IBM PC Convertible Technical Reference Volume 2

55X8817

PC Convertible Technical Reference

Volume 2

First Edition (February 1986)

Preface

The IBM PC Convertible Technical Reference consists of two volumes. Volume 1 describes the hardware design and provides interface information for the IBM PC Convertible. Volume 1 also has information about the basic input/output system (BIOS) and programming support. Volume 2 contains the BIOS listings.

The information in these volumes is both descriptive and reference-oriented and is intended for hardware and software designers, programmers, engineers, and other interested persons who need to understand the design and operation of the IBM PC Convertible. These users should be familiar with the use of the IBM PC Convertible and understand the concepts of computer architecture and programming.

Volume 1 has five sections:

- Section 1, "Introduction" is an overview of the system and the available options.
- Section 2, "System Unit" describes each functional part of the base system. This section also contains the description of the interfaces. Programming considerations are supported by command code and register descriptions.
- Section 3, "System Options" describes each available option.
- Section 4, "System BIOS and Usage" describes the basic input/output system (BIOS) and its use. This section also contains the software interrupt listing, a system memory map, descriptions of vectors with special meanings, and a set of low-storage maps. In addition, Section 4 describes keyboard encoding and usage.
- Section 5, "Compatibility with the IBM Personal Computer Family" describes programming concerns for maintaining compatibility among the IBM PC Convertible system and the other IBM Personal Computers.

Volume 1 has the following appendixes:

- Appendix A, "Characters and Keystrokes"
- Appendix B, "Unit Specifications"
- Appendix C, "Logic Timing Diagrams"
- Appendix D, "Power-On Self Test Error Codes".

Volume 2 contains the BIOS listing and is to be used in conjunction with Volume 1.

Prerequisite Publications

Guide to Operations: for the IBM PC Convertible.

Suggested Reading

- *BASIC* for the IBM Personal Computer
- *Disk Operating System*
- *Hardware Maintenance and Service* for the IBM PC Convertible
- *MACRO Assembler* for the IBM Personal Computer.

Contents

Notes:

Section 1. Power-On Self-Test (POST)

The IBM PC Convertible uses a set of routines called the power-on self-test (POST) routines. These routines automatically activate system power, perform basic function and device control tests for the major system components, and initialize the system to the ready state. Because the operation of these routines is automatic and cannot be changed by an application programmer, the actual program listings are not included in this manual. However, it may be useful to understand the function of these routines and the initial values they may establish.

Figure 1-1 on page 1-2 shows the sequence of the steps the power-on routines, the error indicators returned, and the initial values established. Figure 1-2 on page 1-9 and Figure 1-3 on page 1-12 provide additional information on testing for the presence of devices.

Power-On Self-Test Sequence

Description	Error Indication
Processor test: Verifies processor operation.	System power off.
System ROM checksum test: Verifies system ROM.	1-long and 1-short beep, then system power off.
Real-time clock RAM test: Verifies the clock. If contents have been altered, RAM is cleared and system profile defaults are loaded.	1-long and 1-short beep, then system power off.
Power source and battery check: Checks for external power. If no external power, checks for good battery.	3-short beeps, then system power off.
Base 8K RAM test: Validates first 8K of RAM and clears it to zero.	1-long and 1-short beep, then system power off.
Internal modem power on: Checks for internal modem presence and applies power to it if present. See Figure 1-2 on page 1-9 for additional information.	

Figure 1-1 (Part 1 of 7). Power-on Self-Test Sequence

Description	Error Indication
Video adapter presence test: Checks for attached video adapters by nondestructive writing and reading hex A55A to address hex B0000 and B8000. Checks for video adapters with valid rom. Checks for LCD by determining if the 'LCD sense' line is set.	
Video interrupt vectors initialization: Initializes hex 10, 1D, 1F and 44 video interrupt vectors to system default values.	
Video ROM validity tests: Checks for video adapter with valid ROM. See Volume 1 for additional information about adapters with system accessible ROM. Links to ROM if valid.	
LCD function test: Runs LCD function test if the LCD is present.	Error 5001, 5002, or 5003
CRT adapter without ROM presence and function tests: If a CRT adapter without ROM is present, test vertical and horizontal syncs.	Error 0501
Operable display presence test: Checks to ensure that an operable display is present.	1-long and 2-short beeps, then system power off.

Figure 1-1 (Part 2 of 7). Power-on Self-Test Sequence

Description	Error Indication
Display initialization: If not a resume sequence, initialize the display for POST. If the LCD is present, loads the default font from vectors hex 1F and 44.	
Interrupt controller test: Tests the interrupt controller and initializes it to edge triggered, software end-of-interrupt, and interrupt types 8-15.	Error 0101 with icon, 1-long and 1-short beep, then waits for power off or Fn/Ctl/Del sequence.
System timers tests: Tests the system timers. Timer-0 is initialized to 18.2 interrupts per second rate.	One of the following: Error 0103 with icon, 1-long and 1-short beep, then waits for power off or Fn/Ctl/Del sequence. Error 0102 with icon and F1 prompt; POST continues.
DMA controller test: Tests the DMA controller.	Error 0105 with icon, 1-long and 1-short beep, then waits for power off or Fn/Ctl/Del sequence.

Figure 1-1 (Part 3 of 7). Power-on Self-Test Sequence

Description	Error Indication
Keyboard controller test: Tests the keyboard controller.	One of the following: Error 0303 or 0304 with icon, 1-long and 1-short beep, then waits for power off or Fn/Ctl/Del sequence. Error 0102 with icon and F1 prompt; POST continues.
Interrupt vectors initialization: Initializes interrupt vectors 0 through 32 excluding the video interrupt vectors.	
Serial/parallel initializations: Initializes the serial and parallel adapter timeout values.	
Internal modem initialization: Initializes the internal modem. See Volume 1 for additional information about initializing the internal modem.	
RAM block check: Checks RAM for good contiguous 8K-bytes of RAM.	Error 0164 with icon and F1 prompt.

Figure 1-1 (Part 4 of 7). Power-on Self-Test Sequence

Description	Error Indication
RAM function test: Checks for functional RAM in 64K-byte increments and clears it to zeroes. This test is not performed during either a resume or Ctl/Alt/Del sequence.	One of the following: When less than 64K-bytes of RAM is functional, error 0201 or 0202 with icon, 1-long and 1-short beep, then waits for power off or Fn/Ctl/Del sequence. When more than 64K-bytes of RAM is functional, error 0201 or 0102 with icon and F1 prompt. POST continues.
Diskette drive function test: Tests the operation of the diskette drives.	One of the following: Error 0601 with icon, 1-long and 1-short beep, then waits for power off or Fn/Ctl/Del sequence. Error 0601 with icon and F1 prompt.

Figure 1-1 (Part 5 of 7). Power-on Self-Test Sequence

Description	Error Indication
Real-time clock test: Tests the update and interrupt functions. Zeros and initializes time and date, and sets alarm to zeros if the clock has not been updated or if power was lost.	Error 0163 with icon and F1 prompt.
Portable printer interface test: Checks the function of the adapter.	Error 5101 with icon and F1 prompt.
Printer/communications presence test: Checks to determine what printer and communications features are attached. See Figure 1-3 on page 1-12 for additional information.	
Internal modem test: Checks the function of the internal modem.	Error 1101 or 1102 with icon and F1 prompt.
Serial adapter test: Tests the serial adapter portion of the IBM PC Convrtible Serial/Parallel Adapter if it is installed.	Error 1101 or 1201 with icon and F1 prompt.
Feature adapter ROM presence: External adapters are tested for the presence of ROM. See Volume 1 for additional information about adapters with system accessible ROM. Links to the ROM if valid.	Checksum or feature error.

Figure 1-1 (Part 6 of 7). Power-on Self-Test Sequence

Description	Error Indication
Set time of day: Sets time of day data area from real-time clock.	Clock icon if clock has not been set by the user.
Low battery test: Checks for good battery.	Battery icon if battery is low and warning is enabled in the system profile.
Complete POST: If a system suspend was active and the resume was not cancelled, the system state is restored and the application resumed. Otherwise, the LCD default font is loaded from vectors hex 1F and 44, and the system is booted from drive 0.	2-short beeps if any warning errors occurred.

Figure 1-1 (Part 7 of 7). Power-on Self-Test Sequence

Modem Presence Test

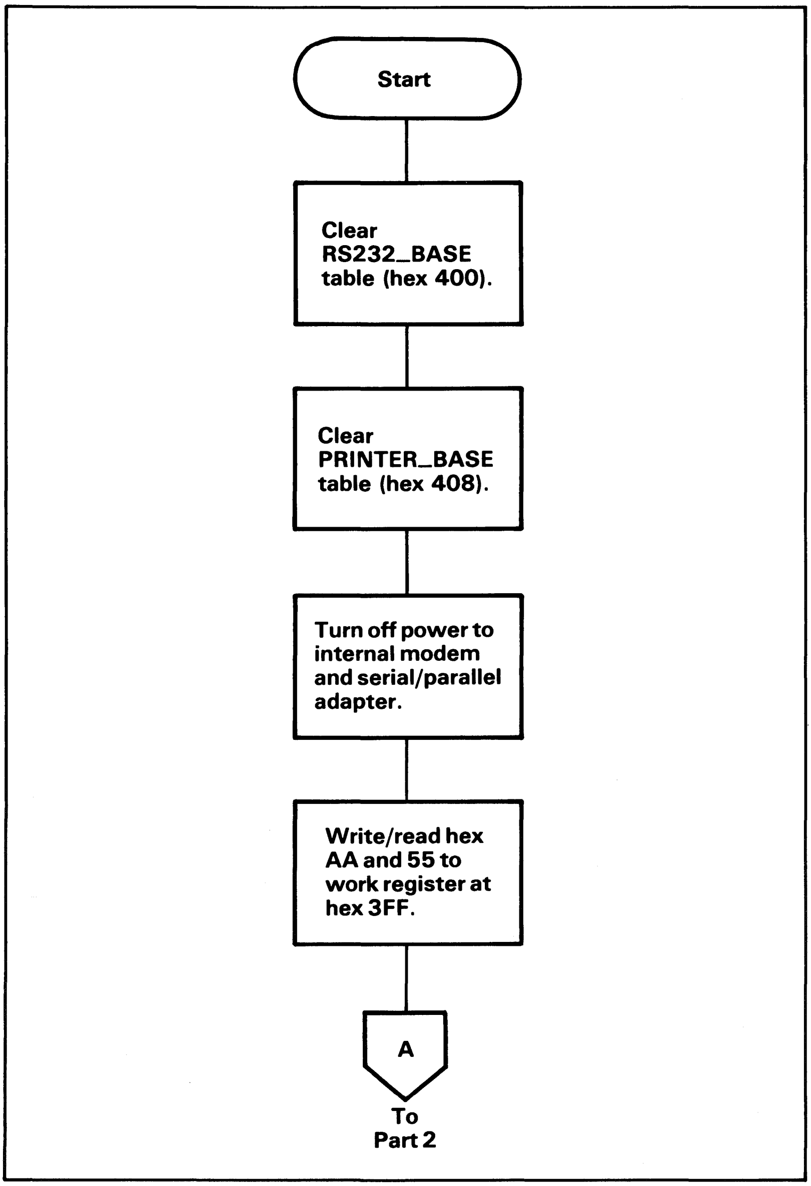

Start
Clear RS232_BASE table (hex 400).
Clear PRINTER_BASE table (hex 408).
Turn off power to internal modem and serial/parallel adapter.
Write/read hex AA and 55 to work register at hex 3FF.
A
To Part 2

Figure 1-2 (Part 1 of 3). Internal Modem Presence Test

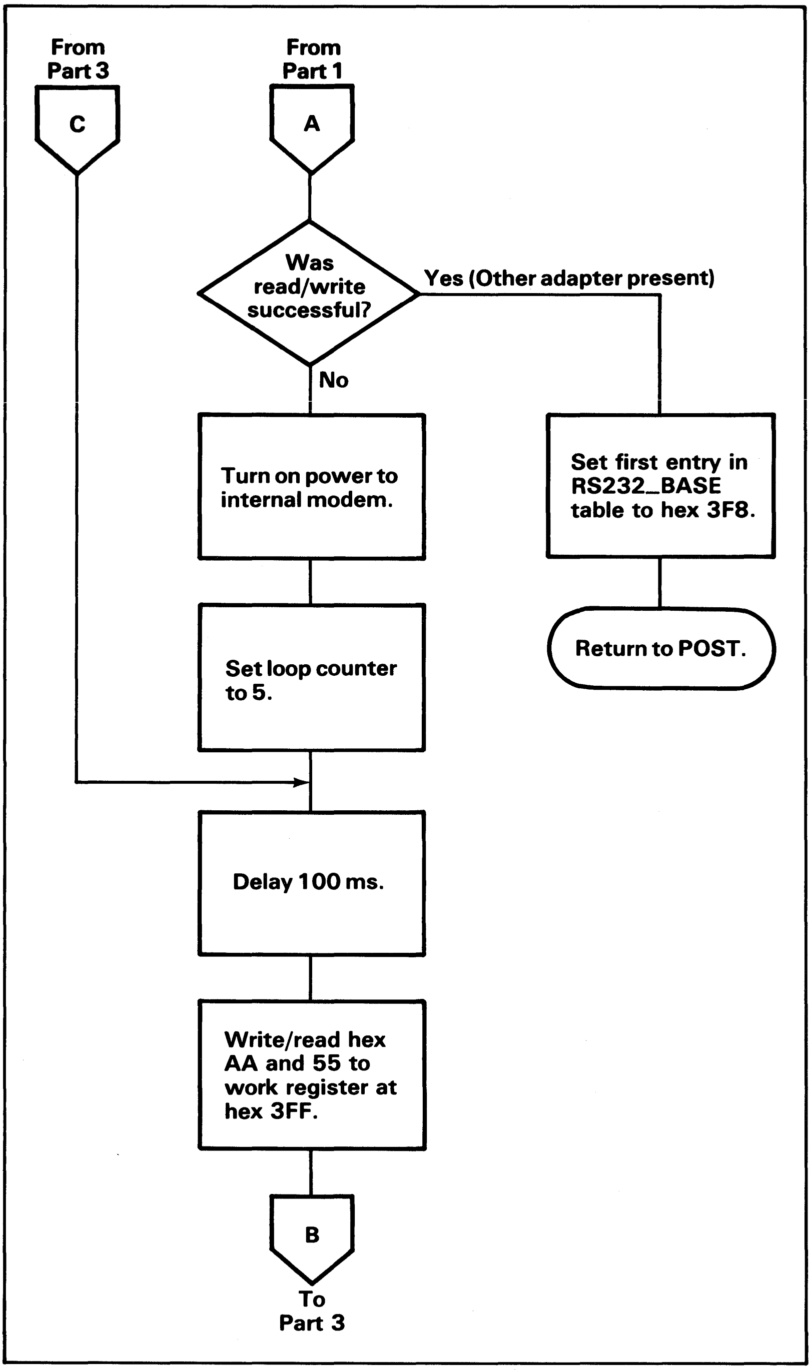

From Part 3
C
From Part 1
A
Was read/write successful?
Yes (Other adapter present)
No
Turn on power to internal modem.
Set first entry in RS232_BASE table to hex 3F8.
Return to POST.
Set loop counter to 5.
Delay 100 ms.
Write/read hex AA and 55 to work register at hex 3FF.
B
To Part 3

Figure 1-2 (Part 2 of 3). Internal Modem Presence Test

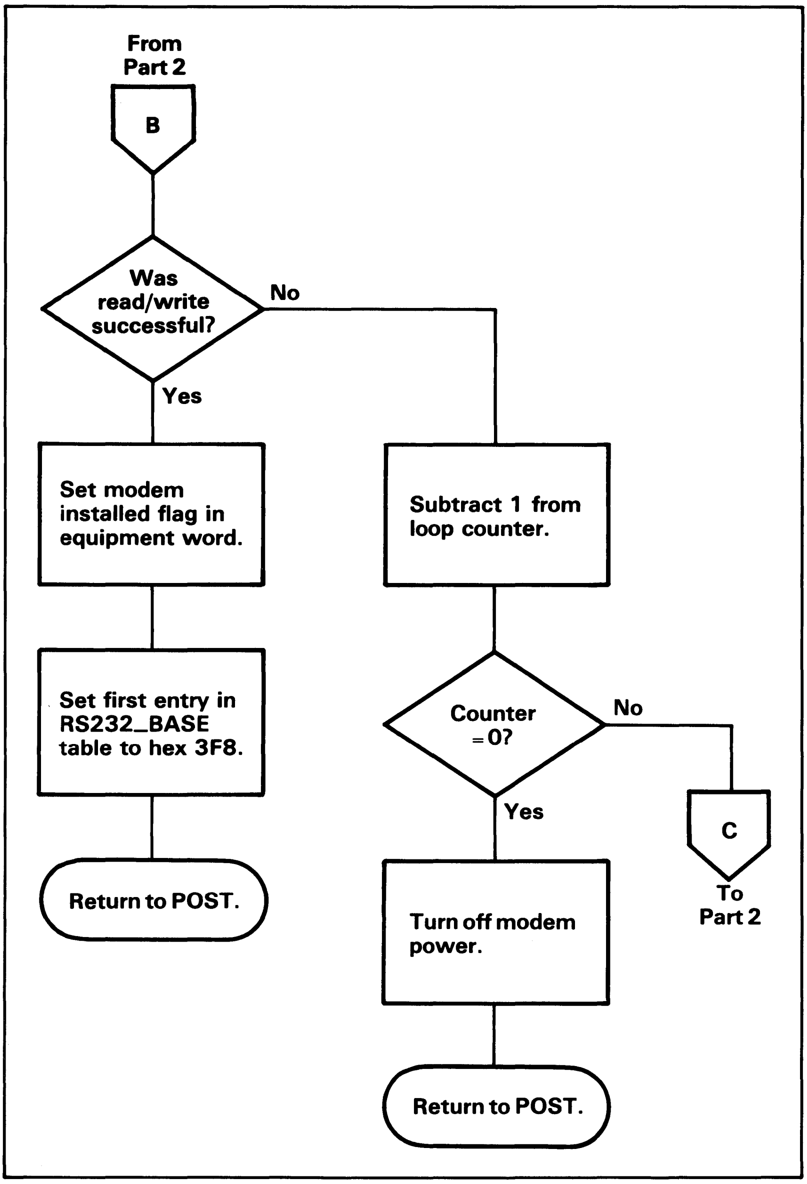
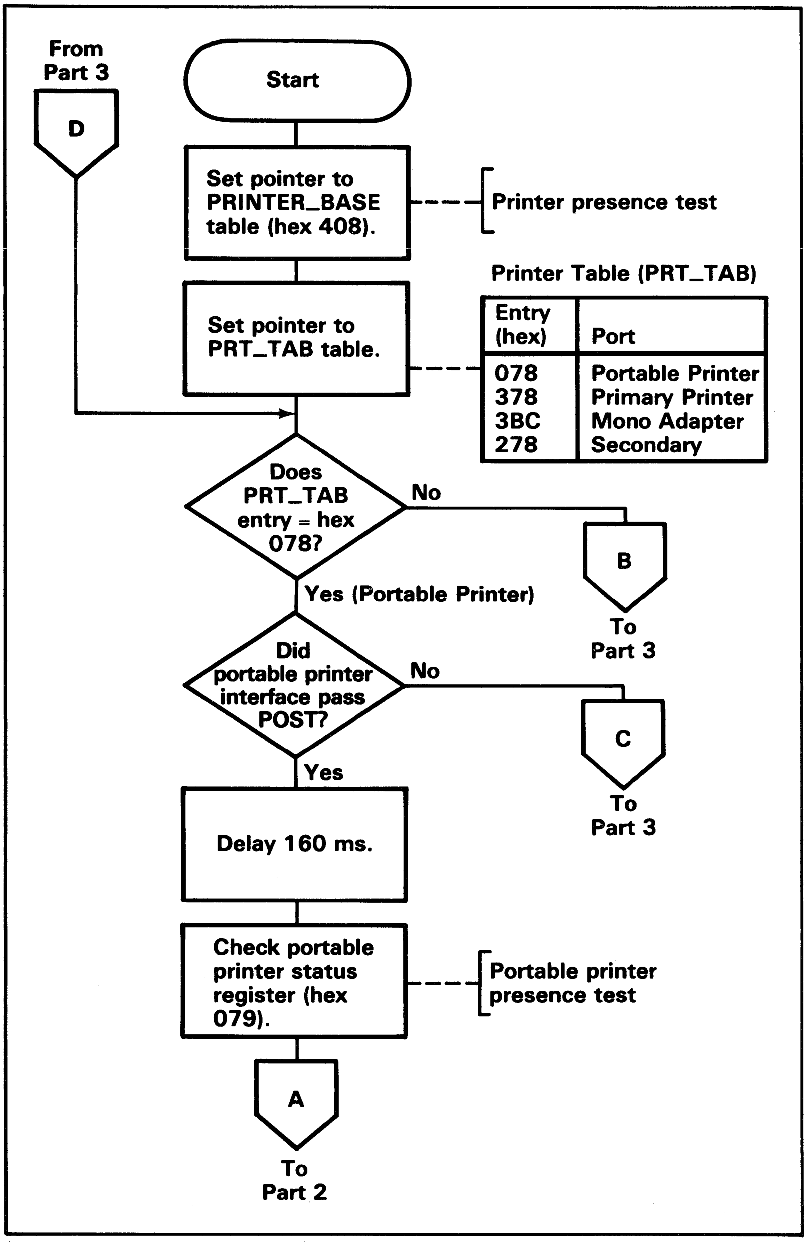

From
Part 2
B
Was read/write successful?
No
Yes
Set modem installed flag in equipment word.
Subtract 1 from loop counter.
Set first entry in RS232_BASE table to hex 3F8.
Counter =0?
No
Yes
C
To
Part 2
Return to POST.
Turn off modem power.
Return to POST.

