING
;                 REGISTER AND RECEIVER LINE STATUS INTERRUPTS
;           (4) = MODEM STATUS INTERRUPT
;       (DX) = ADDRESS OF THE LINE STATUS OR MODEM STATUS REGISTER
;              IT RETURNS:
;       (AL) = 0FFH IF TEST FAILS - EITHER NO INTERRUPT OCCURRED OR
;              THE WRONG INTERRUPT OCCURRED
;    OR
;       (AL) = CONTENTS OF THE INTERRUPT ID REGISTER FOR RECEIVED
;          DATA AVAILABLE AND TRANSMITTER HOLDING REGISTER
;          EMPTY INTERRUPTS
;          -OR-
;          CONTENTS OF THE LINE STATUS OR MODEM STATUS REGISTER
;              DEPENDING ON WHICH ONE WAS TESTED.
;       (DX) = ADDRESS OF INTERRUPT ID REGISTER FOR RECEIVED DATA
;              AVAILABLE OR TRANSMITTER HOLDING REGISTER EMPTY
;              INTERRUPTS
;    OR
;       (DX) = ADDRESS OF THE LINE STATUS OR DATA SET STATUS
;              REGISTER (DEPENDING ON WHICH INTERRUPT WAS TESTED)
;    NO OTHER REGISTERS ARE ALTERED.
;-----------------------------------------------------------------------
12B4                    ICT     PROC    NEAR
12B4  EC                        IN      AL,DX           ; READ STATUS REGISTER
12B5  EB 00                     JMP     $+2             ; I/O DELAY
12B7  0A C4                     OR      AL,AH           ; SET TEST BIT
12B9  EE                        OUT     DX,AL           ; WRITE IT TO THE STATUS REGISTER
12BA  2B D1                     SUB     DX,CX           ; POINT TO INTERRUPT ID REGISTER
12BC  51                        PUSH    CX
```

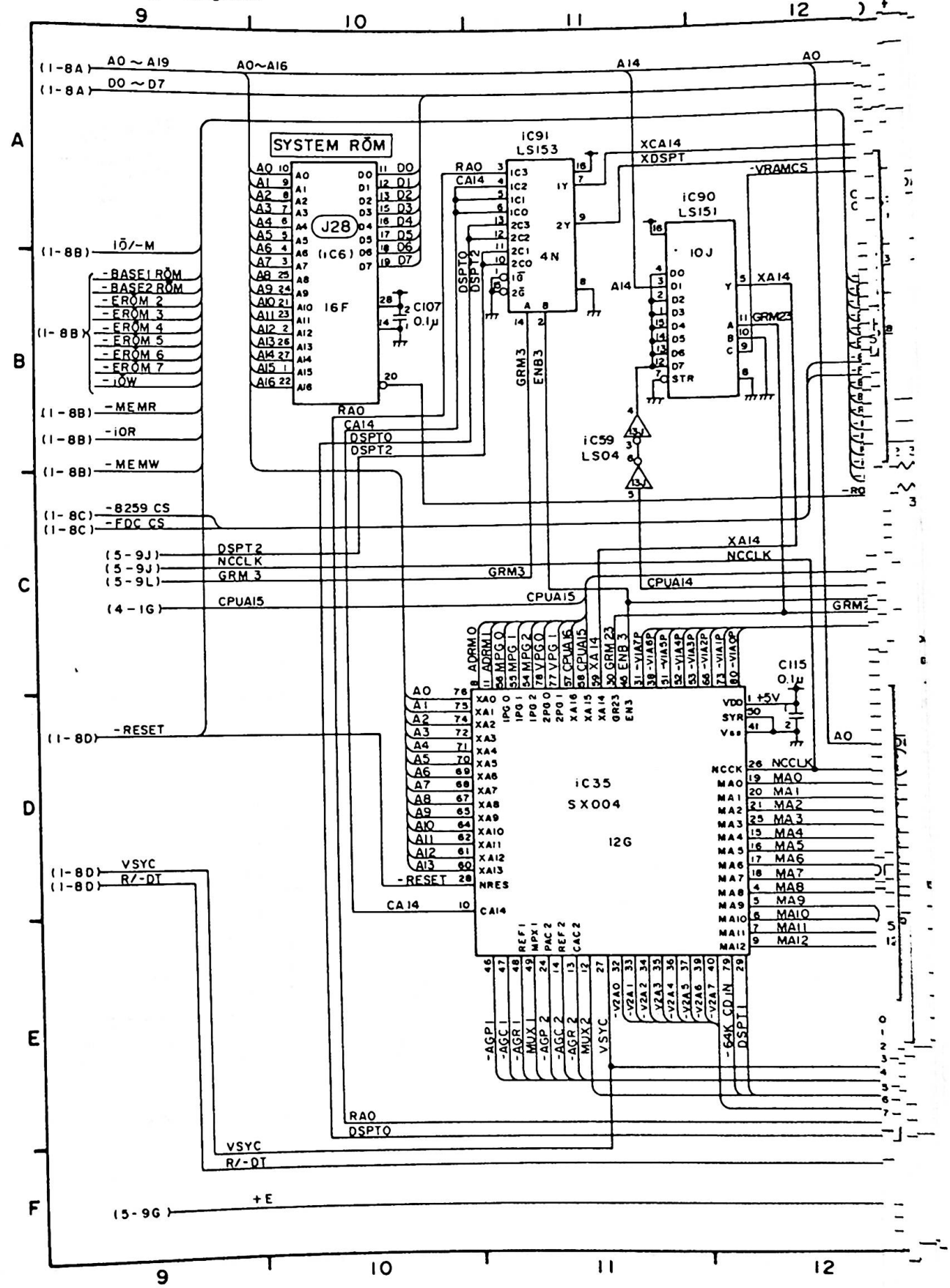
```
12BD  2B C9                     SUB     CX,CX           ; WAIT FOR 8250 INTERRUPT TO OCCUR
12BF  EC                AT21:   IN      AL,DX           ; READ INTR ID REG
12C0  A8 01                     TEST    AL,1            ; INTERRUPT PENDING?
12C2  74 02                     JE      AT22            ; YES -RETURN W/ INTERRUPT ID IN AL
12C4  E2 F9                     LOOP    AT21            ; NO - TRY AGAIN
12C6  59                AT22:   POP     CX              ; AL = 1 IF NO INTERRUPT OCCURRED
12C7  3A C7                     CMP     AL,BH           ; INTERRUPT WE'RE LOOKING FOR?
12C9  75 08                     JNE     AT23            ; NO
12CB  0A DB                     OR      BL,BL           ; DONE WITH TEST FOR THIS INTERRUPT
12CD  74 06                     JE      AT24            ; RETURN W/ CONTENTS OF INTR ID REG
12CF  03 D1                     ADD     DX,CX           ; READ STATUS REGISTER TO CLEAR THE

12D1  EC                        IN      AL,DX           ; INTERRUPT (WHEN BL=1)
12D2  C3                        RET                     ; RETURN CONTENTS OF STATUS REG

12D3  B0 FF             AT23:   MOV     AL,0FFH         ; SET ERROR INDICATOR
12D5  C3                AT24:   RET
12D6                    ICT     ENDP
                        ;-----------------------------------------------------------------
                        ; SUBROUTINE
                        ;       SET UP CONDITIONS FOR 8250 TESTING
                        ; DESCRIPTION
                        ;       TO SET UP CONDITIONS FOR THE TESTING OF 8250 AND
                        ;       8259 INTERRUPTS.  ENABLES MASKABLE EXTERNAL INTERRUPTS,
                        ;       CLEARS THE 8259 INTR RECEIVED FLAG BIT, AND ENABLES THE
                        ;       DEVICE'S 8259 INTR (WHICHEVER IS BEING TESTED).