Figure 1-2 (Part 3 of 3). Internal Modem Presence Test

Printer/Communications Presence Test


From Part 3
D
Start
Set pointer to PRINTER_BASE table (hex 408).
Printer presence test
Set pointer to PRT_TAB table.
Does PRT_TAB entry = hex 078?
No
B
To Part 3
Yes (Portable Printer)
Did portable printer interface pass POST?
No
C
To Part 3
Yes
Delay 160 ms.
Check portable printer status register (hex 079).
Portable printer presence test
A
To Part 2

Printer Table (PRT_TAB)

Entry (hex)	Port
078	Portable Printer
378	Primary Printer
3BC	Mono Adapter
278	Secondary

Figure 1-3 (Part 1 of 7). Printer/Communications Presence Test

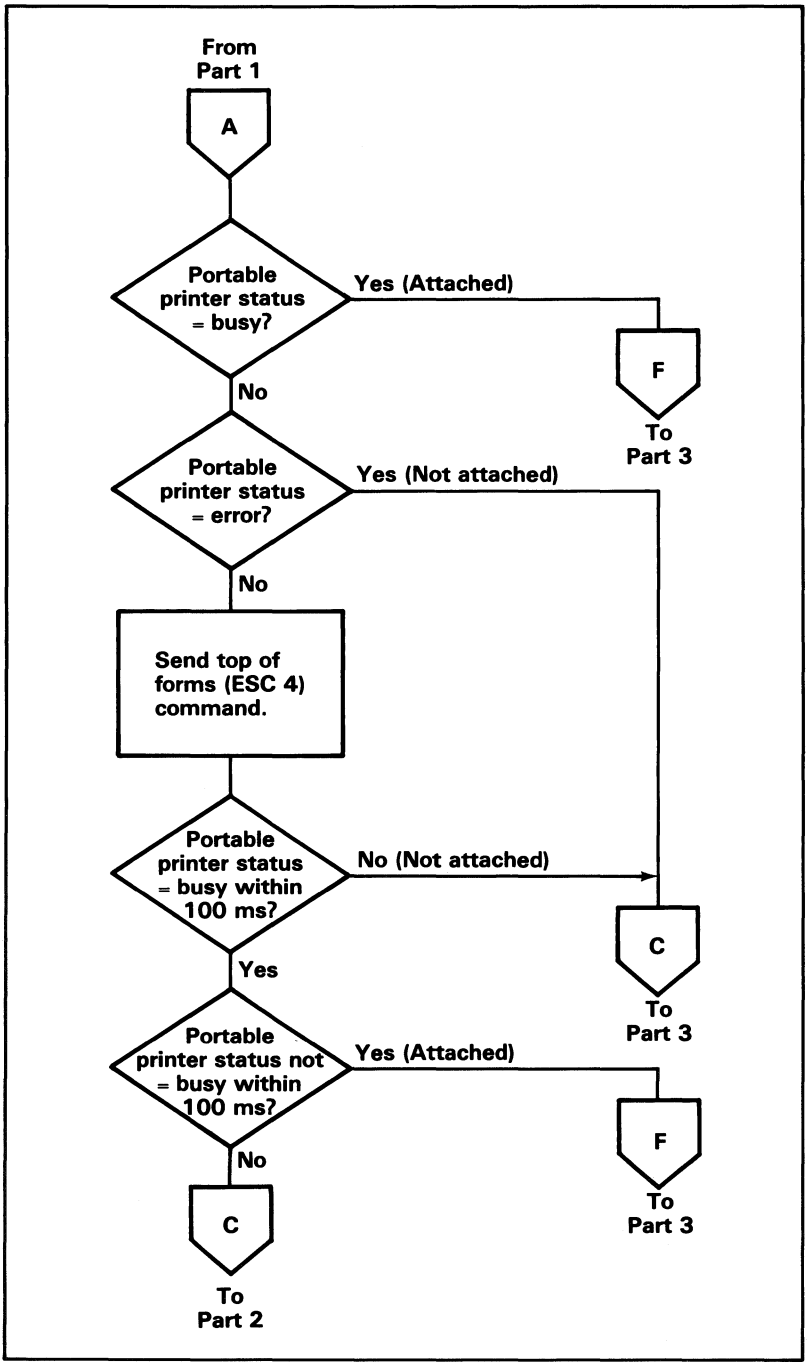

From Part 1
A
Portable printer status = busy?
Yes (Attached)
F
To Part 3
No
Portable printer status = error?
Yes (Not attached)
No
Send top of forms (ESC 4) command.
Portable printer status = busy within 100 ms?
No (Not attached)
C
To Part 3
Yes
Portable printer status not = busy within 100 ms?
Yes (Attached)
F
To Part 3
No
C
To Part 2

Figure 1-3 (Part 2 of 7). Printer/Communications Presence Test

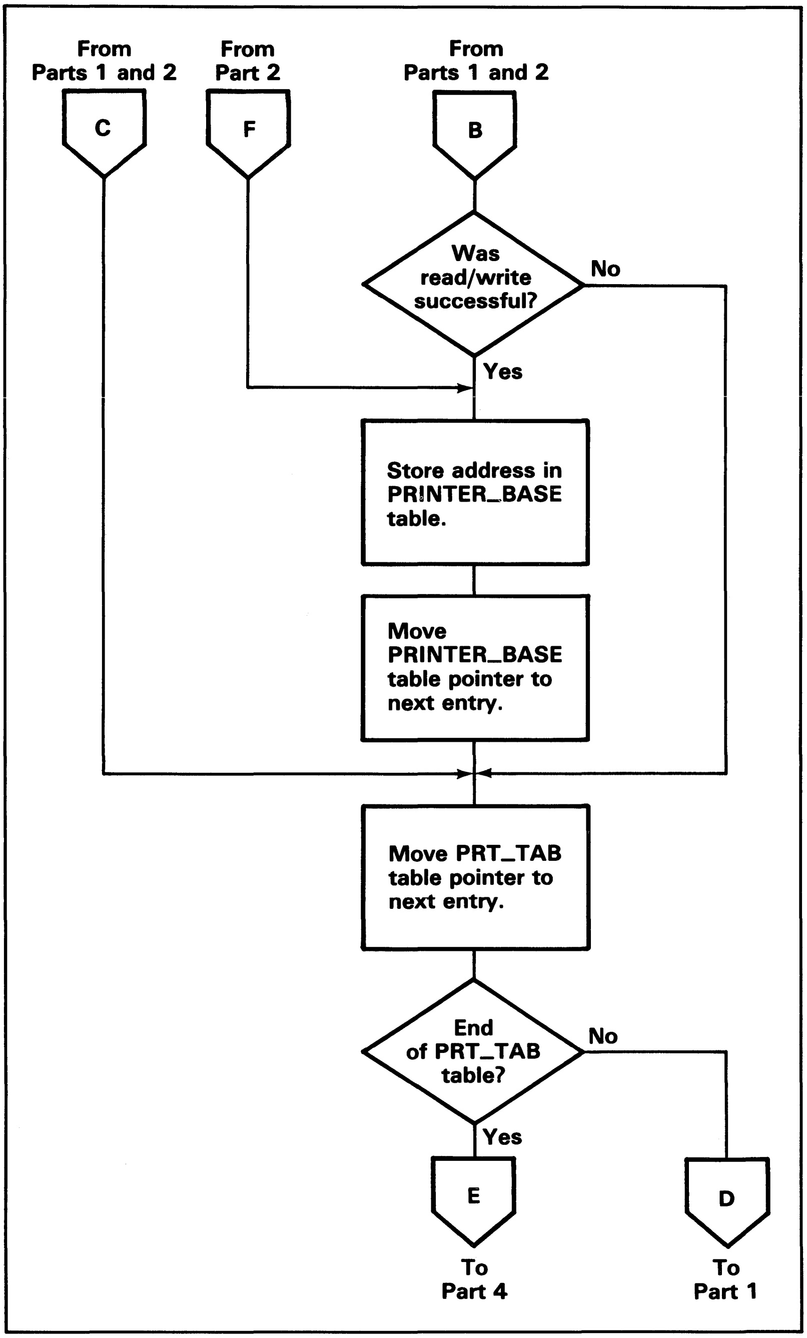

From Parts 1 and 2
From Part 2
From Parts 1 and 2
C
F
B
Was read/write successful?
No
Yes
Store address in PRINTER_BASE table.
Move PRINTER_BASE table pointer to next entry.
Move PRT_TAB table pointer to next entry.
End of PRT_TAB table?
No
Yes
E
D
To Part 4
To Part 1

Figure 1-3 (Part 3 of 7). Printer/Communications Presence Test

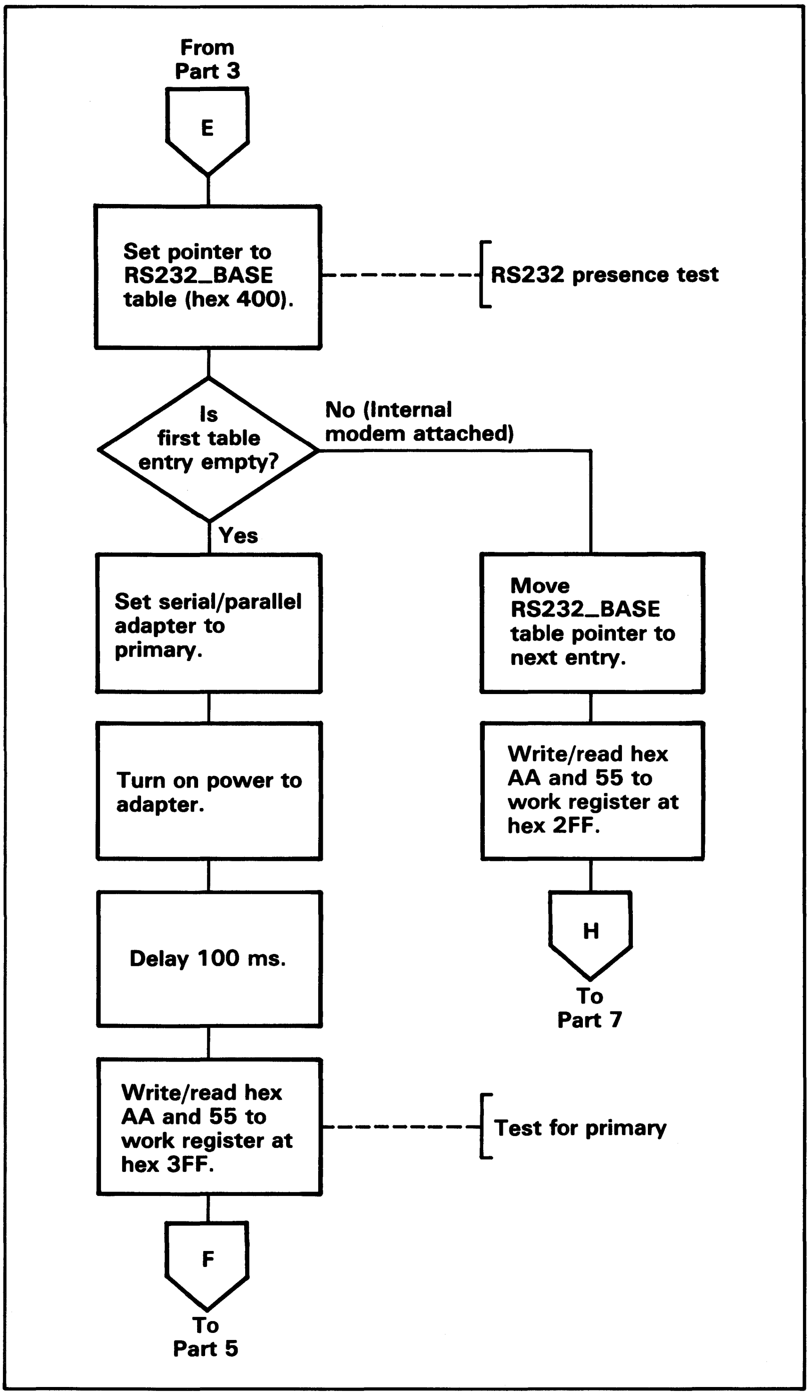

From Part 3
E
Set pointer to RS232_BASE table (hex 400).
RS232 presence test
Is first table entry empty?
No (Internal modem attached)
Yes
Set serial/parallel adapter to primary.
Turn on power to adapter.
Delay 100 ms.
Write/read hex AA and 55 to work register at hex 3FF.
Test for primary
F
To Part 5
Move RS232_BASE table pointer to next entry.
Write/read hex AA and 55 to work register at hex 2FF.
H
To Part 7

Figure 1-3 (Part 4 of 7). Printer/Communications Presence Test

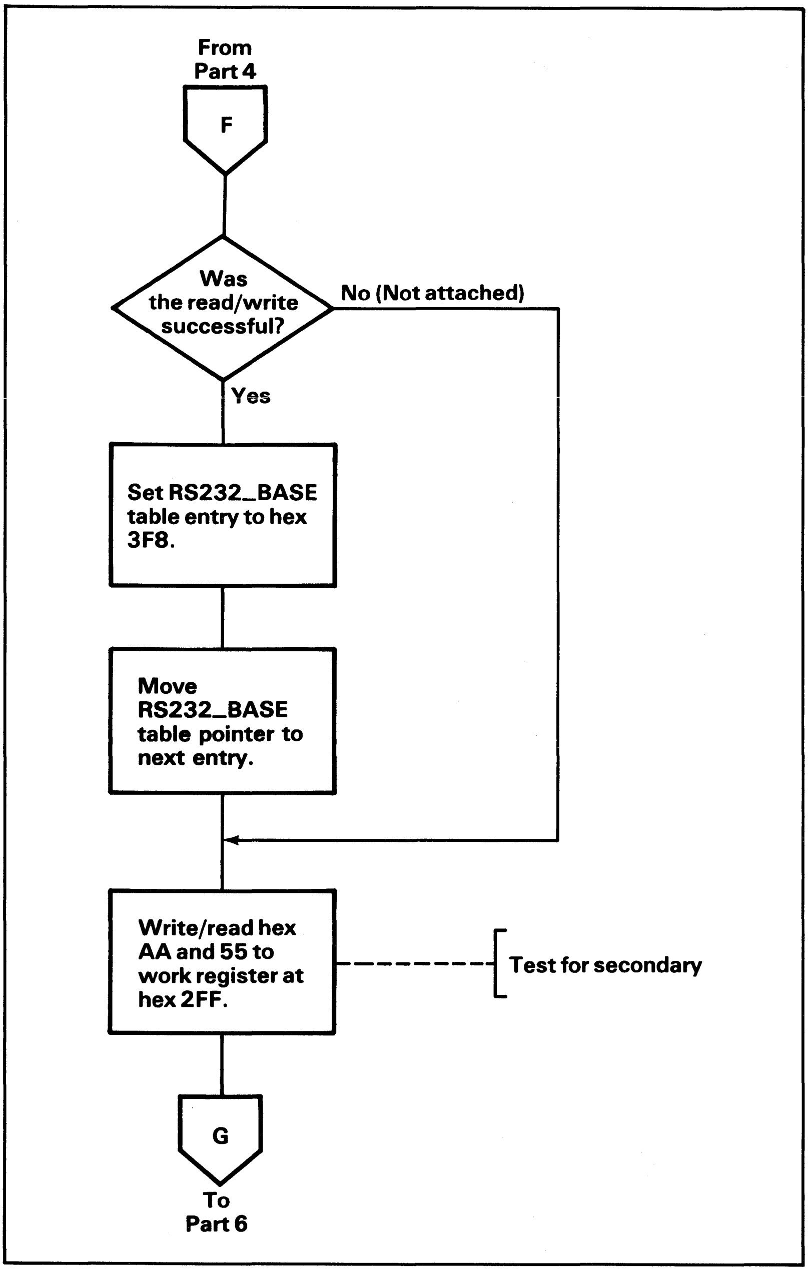


Figure 1-3 (Part 5 of 7). Printer/Communications Presence Test

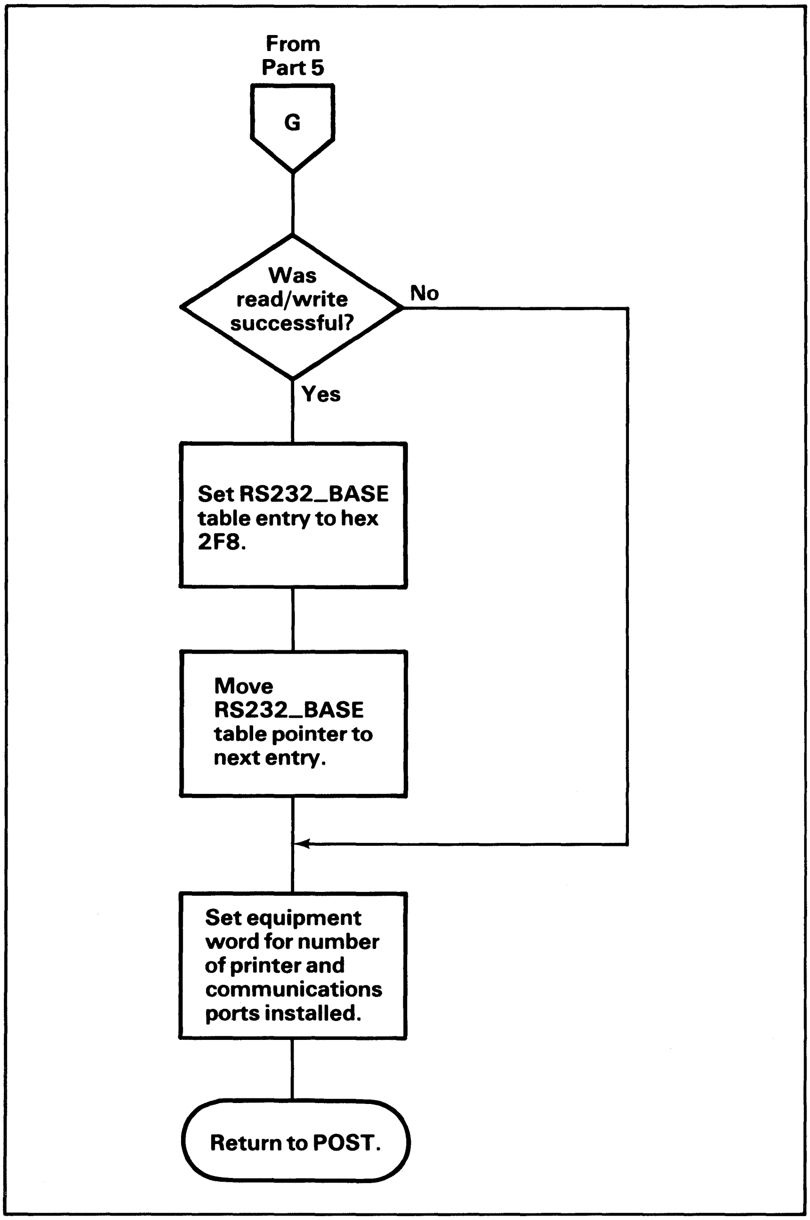

From Part 5
G
Was read/write successful?
No
Yes
Set RS232_BASE table entry to hex 2F8.
Move RS232_BASE table pointer to next entry.
Set equipment word for number of printer and communications ports installed.
Return to POST.