                        ; IT EXPECTS TO BE PASSED:
                        ;       (DS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED
                        ;       (DI) = OFFSET OF THE INTERRUPT BIT MASK
                        ; UPON RETURN:
                        ;       INTR_FLAG BIT FOR THE DEVICE = 0
                        ; NO REGISTERS ARE ALTERED.
                        ;-----------------------------------------------------------------
12D6                    SUI     PROC    NEAR
12D6  50                        PUSH    AX
12D7  FB                        STI                     ; ENABLE MASKABLE EXTERNAL
                                                        ; INTERRUPTS
12D8  2E: 8A 25                 MOV     AH,CS:[DI]      ; GET INTERRUPT BIT MASK
12DB  20 26 0084 R              AND     INTR_FLAG,AH    ; CLEAR 8259 INTERRUPT REC'D FLAG
                                                        ; BIT
12DF  E4 21                     IN      AL,INTA01       ; CURRENT INTERRUPTS
12E1  22 C4                     AND     AL,AH           ; ENABLE THIS INTERRUPT, TOO
12E3  E6 21                     OUT     INTA01,AL       ; WRITE TO 8259 (INTERRUPT
                                                        ; CONTROLLER)
12E5  58                        POP     AX
12E6  C3                        RET
12E7                    SUI     ENDP
                        ;-----------------------------------------------------------------
                        ; SUBROUTINE
                        ;       CHECKS IF A 8259 INTERRUPT IS GENERATED BY THE
                        ;       8250 INTERRUPT.
                        ; EXPECTS TO BE PASSED:
                        ;       (DI) = OFFSET OF INTERRUPT BIT MASK
                        ;       (DS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED.
                        ; RETURNS:
                        ;       (CF) = 1 IF NO INTERRUPT IS GENERATED
                        ;              0 IF THE INTERRUPT OCCURRED
                        ;       (AL) = COMPLEMENT OF THE INTERRUPT MASK
                        ; NO OTHER REGISTERS ARE ALTERED.
                        ;-----------------------------------------------------------------
12E7                    C5059   PROC    NEAR
12E7  51                        PUSH    CX
12E8  2B C9                     SUB     CX,CX           ; SET PROGRAM LOOP COUNT
12EA  2E: 8A 05                 MOV     AL,CS:[DI]      ; GET INTERRUPT MASK
12ED  34 FF                     XOR     AL,0FFH         ; COMPLEMENT MASK SO ONLY THE INTR
                                                        ; TEST BIT IS ON
12EF  84 06 0084 R      AT25:   TEST    INTR_FLAG,AL    ; 8259 INTERRUPT OCCUR?
12F3  75 03                     JNE     AT27            ; YES - CONTINUE
12F5  E2 F8                     LOOP    AT25            ; WAIT SOME MORE
12F7  F9                        STC                     ; TIME'S UP - FAILED
                    MSCBIO10.INC

12F8  59                AT27:   POP     CX
12F9  C3                        RET
12FA                    C5059   ENDP
                        ;-----------------------------------------------------------------
                        ; SUBROUTINE
                        ;       TO WAIT FOR ALL ENABLED 8250 INTERRUPTS TO CLEAR (SO
                        ;       NO INTRS WILL BE PENDING).  EACH INTERRUPT COULD TAKE UP TO
                        ;       1 MILLISECOND TO CLEAR. THE INTERRUPT IDENTIFICATION
                        ;       REGISTER WILL BE CHECKED UNTIL THE INTERRUPT(S) IS CLEARED
                        ;       OR A TIMEOUT OCCURS.
                        ; EXPECTS TO BE PASSED:
                        ;       (DX) = ADDRESS OF THE INTERRUPT ID REGISTER
                        ; RETURNS:
                        ;       (AL) = CONTENTS OF THE INTR ID REGISTER
                        ;       (CF) = 1  IF INTERRUPTS ARE STILL PENDING
                        ;              0  IF NO INTERRUPTS ARE PENDING (ALL CLEAR)
                        ; NO OTHER REGISTERS ARE ALTERED.
                        ;-----------------------------------------------------------------
12FA                    W8250C  PROC    NEAR
12FA  51                        PUSH    CX
12FB  2B C9                     SUB     CX,CX
12FD  EC                AT28:   IN      AL,DX           ; READ INTR ID REG
12FE  3C 01                     CMP     AL,1            ; INTERRUPTS STILL PENDING?
1300  74 05                     JE      AT29            ; NO - GOOD FINISH
1302  E2 F9                     LOOP    AT28            ; KEEP TRYING
1304  F9                        STC                     ; TIME'S UP - ERROR
1305  EB 01                     JMP     SHORT AT30
1307  F8                AT29:   CLC
1308  59                AT30:   POP     CX
1309  C3                        RET
```

```
130A                                    W8250C  ENDP
                                        ;--------------------------------------------------------------
                                        ; SUBROUTINE
                                        ;       TO SET AN IMS8250 CHIP'S BAUD RATE TO 9600 BPS AND
                                        ;       DEFINE IT'S DATA WORD AS HAVING 8 BITS/WORD, 2 STOP BITS,
                                        ;       AND ODD PARITY.
                                        ; EXPECTS TO BE PASSED:
                                        ;       (DX) = LINE CONTROL REGISTER
                                        ; UPON RETURN:
                                        ;       (DX) = TRANSMIT/RECEIVE BUFFER ADDRESS
                                        ; ALTERS REGISTER AL. ALL OTHERS REMAIN INTACT.
                                        ;--------------------------------------------------------------
130A                                    S8250   PROC    NEAR
130A  B0 80                                     MOV     AL,80H          ; SET DLAB = 1
130C  EE                                        OUT     DX,AL
130D  EB 00                                     JMP     $+2             ; I/O DELAY
130F  83 EA 03                                  SUB     DX,3            ; LSB OF DIVISOR LATCH
1312  B0 0C                                     MOV     AL,12           ; DIVISOR = 12 PRODUCES 9600 BPS
1314  EE                                        OUT     DX,AL           ; SET LSB
1315  EB 00                                     JMP     $+2             ; I/O DELAY
1317  42                                        INC     DX              ; MSB OF DIVISOR LATCH
1318  B0 00                                     MOV     AL,0            ; HIGH ORDER OF DIVISORS
131A  EE                                        OUT     DX,AL           ; SET MSB

131B  EB 00                                     JMP     $+2             ; I/O DELAY
131D  42                                        INC     DX
131E  42                                        INC     DX              ; LINE CONTROL REGISTER
131F  B0 0F                                     MOV     AL,00001111B    ; 8 BITS/WORD, 2 STOP BITS, ODD
                                                                        ; PARITY
1321  EE                                        OUT     DX,AL
1322  EB 00                                     JMP     $+2             ; I/O DELAY
1324  83 EA 03                                  SUB     DX,3            ; RECEIVER BUFFER
1327  EC                                        IN      AL,DX           ; IN CASE WRITING TO PORT LCR
                                                                        ; CAUSED DATA READY TO GO HIGH!
1328  C3                                        RET
1329                                    S8250   ENDP
                                        ;--------------------------------------------------------------
                                        ; SUBROUTINE
                                        ;       TO READ AN 8250 REGISTER.  MAY ALSO BUMP ERROR
                                        ;       REPORTER (BL) AND/OR REG DX (PORT ADDRESS) DEPENDING ON
                                        ;       WHICH ENTRY POINT IS CHOSEN.
                                        ;       THIS SUBROUTINE WAS WRITTEN TO AVOID MULTIPLE USE OF I/O
                                        ;       TIME DELAYS FOR THE 8250.  IT WAS THE MOST EFFICIENT WAY TO
                                        ;       INCLUDE THE DELAYS.