Figure 1-3 (Part 6 of 7). Printer/Communications Presence Test

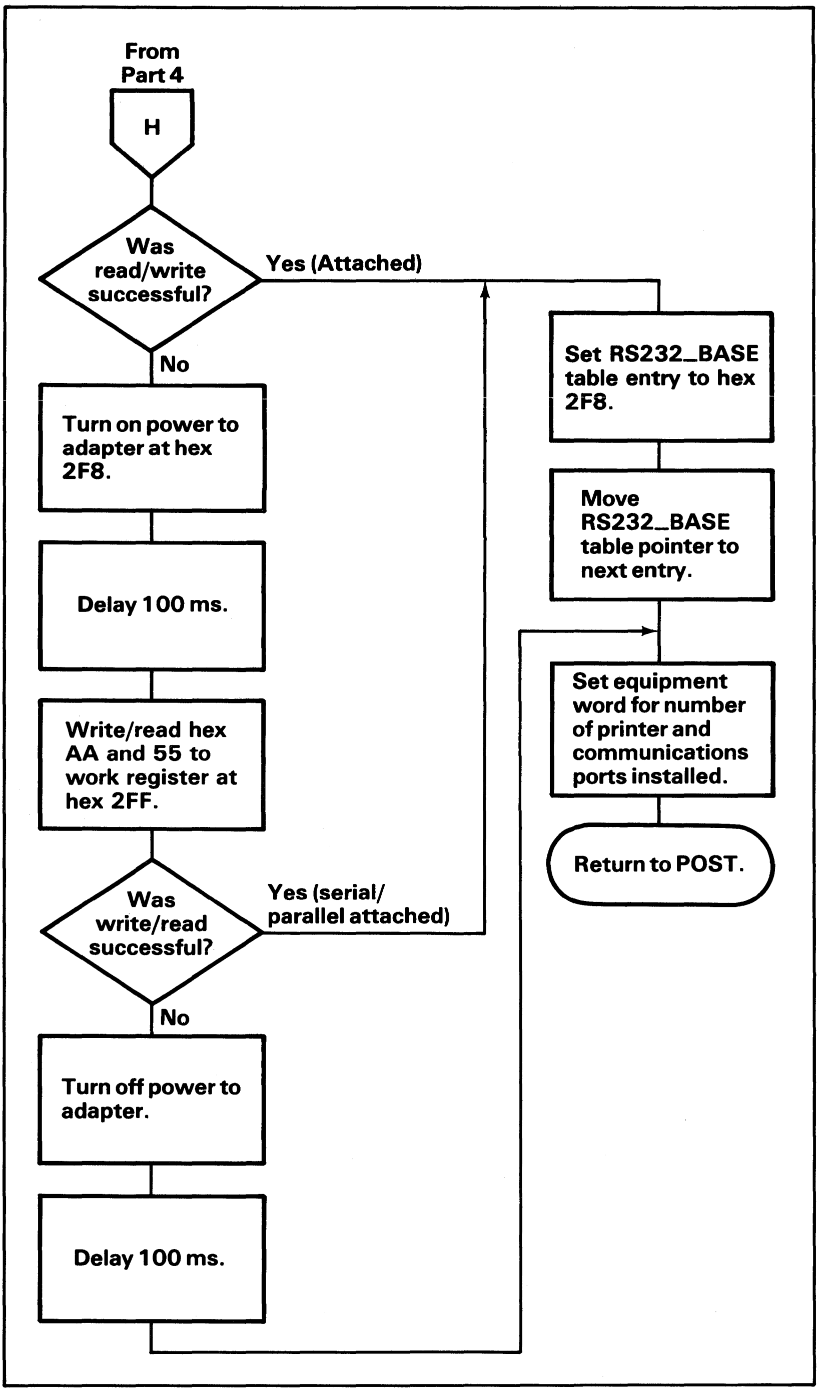

From Part 4
H
Was read/write successful?
Yes (Attached)
No
Turn on power to adapter at hex 2F8.
Delay 100 ms.
Write/read hex AA and 55 to work register at hex 2FF.
Was write/read successful?
Yes (serial/ parallel attached)
No
Turn off power to adapter.
Delay 100 ms.
Set RS232_BASE table entry to hex 2F8.
Move RS232_BASE table pointer to next entry.
Set equipment word for number of printer and communications ports installed.
Return to POST.

Figure 1-3 (Part 7 of 7). Printer/Communications Presence Test

Section 2. ROM BIOS Listing

ROM Map

```
Start  Stop   Length  Name
F0000H FFFFEH FFFFH  ROMCODE

 Address          Publics by Name          Address          Publics by Value

F000:435D         ACT_DISP_PAGE            F000:0008  Abs  K6L
F000:0C72         ADDR_TST                 F000:0010  Abs  M4
F000:E229         BAD_DSKT_ICON            F000:0014  Abs  KBFUNL
F000:E7E6         BAT_SAV_SETUP            F000:001E  Abs  KBPADL
F000:E729         BAUD_TABLE               F000:0020  Abs  M1L
F000:E75F         BEEP                     F000:002C       POSTMAIN
F000:E762         BEEP_SUB                 F000:0043       POST_LOOP
F000:E6F2         BOOT_STRAP               F000:03DB       POSTCNTL
F000:F859         CASSETTE_IO              F000:0A46       MEM_TEST
F000:EFE3         CGA_TBL                  F000:0B40       STG_TST
F000:E2C6         CHAR_GEN_HI              F000:0BDD       MEM_SIZE_CHK
F000:FA6E         CHAR_GEN_LO              F000:0C72       ADDR_TST
F000:2D30         CHECK_FOR_8250           F000:0CCF       CLR_STG
F000:0DEE         CHECK_INT                F000:0D0A       ROM_CHECKSUM
F000:0CCF         CLR_STG                  F000:0D15       ROM_SCAN
F000:2B61         CMPT_ADAPT_TEST          F000:0D5A       VALID_ROMCHK
F000:530A         COMMO_IO                 F000:0D6F       ROM_LINK
F000:53DF         COM_POWER                F000:0DEE       CHECK_INT
F000:2473         CRT_TEST                 F000:134C       TMR_TST
F000:FF23         D11                      F000:18AA       DMA_CHECK
F000:E85C         DDS                      F000:1B49       KYBD_TST
F000:E261         DEF_SYS_PROF             F000:1E9A       DSP_CONFIG
F000:E79B         DISABLE_NMI              F000:1F90       LCDBUF_TST
F000:EC59         DISKETTE_IO              F000:20EE       LCDCTL_TST
F000:4C95         DISK_RESET               F000:22E2       DSP_INIT
F000:18AA         DMA_CHECK                F000:2304       DSP_FSETM
F000:EFC7         DSKT_BASE                F000:2384       LCDINIT
F000:E214         DSKT_ICON                F000:2473       CRT_TEST
F000:EF57         DSKT_INT                 F000:2548       VIDEO_LINK
F000:4FB0         DSKT_INTE                F000:261F       DSKT_TST
F000:4C2B         DSKT_IO                  F000:2857       PTR_COM_PRESENCE
F000:261F         DSKT_TST                 F000:2966       MODEM_POWER_ON
F000:E6C6         DSP_BYTE                 F000:29EA       MODEM_CONFIG
F000:1E9A         DSP_CONFIG               F000:2B1A       SEND_C
F000:2304         DSP_FSETM                F000:2B26       RECV_S
F000:E6E7         DSP_HEX                  F000:2B4B       MODEM_INIT
F000:22E2         DSP_INIT                 F000:2B61       CMPT_ADAPT_TEST
F000:FF53         DUMMY_RETURN             F000:2D30       CHECK_FOR_8250
F000:E7A3         ENABLE_NMI               F000:2D70       RS232_TEST
F000:F84D         EQUIPMENT                F000:2DCB       MODEM_TEST
F000:E73C         ERR_BEEP                 F000:3174       SEND_COM
F000:5A99         EXT_EVENT                F000:3192       NMI_FLIH
F000:EC5C         E_MSG                    F000:38D9       GET_RTC_NMI
F000:E128         F1_ICON                  F000:38E2       PUT_RTC_NMI
F000:5D4F         FILL                     F000:3AC4       RESUME
F000:4E6C         GET_PARM                 F000:3D40       KYBD_IO
F000:38D9         GET_RTC_NMI              F000:3DA4       KYBD_INT9
F000:E7AA         GET_RTC_REG              F000:411D       VIDEO_IO_1
F000:E864         GET_VECTOR@              F000:415A       SET_MODE
F000:ED5F         ICON_PR                  F000:42AB       SET_CTYPE
F000:570C         INITIALIZE_STATUS        F000:4332       SET_CPOS
F000:E8E1         K10                      F000:435D       ACT_DISP_PAGE
F000:E91B         K11                      F000:4381       READ_CURSOR
F000:E955         K12                      F000:439C       SET_COLOR
F000:E95F         K13                      F000:43C2       VIDEO_STATE
F000:E969         K14                      F000:43E8       SCROLL_UP
F000:E976         K15                      F000:446B       SCROLL_DOWN
F000:EA87         K30                      F000:44A7       READ_AC_CURRENT
F000:E87E         K6                       F000:44ED       WRITE_AC_CURRENT
F000:0008   Abs   K6L                      F000:4521       WRITE_C_CURRENT
F000:E886         K7                       F000:4554       READ_DOT
```

F000:E88E		K8
F000:E8C8		K9
F000:0014	Abs	KBFUNL
F000:EAC9		KBFUN_TBL
F000:EADD		KBNMI_TBL
F000:001E	Abs	KBPADL
F000:EAAB		KBPAD_TBL
F000:E987		KB_INT
F000:E6F5		KB_NOISE
F000:E82E		KEYBOARD_IO
F000:3DA4		KYBD_INT9
F000:3D40		KYBD_IO
F000:E831		KYBD_RESET
F000:1B49		KYBD_TST
F000:1F90		LCDBUF_TST
F000:20EE		LCDCTL_TST
F000:2384		LCDINIT
F000:EFF1		LCD_CGA_TBL
F000:EFFF		LCD_MONO_TBL
F000:F045		M1
F000:0020	Abs	M1L
F000:0010	Abs	M4
F000:F0E4		M5
F000:F0EC		M6
F000:F0F4		M7
F000:F841		MEMORY_SIZE_DET
F000:E06B		MEM_MSG
F000:0BDD		MEM_SIZE_CHK
F000:0A46		MEM_TEST
F000:29EA		MODEM_CONFIG
F000:2B4B		MODEM_INIT
F000:2966		MODEM_POWER_ON
F000:2DCB		MODEM_TEST
F000:EFD5		MONO_TBL
F000:4E3F		NEC_OUTPUT
F000:3192		NMI_FLIH
F000:E2C3		NMI_INT
F000:EF62		PARMS_TPI135
F000:EF5A		PARMS_TPI48
F000:E18F		PAR_CHK
F000:03DB		POSTCNTL
F000:002C		POSTMAIN
F000:0043		POST_LOOP
F000:EFD2		PRINTER_IO
F000:FF54		PRINT_SCREEN
F000:547A		PRT_IO
F000:4B4F		PRT_SCRN
F000:E26D		PRT_TAB
F000:E275		PRT_TAB_END
F000:2857		PTR_COM_PRESENCE
F000:38E2		PUT_RTC_NMI
F000:E7C8		PUT_RTC_REG
F000:FFF0		P_O_R
F000:44A7		READ_AC_CURRENT
F000:4381		READ_CURSOR
F000:4554		READ_DOT
F000:48C4		READ_LPEN
F000:2B26		RECV_S
F000:E05B		RESET
F000:4FC6		RESULTS
F000:3AC4		RESUME
F000:EB70		RES_ERR_CHK
F000:6000		ROM_BASIC
F000:0D0A		ROM_CHECKSUM
F000:0D6F		ROM_LINK
F000:E072		ROM_MSG
F000:0D15		ROM_SCAN
F000:E739		RS232_IO
F000:2D70		RS232_TEST
F000:57BA		RTC2_TST
F000:E269		RTC_SIG_SAVE
F000:446B		SCROLL_DOWN
F000:43E8		SCROLL_UP
F000:4E7B		SEEK

F000:4565	WRITE_DOT
F000:4840	WRITE_TTY
F000:48C4	READ_LPEN
F000:4B4F	PRT_SCRN
F000:4C2B	DSKT_IO
F000:4C95	DISK_RESET
F000:4E3F	NEC_OUTPUT
F000:4E6C	GET_PARM
F000:4E7B	SEEK
F000:4FB0	DSKT_INTE
F000:4FC6	RESULTS
F000:51B6	SYS_BOOT
F000:530A	COMMO_IO
F000:53DF	COM_POWER
F000:547A	PRT_IO
F000:5527	TOD_PROC
F000:570C	INITIALIZE_STATUS
F000:5739	SET_TOD
F000:57BA	RTC2_TST
F000:5864	TMR0_INT8
F000:58F7	SYS_SERVICES
F000:5A99	EXT_EVENT
F000:5D4F	FILL
F000:6000	ROM_BASIC
F000:E05B	RESET
F000:E05B	START
F000:E06B	MEM_MSG
F000:E072	ROM_MSG
F000:E128	F1_ICON
F000:E18F	PAR_CHK
F000:E1BF	SYS_DSKT_ICON
F000:E214	DSKT_ICON
F000:E229	BAD_DSKT_ICON
F000:E257	SYS_DESCR_TABLE
F000:E261	DEF_SYS_PROF
F000:E269	RTC_SIG_SAVE
F000:E26D	PRT_TAB
F000:E275	PRT_TAB_END
F000:E275	STR_CON
F000:E2C3	NMI_INT
F000:E2C6	CHAR_GEN_HI
F000:E6C6	DSP_BYTE
F000:E6E0	XLAT_NIB
F000:E6E7	DSP_HEX
F000:E6F2	BOOT_STRAP
F000:E6F5	KB_NOISE
F000:E729	BAUD_TABLE
F000:E739	RS232_IO
F000:E73C	ERR_BEEP
F000:E75F	BEEP
F000:E762	BEEP_SUB
F000:E79B	DISABLE_NMI
F000:E7A3	ENABLE_NMI
F000:E7AA	GET_RTC_REG
F000:E7C8	PUT_RTC_REG
F000:E7E6	BAT_SAV_SETUP
F000:E82E	KEYBOARD_IO
F000:E831	KYBD_RESET
F000:E85C	DDS
F000:E864	GET_VECTOR@
F000:E87E	K6
F000:E886	K7
F000:E88E	K8
F000:E8C8	K9
F000:E8E1	K10
F000:E91B	K11
F000:E955	K12
F000:E95F	K13
F000:E969	K14
F000:E976	K15
F000:E987	KB_INT
F000:E98A	SYS_SETUP
F000:EA0C	VECTOR_SETUP
F000:EA87	K30

```
 F000:2B1A      SEND_C                      F000:EAAB      KBPAD_TBL
 F000:3174      SEND_COM                    F000:EAC9      KBFUN_TBL
 F000:439C      SET_COLOR                   F000:EADD      KBNMI_TBL
 F000:4332      SET_CPOS                    F000:EB43      SYS_CHK
 F000:42AB      SET_CTYPE                   F000:EB70      RES_ERR_CHK
 F000:415A      SET_MODE                    F000:EC59      DISKETTE_IO
 F000:5739      SET_TOD                     F000:EC5C      E_MSG
 F000:E05B      START                       F000:ED5F      ICON_PR
 F000:0B40      STG_TST                     F000:EF57      DSKT_INT
 F000:E275      STR_CON                     F000:EF5A      PARMS_TPI48
 F000:51B6      SYS_BOOT                    F000:EF62      PARMS_TPI135
 F000:EB43      SYS_CHK                     F000:EFC7      DSKT_BASE
 F000:E257      SYS_DESCR_TABLE             F000:EFD2      PRINTER_IO
 F000:E1BF      SYS_DSKT_ICON               F000:EFD5      MONO_TBL
 F000:58F7      SYS_SERVICES                F000:EFE3      CGA_TBL
 F000:E98A      SYS_SETUP                   F000:EFF1      LCD_CGA_TBL
 F000:FEA5      TIMER_INT                   F000:EFFF      LCD_MONO_TBL
 F000:FE6E      TIME_OF_DAY                 F000:F045      M1
 F000:5864      TMR0_INT8                   F000:F065      VIDEO_IO
 F000:134C      TMR_TST                     F000:F0A4      VIDEO_PARMS
 F000:5527      TOD_PROC                    F000:F0E4      M5
 F000:0D5A      VALID_ROMCHK                F000:F0EC      M6
 F000:EA0C      VECTOR_SETUP                F000:F0F4      M7
 F000:FEF3      VECTOR_TABLE                F000:F841      MEMORY_SIZE_DET
 F000:F065      VIDEO_IO                    F000:F84D      EQUIPMENT
 F000:411D      VIDEO_IO_1                  F000:F859      CASSETTE_IO
 F000:2548      VIDEO_LINK                  F000:FA6E      CHAR_GEN_LO
 F000:F0A4      VIDEO_PARMS                 F000:FE6E      TIME_OF_DAY
 F000:43C2      VIDEO_STATE                 F000:FEA5      TIMER_INT
 F000:44ED      WRITE_AC_CURRENT            F000:FEF3      VECTOR_TABLE
 F000:4521      WRITE_C_CURRENT             F000:FF23      D11
 F000:4565      WRITE_DOT                   F000:FF53      DUMMY_RETURN
 F000:4840      WRITE_TTY                   F000:FF54      PRINT_SCREEN
 F000:E6E0      XLAT_NIB                    F000:FFF0      P_O_R

Program entry point at F000:E05B
```

Common Equates and Data Areas

```
                      TITLE   COMMON EQUATES/ DATA AREAS
                              INCLUDE SROMEQUS.INC
                      ; DATE LAST MODIFIED: 09/13/1985
                      ;
                      ;*******************************************************
                      ;  G L O B A L   E Q U A T E S   F O R   I / O   P O R T S
                      ;*******************************************************
```

Common Equates

```
                      ;--------------------------------------------------
                      ; D M A   C O N T R O L L E R   P O R T   00 -
                      ;--------------------------------------------------
= 0000                        DMA        EQU     0          ; DMA CONTROLLER BASE ADDRESS
                      ;--------------------------------------------------
                      ; I N T E R R U P T   C O N T R O L L E R  - 20,21H
                      ;--------------------------------------------------
= 0020                        INTA00     EQU     20H        ; INTERRUPT CONTROLLER PORT 1
= 0020                        EOI        EQU     20H        ; NON SPECIFIC END OF INTERRUPT
= 000B                        READ_ISR   EQU     0BH        ; READ IN SERV REG AT INTA00
= 000A                        READ_IRR   EQU     0AH        ; READ INT REQ REG AT INTA00

= 0021                        INTA01     EQU     21H        ; IRPT CONTROLLER MASK REG

                      ;-----------------------------------------
                      ; S Y S T E M   T I M E R S    - 40,42,43H
                      ;-----------------------------------------
= 0040                        TIMER0     EQU     40H        ; TIMER 0 COUNTER PORT ADDR
= 0042                        TIMER2     EQU     42H        ; TIMER 2 COUNTER PORT ADDR

= 0043                        TIMER_CTL  EQU     43H        ; TIMER 0,2 CONTROL PORT ADDR
= 0080                        TIMER2_CTL EQU     80H        ; ACCESS TIMER2 CONTROLS
= 0030                        TLATCH     EQU     30H        ; LATCH TMR VAL R/W LEAST, 1ST

                      ;-------------------------------------------------
                      ; K E Y B O A R D   S C A N   C O D E   P O R T - 60H
                      ;-------------------------------------------------
= 0060                        KB_DATA    EQU     60H        ; MAIN KEYBOARD SCAN CODE PORT

                      ;----------------------------------------
                      ; N M I   C O N T R O L   P O R T    - 61H
                      ;----------------------------------------
= 0061                        NMI_CNTL   EQU     61H        ;NMI CONTROL PORT
= 0061                        KB_CTL     EQU     61H        ;PORT 60H CONTROL REGISTER
= 0080                        CLR_KEYBD  EQU     80H        ;CLEAR KEYBOARD SIGNAL
= 0040                        DIS_COPROC EQU     40H        ;DISABLE COPROCESSOR NMI
= 0020                        DIS_IOCHK  EQU     20H        ;DISABLE I/O CHANNEL CHECK NMI
= 0008                        DIS_ALARM  EQU     08H        ;DISABLE RTC ALARM NMI
= 0004                        EN_SPKR    EQU     04H        ;ENABLE SPEAKER    (NOT AN NMI)
= 0002                        SPKR_DATA  EQU     02H        ;SPEAKER DATA LINE (NOT AN NMI)
= 0001                        TMR2_GATE  EQU     01H        ;GATE TIMER 2      (NOT AN NMI)
```

```
                    ;-----------------------------------------
                    ; N M I  S O U R C E  R E G        - 62H
                    ;-----------------------------------------
= 0062                      NMI_SRC         EQU     62H    ; NMI SOURCE PORT ADDRESS
= 0080                      DSKT_NMI        EQU     80H    ; DISKETTE NMI
= 0040                      IOCHK_NMI       EQU     40H    ; I/O CHANNEL CHECK NMI
= 0020                      TIMER2_SN       EQU     20H    ; TMR 2 SNE (NOT AN NMI SOURCE)
= 0010                      KBCLR_NMI       EQU     10H    ; KYBD PORT 60 AVAILABLE NMI
= 0008                      SYS_SUSP_NMI    EQU     08H    ; SYSTEM SUSPEND NMI
= 0004                      RTC_ALRM_NMI    EQU     04H    ; REAL TIME CLOCK ALARM NMI
= 0001                      KBDATA_NMI      EQU     01H    ; KEYBOARD DATA READY NMI

                    ;-----------------------------------------
                    ; SYSTEM CLOCK CONTROL REGISTER    - 72H
                    ;-----------------------------------------
= 0072                      CLOCK_CTL       EQU     72H    ; SLEEP CLOCK CONTROL REGISTER
= 0020                      GLOBAL_NMI      EQU     20H    ; GLOBAL NMI ENABLE
= 0004                      DISABLE_SLEEP   EQU     04H    ; DISABLE SLEEP CLOCK BIT
= 0003                      CLOCK_RUN       EQU     03H    ; SLEEP CLOCK RUN STATE
= 0000                      CLOCK_STOP      EQU     0      ; CLOCK STOP VALUE

                    ;------------------------------------------------
                    ; L C D  C O N T R O L L E R  I N D E X     - 74H
                    ;------------------------------------------------
= 0074                      LCD_INDX        EQU     74H    ; INDEX REG FOR ACCESING LCDC
                                                           ; REGS
= 0000                      LCD_FUNCT       EQU     00     ; LCDC FUNCT CONTROL REGISTER
= 0001                      LCD_WRAP        EQU     01     ; LCDC DIAG CONTROL REGISTER

                    ;------------------------------------------------
                    ; L C D  C O N T R O L L E R  D A T A       - 75H
                    ;------------------------------------------------
= 0075                      LCD_DATA        EQU     75H    ; DATA REG FOR WRITING TO LCDC
                                                           ; REGS

= 0001                      LCD_2PAN        EQU     01H    ; ON = TWO PANEL LCD
                    ;                                      ; OFF = ONE PANEL LCD
= 0002                      LCD_CGA         EQU     02H    ; ON = LCD EMULATING CGA
                    ;                                      ; OFF = LCD EMULATING MONO
= 0004                      LCD_NORM        EQU     04H    ; ON = LCD IN NORMAL MODE
                    ;                                      ; OFF = LCD IN DIAGNOSTIC MODE
= 0008                      LCD_ENAB        EQU     08H    ; ON = LCD ADDR DECODE ENABLED
                    ;                                      ; OFF = LCD DISABLED
= 0010                      LCD_FONT        EQU     10H    ; ON = ACCESS LCD FONT STORAGE
                    ;                                      ; OFF = ACCESS REGEN BUFFER
= 0020                      SYNC_ENABLE     EQU     20H    ; ON = ENABLE SYNCS
                    ;                                      ; OFF = DISABLE SYNCS
= 0040                      PANEL_ENABLE    EQU     40H    ; ON = LCD PANEL POWER ENABLED
                    ;                                      ; OFF = LCD PANEL PWR DISABLE
= 0080                      LCD_SENSP       EQU     80H    ; ON = TWO PANEL LCD SENSED BY
                                                           ; HRDWARE

                    ;---------------------------------------------------
                    ; DISKETTE CONTROL  PORT
                    ;---------------------------------------------------
= 0077                      DSKT_CNTL       EQU     77H    ; DSKT POWER CONTROL REGISTER
= 0080                      DSKT_NMI        EQU     80H    ; ENABLE DSKT NMI POWER ON REQ

= 0040                      FDC_PWR         EQU     40H    ; DSKT CONTROLLER POWER ENABLE

= 0020                      DSKT_DEGATE     EQU     20H    ; DSKT DRIVE DEGATE FROM CNTLR

= 0010                      RD_CNTL         EQU     10H    ; READ CONTROLLER REGISTER
                    ;* THIS BIT OFF ALLOWS READING THE DRIVE TRK POSTN FROM DSKT_CNTL REG

= 0008                      DRO_TRK_SEL     EQU     08H    ; READ DRIVE 0 TRACK POSITION
                    ;* THIS BIT OFF ALLOWS READING THE DRIVE 1 TRACK POSITION

= 0002                      CNTL_SEL        EQU     02H    ; BIT MUST ALWAYS BE ON FOR
                                                           ; ANY WRITE TO THIS I/O PORT
                    ;------------------------------------------------
                    ; P O R T A B L E   P R I N T E R   P O R T S
                    ;------------------------------------------------
 = 0078                     CPRT_DATA        EQU     78H    ; PORTABLE PTR XMIT PORT ADDR

 = 0079                     CPRT_STAT        EQU     79H    ; PORTABLE PTR STAT PORT ADDR
 = 0080                     NOT_BUSY         EQU     80H    ; NOT BUSY BIT IN STATUS PORT
 = 0008                     NOT_ERROR        EQU     08H    ; NOT ERROR BIT IN STATUS PORT

 = 007A                     CPRT_MODE        EQU     7AH    ; PORTABLE PTR MODE PORT ADDR
 = 0040                     ACK              EQU     40H    ; ACKNOWLEDGE BIT (NOT USED)
 = 0008                     SELECT           EQU     08H    ; SLCT BIT IN MD PRT (NOT USED)
 = 0004                     NO_INIT          EQU     04H    ; INITIALIZE BIT IN MODE PORT
 = 0001                     STROBE           EQU     01H    ; STRB BIT IN MD PRT (NOT USED)

                    ;-----------------------------------------------
                    ; K E Y B O A R D  &  F E A T U R E  C N T L - 7CH
                    ;-----------------------------------------------
 = 007C                     KYBD_CNTL        EQU     7CH    ; KYBD AND FEAT CONTROL ADDR
 = 0080                     EN_KYBD_NMI      EQU     80H    ; ENABLE KEYBOARD NMI
 = 0040                     KYBD_ATTACH      EQU     40H    ; KEYBOARD ATTACHED SENSE
 = 0020                     SET_KYBD_DIAG    EQU     20H    ; ACT KYBD DIAGNOSTIC MODE
 = 0004                     ACT_RS232        EQU     04H    ; ACT RS232 FEATURE POWER
 = 0002                     ACT_MODEM        EQU     02H    ; ACTIVATE MODEM FEATURE POWER
 = 0001                     SET_RS232_PRIM   EQU     01H    ; SET RS232 FEAT TO PRI ADDR.

                    ;------------------------------------------
                    ; P O W E R   I N T E R F A C E   R E G - 7FH
                    ;------------------------------------------

 = 007F                     PWR_STAT         EQU     7FH    ; POWER INTERFACE PORT ADDRESS
 = 0080                     LOW_BAT          EQU     80H    ; LOW BATTERY STATUS FLAG
 = 0040                     EXT_PWR          EQU     40H    ; EXTERNAL POWER SUPPLIED
 = 0020                     SYS_POR          EQU     20H    ; SYSTEM POR REQSTD (ALT CTL R)
 = 0010                     PON_ALRM         EQU     10H    ; POWER ACTIVATED BY RTC ALARM
 = 0008                     HDWR_RESET       EQU     08H    ; CAUSE POWER-ON-RESET
 = 0004                     EN_SUS_NMI       EQU     04H    ; ENABLE SYSTEM SUSPEND NMI
 = 0002                     REQ_POFF         EQU     02H    ; REQUEST SYSTEM POWER OFF
 = 0001                     EN_PON_ALRM      EQU     01H    ; ENABLE POWER ON BY RTC ALARM

                    ;------------------------------------------
                    ; D M A   P A G E   R E G I S T E R S
                    ;------------------------------------------
 = 0083                     DMA_PAGE1        EQU     83H    ; DMA PAGE REGISTER CHANNEL 1
 = 0081                     DMA_PAGE2        EQU     81H    ; DMA PAGE REGISTER CHNL 2 (DSKT)
 = 0082                     DMA_PAGE3        EQU     82H    ; DMA PAGE REGISTER CHANNEL 3

                    ;------------------------------------------
                    ; M A N U F A C T U R I N G   P O R T S
                    ;------------------------------------------
 = 00A1                     MFG_CHKPT        EQU     0A1H   ; MFG CHECKPOINT PORT
 = 00A2                     MFG_ERR_HI       EQU     0A2H   ; MFG ERROR CODE PORT HIGH
 = 00A3                     MFG_ERR_LO       EQU     0A3H   ; MFG ERROR CODE PORT LOW
 = 0080                     MEM_CTL          EQU     80H    ; MEMORY DECODE CONTROL PORT
 = 0001                     MEM_SUB_MODE     EQU     01H    ; BIT SET IN MEM_CTL FOR SPECIAL
                                                            ; MEM SUBSTITUTE OF LCD DISPLAY
                                                            ; RAM FOR MAIN MEM FOR MFG MODE

                    ;--------------------------------------------
                    ; I / O  C H K  - 8 0 8 7  C O N T R O L - A0H
                    ;--------------------------------------------
 = 00A0                     IONMI_CNTL       EQU     0A0H   ; I/O CHAN CHK NMI ENAB/DISAB
 = 0080                     EN_IOCHK         EQU     80H    ; ENABLE 8087 AND IO CHECK NMI
 = 0007                     INT_LEVEL        EQU     07H    ; MASK FOR CURRENT IRPT LVL SNE
```

```
                  ;-------------------------------------------------
                  ; D I S P L A Y   A D A P T E R   C O N T R O L   R E G S
                  ;-------------------------------------------------
= 03D8                    CGA_CNTL        EQU     03D8H  ; COLOR GRAPHICS CONTROL PORT
= 03B8                    MONO_CNTL       EQU     03B8H  ; MONO DISPLAY CONTROL PORT

                  ;-------------------------------------------------
                  ; F L O P P Y   D I S K E T T E   C O N T R O L   R E G S
                  ;-------------------------------------------------
                  ;
                  ; DRIVE MOTOR, SELECT CONTROL PORT
                  ;
= 03F2                    DRIVE_CNTL      EQU     03F2H  ; FLOPPY DISKETTE DRIVE CONTROL
                                                         ; *****WRITE ONLY REGISTER*****
= 0020                    DR1_MOTOR       EQU     20H    ; DRIVE 1 MOTOR ENABLE
= 0010                    DR0_MOTOR       EQU     10H    ; DRIVE 0 MOTOR ENABLE
= 0008                    FDC_DMA_ENAB    EQU     08H    ; ENABLE FDC DMA AND INTERRUPTS
= 0004                    FDC_RUN         EQU     04H    ; DISKETTE CONTROLLER RUN BIT
                  ; IF THE ABOVE BIT IS OFF THE DISKETTE CONTROLLER IS RESET

= 0001                    DR1_SELECT      EQU     01H    ; DRIVE 1 SELECT
= 0000                    DR0_SELECT      EQU     00H    ; DRIVE 0 SELECT
                  ;
                  ; DISKETTE CONTROLLER MAIN STATUS REGISTER
                  ;
= 03F4                    FDC_STATUS      EQU     03F4H  ; DSKT CONTROLLER MASTER STATUS
                                                         ; *** READ ONLY REGISTER *****
= 0080                    REQ_MASTER      EQU     080H   ; REQUEST FOR MASTER
= 0040                    DATA_READY      EQU     040H   ; DATA RDY TO BE RD FROM CNTLR
= 0010                    FDC_BUSY        EQU     010H   ; CONTROLLER IS BUSY

                  ;
                  ; DISKETTE CONTROLLER DATA INPUT/OUTPUT REGISTER
                  ;
= 03F5                    FDC_DATA        EQU     03F5H  ; DISKETTE CONTROLLER DATA PORT

                  ;
                  ; DRIVE MOTOR, SELECT TRACK 0 SENSE PORT
                  ;
= 03F7                    DRIVE_SENSE     EQU     03F7H  ; DSKT DRIVE CONTROL LINE SENSE
                                                         ; ***** READ ONLY REGISTER *****
= 0080                    CHG_LINE        EQU     080H   ; MEDIA CHNG LINE ACTIVE SENSE
= 0040                    DR0_SEL_SENSE   EQU     040H   ; DRIVE 0 SELECTED SENSE BACK
= 0020                    DR1_SEL_SENSE   EQU     020H   ; DRIVE 1 SELECTED SENSE BACK
= 0010                    DR0_MOT_SENSE   EQU     010H   ; DRIVE 0 MOTOR ON SENSE BACK
= 0008                    DR1_MOT_SENSE   EQU     008H   ; DRIVE 1 MOTOR ON SENSE BACK
= 0001                    TRACK0_SENSE    EQU     001H   ; TRACK 0 IND FOR SLCTD DRIVE

                  ;
                  ;  COMMANDS SENT TO DISKETTE CONTROLLER BY BIOS
                  ;
= 0008                    READ_INT_STATUS EQU     08H    ; READ INTERRUPT STATUS COMMAND
= 0003                    SPECIFY         EQU     03H    ; SPECIFY COMMAND
= 0007                    RECALIBRATE     EQU     07H    ; RECALIBRATE DRIVE COMMAND
= 000F                    SEEK_CMD        EQU     0FH    ; SEEK DRIVE COMMAND
= 00E6                    READ_CMND       EQU     0E6H   ; READ COMMAND
= 00C5                    WRITE_CMND      EQU     0C5H   ; WRITE COMMAND
= 004D                    FORMAT_CMND     EQU     04DH   ; FORMAT COMMAND
                  ;
                  ; DMA SETUP COMMANDS FOR DISKETTE
                  ;
= 0046                    DMA_READ        EQU     46H    ; SETUP DMA FOR DISKETTE READ
= 004A                    DMA_WRITE       EQU     4AH    ; SETUP DMA FOR DISKETTE WRITE
= 0042                    DMA_VERIFY      EQU     42H    ; SETUP DMA FOR NO XFER (VRFY)
```

```
                 ;*********************************************************************
                 ; G L O B A L   E Q U A T E S   F O R   R E A L   T I M E   C L O C K
                 ;*********************************************************************
                 ;
                 ; --- RTC REGISTER INDEX PORT
                 ;
= 0070                  RTCR_PORT       EQU     070H    ; REAL TIME CLOCK IX REG PORT
                 ;
                 ; --- RTC REGISTER DATA PORT
                 ;
= 0071                  RTCD_PORT       EQU     071H    ; REAL TIME CLK DATA REG PORT
                 ;
                 ; --- RTC TIME,DATE, AND ALARM REGISTERS
                 ;
= 0000                  RTC_TSEC        EQU     0       ; TIME SECONDS REG ADDR
= 0001                  RTC_ASEC        EQU     1       ; ALARM SECONDS REG ADDR
= 0002                  RTC_TMIN        EQU     2       ; TIME MINUTES REG ADDR
= 0003                  RTC_AMIN        EQU     3       ; ALARM MINUTES REG ADDR
= 0004                  RTC_THRS        EQU     4       ; TIME HOURS REG ADDR
= 0005                  RTC_AHRS        EQU     5       ; ALARM HOURS REG ADDR
= 0006                  RTC_WDAY        EQU     6       ; DAY OF WEEK (SUNDAY = 1)
= 0007                  RTC_MDAY        EQU     7       ; DAY OF MONTH
= 0008                  RTC_MON         EQU     8       ; MONTH
= 0009                  RTC_YEAR        EQU     9       ; YEAR
                 ;
                 ; --- RTC STATUS REGISTER
                 ;
= 000A                  RTC_UP_STAT     EQU     10      ; CLOCK UPDATE STATUS REG ADDR
= 0080                  RTC_UIP         EQU     080H    ; UPDATE IN PROGRESS BIT
                 ;
                 ; --- RTC INTERRUPT AND MODE SET REGISTER
                 ;
= 000B                  RTC_MODE        EQU     11      ; MODE REG ADDR
= 0080                  SET_CLOCK       EQU     080H    ; SET CLOCK BIT
= 0040                  PIE_ENABLE      EQU     040H    ; PERIODIC INTERRUPT ENABLE
= 0020                  AIE_ENABLE      EQU     020H    ; ALARM INTERRUPT ENABLE
= 0010                  UIE_ENABLE      EQU     010H    ; UPDATE ENDED IRPT ENABLE
= 0004                  SET_BIN         EQU     004H    ; SET BINARY MODE
= 0002                  SET_24HR        EQU     002H    ; SET 24 HOUR MODE
= 0001                  SET_DAYLIGHT    EQU     001H    ; SET DAYLIGHT SAVINGS MODE
                 ;
                 ; --- RTC INTERRUPT STATUS REGISTER
                 ;
= 000C                  RTC_INT_STAT    EQU     12      ; INTERRUPT STATUS REG ADDR
= 0080                  RTC_IRQ         EQU     080H    ; INTERRUPT REQUEST SET
= 0040                  PI_INT          EQU     040H    ; PERIODIC  REQUEST
= 0020                  AL_INT          EQU     020H    ; ALARM REQUEST
= 0010                  UE_INT          EQU     010H    ; UPDATE ENDED REQUEST
                 ;
                 ; --- RTC CONDITION STATUS REGISTER
                 ;
= 000D                  RTC_COND_STAT   EQU     13      ; RTC CONDITION STATUS
= 0080                  VALID_TIME      EQU     080H    ; RTC HAS NOT LOST POWER
                 ;
                 ;--- RTC DIAG_STATUS FLAGS ---
                 ;
= 000E                  RTC_DIAG_STAT   EQU     14      ; DIAG STATUS BYTE IN RTC RAM
= 0080                  RTC_TIME_BAD    EQU     080H    ; STANDBY POWER LOST FLAG
= 0040                  BAD_RTC_SIG     EQU     040H    ; REAL TIME CLK SIGNATURE BAD
= 0020                  BAD_STOR_CKSUM  EQU     020H    ; BASE 128K STG CHECKSUM BAD
= 0010                  LCD_ALT_FAILED  EQU     010H    ; LCD ALT DISPLAY MODE FAILED
= 0008                  LCD_CHANGE      EQU     08H     ; LCD CONFIGURATION CHANGED
= 0004                  RTC_FAILED      EQU     04H     ; REAL TIME CLK DID NOT UPDATE
                                                        ; OR VALUES OUT OF LIMITS
= 0002                  LCD_NOT_ACTIVE  EQU     02H     ; LCD NOT ACT WHEN SUSPENDED
= 0001                  DSKT_ACTIVE     EQU     01H     ; DSKT WAS ACT AT SUSPEND TIME
```

```
                        ;
                        ;---- BEGINNING OF RTC CMOS DATA AREA ----
                        ; REGISTERS FROM HERE TO RTC_MEM_END ARE KEPT CHECKSUMMED
                        ;
 = 000F                         RTC_MEM_START   EQU     15      ; START OF RTC REGISTER STACK


                        ;
                        ; --- DISKETTE TYPE INDENTIFIER --- UPPER 4 BITS = DRV 0, LOWER = DRV 1
                        ;
 = 0010                         RTC_DSKT_CON    EQU     16      ; DSKT CONFIG INFO (0 & 1)
 = 0011                         RTC_DSKT_CON2   EQU     17      ; DSKT CONFIG INFO (2 & 3)
 = 0000                         NO_DRIVE        EQU     0
 = 0001                         TPI_48          EQU     1       ; 48 TRACK PER INCH DRIVE
 = 0002                         TPI_96          EQU     2       ; 96 TRACK PER INCH DRIVE
 = 0003                         TPI_135         EQU     3       ; 135 TRACK PER INCH DRIVE
                        ;
                        ; --- SYSTEM EQUIPMENT WORD (COPY OF EQUIPMENT_WORD IN DATA SEGMENT)
                        ;
 = 0013                         RTC_EQUIP_LO    EQU     19      ; SYSTEM EQUIPMENT FLAG (LOW)
 = 0014                         RTC_EQUIP_HI    EQU     20      ; SYSTEM EQUIPMENT FLAG (HI)
                        ;  --------------------------------------------------
                        ;  - 15-14 - 13 - 12 - 11-9 - 8 - 7-6 - 5-4 - 3-1 - 0 -
                        ;  --------------------------------------------------
                        ;      |     |    |     |    |    |     |     |    |
                        ;      |     |    |     |    |    |     |     |    IPL DSKT INSTALLED
                        ;      |     |    |     |    |    |     |     -- UNUSED
                        ;      |     |    |     |    |    |     -- INITIAL VIDEO MODE
                        ;      |     |    |     |    |    -- NUMBER OF DISKETTE DRIVES
                        ;      |     |    |     |    -- UNUSED
                        ;      |     |    |     -- NUMBER OF COM DEVICES
                        ;      |     |    -- JOYSTICK ATTACHED
                        ;      |     -- INTERNAL MODEM INSTALLED
                        ;     ----  NUMBER OF PRINTERS ATTACHED
                        ;
                        ; --- SYSTEM MEMORY SIZE
                        ;
 = 0015                         RTC_MEMS_LO     EQU     21      ; IO MEMORY SIZE LO BYTE (POST)
 = 0016                         RTC_MEMS_HI     EQU     22      ; IO MEMORY SIZE HI BYTE (POST)
                        ;
                        ; --- SYSTEM PROFILE INFORMATION
                        ;          SYS_PROF1                           SYS_PROF2
                        ;  ------------------------------------      -------------
                        ;  | 15 | 14 | 13-12 | 11-10 |  9  | 8 |      |  7 - 0    |
                        ;  ------------------------------------      -------------
                        ;     |    |     |       |      |    |              |
                        ;     |    |     |       |      |    |              -- UNUSED
                        ;     |    |     |       |      |    |
                        ;     |    |     |       |      |    |
                        ;     |    |     |       |      |    -- RS232/PARL AVAIL ON BATTERY
                        ;     |    |     |       |      -- MODEM AVAILABLE ON BATTERY
                        ;     |    |     |       -- LCD HIGH INTENSITY MAPPING
                        ;     |    |     - INITIAL VIDEO MODE
                        ;     |    -- ENABLE LOW BATTERY WARNING
                        ;    ----  SYSTEM WARM START SELECTED
                        ;
 = 0017                         RTC_SYS_PROF1   EQU     23      ; SYSTEM PROFILE BYTE 1
 = 0080                         RESUME_ENABLE   EQU     80H     ; SYSTEM RESUME ENABLE FLAG
 = 0040                         LOWBAT_ENABLE   EQU     40H     ; LOW BAT WARNING ENABLE FLAG
 = 0002                         MODEM_BATT      EQU     02      ; MODEM AVAILABLE ON BATTERY
 = 0001                         RS232_BATT      EQU     01      ; RS232/PARALLEL  AVAILABLE ON
                                                                ; BATTERY
 = 0018                         RTC_SYS_PROF2   EQU     24      ; SYSTEM PROFILE BYTE 2
 = 0019                         RTC_LCD_INACT   EQU     25      ; INACT TIME TO DISPLAY BLANK
                                                                ; (2 BYTES)
 = 001B                         RTC_SYS_INACT   EQU     27      ; INACT TIME TO SYS POWER OFF
                                                                ; (2 BYTES)
```

```
                         ;
                         ; --- MODEM  PROFILE INFORMATION
                         ;         MOD_PROF1                      MOD_PROF2
                         ;
                         ;  -------------------------           -------------
                         ;  | 15-14 | 13 |12-10 |9-8 |          |  7 - 0    |
                         ;  -------------------------           -------------
                         ;     |      |     |     |                  |
                         ;     |      |     |     |                  -- UNUSED
                         ;     |      |     |     |
                         ;     |      |     |     |
                         ;     |      |     |     |
                         ;     |      |     |     |
                         ;     |      |     |     -- DATA RATE
                         ;     |      |     --- PARITY / FRAMING
                         ;     |      ---- 0 = MANUAL ANSWER, 1 = AUTO ANSWER
                         ;     ----  RESERVED
= 001D                         RTC_MOD_PROF1   EQU     29      ; MODEM SETUP PROFILE BYTE 1
= 001E                         RTC_MOD_PROF2   EQU     30      ; MODEM SETUP PROFILE BYTE 2

                         ;
                         ; --- FEATURE DEVICE CONFIGURATION INFORMATION
                         ;
= 001F                         RTC_FEAT_CON    EQU     31      ; FEATURE CONFIGURATION REG @
= 0080                         SERPLL_INST     EQU     80H     ; SERIAL PARAL CARD INSTALLED
= 0040                         INTMOD_INST     EQU     40H     ; INTERNAL MODEM INSTALLED
= 0020                         MOD12_INST      EQU     20H     ; 1200BPS MODEM INSTALLED
= 0010                         PRI_INST        EQU     10H     ; SER CARD INSTALLED AS PMRY
= 0008                         CMPT_PP_OK      EQU     08H     ; COMPACT PTR PORT PASSED TEST
                         ;
                         ; --- LCD/CRT ADAPTER CONFIGURATION INFORMATION
                         ;
= 0020                         RTC_DSP_CON     EQU     32      ; DISPLAY CONFIGURATION REG  @
= 0080                         DSP_LCD_PRES    EQU     80H     ; ON = LCD PANEL IS PRESENT
= 0010                         DSP_VIDEO_ROM   EQU     10H     ; ON = FEAT VIDEO ROM SENSED
= 0008                         DSP_MONO        EQU     08H     ; ON = MONOCHROME ADAPT PRESENT
= 0004                         DSP_CGA         EQU     04H     ; ON = CGA ADAPTER PRESENT
= 0002                         DSP_MLCD        EQU     02H     ; ON = LCD CONFIGED AS MONO ADA
= 0001                         DSP_CLCD        EQU     01H     ; ON = LCD CONFIGED AS CGA ADA

= 0021                         RTC_SYS_STAT    EQU     33      ; SYSTEM POWER ON STATUS
                         ;
                         ; FOR EQUATES SEE DEFINITION OF PWR_STAT REGISTER 7FH
                         ;
                         ;
                         ; --- LCD/CRT ADAPTER STATUS INFO
                         ;
= 0022                         RTC_DSP_STAT    EQU     34      ; DISPLAY STATUS REGISTER
= 0080                         DIAG_FORCE_SUS  EQU     80H     ; USED TO FORCE RESUME W/O LCD
                                                               ;  (DSP_MLCD & DSP_CLCD = 0)
= 0004                         MONO_BAD        EQU     04H     ; MONO ADAPT PRESENT, BUT BAD
= 0002                         CGA_BAD         EQU     02H     ; CGA PRESENT, BUT BAD
= 0001                         LCD_BAD         EQU     01H     ; LCD WAS CONFIG, BUT BAD
                         ;
                         ; --- BASE 128K STORAGE CHECKSUM
                         ;
= 0023                         RTC_BMEM_CKSL   EQU     35      ; LOW BYTE OF BASE STGE CHECKSUM
= 0024                         RTC_BMEM_CKSH   EQU     36      ; HI BYTE OF BASE STG CHECKSUM
                         ;
                         ;
                         ;
= 0024                         RTC_MEM_END     EQU     36      ; END OF RTC
                         ;
                         ; --- RTC MEMORY GOOD SIGNATURE AREA (MUST BE "RTCG" FOR VALID RTC
                         ;

= 002E                         RTC_SIGNATURE   EQU     46      ; 46-49 VAL RTC SIGNATURE AREA

= 0032                         RTC_CENTURY     EQU     50      ; RTC CENTURY BYTE SAVE AREA
```

```
                      ;*******************************************************
                      ;  G L O B A L   E Q U A T E S   F O R   B I O S / P O S T
                      ;*******************************************************
                      ;
                      ; SPECIAL KEY EQUATES
                      ;
= 0045                        NUM_KEY         EQU     69      ; PC1 SCAN CODE FOR NUM LOCK
= 0046                        SCROLL_KEY      EQU     70      ; PC1 SCAN CODE FOR SCL LK KEY
= 0038                        ALT_KEY         EQU     56      ; PC1 SCAN CODE FOR ALT SFT KEY
= 001D                        CTL_KEY         EQU     29      ; PC1 SCAN CODE FOR CNTL KEY
= 003A                        CAPS_KEY        EQU     58      ; PC1 SCAN CODE FOR SHIFT LOCK
= 002A                        LEFT_KEY        EQU     42      ; PC1 SCAN CODE FOR LEFT SHIFT
= 0036                        RIGHT_KEY       EQU     54      ; PC1 SCAN CODE FOR RGHT SHIFT
= 0052                        INS_KEY         EQU     82      ; PC1 SCAN CODE FOR INSERT KEY
= 0053                        DEL_KEY         EQU     83      ; PC1 SCAN CODE FOR DELETE KEY
= 000F                        TAB_KEY         EQU     15      ; PC1 SCAN CODE FOR TAB KEY
= 0054                        SYSREQ_MAKE     EQU     054H    ; PC1 SCAN CODE SYS_REQ (MAKE)
= 00D4                        SYSREQ_BREAK    EQU     0D4H    ; PC1 SCAN CODE SYS_REQ (BRK)
= 0052                        FN_KEY          EQU     052H    ; NMI SCAN CODE FOR FN KEY
= 003B                        F1_KEY          EQU     59      ; PC1 SCAN CODE FOR F1 KEY
= 00E0                        HIDN_CODE_E0    EQU     0E0H    ; HIDDEN CODE SEQUENCE ID
                      ;----------------------------------------------------
                      ;  KEYBOARD EQUATES USED IN KEYBOARD SUPPORT TABLES  |
                      ;----------------------------------------------------
                      ;
                      ; -- CTL + KEYPAD KEYS TO GIVE ASCII CONTROL CODES
                      ;    (EXCEPT NUL)
= 0000                        NUL         EQU     000
= 0001                        SOH         EQU     001
= 0002                        STX         EQU     002
= 0003                        ETX         EQU     003
= 0004                        EOT         EQU     004
= 0005                        ENQ         EQU     005
= 0006                        ACK006      EQU     006
= 0007                        BEL         EQU     007
= 0009                        HT          EQU     009
= 000A                        LF          EQU     010
= 000B                        VT          EQU     011
= 000C                        FF          EQU     012
                      ;       CR          EQU     013
= 000E                        SO          EQU     014
= 000F                        SI015       EQU     015
= 0010                        DLE         EQU     016
= 0011                        DC1         EQU     017
= 0012                        DC2         EQU     018
= 0013                        DC3         EQU     019
= 0014                        DC4         EQU     020
= 0015                        NAK         EQU     021
= 0016                        SYN         EQU     022
= 0017                        ETB         EQU     023
= 0018                        CAN         EQU     024
= 0019                        EM          EQU     025
= 001A                        SUB         EQU     026
                      ;       ESC         EQU     027
= 001C                        FS          EQU     028
= 001D                        GS          EQU     029
= 001E                        RS          EQU     030
= 001F                        US          EQU     031
= 0020                        ENTER       EQU     032
= 007F                        DEL         EQU     127
```

```
                   ;
                   ; -- KEYPAD BASE PC1 SCAN CODES
                   ;
= 0047                      HOME_KEY   EQU    71
= 0048                      CUR_UP     EQU    72
= 0049                      PGUP       EQU    73
= 004A                      KYPD_MINUS EQU    74
= 004B                      CUR_LFT    EQU    75
= 004D                      CUR_RHT    EQU    77
= 004E                      KYPD_PLUS  EQU    78
= 004F                      END_KEY    EQU    79
= 0050                      CUR_DN     EQU    80
= 0051                      PGDN       EQU    81
                   ;        INS_KEY    EQU    82
                   ;        DEL_KEY    EQU    83
                   ;
                   ; -- CTL + BASE KEYPAD KEYS  EXTENDED ASCII CODE
                   ;
= 0077                      CTL_HOME        EQU   119
= 0084                      CTL_PGUP        EQU   132
= 0073                      CTL_CUR_LFT     EQU   115
= 0074                      CTL_CUR_RHT     EQU   116
= 0075                      CTL_END         EQU   117
= 0076                      CTL_PGDN        EQU   118
                   ;
                   ; -- ASCII CODES
                   ;
= 001B                      ESC        EQU    1BH
= 0008                      BKSPC      EQU    08H
= 0009                      TAB        EQU    09H
= 000D                      ENTER      EQU    0DH
= 0027                      APOSTR     EQU    27H
= 0020                      SPACE      EQU    20H
= 0000                      PSEUDO     EQU    00H          ; BUILD PSEUDO SCAN CODES
                   ;----------------------------------------------
                   ;  BIOS I/O ROUTINE INTERRUPT ASSIGNMENTS
                   ;----------------------------------------------
= 0005                      PRTSC_FN        EQU     05     ; PRINT SCREEN FUNCTION CALL
= 0010                      VIDEO_FN        EQU     10H    ; VIDEO I/O
= 0013                      DSKT_FN         EQU     13H    ; DISKETTE
= 0014                      RS232_FN        EQU     14H    ; COMMUNICATIONS
= 0015                      SYSSERV_FN      EQU     15H    ; SYSTEM SERVICES
= 0016                      KEYBD_FN        EQU     16H    ; KEYBOARD
= 0017                      PRINTER_FN      EQU     17H    ; PRINTER
= 0019                      BOOT_FN         EQU     19H    ; SYSTEM BOOT STRAP
= 001A                      TOD_FN          EQU     1AH    ; TIME OF DAY
                   ;
                   ; EQUATES FOR ERRORS IN RETURN CODE OF POST ROUTINES
                   ;
                   ;        FATAL_ERROR     EQU     80H    ; FATAL ERROR  (POST_STATUS)
                   ;        NON_FATAL_ERR   EQU     40H    ; NON FATAL ERR  (POST_STATUS)
= 0020                      POST_MSG        EQU     20H    ; ERR MSG POINTED TO BY ES:DX
= 0010                      RE_DISPATCH     EQU     10H    ; RE-DSPTCH LST RTN AFTER STAT
= 0008                      NON_FATAL_NW    EQU     08H    ; NON FATAL BUT NOT WARMSTART
= 0004                      DISPLAY_ERR     EQU     04H    ; ERROR COMES FROM DSPLY TEST
                   ;
                   ; GENERAL DELAY LOOP COUNTS USING LOOP $ WITH INTERRUPTS DISABLED
                   ;
= 0102                      MS_DELAY        EQU     258    ; 1 MILLISECOND DELAY COUNT
= 006B                      DELAY_415US     EQU     107    ; 415 USEC DLY COUNT FOR COMMO
```

```
                                  INCLUDE SROMDATA.INC
```