                                        ; UPON RETURN
                                        ;       REG AL WILL CONTAIN THE CONTENTS OF PORT(DX)
                                        ;--------------------------------------------------------------
1329                                    RR1     PROC    NEAR
1329  32 C0                                     XOR     AL,AL
132B  EE                                        OUT     DX,AL           ; DISABLE ALL INTERRUPTS
132C  FE C3                                     INC     BL              ; BUMP ERROR REPORTER
132E  42                                RR2:    INC     DX              ; INCR PORT ADDR
132F  EC                                RR3:    IN      AL,DX           ; READ REGISTER
1330  C3                                       RET
1331                                    RR1     ENDP
                                        ;-----------------------------------------------
                                        ; THESE ARE THE VECTORS WHICH ARE MOVED INTO     :
                                        ; THE 8086 INTERRUPT AREA DURING POWER ON.       :
                                        ; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE      :
                                        ; SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT  :
                                        ; WHERE NOTED.                                   :
                                        ;-----------------------------------------------
                                                ASSUME  CS:CODE
                                                ORG     BEGIN+15C0H
15C0                                    VECTOR_TABLE    LABEL   WORD ; VECTOR TABLE FOR MOVE TO INTERRUPTS
15C0  0000 E                                    DW      OFFSET TIMER_INT                ; INTERRUPT 08H
15C2  0000 E                                    DW      OFFSET  KB_INT                  ; INTERRUPT 09H
15C4  0599 R                                    DW      OFFSET  D11                     ; INTERRUPT 0AH
15C6  0599 R                                    DW      OFFSET  D11                     ; INTERRUPT 0BH
15C8  0599 R                                    DW      OFFSET  D11                     ; INTERRUPT 0CH
15CA  0599 R                                    DW      OFFSET  D11                     ; INTERRUPT 0DH
15CC  0000 E                                    DW      OFFSET DISK_INT                 ; INTERRUPT 0EH
15CE  0599 R                                    DW      OFFSET  D11                     ; INTERRUPT 0FH
15D0  0000 E                                    DW      OFFSET VIDEO_IO                 ; INTERRUPT 10H
15D2  0000 E                                    DW      OFFSET EQUIPMENT                ; INTERRUPT 11H
15D4  0000 E                                    DW      OFFSET MEMORY_SIZE_DETERMINE    ; INTERRUPT 12H
15D6  0000 E                                    DW      OFFSET DISKETTE_IO              ; INTERRUPT 13H
15D8  0000 E                                    DW      OFFSET RS232_IO                 ; INTERRUPT 14H
15DA  0000 E                                    DW      CASSETTE_IO                     ; INTERRUPT 15H
15DC  0000 E                                    DW      OFFSET KEYBOARD_IO              ; INTERRUPT 16H
15DE  0000 E                                    DW      OFFSET PRINTER_IO               ; INTERRUPT 17H
15E0  0000                                      DW      00000H                          ; INTERRUPT 18H
15E2  0000 E                                    DW      OFFSET BOOT_STRAP               ; INTERRUPT 19H
15E4  0000 E                                    DW      TIME_OF_DAY                     ; INTERRUPT 1AH
15E6  05C0 R                                    DW      DUMMY_RETURN                    ; INTERRUPT 1BH - KEYBD BREAK ADDR
15E8  05C0 R                                    DW      DUMMY_RETURN                    ; INTERRUPT 1CH - TIMER BREAK ADDR
15EA  0000 E                                    DW      VIDEO_PARMS                     ; INTERRUPT 1DH
15EC  0000 E                                    DW      OFFSET DISK_BASE                ; INTERRUPT 1EH
15EE  05C0 R                                    DW      DUMMY_RETURN                    ; INTERRUPT 1FH
                                        ;-----------------------------
                                        ;       POWER ON RESET VECTOR   :
                                        ;-----------------------------
                                        ;
                                        ;----- POWER ON RESET
                                        ;
15F0  EA                                        DB      0EAH                    ; JUMP FAR
15F1  006B R                                    DW      OFFSET RESET
15F3  F800                                      DW      0F800H                  ; CODE SEGMENT

15F5  30 39 2F 31 30 2F                         DB      '09/10/84'              ; RELEASE MARKER
      38 34

15FD  FF                                        DB      0FFH                    ; FILLER
```

```
                        DB      0EDH            ; SYSTEM IDENTIFIER
15FE  ED                DB      0FFH            ; CHECKSUM
                        ;
                POST_END LABEL  FAR
15FF                    ENDS
15FF                    END
```

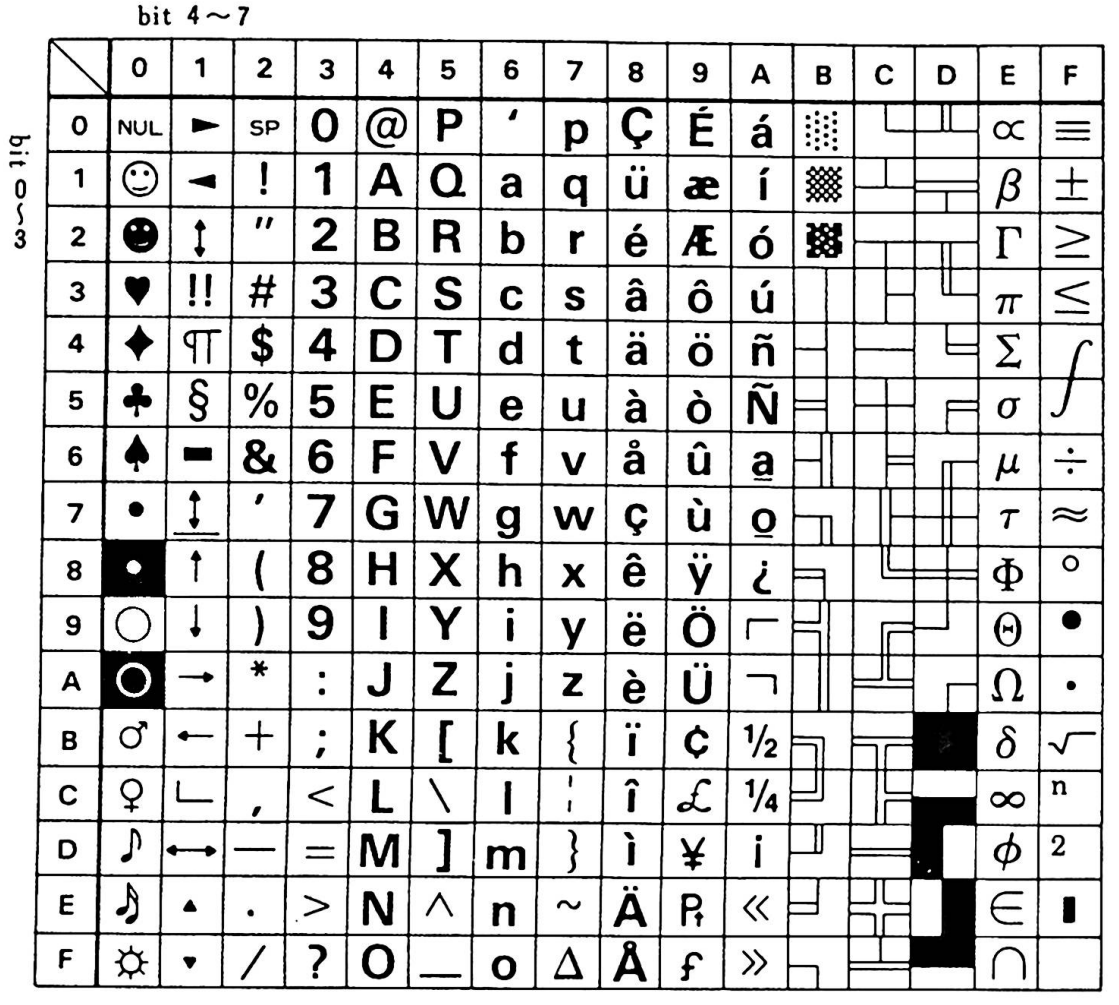
Appendix B Logic Diagrams

This page intentionally left blank.