Processor Interrupt Vector Area

```
              ;--------------------------------------
              ;   8088 INTERRUPT LOCATIONS        :
              ;--------------------------------------

0000                    ABSO    SEGMENT AT 0
0000                    STG_LOCO        LABEL   BYTE
0008                       ORG     2*4
0008                    NMI_PTR         LABEL    WORD
0014                       ORG     5*4
0014                    INT5_PTR        LABEL   WORD    ; PRT SCREEN INTERRUPT VECTOR
0020                       ORG     8*4
0020                    INT_ADDR        LABEL   WORD
0020                    INT_PTR         LABEL   DWORD
0024                       ORG     9*4
0024                    INT9_PTR        LABEL   WORD    ; KEYBOARD INTERRUPT VECTOR
0040                       ORG     10H*4
0040                    VIDEO_INT       LABEL   WORD    ; VIDEO I/O INTERRUPT VECTOR
0070                       ORG     1CH*4
0070                    ONECH      LABEL   WORD         ; USER TIMER VECTOR
0074                       ORG     1DH*4
0074                    PARM_PTR        LABEL   DWORD   ; POINTER TO VIDEO PARMS
0060                       ORG     18H*4
0060                    BASIC_PTR       LABEL   WORD    ; ENTRY FOR RESIDENT BASIC
0078                       ORG     01EH*4
0078                    DISK_POINTER    LABEL   DWORD
007C                       ORG     01FH*4               ; LOCATION OF POINTER
007C                    EXT_PTR         LABEL   DWORD   ; POINTER TO EXTENSION
0110                       ORG     044H*4
0110                    CSET_PTR        LABEL   DWORD   ; PTR TO LOWER 128 CHAR SET
0128                       ORG     04AH*4
0128                    RTCA_PTR        LABEL   DWORD   ; PTR TO USER RTC ALARM VECTOR
01B0                       ORG     06CH*4
01B0                    RESUME_PTR      LABEL   DWORD   ; PTR TO PROGRAM RESUME VECTOR
0400                       ORG     400H
0400                    DATA_AREA       LABEL   BYTE    ; ABS LOCATION OF DATA SEGMENT
0400                    DATA_WORD       LABEL   WORD
0600                       ORG     0600H                ;
0600                    MFG_TEST_RTN    LABEL   FAR
7C00                       ORG     7C00H
7C00                    BOOT_LOCN       LABEL   FAR
7C00                    ABSO    ENDS
```

POST and Bootstrap Temporary Stack

```
              ;----------------------------------------------
              ; STACK -- USED DURING INITIALIZATION ONLY      :
              ;----------------------------------------------

0000                    STACK   SEGMENT AT 30H
0000     7F [              DW      127 DUP(?)
              ????
                    ]

00FE                    TOP_OF_STACK  LABEL   WORD
00FE                    STACK   ENDS
```