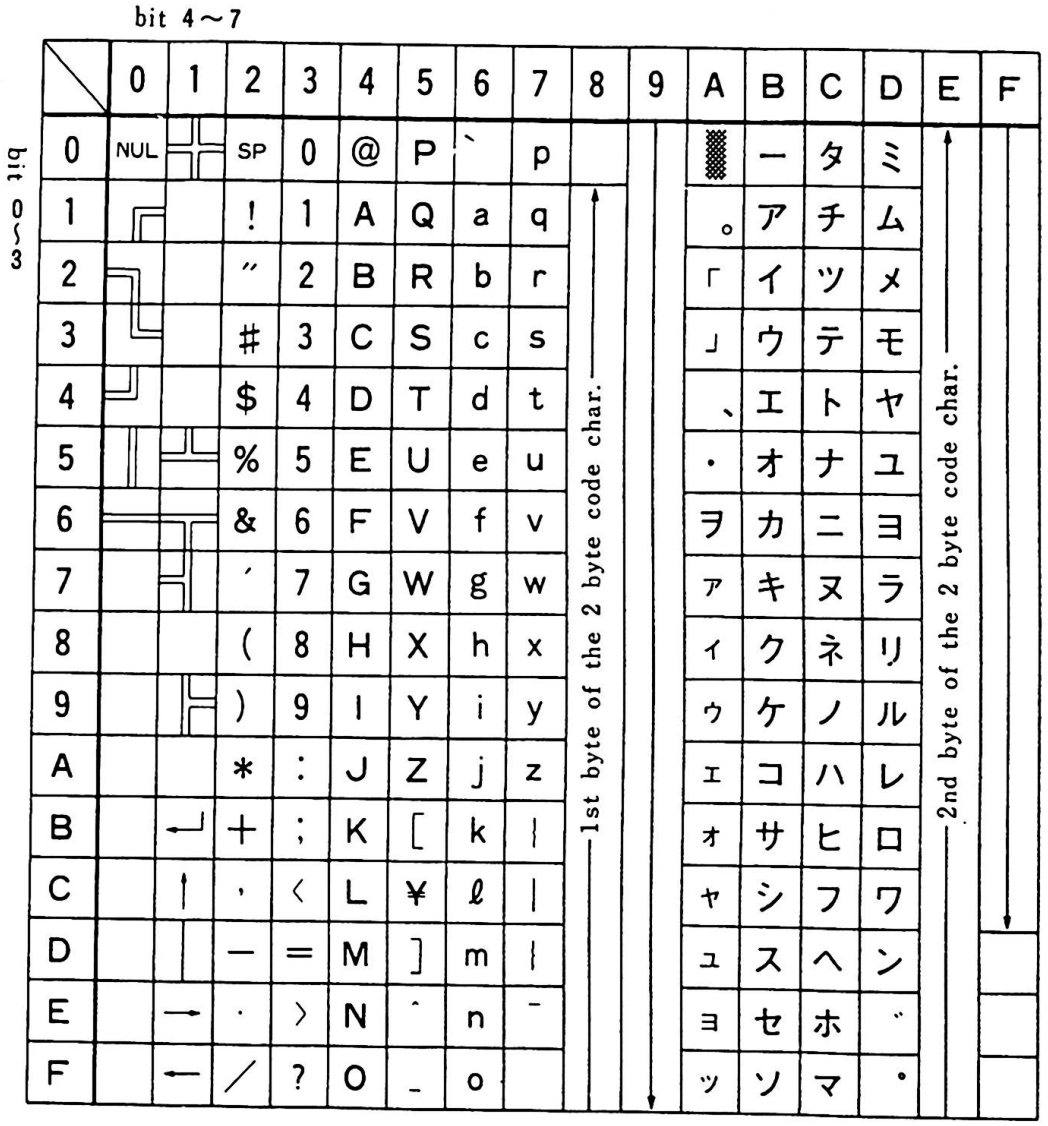

SYSTEM ROM
J28
(1C6)
16F
C107
0.1µ
1C91
LS153
4N
1C90
LS151
10J
1C59
LS04
1C35
SX004
12G
C115
0.1µ
+5V
A0 ~ A19
D0 ~ D7
A0~A16
A14
A0
XCA14
XDSPT
-VRAMCS
XA14
GRM2
IO/-M
-BASE1 ROM
-BASE2 ROM
-EROM 2
-EROM 3
-EROM 4
-EROM 5
-EROM 6
-EROM 7
-IOW
-MEMR
-IOR
-MEMW
-8259 CS
-FDC CS
DSPT2
NCCLK
GRM3
CPUAI5
CPUAI4
-RESET
VSYC
R/-DT
+E
RA0
CA14
DSPT0
ENB3
-RQ
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7
MA8
MA9
MA10
MA11
MA12
-AGP1
-AGC 1
-AGR 1
MUX 1
-AGP 2
-AGC 2
-AGR 2
MUX 2
-64K CDIN
9
10
11
12
A
B
C
D
E
F

Appendix C
Character Code Table/Character Font

■ Display Character Code(AN of CG1)



■ Display Character Code (ANK of CG2)



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL		SP	0	@	P	`	p			▒	ー	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			"	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A			*	:	J	Z	j	z			ェ	コ	ハ	レ		
B		←	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C		↑	,	<	L	¥	ℓ	|			ャ	シ	フ	ワ		
D			−	=	M	]	m	}			ュ	ス	ヘ	ン		
E		→	.	>	N	^	n	~			ョ	セ	ホ	゛		
F		←	/	?	O	_	o				ッ	ソ	マ	゜		

■ Thermal Printer (IBM5513) Character Code

bit 4～7

bit 0～3	0	1	2	3	4	5	6	7	8	9	A	E	F
0	NUL	▶	SP	0	@	P	`	p	Ç	É	á	∞	≡
1		◀	!	1	A	Q	a	q	ü	æ	í	ß	±
2		DC2	"	2	B	R	b	r	é	Æ	ó	Γ	≥
3	♥	‼	#	3	C	S	c	s	â	ô	ú	π	≤
4	♦	DC4	$	4	D	T	d	t	ä	ö	ñ	Σ	⌠
5	♣	§	%	5	E	U	e	u	à	ò	Ñ	σ	⌡
6	♠	▬	&	6	F	V	f	v	å	û	ª	µ	÷
7	•	↨	'	7	G	W	g	w	ç	ù	º	τ	≈
8	◘	CAN	(	8	H	X	h	x	ê	ÿ	¿	Φ	°
9	HT	↓	)	9	I	Y	i	y	ë	Ö	⌐	Θ	∙
A	LF	→	*	:	J	Z	j	z	è	Ü	¬	Ω	·
B	VT	ESC	+	;	K	[	k	{	ï	¢	½	δ	√
C	FF	∟	,	<	L	\	l	|	î	£	¼	∞	n
D	CR	↔	-	=	M	]	m	}	ì	¥	¡	ø	²
E	SO	▲	.	>	N	^	n	~	Ä	₧	«	∈	■
F	SI	▼	/	?	O	_	o	DEL	Å	ƒ	»	∩	SP

Remark) Applied in English Mode

Thermal Transfer Printer (IBM5512) Character Code

The IBM 5512 has three kinds of character sets:

1. PC character set
   corresponds to AN of character generator 1.