BIOS Data Areas

```
                  ;---------------------------------------
                  ;  ROM BIOS DATA AREAS             :
                  ;---------------------------------------

0000                        DATA    SEGMENT AT 40H

0000     04 [               RS232_BASE      DW      4 DUP(?) ; ADDR RS232 ADA (COM1-COM4)
              ????
                    ]


0008     04 [               PRINTER_BASE    DW      4 DUP(?) ; ADDR PRINTERS (LPT1-LPT3)
              ????
                    ]

                  ;
                  ; E Q U I P M E N T   W O R D
                  ;
0010  ????                  EQUIP_FLAG      DW      ?      ; INSTALLED HARDWARE
                  ;  ---------------------------------------------------
                  ;  | 15-14 | 13 | 12 | 11-9 | 8 | 7-6 | 5-4 | 3-1 | 0 |
                  ;  ---------------------------------------------------
                  ;      |      |    |     |    |    |     |     |   |
                  ;      |      |    |     |    |    |     |     |   IPL DSKT INSTALLED
                  ;      |      |    |     |    |    |     |   -- UNUSED
                  ;      |      |    |     |    |    |   -- INITIAL VIDEO MODE
                  ;      |      |    |     |    |   -- NUMBER OF DISKETTE DRIVES
                  ;      |      |    |     |   -- UNUSED
                  ;      |      |    |   -- NUMBER OF COM DEVICES
                  ;      |      |   -- JOYSTICK ATTACHED
                  ;      |     -- INTERNAL MODEM INSTALLED
                  ;     ----  NUMBER OF PRINTERS ATTACHED
                  ;
                  ; P O S T   S T A T U S   F L A G
                  ;
0012  ??                    POST_STATUS     DB      ?      ; POST ERROR FLAGS AND STATUS
                                                           ; (DURING POST)

= 0080                      FATAL_ERROR     EQU     80H    ; POST DETCD FATAL SYS ERROR
= 0040                      NON_FATAL_ERR   EQU     40H    ; POST HAS DETECTED AN ERROR
                                                           ; (NON_FATAL)
= 0020                      BAD_RTC_MEM     EQU     20H    ; RTC MEMORY BAD - DO NOT USE
= 0010                      FEATURE_ERROR   EQU     10H    ; FEAT DEV FAILED (ROM_SCAN)
= 0002                      MFG_MODE        EQU     02H    ; MANUFACTURING MODE ACTIVE
= 0001                      POST_ACTIVE     EQU     01H    ; POST IS CURRENTLY ACTIVE
                  ;
                  ; M E M O R Y   S I Z E   ( 1 K   B Y T E S )
                  ;
0013  ????                  MEMORY_SIZE     DW      ?      ; MEMORY SIZE IN K BYTES
                  ;
                  ; BATTERY CONTROL FLAGS
                  ;
0015  ??                    BAT_STATUS      DB      ?      ; BATTERY SUPPORT STATUS
= 0080                      LOW_BAT_SIG     EQU     80H    ; LOW BATTERY SIGNALLED FLAG
= 0040                      LOW_BAT_HOLD    EQU     40H    ; LOW BATTERY KEY HOLD STATE
= 0020                      LOW_BAT_PEND    EQU     20H    ; LOW BAT PENDING SECOND SENSE
```

```
                ;
                ; B I O S   S T A T U S   F L A G
                ;
0016  ??                 BIOS_STATUS     DB      ?       ; BIOS STATUS FLAGS
= 0080                   DSP_BLANKED     EQU     80H     ; DISPLAY HAS BEEN BLANKED
                                                         ; (KYBD INACTIVE)
= 0040                   F_RESUME        EQU     40H     ; FORCE SYSTEM RESUME MODE REQ
= 0020                   KYBD_ACTIVE     EQU     20H     ; KEBD HAS HAD A KEY PRESSED
= 0010                   BOOT_F1HIT      EQU     10H     ; F1 KEY HIT IN BOOT ROUTINE
= 0004                   DCL_SUPPORTED   EQU     04H     ; DSKT CHANGE LINE SUPPORTED
= 0002                   FORCE_DCL       EQU     02H     ; SYS RESUMED - FORCE DISKETTE
                                                         ; CHANGE LINE ERROR ON NEXT OP
= 0001                   KB_NOISE_ACT    EQU     01H     ; FLAG TO SHOW AUDIO ROUTINE

                ;---------------------------------------
                ;  KEYBOARD DATA AREAS                :
                ;---------------------------------------
0017                     KB_AREA1        LABEL   BYTE    ; KYBD INT 9 AND INT 16 FLAGS
                                                         ; AND BFR (CLRD BY KYBD_RESET
                                                         ; ROUTINE DURING POST)
                ;
                ; K E Y B O A R D   F L A G
                ;
0017  ??                 KB_FLAG         DB      ?       ;KEYBOARD FLAG STATUS BYTE 1

= 0080                   INS_STATE       EQU     80H     ; INSERT STATE IS ACTIVE
= 0040                   CAPS_STATE      EQU     40H     ; CAPS LOCK STATE TOGGLED
= 0020                   NUM_STATE       EQU     20H     ; NUM LOCK STATE TOGGLED
= 0010                   SCROLL_STATE    EQU     10H     ; SCROLL LOCK STATE TOGGLED
= 0008                   ALT_SHIFT       EQU     08H     ; ALTERNATE SHIFT KEY PRESSED
= 0004                   CTL_SHIFT       EQU     04H     ; CONTROL SHIFT KEY PRESSED
= 0002                   LEFT_SHIFT      EQU     02H     ; LEFT SHIFT KEY PRESSED
= 0001                   RIGHT_SHIFT     EQU     01H     ; RIGHT SHIFT KEY PRESSED
                ;
                ; K E Y B O A R D   F L A G   1
                ;
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                   SYS_SHIFT       EQU     04H     ; SYS REQUEST KEY IS DEPRESSED

0019  ??                 ALT_INPUT       DB      ?       ; STORAGE FOR ALT KEYPAD ENTRY
                ;
                ; K E Y B O A R D   A S C I I   B U F F E R   A N D   P O I N T E R S
                ;
001A  ????               BUFFER_HEAD     DW      ?       ; POINTER TO HEAD OF KBD BUFFER

                ;------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY

001C  ????               BUFFER_TAIL     DW      ?       ; POINTER TO TAIL OF KBD BUFFER

001E     10 [            KB_BUFFER       DW      16 DUP(?) ; ROOM FOR 16 ENTRIES
                ????
                      ]

003E                     KB_BUFFER_END   LABEL   WORD    ; LAST BYTE IN KB_AREA1_LNG

= 0027                   KB_AREA1_LNG    EQU     $-KB_AREA1 ; LNTH OF KEYBOARD AREA 1
```

```
                 ;--------------------------------------
                 ;  DISKETTE DATA AREAS            :
                 ;--------------------------------------
                 ;
                 ; S E E K   S T A T U S   A N D   F L A G S
                 ;
003E  ??                 SEEK_STATUS     DB       ?       ; DRIVE RECALIBRATION STATUS
                                                           ; BIT 3-0 = DRV 3-0 NEEDS RECAL
                                                           ; BEFORE NEXT SEEK IF BIT IS = 0
= 0080                   INT_FLAG        EQU      080H    ; BIT 7 = INTERRUPT OCCURRED

                 ;
                 ; M O T O R   S T A T U S   A N D   M O T O R   O F F   D E L A Y   C O U N T
                 ;
003F  ??                 MOTOR_STATUS    DB       ?       ; MOTOR STATUS
= 0080                   WRITE_OP        EQU      080H    ; CURRENT OPERATION IS A WRITE
= 0020                   MOTOR_OK        EQU      020H    ; MOTOR ON FOR 500 MSEC
                                                           ; BIT 3-0 = DRV 3-0 IS CNTRLY
                                                           ;  RUNNING

0040  ??                 MOTOR_COUNT     DB       ?       ; TIME OUT CNTR FOR DRIVE OFF
= 0025                   MOTOR_WAIT      EQU      37      ; 2 SECS OF CNTS FOR MOTOR OFF
                 ;
                 ; D I S K E T T E   O P E R A T I O N   E N D   S T A T U S
                 ;
0041  ??                 DISKETTE_STATUS DB       ?       ; RETURN CODE STATUS BYTE

= 0080                   TIME_OUT        EQU      80H     ; ATTACH 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     ; ATMPT TO DMA CROSS 64K BNDRY
= 0008                   BAD_DMA         EQU      08H     ; DMA OVERRUN ON OPERATION
= 0006                   MEDIA_CHANGE    EQU      06H     ; MEDIA CHANGED ON 3.5" DRIVES
= 0004                   RECORD_NOT_FND  EQU      04H     ; REQUESTED SECTOR NOT FOUND
= 0003                   WRITE_PROTECT   EQU      03H     ; WRITE ATTEMP ON WRT PROT DSK
= 0002                   BAD_ADDR_MARK   EQU      02H     ; ADDRESS MARK NOT FOUND
= 0001                   BAD_CMD         EQU      01H     ; BAD CMD PASSED TO DSKT I/O

                 ;
                 ; D I S K E T T E   C O N T R O L L E R   S T A T U S
                 ;
0042      07 [           NEC_STATUS      DB       7 DUP(?) ; STATUS BYTES FROM CNTLR
                 ??
                     ]

                 ;--------------------------------------
                 ;  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 ADDR IN REGEN BFR

0050      08 [           CURSOR_POSN     DW       8 DUP(?) ; CURSOR EACH OF UP TO 8 PGS
                 ????
                      ]

0060  ????               CURSOR_MODE     DW       ?       ; CURRENT CURSOR MODE SETTING

0062  ??                 ACTIVE_PAGE     DB       ?       ; CURRENT PAGE BEING DISPLAYED

0063  ????               ADDR_6845       DW       ?       ; BASE ADDR FOR ACT DSPLY CARD

0065  ??                 CRT_MODE_SET    DB       ?       ; CRNT SETTING OF THE 3X8 REG

0066  ??                 CRT_PALETTE     DB       ?       ; CRNT PALETTE SETTING COLOR CD
```

```
              ;
              ; THE FOLLOWING FOUR BYTES ARE LOADED IN THE BLANK_CTR THEN SYS_OFF_CTR
              ; BY THE RTC_ALARM INTERRUPT HANDLER. THE ORDER MUST NOT BE CHANGED
              ;
              ;
              ;  AUTO DISPLAY BLANK TIME COUNTER
              ;
0067  ????          DSP_BLANK_CTR   DW      ?      ; TIME DSPLY BLANKED (SECS)
              ;
              ; AUTO SYSTEM OFF TIME COUNTER
              ;
0069  ????          SYS_OFF_CTR     DW      ?      ; TIME SYSTEM IS POWERED OFF

006B  ??            INTR_FLAG       DB      ?      ; FLAG TO SHOW AN INTERRUPT

              ;----------------------------------------
              ;  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 ROLLED SINCE LAST RD

= 0059              SEC_MAX_LIMIT   EQU    59H     ; MAX FOR BCD SECONDS
= 0059              MIN_MAX_LIMIT   EQU    59H     ; MAX FOR BCD MINUTES
= 0023              HRS_MAX_LIMIT   EQU    23H     ; MAX FOR BCD HOURS
= 0002              GET_RTC_TIME    EQU    2       ;
= 0012              COUNTS_SEC      EQU    18      ; TIMER COUNTS PER SECOND
= 0444              COUNTS_MIN      EQU    1092    ; TIMER COUNTS PER MINUTE
= 0007              COUNTS_HOUR     EQU    7       ; 65543-10000H (65543 TOO LARGE
                                                   ; FOR 1 WORD, SO LOW WORD IS
                                                   ; USED IN CALCULATIONS)
=                   COUNTS_DAY      EQU    1573040 ; = 1800B0H  COUNTS PER DAY
= 0018              COUNTS_DAY_HI   EQU    18H     ; HIGH BYTE OF COUNTS PER DAY
= 00B0              COUNTS_DAY_LO   EQU    0B0H    ; LOW BYTE OF COUNTS PER DAY

              ;----------------------------------------
              ;  SYSTEM DATA AREA                  :
              ;----------------------------------------

0071  ??            BIOS_BREAK      DB      ?      ; BREAK KEY FLAGS
= 0080              BREAK_HIT       EQU     80H    ; BREAK KEY HAS BEEN HIT


0072  ????          RESET_FLAG      DW      ?      ; SYS RESET TYPE CONTROL FLAG
= 1234              SOFT_RESET      EQU     1234H  ; RE-IPL WITHOUT STORAGE CLEAR
= 5678              SYS_SUSPEND     EQU     5678H  ; SYS SUCCESSFULLY SUSPENDED
= 9ABC              MFG_MEM_MODE    EQU     9ABCH  ; MANUFACTURING MEMORY MODE
= ABCD              LOOP_MODE       EQU     0ABCDH ; POST LOOP MODE

0074     02 [        DW      2 DUP(?)              ; RESERVED
              ????
                   ]


              ;------------------------------------------------------
              ;  PRINTER AND RS232 TIME-OUT VARIABLES              :
              ;------------------------------------------------------

0078     03 [       PRINT_TIM_OUT   DB      3 DUP(?) ; FOR LPT1-LPT4 RESPECTIVELY
              ??
                  ]


007B  ??            EVENT_TIM_OUT   DB      ?      ; WAIT ON EXT EVNT TIMEOUT CTR

007C     04 [       RS232_TIM_OUT   DB      4 DUP(?) ; FOR COM1-COM4 RESPECTIVELY
              ??
                  ]
              ;--------------------------------------
              ;  ADDITIONAL KEYBOARD DATA AREA   :
              ;--------------------------------------

0080  ????            BUFFER_START    DW      ?      ; PTRS TO 16 BYTE KBD BUFFER

0082  ????            BUFFER_END      DW      ?      ; INIT SET TO KB_BUFFER AND
              ; KB_BUFFER_END BY POST


0084     07 [          DB      7  DUP(?)             ; RESERVED
              ??
                 ]

              ;
              ; DISKETTE LAST DATA RATE INFORMATION
              ;
008B  ??              LAST_DATA_RATE  DB      ?      ; ALWAYS 80H - 250 KBS RATE
= 0080                RATE_250KBS     EQU     80H    ; 250 KB/SEC DATA RATE VALUE
                                                     ; SET BY POST DISKETTE TEST

008C     02 [          DW      2 DUP(?)              ; RESERVED
              ????
                   ]

              ;
              ; DISKETTE DRIVE MEDIA TYPE CODES  (MEDIA TYPE BYTE = 0 IF NO DRIVE)
              ;
              ;  ------------------------------
              ;  | 7 - 6 | 5  | 4  | 3 | 2 - 0 |
              ;  ------------------------------
              ;      |     |    |    |   -----
              ;      |     |    |    |     |
              ;      |     |    |    |     |
              ;      |     |    |    |     |
              ;      |     |    |    |     |
              ;      |     |    |    |     -- MEDIA TYPE CODES (ALWAYS
              ;      |     |    |    |        111 FOR 720KB MEDIA)
              ;      |     |    |    -- RESERVED - ALWAYS 0
              ;      |     |    -- MEDIA TYPE ESTABLISHED - ALWAYS 1
              ;      |     -- DOUBLE STEP REQUIRED - NOT SUPPORTED ALWAYS 0
              ;     ----  DATA TRANSFER RATE FOR THIS DRIVE  - ALWAYS 10 (250 KBS)
              ;
0090  ??              MEDIA_TYPE_DR0  DB      ?      ; DISKETTE MEDIA TYPE DRIVE 0
0091  ??              MEDIA_TYPE_DR1  DB      ?      ; DISKETTE MEDIA TYPE DRIVE 1
= 0097                MEDIA_720KB     EQU     97H    ; MEDIA TYPE FOR 720 KB DSKT
0092  ??               DB      ?                     ; RESERVED
              ;
              ; LOW BATTERY WARNING COUNTER
              ;
0093  ??              LOW_BAT_CTR     DB      ?      ; LOW BATTERY WARNING COUNTER

0094  ????             DW      ?                     ; RESERVED

              ;
              ; KEYBOARD CONTROL FLAG 3 USED BY INTERRUPT 9 KEYBOARD PROCESSING
              ;
0096  ??              KB_FLAG_3       DB      ?      ; KEYBOARD TRACKING FLAG 3
                                                     ; (CLEARED BY KYBD_RESET
                                                     ; ROUTINE IN POST)
= 0002                LC_HC      EQU     02H         ; LAST CODE HIDDEN CODE FLAG
0097  ??               DB      ?                     ; RESERVED
```

```
                ;-------------------------------------
                ; EVENT POST/WAIT DATA AREA           :
                ;-------------------------------------
                ;
                ; POINTER TO USERS POST FLAG
                ;
0098                    IO_ROM_INIT     LABEL   WORD    ; REDEFINITION OF WORD DURING
                                                        ; POST ROM_SCAN
0098  ????              USER_FLAG       DW      ?       ; OSET ADDR OF USERS WAIT FLAG

009A                    IO_ROM_SEG      LABEL   WORD    ; REDEFINITION OF WORD DURING
                                                        ; POST ROM_SCAN
009A  ????              USER_FLAG_SEG   DW      ?       ; SEG ADDR OF USERS WAIT FLAG

= 0080                  EVENT_POSTED    EQU     080H    ; FLAG SET WHEN WAIT TIME EXP
                                                        ; IN USERS FLAG
                ;
                ; I N T E R V A L  W A I T  T I M E  S A V E  A R E A
                ;
009C  ????              RTC_LOW         DW      ?       ; LOW WORD OF USER WAIT COUNT
009E  ????              RTC_HIGH        DW      ?       ; HIGH WORD OF USER WAIT COUNT
                ;
                ; P O S T / W A I T  F L A G S
                ;
00A0  ??                RTC_WAIT_FLAG   DB      ?       ; WAIT ACTIVE FLAG
= 0080                  POSTED      EQU     80H         ; POST ON INTERVAL OCCURRED
= 0004                  PON_ALRM_PEND   EQU     04H     ; POWER ON BY ALARM PENDING
= 0002                  ALARM_PEND      EQU     02H     ; USER ALRM INT 4AH PNDNG SPND
= 0001                  INTERVAL_WAIT   EQU     01H     ; INTVL WAIT CURRENTLY ACTIVE


                ;
                ; KEYBOARD NMI PRE-PROCESSING FUNCTION CONTROL SAVE AREA B4H-CCH
                ;
00B4                     ORG     0B4H
00B4                    KB_AREA2        LABEL   BYTE    ; KBD AREA 2 (CLRD KYBD_RESET
                                                        ; ROUTINE IN POST)
00B4  ??                KB_NMI_CNTL     DB      ?       ; KBD PREPROCESSOR CTRLL FLAGS
= 0080                  P60_LOADED      EQU    80H      ; KEYBOARD PORT 60 IS LD FLAG
= 0040                  FUNC_STATE      EQU    40H      ; KEYBOARD FUNCTION KEY STATE
= 0010                  XLATE_BUSY      EQU    10H      ; KEYBOARD TRANSLATION ACTIVE
= 0008                  CLICK_ON        EQU    08H      ; KEYBOARD CLICKER IS ENABLED
= 0004                  KEYPAD_STATE    EQU    04H      ; KEYBOARD KEYPAD STATE

00B5  ????              B_PEND1         DW      ?       ; KEYBOARD BRK PENDING FLAGS 1
00B7  ????              B_PEND2         DW      ?       ; KEYBOARD BRK PENDING FLAGS 2
00B9  ??                P60_HOLD_BYTE   DB      ?       ; PORT 60 SINGLE BYTE QUEUE
00BA  ??                LAST_CLICK_KEY  DB      ?       ; AREA FOR SCAN CD OF LAST KEY
                                                        ; CLICKED
00BB  ??                KB_NMI_HEAD     DB      ?       ; PTR TO HEAD OF PREPROC BFR
00BC  ??                KB_NMI_TAIL     DB      ?       ; PTR TO END OF PREPROC BFR

00BD      10 [          KB_NMI_BUFFER   DB      16 DUP(?) ; ROOM FOR 16 ENTRIES
              ??
                ]

= 0010                  KB_NMI_BLTH     EQU     $-KB_NMI_BUFFER ; LENGTH OF NMI BFR

= 0019                  KB_AREA2_LNG    EQU     $-KB_AREA2 ; LENGTH OF KYBD AREA 2

00CD                    DATA    ENDS
```

POST Temporary Data Area

```
                ;--------------------------------------------------------
                ; EXTRA DATA SEGMENT - FOR POST AND PRINT SCREEN STATUS ONLY
                ;--------------------------------------------------------

0000                    XXDATA  SEGMENT AT 50H
0000  ??                STATUS_BYTE     DB      ?      ; PRINT SCREEN ACTIVE STATUS
= 0001                  PRTSC_ACTIVE    EQU     01     ; BIT ON WHEN PRTSC IS ACTIVE
= 00FF                  PRTSC_ERROR     EQU     0FFH   ; PRTSC ERROR - CANCELLED

0001  ????              POST_PTR        DW      ?      ; PTR TO CURRENT TEST ROUTINE

0003  ??                POST_MASK       DB      ?      ; PWR ON SELF TST ROUTINE MASK
= 0080                  SOFT_MODE       EQU     80H    ; SYSTEM SOFT START MODE
= 0040                  WARM_MODE       EQU     40H    ; SYSTEM WARM START MODE
= 0020                  COLD_MODE       EQU     20H    ; SYSTEM COLD START MODE
= 0010                  MFG_TST         EQU     10H    ; MANUFACTURING TEST MODE

0004  ??                POST_DEVID      DB      ?      ; DEV ID FOR POST ROUTINE CALLS
= 0001                  SYS_ID     EQU     01          ; SYSTEM UNIT ID
= 0002                  MEM_ID     EQU     02          ; MEMORY ID
= 0003                  KYBD_ID         EQU     03     ; KEYBOARD ID
= 0004                  MONO_ID         EQU     04     ; MONOCHROME CRT ADAPTER ID
= 0005                  COLOR_ID        EQU     05     ; COLOR CRT ADAPTER ID
= 0006                  DSKT_ID         EQU     06     ; DISKETTE ID
= 0009                  PPRT_ID         EQU     09     ; PARALLEL PRINTER ID
= 0010                  APRT_ID         EQU     10H    ; ALT PARALLEL PRINTER ID
= 0012                  RS232_ID        EQU     12H    ; RS232 PORT ID
= 0011                  MODEM_ID        EQU     11H    ; MODEM_ID
= 0050                  LCD_ID     EQU     050H        ; LCD DEVICE ID
= 0051                  CPRT_ID         EQU     051H   ; COMPACT PRINTER PORT ID

0005  ??                POST_ICON       DB      ?      ; ICON NUMBER FOR FAILING AREA
= 0001                  SYSTEM     EQU     01          ; SYSTEM UNIT ICON
= 0002                  FEATURE         EQU     02     ; SYSTEM FEATURE ICON
= 0003                  SETUP      EQU     03          ; SYSTEM SETUP
= 0004                  BATTERY         EQU     04     ; BATTERY LOW ICON
= 0005                  PCHECK2         EQU     05     ; SYSTEM PARITY CHECK 2
= 0006                  PROMPT     EQU     06          ; USER PROMPT ICON

0006  ????              MFG_ERR_CODE    DW      ?      ; MANUFACTURING ERROR CODE
0008  ??                MFG_CKPT        DB      ?      ; MANUFACTURING CHECKPOINT

0009                    TEMP       LABEL   BYTE        ; USED DURING SELF TEST ONLY
0009      10 [           DB      16  DUP(?)
              ??
                 ]

0019                    ASC_STR         LABEL   BYTE
0019      11 [           DB      17 DUP(?)            ; ASCII STRING FOR ERROR MSG
              ??
                 ]

002A      07 [          ICON_DIS        DB      7 DUP(?) ; AREA USED FOR E_MSG ONLY
              ??
                 ]

0031      07 [          ICON_MSG        DB      7 DUP(?) ; AREA USED FOR E_MSG ONLY
              ??
                 ]
```

```
0038  ??                 DSPTEST_MASK    DB       ?       ; DISPATCH MASK FOR DSP_TEST
= 0080                   VROM_CGA        EQU      80H     ; VIDEO ROM CGA AVAILABLE
= 0040                   VROM_MONO       EQU      40H     ; VIDEO ROM MONO AVAILABLE
= 0008                   VIDEO_ROM       EQU      08H     ; DISPATCH VIDEO_LINK
= 0004                   MONO       EQU     04H           ; MONO AVAILABLE
= 0002                   CGA        EQU     02H           ; CGA AVAIL / DSPATCH CRT_TEST
= 0001                   LCD        EQU     01H           ; LCD AVAIL / DSPATCH LCD TSTS

0039  ??                 SUSP_DSP_CON    DB       ?       ; HLDS RTC_DSP_CON VALUE

003A                     XXDATA  ENDS

               ;-----------------------------------------
               ;  VIDEO DISPLAY BUFFER            :
               ;-----------------------------------------

0000                     VIDEO_RAM       SEGMENT AT 0B800H

0000                     REGEN   LABEL   BYTE
0000                     REGENW  LABEL   WORD

0000  4000 [                DB      16384 DUP(?)          ; 16K BYTE DISPLAY BUFFER
               ??
                  ]


4000                     VIDEO_RAM       ENDS
                            INCLUDE SROMMACS.MAC
```

Common Macros

```
IDENT MACRO    SNAME,SEQ,REV
NAME  SNAME
 ENDM

SAVE    MACRO
   PUSH    AX
   PUSH    BX
   PUSH    CX
   PUSH    DX
   PUSH    BP
   PUSH    SI
   PUSH    DI
   PUSH    ES
   PUSH    DS
   ENDM

RESTORE MACRO
   POP     DS
   POP     ES
   POP     DI
   POP     SI
   POP     BP
   POP     DX
   POP     CX
   POP     BX
   POP     AX
   ENDM
```

```
;
; THE FOLLOWING MACROS MAY BE USED WHEN A CONDITIONAL JUMP OF MORE THAN
; +/- 128 BYTES IS REQUIRED. THE MACROS ARE NAMED THE SAME WAY AS THE
; CONDITIONAL JUMP EXCEPT J BECOMES JF.
;
        JFA   MACRO    JLABEL
         JNA      $+5                   ; IF NOT ABOVE JUMP AROUND JMP
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFNA  MACRO    JLABEL
         JA       $+5                   ; IF NOT NOT ABOVE JUMP AROUND
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFB   MACRO    JLABEL
         JNB      $+5                   ; IF NOT BELOW JUMP AROUND JMP
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFNB  MACRO    JLABEL
         JB       $+5                   ; IF NOT NOT BELOW JUMP AROUND
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFG   MACRO    JLABEL
         JNG      $+5                   ; IF NOT GREATER JUMP AROUND
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFNG  MACRO    JLABEL
         JG       $+5                   ; IF NOT NOT GTR JUMP AROUND
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFL   MACRO    JLABEL
         JNL      $+5                   ; IF NOT LESS JUMP AROUND JUMP
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFNL  MACRO    JLABEL
         JL       $+5                   ; IF NOT NOT LESS JUMP AROUND
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFO   MACRO    JLABEL
         JNO      $+5                   ; IF NOT OVERFLOW JUMP AROUND
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFNO  MACRO    JLABEL
         JO       $+5                   ; IF NOT NOT OFLOW JMP AROUND
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFP   MACRO    JLABEL
         JNP      $+5                   ; IF NOT PRTY E JMP AROUND JMP
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFNP  MACRO    JLABEL
         JP       $+5                   ; IF PARITY E JMP AROUND JMP
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFS   MACRO    JLABEL
         JNS      $+5                   ; IF NOT SIGN NEG JMP AROUND
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFNS  MACRO    JLABEL
         JS       $+5                   ; IF SIGN NEGATIVE JUMP AROUND
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFZ   MACRO    JLABEL
         JNZ      $+5                   ; IF NOT ZERO JUMP AROUND JUMP
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
        JFNZ  MACRO    JLABEL
         JZ       $+5                   ; IF NOT NOT 0 JMP AROUND JMP
         JMP      JLABEL                ; ELSE TAKE A LONG JUMP
         ENDM
           END
```

Nonmaskable Interrupt Handler (B10NMIH)

```
                ;****************************
                ;   P U B L I C S
                ;****************************
                            PUBLIC  NMI_FLIH
                            PUBLIC  PUT_RTC_NMI
                            PUBLIC  GET_RTC_NMI
                ;****************************
                ;   E X T E R N A L S
                ;****************************
                            EXTRN   PAR_CHK:BYTE
                            EXTRN   ICON_PR:NEAR
                            EXTRN   ENABLE_NMI:NEAR
                            EXTRN   DISABLE_NMI:NEAR

0000                     ROMCODE SEGMENT BYTE PUBLIC
                            ASSUME CS:ROMCODE
                            IDENT B10NMIH,10,00
```