2. Nihongo DOS character set
   corresponds to ANK of character generator 2. "7F", however, is ⧖ and "A0" is blank.
3. Kana character set (See the following chart)

bit 4～7

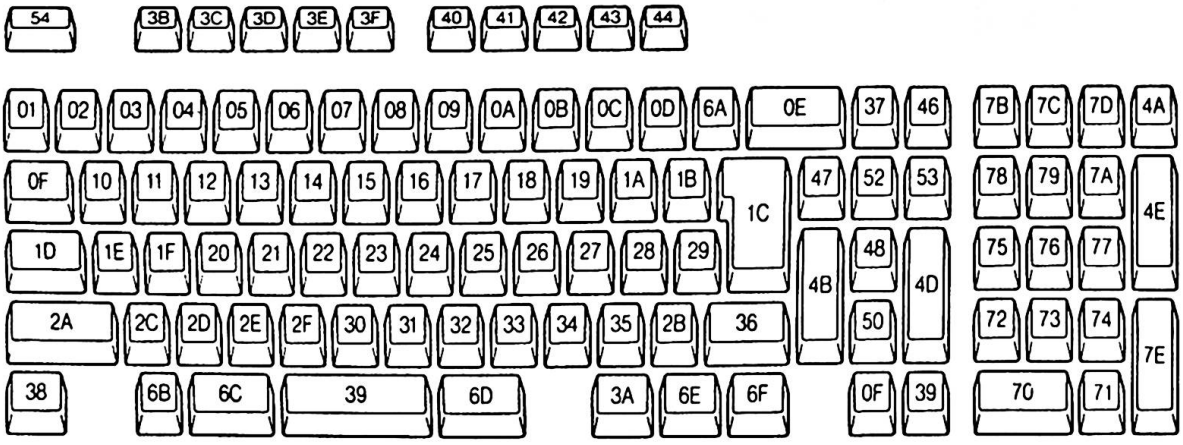
bit 0～3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	►	SP	0	@	P	‘	p			á	ー	タ	ミ		
1	☺	◄	!	1	A	Q	a	q			。	ア	チ	ム		円
2	☻	↕	"	2	B	R	b	r			「	イ	ツ	メ		年
3	♥	!!	#	3	C	S	c	s			」	ウ	テ	モ		月
4	♦	¶	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	♣	§	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	♠	▬	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	•	↨	’	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	◘	↑	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	干
9	○	↓	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	市
A	◙	→	*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	区
B	♂	←	+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	町
C	♀	∟	,	<	L	\	l	¦			ャ	シ	フ	ワ	●	村
D	♪	↔	—	=	M	]	m	}			ュ	ス	ヘ	ン	○	人
E	♫	▲	.	>	N	^	n	~			ョ	セ	ホ	゛		▒
F	☼	▼	/	?	O	_	o	Δ			ッ	ソ	マ	゜		

■ Scan Code

The following chart shows scan codes for Native and English modes. ( ) is for English mode.

Scan Code	Key	Scan Code	Key	Scan Code	Key	Scan Code	Key	Scan Code	Key	Scan Code	Key
01	ESC	12	E	23	H	34	.	46	Scroll	6E(**)	カタカナ
02	1	13	R	24	J	35	/		Lock	6F(**)	ひらがな
03	2	14	T	25	K	36	⇧ 右	47	↖	70 (52)	0
04	3	15	Y	26	L	37	*	48	↑	71 (53)	.
05	4	16	U	27	;	38	前面	4A	−	72 (4F)	1
06	5	17	I	28	'	39	スペース	4B	←	73 (50)	2
07	6	18	O	29	`	3A	英数	4D	→	74 (51)	3
08	7	19	P	2A	⇧ 左	3B	PF1	4E	+	75 (4B)	4
09	8	1A	[	2B	\	3C	PF2	50	↓	76 (4C)	5
0A	9	1B	]	2C	Z	3D	PF3	52	挿入	77 (4D)	6
0B	0	1C	↵	2D	X	3E	PF4	53	削除	78 (47)	7
0C	−	1D	Ctrl	2E	C	3F	PF5	54	Fn	79 (48)	8
0D	=	1E	A	2F	V	40	PF6	55	Invalid	7A (49)	9
0E	後退	1F	S	30	B	41	PF7	6A(**)	¥	7B (37)	*
0F	⇆	20	D	31	N	42	PF8	6B(**)	漢字	7C (35)	/
10	Q	21	F	32	M	43	PF9	6C (39)	無変換	7D (33)	,
11	W	22	G	33	,	44	PF10	6D (39)	変換	7E (1C)	↵

**……「Not used」



■ Keyboard Hankaku (half-size) Character Code (1 of 2)

Alphanumeric				Katakana				CTRL	Alt
Lower Case		Upper Case		Lower Case		Upper Case			
Esc	1B	Esc	1B	Esc	1B	Esc	1B	(Cassette)	−1
1	31	!	21	ヌ	C7		−1	−1	Null+78
2	32	@	40	フ	CC		−1	NUL(00)	Null+79
3	33	#	23	ア	B1	ァ	A7	−1	Null+7A
4	34	$	24	ウ	B3	ゥ	A9	−1	Null+7B
5	35	%	25	エ	B4	ェ	AA	−1	Null+7C
6	36	^	5E	オ	B5	ォ	AB	RSO(1E)	Null+7D
7	37	&	26	ヤ	D4	ャ	AC	−1	Null+7E
8	38	*	2A	ユ	D5	ュ	AD	−1	Null+7F
9	39	(	28	ヨ	D6	ョ	AE	−1	Null+80
0	30	)	29	ワ	DC	ヲ	A6	−1	Null+81
-	2D	_	5F	ホ	CE		−1	US(1F)	Null+82
=	3D	+	2F	ヘ	CD		−1	−1	Null+83
後退*	08	後退*	08	後退*	08	後退*	08	DEL(7F)	−1
→|	09	|←	Null+0F	→|	09	|←	Null+0F	−1	−1
q	71	Q	51	タ	C0		−1	DC1(11)	Null+10
w	77	W	57	テ	C3		−1	ETB(17)	Null+11
e	65	E	45	イ	B2	ィ	A8	ENQ(05)	Null+12
r	72	R	52	ス	BD		−1	DC2(12)	Null+13
t	74	T	54	カ	B6		−1	DC4(14)	Null+14
y	79	Y	59	ン	DD		−1	EM(19)	Null+15
u	75	U	55	ナ	C5		−1	NAK(15)	Null+16
i	69	I	49	ニ	C6		−1	HT(09)	Null+17
o	6F	O	4F	ラ	D7		−1	SI(0F)	Null+18
p	70	P	50	セ	BE		−1	DLE(10)	Null+19
[	5B	{	7B	"	DE		−1	Esc(1B)	−1
]	5D	}	7D	。	DF	「	A2	GS(1D)	−1
改行**	0D	改行**	0D	改行**	0D	改行**	0D	LF(0A)	−1

Remark : 「−1」 is 「Not Used」.