First Level Interrupt Handler (NMI_FLIH)

```
                ;***********************************************************************
                ;
                ; MODULE-NAME :     NMI_FLIH
                ; DATE LAST MODIFIED : 09/12/85
                ;
                ; DESCRIPTIVE-NAME : THIS ROUTINE SUPPORTS THE SIX TYPES OF NMI SOURCES
                ;
                ; COPYRIGHT : 7396-917 (C) COPYRIGHT IBM CORP. 1985
                ;             REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
                ;
                ; CHANGE LEVEL: EC000
                ;
                ; FUNCTION:   THIS MODULE HANDLES NON-MASKIBLE INTERRUPTS. THE
                ;             SIX TYPES OF NMI'S ARE:
                ;                DISKETTE NMI - I/O INSTR. ISSUED WITH CONTROLLER OFF
                ;                I/O CHANNEL CHECK - I/O DEVICE CHANNEL ERROR
                ;                SYSTEM SUSPEND - 2 SECONDS BEFORE POWER OFF OCCURS
                ;                KYBD_DATA      - KEYBOARD HAS DATA IN PORT 7DH
                ;                KYBD_CLEAR     - INT 9 HANDLER ISSUED KYBD CLEAR I/O
                ;                RTC_ALARM      - REAL TIME CLOCK INTERRUPT
                ;
                ; MODULE SIZE: 2618 BYTES
                ;
```

```
                    ; ENTRY CONDITIONS:
                    ;
                    ;       PURPOSE OF ENTRY: SERVICE NON-MASKIBLE INTERRUPT
                    ;
                    ;       INPUT CONDITIONS: NON-MASKIBLE INTERRUPT OCCURRED
                    ;
                    ;       RESTRICTIONS: NONE
                    ;
                    ;       INTERNALLY REFERENCED ROUTINES:
                    ;            DSKT_RESYNC      KYBD_PREP       KYBD_CLR
                    ;            KYBD_XLT         RTC_ALARM_NMILOW_BAT_CHK
                    ;            CHAN_CHK         SUSPEND         RESUME
                    ;
                    ; EXTERNALLY REFERENCED ROUTINES: REFER TO EXTRN LIST
                    ;
                    ; CHANGE ACTIVITY:  NONE
                    ;
                    ;**********************************************************************
                            REGSAVE STRUC
0000      05 [                 DW      5 DUP(?)
                    ????
                          ]

000A  ????                  DXSAVE  DW      ?               ; DX SAVE AREA ON STACK
000C  ????                  CXSAVE  DW      ?               ; CX SAVE AREA ON STACK
000E  ????                  BXSAVE  DW      ?               ; BX SAVE AREA ON STACK
0010  ????                  AXSAVE  DW      ?               ; AX SAVE AREA ON STACK
0012  ????                  OFFSAVE DW      ?               ; OFFSET SAVE AREA ON STACK
0014  ????                  SEGSAVE DW      ?               ; SEGMENT SAVE AREA ON STACK
0016  ????                  FLGSAVE DW      ?               ; FLAGS SAVE AREA ON STACK
0018                        REGSAVE ENDS

                            REGLSAV STRUC
0000      08 [                 DW      8 DUP(?)
                    ????
                          ]

0010  ??                    ALSAVE  DB      ?               ; AL SAVE AREA ON STACK
0011  ??                    AHSAVE  DB      ?               ; AH SAVE AREA ON STACK
0012                        REGLSAV ENDS

= 0200                      I_FLAG          EQU     0200H   ; INTERRUPT FLAG IN FLAGS
                                                            ; REGISTER
= 00EC                      IN_INSTR        EQU     0ECH    ; OBJECT CODE FOR IN AL,DX INSTR
= 00EE                      OUT_INSTR       EQU     0EEH    ; OBJECT CODE FOR OUT DX,AL INSTR

                             ASSUME  DS:DATA,ES:NOTHING
0000                        NMI_FLIH        PROC    FAR
0000  50                     PUSH    AX                     ; SAVE REGISTERS
0001  B0 07                  MOV     AL,DISABLE_SLEEP+CLOCK_RUN ; DISABLE NMIS
0003  E6 72                  OUT     CLOCK_CTL,AL           ;
0005  53                     PUSH    BX
0006  51                     PUSH    CX
0007  52                     PUSH    DX
0008  55                     PUSH    BP
0009  56                     PUSH    SI
000A  57                     PUSH    DI
000B  06                     PUSH    ES
000C  1E                     PUSH    DS
000D                        NMI_REDRIVE:
000D  FC                     CLD                            ; CLEAR DIRECTION
000E  B8 ---- R              MOV     AX,DATA
0011  8E D8                  MOV     DS,AX
0013  8B EC                  MOV     BP,SP                  ; SET BP TO STACK PTR
0015  E4 62                  IN      AL,NMI_SRC             ; READ NMI SOURCE FLAGS
0017  24 DF                  AND     AL,NOT TIMER2_SN       ; SAVE ALL BUT TIMER SENSE
0019  74 46                  JZ      NMIH_OUT

001B  8A D8                  MOV     BL,AL                  ; BL <--- NMI SOURCE FLAGS
001D  E4 61                  IN      AL,NMI_CNTL            ;
001F  0C 08                  OR      AL,DIS_ALARM           ; DISABLE RTC ALARM NMI
0021  E6 61                  OUT     NMI_CNTL,AL            ;
```

```
0023  F6 C3 40            TEST    BL,IOCHK_NMI            ; CHANNEL CHECK?
0026  74 03               JZ      NMI_00
0028  E9 0536 R           JMP     CHAN_CHK                ; PROCESS CHANNEL CHECK
002B                     NMI_00:
002B  F6 C3 08            TEST    BL,SYS_SUSP_NMI         ; SYSTEM SUSPEND?
002E  74 03               JZ      NMI_01
0030  E9 0797 R           JMP     SUSPEND                 ; RUN SYSTEM SUSPEND ROUTINE
0033                     NMI_01:
0033  F6 C3 80            TEST    BL,DSKT_NMI             ;
0036  75 37               JNZ     DSKT_RESYNC             ; PROCESS DISKETTE NMI

0038  F6 C3 01            TEST    BL,KBDATA_NMI           ; KEYBOARD DATA READY?
003B  74 03               JZ      NMI_02
003D  E9 025F R           JMP     KYBD_PREP               ; GO TO PROCESS DATA
0040                     NMI_02:
0040  F6 C3 10            TEST    BL,KBCLR_NMI            ; KEYBOARD CLEAR?
0043  74 03               JZ      NMI_03
0045  E9 02F6 R           JMP     KYBD_CLR                ; PROCESS KEYBOARD CLEAR
0048                     NMI_03:
0048  F6 C3 04            TEST    BL,RTC_ALRM_NMI
004B  74 03               JZ      NMIH_EXIT               ; EXIT IF NOT ALARM
004D  E9 0551 R           JMP     RTC_ALARM_NMI           ; OTHERWISE MUST BE ALARM
               ;
               ; KYBD_PREP, KYBD_CLR , DSKT_RESYNC, RTC_ALARM_NMI ROUTINES RETURN HERE
               ;
0050                     NMIH_EXIT:
0050  FA                  CLI
0051  B0 07               MOV     AL,DISABLE_SLEEP+CLOCK_RUN ; DISABLE NMIS
0053  E6 72               OUT     CLOCK_CTL,AL

0055  E4 61               IN      AL,NMI_CNTL             ;
0057  24 F7               AND     AL,NOT DIS_ALARM        ; RE-ENABLE ALARM NMI
0059  E6 61               OUT     NMI_CNTL,AL             ;

005B  E4 62               IN      AL,NMI_SRC              ; READ NMI SOURCE FLAGS
005D  24 DF               AND     AL,NOT TIMER2_SN        ; SAVE ALL BUT TIMER SENSE
005F  75 AC               JNZ     NMI_REDRIVE
0061                     NMIH_OUT:
0061  1F                  POP     DS                      ; RESTORE REGISTERS
0062  07                  POP     ES
0063  5F                  POP     DI
0064  5E                  POP     SI
0065  5D                  POP     BP
0066  5A                  POP     DX
0067  59                  POP     CX
0068  5B                  POP     BX
0069  B0 27               MOV     AL,DISABLE_SLEEP+CLOCK_RUN+GLOBAL_NMI ; ENABLE NMIS
006B  E6 72               OUT     CLOCK_CTL,AL
006D  58                  POP     AX
006E  CF                  IRET
006F                     NMI_FLIH        ENDP


                          INCLUDE DSKTNMI.INC
                         SUBTTL DISKETTE RESYNC DRIVER

                            EXTRN   NEC_OUTPUT:NEAR
                            EXTRN   RESULTS:NEAR
                            EXTRN   DISK_RESET:NEAR
                            EXTRN   GET_PARM:NEAR
                            EXTRN   GET_VECTOR@:NEAR
```