*…… Backspace

**… Carriage Return

■ Keyboard Hankaku (half-size) Character Code (2 of 2)

Alphanumeric				Katakana				CTRL	Alt
Lower Case		Upper Case		Lower Case		Upper Case			
Ctrl	−1		−1		−1		−1	−1	−1
a	61	A	41	チ	C1		−1	SOH(01)	Null+1E
s	73	S	53	ト	C4		−1	DC3(13)	Null+1F
d	64	D	44	シ	BC		−1	EOT(04)	Null+20
f	66	F	46	ハ	CA		−1	ACK(06)	Null+21
g	67	G	47	キ	B7		−1	BELL(07)	Null+22
h	68	H	48	ク	B8		−1	BS(08)	Null+23
j	6A	J	4A	マ	CF		−1	LF(0A)	Null+24
k	6B	K	4B	ノ	C9		−1	VT(0B)	Null+25
l	6C	L	4C	リ	D8		−1	FF(0C)	Null+26
;	3B	:	3A	レ	DA		−1	−1	−1
'	27	"	22	ケ	B9		−1	−1	−1
‘	60	(~)	−1	ム	D1	」	A3	−1	−1
Left Shift	−1		−1		−1		−1	−1	−1
(\)	−1	|	7C	ロ	DB		−1	−1	−1
z	7A	Z	5A	ツ	C2	ッ	AF	SUB(1A)	Null+2C
x	78	X	58	サ	BB		−1	CAN(18)	Null+2D
c	63	C	43	ソ	BF		−1	EXT(03)	Null+2E
v	76	V	56	ヒ	CB		−1	SYN(16)	Null+2F
b	62	B	42	コ	BA		−1	STX(02)	Null+30
n	6E	N	4E	ミ	D0		−1	SO(02)	Null+31
m	6D	M	4D	モ	D3		−1	CR(0D)	Null+32
,	2C	<	3C	ネ	C8	・	A4	−1	−1
.	2E	>	3E	ル	D9	。	A1	−1	−1
/	2F	?	3F	メ	D2	、	A5	−1	−1
Right Shift	−1		−1		−1		−1	−1	−1
*	2A	Prt Screen	−1	*	2A	Prt Screen	−1	Echo −1	−1
¥	5C	―	7E	−	B0		−1	−1	−1

Remark : 「−1」 is 「Not Used」.
: (\) and (~) are valid only in English mode.

■ Keyboard Zenkaku (full-size) Character Code (1 of 2)

Scan Code	Alphanumeric		Katakana		Hiragana	
	Lower Case	Upper Case	Lower Case	Upper Case	Lower Case	Upper Case
02	8250	8149	836B	−1	82CA	−1
03	8251	8197	8374	−1	82D3	−1
04	8252	8194	8341	8340	82A0	829F
05	8253	8190	8345	8344	82A4	82A3
06	8254	8193	8347	8346	82A6	82A5
07	8255	814F	8349	8248	82A8	82A7
08	8256	8195	8384	8383	82E2	82E1
09	8257	8196	8386	8385	82E4	82E3
0A	8258	8169	8388	8387	82E6	82E5
0B	824F	816A	838F	8392	82ED	82F0
0C	817C	8151	837A	8192	82D9	8192
0D	8181	817B	8377	8158	82D6	8158
10	8291	8270	835E	−1	82BD	−1
11	8297	8276	8365	−1	82C4	−1
12	8285	8264	8343	8342	82A2	82A1
13	8292	8271	8358	−1	82B7	−1
14	8284	8273	834A	−1	82A9	−1
15	8299	8278	8393	−1	82F1	−1
16	8295	8274	8369	−1	82C8	−1
17	8289	8268	836A	−1	82C9	−1
18	828F	826E	8389	−1	82E7	−1
19	8290	826F	835A	8177	82B9	8177
1A	816D	816F	814A	−1	814A	−1
1B	816E	8170	814B	8175	814B	8175
1E	8281	8260	8360	−1	82BF	−1
1F	8293	8272	8367	−1	82C6	−1
20	8284	8263	8356	−1	82B5	−1
21	8286	8265	836E	−1	82CD	−1
22	8287	8266	834C	−1	82AB	−1
23	8288	8267	834E	−1	82AD	−1

Remark : 「− 1 」 is 「Not Used」.

■ Keyboard Zenkaku (full-size) Character Code (2 of 2)

Scan Code	Alphanumeric		Katakana		Hiragana	
	Lower Case	Upper Case	Lower Case	Upper Case	Lower Case	Upper Case
24	828A	8269	837D	−1	82DC	−1
25	828B	826A	836D	−1	82CC	−1
26	828C	826B	838A	−1	82E8	−1
27	8147	8146	838C	8178	82EA	8178
28	814C	818D	8350	8396	82AF	8396
29	814D	8160	8380	8176	82DE	8176
2B	815F	8162	838D	−1	82EB	−1
2C	829A	8279	8363	8362	82C2	F2C1
2D	8298	8277	8354	−1	82B3	−1
2E	8283	8262	835C	−1	82BB	−1
2F	8296	8275	8371	−1	82D0	−1
30	8282	8261	8352	−1	82B1	−1
31	828E	826D	837E	−1	82DD	−1
32	828D	826C	8382	−1	82E0	−1
33	8143	8271	836C	8141	82CB	8141
34	8144	8172	828B	8142	82E9	8142
35	815E	8148	8381	8145	82DF	8145
6A	818F	8150	815B	−1	815B	−1

Remark : 「— 1 」 is 「Not Used」.

■ Keyboard Character Code

Lower Case		Upper Case		CTRL		Alt	
Alt	−1		−1		−1		−1
Space	20	Space	20	Space	20	Space	20
AN	−1		−1		−1		−1
PF1	Null+3B	PF11	Null+54	PF21	Null+5E	PF31	Null+68
PF2	Null+3C	PF12	Null+55	PF22	Null+5F	PF32	Null+69
PF3	Null+3D	PF13	Null+56	PF23	Null+60	PF33	Null+6A
PF4	Null+3E	PF14	Null+57	PF24	Null+61	PF34	Null+6B
PF5	Null+3F	PF15	Null+58	PF25	Null+62	PF35	Null+6C
PF6	Null+40	PF16	Null+59	PF26	Null+63	PF36	Null+6D
PF7	Null+41	PF17	Null+5A	PF27	Null+64	PF37	Null+6E
PF8	Null+42	PF18	Null+5B	PF28	Null+65	PF38	Null+6F
PF9	Null+43	PF19	Null+5C	PF29	Null+66	PF39	Null+70
PF10	Null+44	PF20	Null+5D	PF30	Null+67	PF40	Null+71
Scroll Lock	−1		−1	Break	−1		−1
Home	Null+47	Home	Null+47		Null+77		−1
Cur up	Null+48	Cur up	Null+48		−1		−1
−	2D	−	2D		−1		−1
Cur left	Null+4B	Cur left	Null+4B		Null+73		−1
Cur right	Null+4D	Cur right	Null+4D		Null+74		−1
+	2B	+	2B		−1		−1
Cur down	Null+50	Cur down	Null+50		−1		−1
挿入	Null+52	挿入	Null+52		−1		−1
削除	Null+53	(Num Pad '.')			−1		−1
Fn	−1		−1		−1		−1
漢字	−1	漢字	−1		−1		−1
無変換	20	無変換	20		−1		−1
変換	20	変換	20		−1		−1
カタカナ	−1	カタカナ	−1		−1	半角	−1
ひらがな	−1	ひらがな	−1		−1	全角	−1

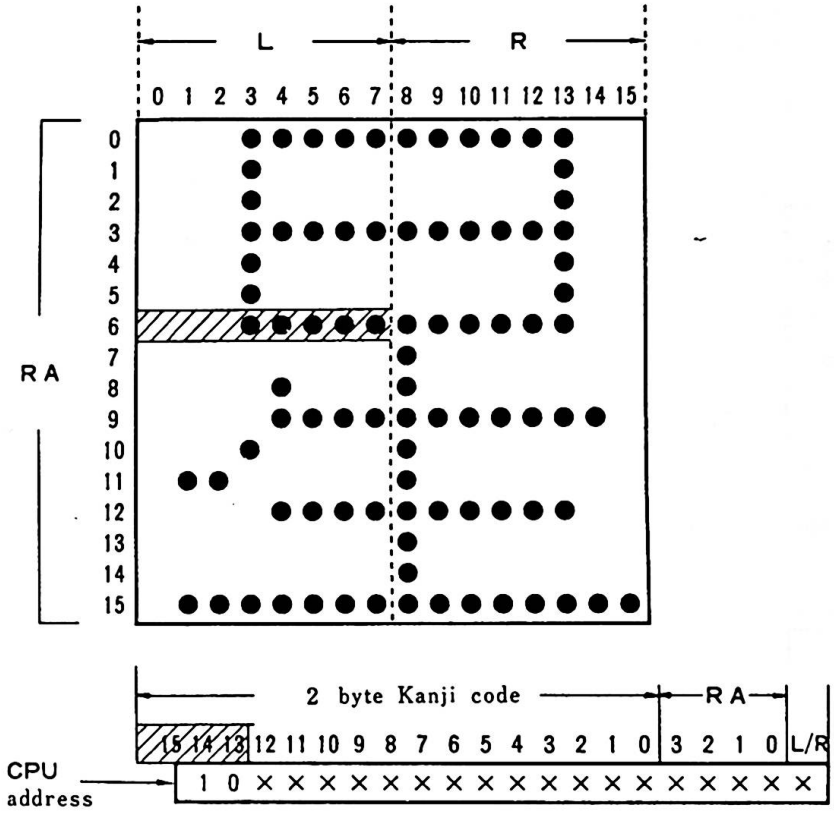
Remark：「−1」is「Not Used」.

- Combined key functions

Key combination	Function sample
[Ctrl] +Scroll Lock	Break
[Ctrl] +印刷	Echo
[⇧] +印刷	Print Screen
Fn+Q	Pause
[前面] +Fn+N	Numeric Lock
Fn+ [←]	Page Up
Fn+ [→]	Page Down
[↖] or Fn+ [↑]	Home Position
Fn+ [↓]	End
[Ctrl] + [↖]	Screen Clear (BASIC)
Fn+ [↵]	Execution
Fn+H	Interrupt
[前面] + [Ctrl] + [→]	Screen right
[前面] + [Ctrl] + [←]	Screen left
[前面] + [Ctrl] + [挿入]	System Reset
[前面] + [Ctrl] + [削除]	System Reset
[Ctrl] +ESC	Cassette auto-load
Fn+T	Termination

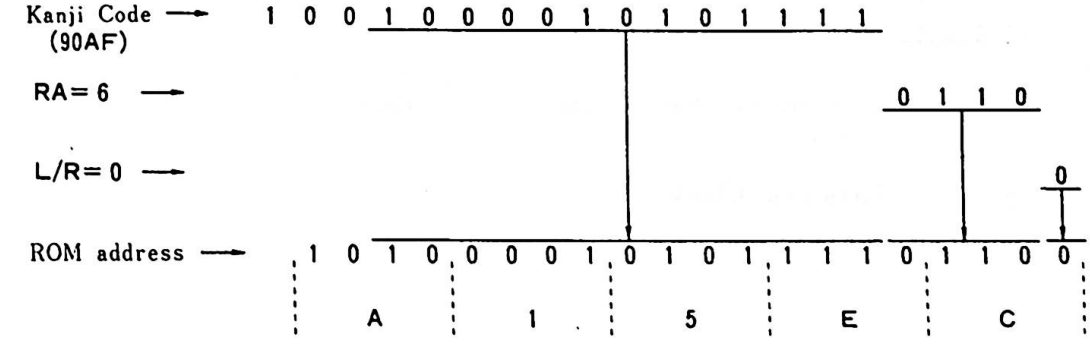
Remark) Functions may differ from application to application.
This is a sample.

■ Kanji ROM address calculation



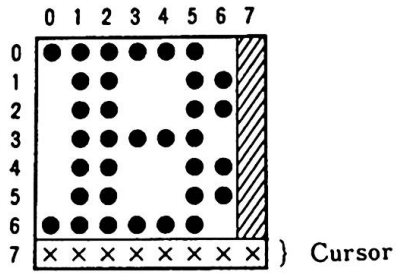
Calculation Sample

The Kanji ROM address for the shaded dot pattern can be calculated as follows :

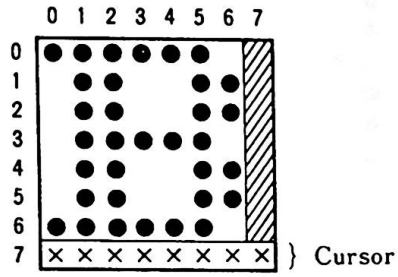


■ Character Font

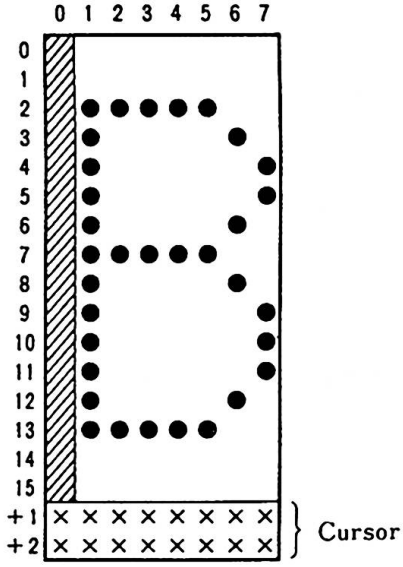
CG1 Alphanumeric · Special Character 8×8
CG2 Alphanumeric · Special Char. and Katakana 8×8
Alphanumeric · Special Char. Hankaku Char. of Katakana 8×16
Hiragana and Kanji 16×16



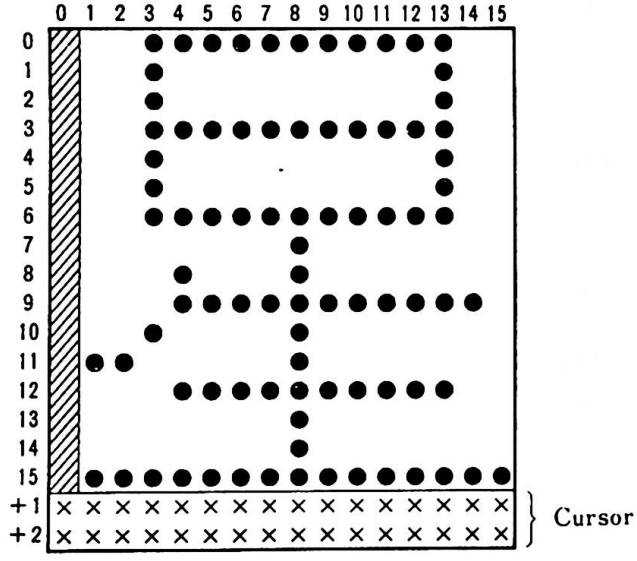
CG1 Sample : B
7×7 (8×8)
Alphanumeric · Special Char.



CG2 Sample : B
7×7 (8×8)
Alphanumeric · Spcial Char. · Katakana



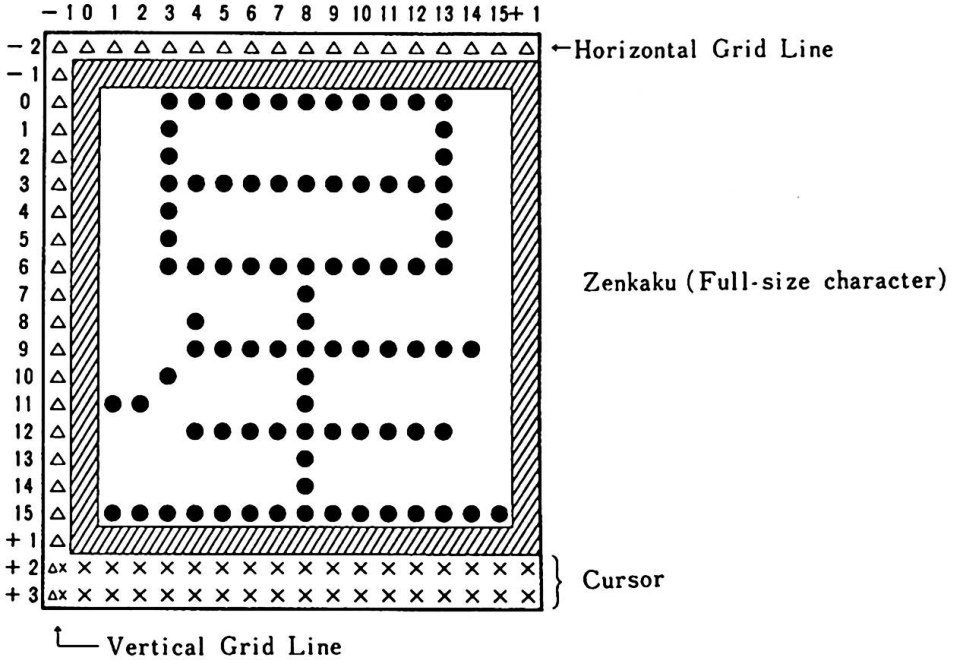
CG2 7×16 (8×16)
Hankaku : Alphanumeric · Special Char.
Katakana

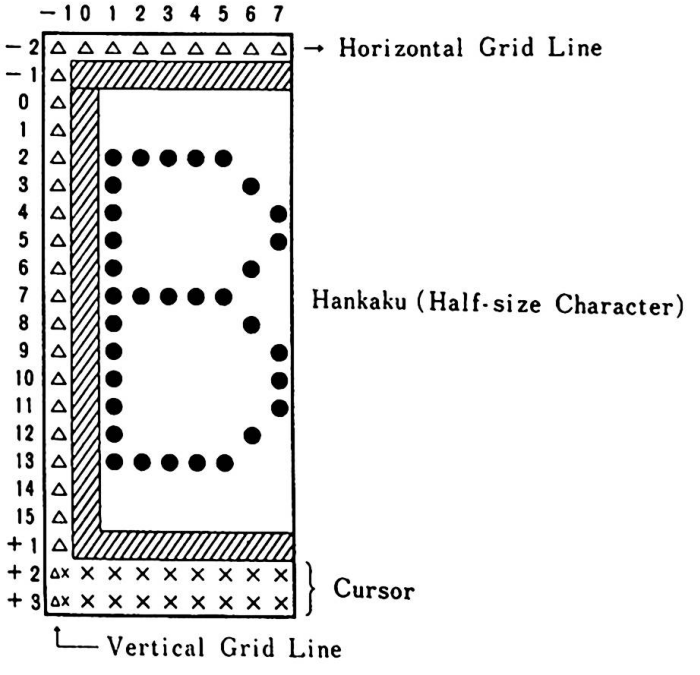


CG2 15×16 (16×16)
Kanji, Hiragana

Remark (▨ : 「always blank」
XX : cursor dot

■ Character Display Format of Extension Video mode





Remarks) ▨ : 「always blank」 portion
XX : Cursor dot
△ : Grid Line dot

Index

1
12" Color Display, 1-4, 4-33
12" Monochrome Display, 1-4, 4-33
128KB RAM Card, 1-3, 4-6
14" Color Display, 1-4, 4-33

6
64KB RAM Card, 1-3, 4-2

A
Address Change, 2-19
Asynchronous communication, 4-26
Attribute, 3-11, 3-12, 3-29
Audio Interface, 2-38

B
Beep, 2-38
BIOS, 5-1, 5-5, 5-43
Block definition value, 2-16
Border Color Register, 3-20

C
Cartridge, 2-47
Cassette, 2-27
CG1, 3-8
CG2, 3-8
Character Generator, 2-15, 3-8
Character Generator 1, 3-8
Character Generator 2, 3-8
Clock, 2-8, 3-26
CMT Cable, 1-4, 4-34
Compatibility, 6-1
Connector, 2-5, 2-6
CPU, 2-7
CRT Controller, 3-39

D
Diskette Compatibility, 6-7
Diskette controller, 4-13
Diskette Drive, 1-3, 4-23, 6-4
Diskette Drive Adapter, 1-3, 4-12
Display, 4-33
DOS, 2-50

E
English Mode, 1-2, 3-2
Expanded memory, 3-4
Expansion Board, 1-5, 4-36
Expansion Channel, 2-22
Expansion Unit, 1-5, 4-37
Extension Video Card, 1-3, 4-7
Extension Video mode, 3-2, 4-7, 5-3

Extension Video Mode BIOS, 5-35

G
Gaiji RAM, 2-15
Gate Array, 3-18, 3-33, 4-7
General-use memory, 2-14
Graphics, 3-13, 3-27, 3-31

I
I/O address, 3-42, 5-49, 5-50
Infrared Receiver, 2-42
Interrupt, 2-8, 4-14
Interrupt vector, 5-6

J
Joystick, 1-4, 4-35
Joystick Interface, 2-44

K
Keyboard, 2-59
Keyboard Cable, 1-4, 4-25
Keyboard Cable Interface, 2-43
Keyboard data, 2-39, 2-60
Keyboard Interface, 2-39

L
Light Pen Interface, 3-40

M
Memory, 2-14, 6-6
Memory Map, 3-5, 5-46
Memory Space, 2-16, 5-47, 5-48
Mode Control 1 Register, 3-19
Mode Control 2 Register, 3-21

N
Native mode, 3-2
Native Mode BIOS, 5-8
Nativemode, 5-3
NMI, 2-13

O
Operation Mode, 1-2, 5-3
Optional Features, 4-1

P
Page, 1-3, 3-4
Page Register, 3-4
Palette, 3-24
Palette Mask Register, 3-20
Parallel Port, 2-10
Port A0, 2-13, 2-41
Power Unit, 2-62
Printer, 1-4

Printer Cable, 1-5
Printer Interface, 2-55
Processing speed, 6-6
Processor, 2-7

R
Register 0, 3-34
Register 1, 3-34
Register 2, 3-34
Register 3, 3-35
Register 4, 3-35
Register 5, 3-36
Register 6, 3-36
Register 7, 3-37
Reset Register, 3-22
ROM cartridge, 2-47
RS-232C, 1-4, 4-26, 6-5
RS-232C Cable, 1-4, 4-32

S
Scan Code, 2-61, 5-45
Self-diagnostic test, 2-48, 2-50
Sound Generator, 2-31, 2-33, 6-4
Sound Source, 2-32
Sound Subsystem, 2-31
Superimpose, 3-23, 3-26
System Board, 2-3
System ROM, 2-14, 2-47
System Software, 5-3

T
Text Display, 3-11, 3-12, 3-29
Timer, 2-7, 6-5
Transparent Palette Register, 3-22
TV Adapter, 1-4, 4-24

U
User memory, 6-4, 6-5

V
Video processor, 3-2
Video RAM, 2-15, 3-4, 3-27, 3-28, 3-38, 4-7
Video Subsystem (VSS), 3-1
Virtual address, 3-4
VP1, 3-1
VP2, 3-1
VP3, 3-1, 4-7
VRAM 1, 3-4
VRAM 2, 3-4
VSS, 3-1