Diskette Resync (DSKT_RESYNC)

```
                 ;**************************************************
                 ;
                 ; MODULE-NAME : DSKT_RESYNC
                 ;
                 ; DESCRIPTIVE-NAME : DISKETTE RESYNC ROUTINES
                 ;
                 ; COPYRIGHT : 7396-917 (C) COPYRIGHT IBM CORP. 1985
                 ;             REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
                 ;
                 ; CHANGE LEVEL: 0.0
                 ;
                 ; FUNCTION : TO POWER ON AND RE-SYNCHRONIZE THE STATE OF THE DISKETTE
                 ;            CONTROLLER WITH THE CURRENT STATE OF THE DISKETTE DRIVES
                 ;
                 ; MODULE SIZE: 660 BYTES
                 ;
                 ; ENTRY CONDITIONS:
                 ;
                 ;     PURPOSE OF ENTRY: TO POWER UP THE FDC WHEN THE FDC IS POWERED OFF
                 ;
                 ;     INPUT CONDITIONS: DS POINTING TO ROM DATA AREA, BP POINTING
                 ;                       TO STACK
                 ;
                 ;  RESTRICTIONS: NONE
                 ;
                 ;  THIS ROUTINE IS INVOKED WHENEVER AN I/O ACCESS IS MADE TO THE FDC
                 ;  REGISTERS 3F2,3F4,3F5 WHEN POWER IS OFF TO THE FDC. IF THE FDC IS
                 ;  POWERED ON BY THIS ROUTINE AND THE RETURN ADDRESS-1 IS POINTING TO AN
                 ;  IN  AL,DX OR OUT DX,AL INSTRUCTION, THE RETURN ADDRESS IS ADJUSTED TO
                 ;  RE-EXECUTE THAT INSTRUCTION UPON NMI EXIT.
                 ;
                 ; EXIT CONDITIONS:
                 ;
                 ;      NORMAL EXIT CONDITIONS: POWER RESTORED TO THE FDC
                 ;                      TRACK COUNTER(S) FOR THE FDC AND ARRAY IN SYNC
                 ;                      NMI'S ENABLED (PORT 77H)
                 ;                      SETUP TO REISSUE THE DISKETTE I/O COMMAND
                 ;
                 ;      ERROR EXIT CONDITIONS:  NONE
                 ;
                 ;      REGISTERS MODIFIED:     AX, BX, CX, DX, ES, AND SI
                 ;
                 ; INTERNAL DATA AREAS / TABLES: BIOS DATA AREA AT SEGMENT 40H
                 ;
                 ; EXTERNALLY REFERENCED ROUTINES: REFER TO EXTRN LIST
                 ;
                 ; EXTERNALLY REFERENCED DATA AREAS:     REFER TO EXTRN LIST
                 ;
                 ;*************************************************************************
006F                        DSKT_RESYNC PROC NEAR
                            ASSUME CS:ROMCODE
                            ASSUME DS:DATA,ES:ABSO

006F  81 7E 0A 03F2         CMP   DXSAVE[BP],DRIVE_CNTL   ; CHECK FOR DX = RESET PORT
0074  75 14                 JNE   DSKT_ON                 ; JUMP IF NOT
0076  F6 46 10 04           TEST  ALSAVE[BP],FDC_RUN      ; RESET BIT ACTIVE IN AL?
007A  74 0E                 JZ    DSKT_ON                 ; YES THEN POWER ON CNTLR
                 ;
                 ; IF NOT A RESET COMMAND TO 3F2H THEN RESET DISKETTE NMI AND EXIT LEVEL
                 ;
007C  B0 B2                 MOV   AL,DSKT_NMI+DSKT_DEGATE+RD_CNTL+CNTL_SEL
007E  E6 77                 OUT   DSKT_CNTL,AL            ;
0080  0C 40                 OR    AL,FDC_PWR              ; TOGGLE POWER ON THEN OFF TO
0082  E6 77                 OUT   DSKT_CNTL,AL            ; RESET NMI BUT DO NOT POWER
0084  24 BF                 AND   AL,NOT FDC_PWR          ; ON CONTROLLER
0086  E6 77                 OUT   DSKT_CNTL,AL            ;
0088  EB C6                 JMP   NMIH_EXIT               ; EXIT LEVEL
                 ;
                 ; SAVE DISKETTE MOTOR STATE BUT DISABLE DRIVE SELECTS,DMA, INTERRUPTS
```

```
                       ;
008A  BA 03F7                DSKT_ON:         MOV    DX,DRIVE_SENSE
008D  EC                       IN    AL,DX                 ; READ DRIVES MOTOR/SEL LINES
008E  8A D8                    MOV   BL,AL                 ; BL <--- STARTING SENSE
0090  24 10                    AND   AL,DRO_MOTOR          ; SAVE DRO MOTOR STATE
0092  F6 C3 08                 TEST  BL,DR1_MOT_SENSE      ; WAS DRIVE 1 MOTOR ON?
0095  74 02                    JZ    DSKT_RS01             ; NO THEN JUMP
0097  0C 20                    OR    AL,DR1_MOTOR          ; OTHERWISE TURN ON DRIVE 1 MOTOR
0099                         DSKT_RS01:
0099  BA 03F2                  MOV   DX,DRIVE_CNTL         ; SET PORT TO 3F2
009C  EE                       OUT   DX,AL                 ; TURN OFF DMA/INT LEAVE MOTOR ON
                       ;
                       ; DISABLE POWER AND ACTIVATE DISKETTE DEGATE MODE
                       ;
009D  B0 B2                    MOV   AL,DSKT_NMI+DSKT_DEGATE+RD_CNTL+CNTL_SEL
009F  E6 77                    OUT   DSKT_CNTL,AL          ; SEND TO THE DISKETTE CONTROL REG

                       ; ENABLE CONTROLLER POWER

00A1  0C 40                    OR    AL,FDC_PWR            ; ENABLE POWER
00A3  E6 77                    OUT   DSKT_CNTL,AL

00A5  F6 06 00A0 R 01          TEST  RTC_WAIT_FLAG,INTERVAL_WAIT ; CHECK FOR PERIODIC INT
00AA  74 06                    JZ    DSKT_RS01A            ;
00AC  E4 61                    IN    AL,NMI_CNTL           ;
00AE  24 F7                    AND   AL,NOT DIS_ALARM      ; ALLOW RTC ALARM INTERRUPTS
00B0  E6 61                    OUT   NMI_CNTL,AL           ;
00B2                         DSKT_RS01A:
00B2  E8 076E R                CALL  NMI_CYCLE             ; CYCLE NMI MASK, RESTORE INT FLAGS

00B5  B9 050A                  MOV   CX,5*MS_DELAY         ; DELAY TO WAIT FOR CONTROLLER
00B8  E2 FE                    LOOP  $
00BA  FA                       CLI
                       ;
                       ; CHECK IF NMI CAUSED BY POWER ON DURING MIDDLE OF CONTROLLER RESET
                       ;

00BB  81 7E 0A 03F2            CMP   DXSAVE[BP],DRIVE_CNTL  ; CHECK FOR DX = RESET PORT
00C0  75 13                    JNE   DSKT_RS02             ; JUMP IF NOT

00C2  8A 46 10                 MOV   AL,ALSAVE[BP]         ; GET ORIGINAL AL REG
00C5  A8 04                    TEST  AL,FDC_RUN            ; IS FDC RESET OFF?
00C7  75 0C                    JNZ   DSKT_RS02             ; YES THEN NOT A RESET
                       ;
                       ; POWER ON REQUEST IN THE MIDDLE OF THE CONTROLLER RESET
                       ;
00C9  0C 04                    OR     AL,FDC_RUN           ; SET RESET BACK TO OFF
00CB  EE                       OUT    DX,AL                ;
00CC  E4 77                    IN     AL,DSKT_CNTL         ; TURN OFF DEGATE
00CE  24 DF                    AND    AL,NOT DSKT_DEGATE   ;
00D0  E6 77                    OUT    DSKT_CNTL,AL         ;
00D2  E9 0207 R                JMP    DSKT_RSEXIT          ; EXIT THE NMI TO COMPLETE
                                                           ; THE RESET SEQUENCE
                       ;
                       ; TAKE OVER AND ACTIVATE DISKETTE INTERRUPT
                       ;
00D5                         DSKT_RS02:
00D5  A0 0040 R                MOV   AL,MOTOR_COUNT        ; SAVE CURRENT MOTOR COUNT
00D8  50                       PUSH  AX                    ; SAVE ON STACK
00D9  C6 06 0040 R FF          MOV   MOTOR_COUNT,0FFH      ; KEEP MOTOR FROM GOING OFF
00DE  2B C0                    SUB   AX,AX
00E0  8E C0                    MOV   ES,AX
00E2  B1 06                    MOV   CL,6                  ; GET INT LEVEL 6 VECTOR LOC
00E4  E8 0000 E                CALL  GET_VECTOR@
                       ;
                       ; SI NOW CONTAINS INTERRUPT VECTOR ADDRESS
                       ;
00E7  26: 8B 04                MOV   AX,ES:[SI]       ; GET VECTOR OFFSET
00EA  50                       PUSH  AX                    ; SAVE
00EB  26: 8B 44 02             MOV   AX,ES:[SI+2]     ; GET VECTOR SEG
00EF  50                       PUSH  AX                    ; SAVE
00F0  26: 8C 4C 02             MOV   ES:[SI+2],CS     ; SET CODE SEGMENT
00F4  26: C7 04 024C R         MOV   WORD PTR ES:[SI],OFFSET DSKT_NMIE ;
00F9  56                       PUSH  SI                    ; SAVE VECTOR ADDRESS
```

```
00FA  E4 21                 IN    AL,INTA01
00FC  24 BF                 AND   AL,0BFH               ; ENABLE DISKETTE INTERRUPTS
00FE  E6 21                 OUT   INTA01,AL
                ;
                ;SETUP TO READ THE TRACK COUNTERS AND DEGATE DISKETTES
                ;
0100  B0 EA                 MOV   AL,DSKT_NMI+FDC_PWR+DSKT_DEGATE+DR0_TRK_SEL+CNTL_SEL
0102  E6 77                 OUT   DSKT_CNTL,AL          ;
0104  E4 77                 IN    AL,DSKT_CNTL          ; READ THE TRACK CNTR FOR DR 0
0106  8A C8                 MOV   CL,AL                 ; CL <-- DRIVE 0 POSITION
0108  B0 E2                 MOV   AL,DSKT_NMI+FDC_PWR+DSKT_DEGATE+CNTL_SEL
010A  E6 77                 OUT   DSKT_CNTL,AL
010C  E4 77                 IN    AL,DSKT_CNTL          ; READ THE TRACK COUNTER FOR DR 1
010E  8A E8                 MOV   CH,AL                 ; CH <--- DRIVE 1 POSITION
0110  B0 F2                 MOV   AL,DSKT_NMI+FDC_PWR+DSKT_DEGATE+RD_CNTL+CNTL_SEL
0112  E6 77                 OUT   DSKT_CNTL,AL
                ;
                ;RESET THE FDC AND ISSUE SPECIFY COMMAND TO THE FDC
                ; CH = DRIVE 1 TRACK POSITION, CL= DRIVE 0, BL = DRIVE_SENSE ON NMI ENTRY
                ;
0114  8A C3                 MOV   AL,BL
0116  24 10                 AND   AL,DR0_MOTOR          ; SAVE DR0 MOTOR STATE
0118  F6 C3 08              TEST  BL,DR1_MOT_SENSE      ; WAS DRIVE 1 MOTOR ON?
011B  74 02                 JZ    NMI_SELMOT            ; NO THEN JUMP
011D  0C 20                 OR    AL,DR1_MOTOR          ; OTHERWISE TURN ON DRIVE 1 MOTOR
011F                      NMI_SELMOT:
011F  0C 08                 OR    AL,FDC_DMA_ENAB       ; ENABLE DMA/INTERRUPTS BIT
0121  F6 C3 20              TEST  BL,DR1_SEL_SENSE      ; DRIVE 1 ACTIVE ?
0124  74 02                 JZ    NMI_SELDR             ; JUMP IF NOT
0126  0C 01                 OR    AL,DR1_SELECT
0128                      NMI_SELDR:
0128  BA 03F2               MOV   DX,DRIVE_CNTL
012B  EE                    OUT   DX,AL                 ; RESET THE ADAPTER
012C  0C 04                 OR    AL,FDC_RUN            ; TURN OFF THE RESET
012E  EB 00                 JMP   $ + 2                 ; TIME DELAY
0130  EE                    OUT   DX,AL                 ;
0131  80 26 003E R 7F       AND   SEEK_STATUS,NOT INT_FLAG ; RESET THE INTERRUPT FLAG

0136  FB                    STI                         ; ALLOW INTERRUPTS
0137  E8 0231 R             CALL  WAIT_INTRPT           ; CHECK FOR FDC TO GENERATE AN INT
013A  BF 01F4 R             MOV   DI,OFFSET RESYNC_OUT2 ; ERROR EXIT FOR NEC_OUTPUT
013D  57                    PUSH  DI
013E  8B F9                 MOV   DI,CX                 ; SAVE TRACK COUNTERS
0140  B9 0004               MOV   CX,4                  ; READ 4 DRIVES STATUS
0143                      NMI_RSETLP:
0143  B4 08                 MOV   AH,READ_INT_STATUS    ; READ INTERRUPT STATUS COMMAND
0145  E8 0000 E             CALL  NEC_OUTPUT            ; SEND IT TO THE FDC
0148  E8 0000 E             CALL  RESULTS               ; GET ST0 AND PCN
014B  E2 F6                 LOOP  NMI_RSETLP            ; LOOP UNTIL COMPLETE
014D                      NMI_SPECIFY:
014D  8B CF                 MOV   CX,DI                 ; RESTORE TRACK COUNTERS
014F  B4 03                 MOV   AH,SPECIFY            ; SPECIFY COMMAND
0151  E8 0000 E             CALL  NEC_OUTPUT            ;
0154  BB 0001               MOV   BX,1                  ; FIRST PARM SENT TO THE FDC
0157  E8 0000 E             CALL  GET_PARM              ;
015A  E8 0000 E             CALL  NEC_OUTPUT
015D  BB 0003               MOV   BX,03H                ; SECOND PARM SENT TO THE FDC
0160  E8 0000 E             CALL  GET_PARM              ;
0163  E8 0000 E             CALL  NEC_OUTPUT
0166  E8 0222 R             CALL  DROP_BUSY             ;
                ;
                ;ISSUE OVERLAPPED RECALS TO DR 0 AND 1 IF INSTALLED
                ;NOTE SEE WHAT TO DO WITH SEEK_STATUS BIT 7-INTERRUPT RECEIVED
                ;
0169  B4 07                 MOV   AH,RECALIBRATE        ; RECAL COMMAND FOR THE FDC
016B  E8 0000 E             CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
016E  B4 00                 MOV   AH,00                 ; RECAL DRIVE 0 FIRST
0170  E8 0000 E             CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
0173  E8 0222 R             CALL  DROP_BUSY             ; CHK FOR FDC BUSY TO DROP FOR OVLP
                ;
                ;SEE IF THERE IS A SECOND DRIVE IF SO RECAL IT
                ;
0176  F7 06 0010 R 0040     TEST  EQUIP_FLAG,40H        ; SEE IF SECOND DRIVE IS INSTALLED
017C  74 0A                 JZ    RECAL_DONE            ; ONLY DRIVE 0 INSTALLED
```

```
017E  B4 07              MOV   AH,RECALIBRATE        ; RECAL COMMAND FOR THE FDC
0180  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
0183  B4 01              MOV   AH,01                 ; RECAL DRIVE 1
0185  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
0188                   RECAL_DONE:
0188  E8 0231 R          CALL  WAIT_INTRPT           ; CHK FOR FDC TO GENERATE AN IRPT
018B  B4 08              MOV   AH,READ_INT_STATUS    ; READ DRIVE STATUS TO THE FDC
018D  E8 0000 E          CALL  NEC_OUTPUT
0190  E8 0000 E          CALL  RESULTS
0193  80 C9 80           OR    CL,80H                ; RECAL STATUS REQUESTED FLAG
0196  F7 06 0010 R 0040  TEST  EQUIP_FLAG,40H        ; SEE IF TWO DRIVES INSTALLED
019C  74 0B              JZ    NMI_SEEK
019E  E8 0231 R          CALL  WAIT_INTRPT           ; CHK FOR FDC TO GENERATE AN IRPT
01A1  B4 08              MOV   AH,READ_INT_STATUS    ; READ DRIVE STATUS TO THE FDC
01A3  E8 0000 E          CALL  NEC_OUTPUT
01A6  E8 0000 E          CALL  RESULTS
                ;
                ;ISSUE OVERLAPPED SEEKS TO DR 0 AND 1 IF INSTALLED
                ;
01A9                   NMI_SEEK:
01A9  80 E1 7F           AND   CL,07FH               ; TURN OFF THE RECAL SWITCH
01AC  B4 0F              MOV   AH,SEEK_CMD           ; SEEK COMMAND FOR THE FDC
01AE  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
01B1  B4 00              MOV   AH,00                 ; SEEK DRIVE 0 FIRST
01B3  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
01B6  8A E1              MOV   AH,CL                 ; DR 0 TRACK TO SEEK TO
01B8  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT TO THE FDC
01BB  E8 0222 R          CALL  DROP_BUSY             ; CHK FOR FDC BUSY TO DROP FOR OVLP
                ;
                ;SEE IF THERE IS A SECOND DRIVE IF SO SEEK
                ;
01BE  F7 06 0010 R 0040  TEST  EQUIP_FLAG,40H        ; SEE IF SECOND DRIVE IS INSTALLED
01C4  74 0F              JZ    SEEK_DONE             ; ONLY DRIVE 0 INSTALLED
01C6  B4 0F              MOV   AH,SEEK_CMD           ; SEEKL COMMAND FOR THE FDC
01C8  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
01CB  B4 01              MOV   AH,01                 ; SEEK  DRIVE 1
01CD  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
01D0  8A E5              MOV   AH,CH                 ; DR 0 TRACK TO SEEK TO
01D2  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT TO THE FDC
01D5                   SEEK_DONE:
01D5  E8 0231 R          CALL  WAIT_INTRPT           ; CHECK FOR FDC TO GENERATE AN IRPT
                ;
                ;GET THE STATUS OF THE SEEK AND THE TRACK VALUE RETURNED FROM THE FDC
                ;
01D8  B4 08              MOV   AH,READ_INT_STATUS    ; READ DRIVE STATUS CMD TO THE FDC
01DA  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
01DD  E8 0000 E          CALL  RESULTS               ; GET STO AND PCN FROM THE FDC
                ;
                ;IF A SECOND DRIVE IS INSTALLED GET ITS RESULTS
                ;
01E0  F7 06 0010 R 0040  TEST  EQUIP_FLAG,40H        ; SEE IF TWO DRIVES
01E6  74 0B              JZ    RESYNC_OUT1
01E8  E8 0231 R          CALL  WAIT_INTRPT           ; CHECK FOR FDC TO GENERATE AN IRPT
01EB  B4 08              MOV   AH,READ_INT_STATUS    ; READ DRIVE STATUS COMMAND TO FDC
01ED  E8 0000 E          CALL  NEC_OUTPUT            ; OUTPUT IT TO THE FDC
01F0  E8 0000 E          CALL  RESULTS               ; GET STO AND PCN FROM FDC FOR DR B
                ;
                ; SETUP TO RETURN TO CALLER THE PROPER VALUE IN PORT 77H
                ;
01F3                   RESYNC_OUT1:
01F3  5E                 POP   SI                    ; DISCARD FDC_OUTPUT ERROR ADDRESS
01F4                   RESYNC_OUT2:
01F4  B0 D2              MOV   AL,DSKT_NMI+FDC_PWR+RD_CNTL+CNTL_SEL ; TURN OFF DEGATE
01F6  E6 77              OUT   DSKT_CNTL,AL
                ;
                ; RESTORE THE USERS INTERRUPT VECTOR AND MOTOR_COUNT
                ;
01F8  FA                 CLI
01F9  5E                 POP   SI                    ; RETRIEVE VECTOR ADDRESS
01FA  58                 POP   AX                    ;
01FB  26: 89 44 02       MOV   ES:[SI+2],AX    ; RESTORE VECTOR SEG ADDRESS
01FF  58                 POP   AX
0200  26: 89 04          MOV   ES:[SI],AX      ; RESTORE VECTOR OFFSET ADDRESS
```

```
0203  58                  POP   AX                   ; RETRIEVE MOTOR COUNT
0204  A2 0040 R           MOV   MOTOR_COUNT,AL
              ;
              ;    ADJUST THE STACK POINTER TO RETURN TO USER
              ;
0207                    DSKT_RSEXIT:
0207  1E                  PUSH  DS                   ;
0208  8B 5E 12            MOV   BX,OFFSAVE[BP]       ; GET RETURN OFFSET  ADDR
020B  8B 46 14            MOV   AX,SEGSAVE[BP]       ; GET RETURN SEGMENT ADDR
020E  8E D8               MOV   DS,AX                ;
0210  4B                  DEC   BX                   ; POINT BACK TO INSTR
0211  8A 07               MOV   AL,[BX]              ; GET INSTRUCTION BYTE
0213  1F                  POP   DS                   ;
0214  3C EC               CMP   AL,IN_INSTR          ; IN AL,DX INSTRUCTION?
0216  74 04               JE    DSKT_RTN_ADJ         ; YES THEN ADJUST RETURN
0218  3C EE               CMP   AL,OUT_INSTR         ; OUT DX,AL INSTRUCTION?
021A  75 03               JNE   RTN_EXITNMI          ; NO THEN DO NOT ADJUST RTN
021C                    DSKT_RTN_ADJ:
021C  FF 4E 12            DEC   OFFSAVE[BP]          ; SETUP TO REISSUE THE I/O FOR USER
021F                    RTN_EXITNMI:
021F  E9 0050 R           JMP   NMIH_EXIT            ; RETURN TO NMI_FLIH
              ;
              ;THIS ROUTINE WILL WAIT FOR THE FDC TO DROP BUSY
              ;
0222                    DROP_BUSY PROC NEAR
0222  51                  PUSH  CX                   ; SAVE REGISTER VALUE

0223  BA 03F4             MOV   DX,FDC_STATUS        ; POINT TO MASTER STATUS PORT
0226  2B C9               SUB   CX,CX                ; TIMING COUNT FOR LOOP
0228                    WAIT_BUSY:
0228  EC                  IN    AL,DX                ; READ THE MASTER PORT
0229  A8 10               TEST  AL,FDC_BUSY          ; TEST FOR BUSY TO DROP
022B  74 02               JZ    FDC_DONE             ; FDC NO LONGER IS BUSY
022D  E2 F9               LOOP  WAIT_BUSY
022F                    FDC_DONE:
022F  59                  POP   CX                   ; RESTORE ORIGINAL VALUE
0230  C3                  RET
0231                    DROP_BUSY ENDP
              ;
              ;THIS ROUTINE WILL WAIT FOR AN INTERRUPT FROM THE FDC
              ;
0231                    WAIT_INTRPT PROC NEAR
0231  51                  PUSH  CX                   ;
0232  53                  PUSH  BX                   ; SAVE BX
0233  B3 04               MOV   BL,4                 ; 2 SECONDS WAIT
0235  2B C9               SUB   CX,CX                ; TIMING COUNT FOR LOOP
0237                    WAIT_INT:
0237  F6 06 003E R 80     TEST  SEEK_STATUS,INT_FLAG   ; WAIT FOR AN INTERRUPT
023C  75 06               JNZ   FDC_INTRPT           ; FDC GENERATED AN INTERRUPT
023E  E2 F7               LOOP  WAIT_INT
0240  FE CB               DEC   BL
0242  75 F3               JNZ   WAIT_INT             ; WAIT ON INTERRUPT
0244                    FDC_INTRPT:
0244  80 26 003E R 7F     AND   SEEK_STATUS,NOT INT_FLAG ; RESET THE INTERRUPT FLAG
0249  5B                  POP   BX                   ; RESTORE BX
024A  59                  POP   CX                   ;
024B  C3                  RET
024C                    WAIT_INTRPT ENDP
024C                    DSKT_RESYNC ENDP
              ;-----------------------------------------------------
              ; DSKT_INTE
              ;      THIS ROUTINE HANDLES THE DISKETTE INTERRUPT
              ;
              ; INPUT:  NONE
              ;
              ; OUTPUT:  THE INTERRUPT FLAG SET IS SEEK_STATUS
              ;-----------------------------------------------------

024C                    DSKT_NMIE       PROC    FAR
024C  1E                  PUSH    DS
024D  50                  PUSH    AX
024E  B8 ---- R           MOV     AX,DATA            ; SET UP DATA SEGMENT
0251  8E D8               MOV     DS,AX
0253  80 0E 003E R 80     OR      SEEK_STATUS,INT_FLAG
```

```
0258  B0 20                 MOV    AL,EOI             ; END OF INTERRUPT MARKER
025A  E6 20                 OUT    INTA00,AL          ; INTERRUPT CONTROL PORT
025C  58                    POP    AX
025D  1F                    POP    DS                 ; RECOVER SYSTEM
025E  CF                    IRET                      ; RETURN FROM INTERRUPT
025F                     DSKT_NMIE       ENDP

                          INCLUDE KYBDNMI.INC
                 ;*********************************************************************
                 ;
                 ; MODULE-NAME:       KYBDNMI.INC
                 ;
                 ; DATE LAST MODIFIED:  09/12/1985
                 ;
                 ; DESCRIPTIVE-NAME:  THIS MODULE CONTAINS THE BIOS KYBD NMI INTERRUPT
                 ;            HANDLER AND THE ASSOCIATED KEY SCAN CODE TRANSLATION
                 ;            ROUTINES.
                 ;
                 ;
                 ; COPYRIGHT:  7396-917 (C) COPYRIGHT IBM CORP.  1985
                 ;             REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
                 ;
                 ; CHANGE LEVEL:  0.0
                 ;
                 ; FUNCTION:  KYBD_PREP - KEYBOARD SCAN CODE NMI ROUTINE
                 ;            KYBD_CLR  - KEYBOARD CLEAR NMI ROUTINE
                 ;            KYBD_XLT  - KEYBOARD SCAN CODE TRANSLATION ROUTINE
                 ;
                 ; MODULE SIZE:  816 BYTES
                 ;
                 ; ENTRY CONDITIONS:
                 ;       PURPOSE OF ENTRY: TO PROCESS A KEYSTROKE
                 ;       INPUT CONDITIONS: N/A
                 ;       RESTRICTIONS: N/A
                 ;       INTERNALLY REFERENCED ROUTINES:
                 ;            KYBD_XLT  SCAN CODE TRANSLATION
                 ;
                 ; EXIT CONDITIONS:
                 ;      NORMAL EXIT CONDITIONS: KEYSTROKE PROCESSED
                 ;      ERROR EXIT CONDITIONS: N/A
                 ;      REGISTERS MODIFIED: NONE
                 ;
                 ; INTERNAL DATA AREAS / TABLES: NONE
                 ;
                 ; EXTERNALLY REFERENCED ROUTINES: REFER TO EXTRN LIST
                 ;
                 ; EXTERNALLY REFERENCED DATA AREAS: REFER TO EXTRN LIST
                 ;
                 ; CHANGE ACTIVITY:   NONE
                 ;
                 ;******************************************************************
                 ;******************************************************************
                 ; K E Y B O A R D   E X T E R N A L S
                 ;******************************************************************
                            EXTRN   KBPAD_TBL:WORD
                            EXTRN   KBPADL:ABS
                            EXTRN   KBFUN_TBL:WORD
                            EXTRN   KBFUNL:ABS
                            EXTRN   KBNMI_TBL:BYTE
                            EXTRN   KB_NOISE:NEAR
```

Keyboard Data NMI (KBNMI_DATA)

```
              ;****************************************************************
              ; LOCAL EQUATES FOR KEYBOARD NMI PROCESSING
              ;****************************************************************
= 007D                KBNMI_DATA      EQU   7DH      ; KEYBOARD NMI SCAN CODE INPUT PORT
= 0000                NO_HOLD         EQU   00H      ; VAL OF P60_HOLD_BYTE FOR NOT Q'D
= 004B                NMI_SLASH_SC    EQU   4BH      ; SCAN CODE FOR / MAKE AT NMI LEVEL
= 0038                P60_ALT_SC      EQU  038H      ; SCAN CODE FOR ALT AT P60 LEVEL
= 0008                NMI_R_ALT_BIT3  EQU   08H      ; USED TO DENOTE LEFT & RIGHT
                                                     ; ALT NMI SC. R-ALT HAS THIS BIT SET.
= 0080                BREAK_BIT       EQU   80H      ; USED TO TST AND SET BRK BIT OF SCs
= 0057                P60_F11_SC      EQU  057H      ; F11 MAKE SC FOR P60

              ;****************************************************************
              ; KYBD_PREP - KEYBOARD SCAN CODE NMI PROCESSING
              ;
              ; DESCRIPTION:
              ;
              ; THIS ROUTINE IS JUMPED TO FROM THE FIRST LEVEL NMI INTERRUPT HANDLER
              ; WHEN THE INTERRUPT IS THE RESULT OF A KEYSTROKE. A CLICK IS SOUNDED
              ; WHEN A KEY SCAN CODE 'MAKE' IS RECEIVED AND THE CLICK STATE IS ON.
              ;
              ;   THE SCAN CODE IS READ THEN TESTS ARE MADE TO SEE IF THE BUFFER IS
              ; EMPTY AND PORT 60 IS NOT ACTIVE. IF THESE 2 CONDITIONS ARE MET A CALL
              ; IS MADE TO THE XLATE ROUTINE WITH THE KEY SCAN CODE IN AL. ELSE THE
              ; SCAN CODE IS BUFFERED IN THE 16 POSITION NMI SCAN CODE BUFFER AT THE
              ; BUFFER TAIL. IF PORT 60 IS NOT ACTIVE A CALL IS MADE TO THE XLATE
              ; ROUTINE WITH THE KEY SCAN CODE TO BE TRANSLATED AT THE BUFFER HEAD.
              ;
              ;   THE FIRST LEVEL NMI INTERRUPT HANDLER SAVES REGISTERS AND MOVES THE
              ; ADDRESS OF "DATA" TO THE DS REGISTER BEFORE IT JUMPS TO THIS ROUTINE.
              ; ON EXIT THE FIRST LEVEL INTERRUPT HANDLER RESTORES REGISTERS.
              ;
              ; INPUT:  DATA IN PORT 7DH
              ;         DS = DATA (40H)
              ;
              ; OUTPUT:  STORE SCAN CODE INTO KB_NMI_BUFFER
              ;          UPDATE KB_NMI_HEAD _TAIL
              ;          SAVE LAST_CLICK_KEY
              ;
              ; INTERNAL ROUTINES:  KYBD_XLT
              ; EXTERNAL ROUTINES:  SPKR_ON, KB_NOISE, SPKR_RESTORE
              ;
              ; REGISTERS MODIFIED: N/A
              ;
              ; INTERRUPTS:  LEFT AS ARE
              ;    KYBD NMI INITIALLY DISABLED, ENABLED BEFORE EXIT
              ;
              ; NORMAL EXIT:  JMP NMIH_EXIT
              ;
              ;****************************************************************
025F                  KYBD_PREP PROC    NEAR

                         ASSUME  DS:DATA
                         ASSUME  ES:DATA
025F  1E                 PUSH    DS
0260  07                 POP     ES                 ; DATA POINTER TO ES

0261                  PREP10:
              ;
              ; ENABLE DISPLAY IF DISABLED BY KEYBOARD INACTIVITY
              ;
0261  F6 06 0016 R 80    TEST    BIOS_STATUS,DSP_BLANKED ; IS LCD BLANKED?
0266  74 0F              JZ      PREP_12
              ;
              ; RESET BLANKED FLAG AND ENABLE DISPLAY
```

```
                        ;
0268  80 26 0016 R 7F      AND     BIOS_STATUS,NOT DSP_BLANKED ; RESET FLAG
026D  B0 00                MOV     AL,0
026F  E6 74                OUT     LCD_INDX,AL
0271  E4 75                IN      AL,LCD_DATA         ; GET DISPLAY CONTROL REG
0273  0C 60                OR      AL,SYNC_ENABLE+PANEL_ENABLE ; TURN ON PANEL
0275  E6 75                OUT     LCD_DATA,AL
                        ;
                        ; SET KEYBOARD ACTIVE FLAG (FOR THIS PERIOD)
                        ;
0277                    PREP_12:
0277  80 0E 0016 R 20      OR      BIOS_STATUS,KYBD_ACTIVE ; INDICATE KEYBOARD IS ACTIVE
027C  80 26 0015 R BF      AND     BAT_STATUS,NOT LOW_BAT_HOLD ; RESET LOW BTRY HOLD FLAG

                        ;------------------------------------------------------------
                        ;
                        ; A KEYBOARD DATA READY NMI OCCURED. READING THE DATA RESETS THE DATA
                        ; READY NMI REQUEST.
                        ;
                        ;------------------------------------------------------------

0281  E4 7D                IN      AL,KBNMI_DATA       ; READ SCAN CODE
                        ;
                        ; DETERMINE IF A KEY STROKE CLICK IS REQUIRED FOR THIS KEY.
                        ; A CLICK IS SOUNDED ON THE MAKE OF ALL KEYS EXCEPT THE MAKES OF KEYS
                        ; THAT ARE GENERATED BY TYPAMATIC MODE.
                        ;
0283  A8 80                TEST    AL,BREAK_BIT        ; IS THIS A 'MAKE' SCAN CODE ?
0285  74 07                JZ      PREP20              ; YES, GO TO CLICK PROCESSING
                        ;
                        ; ALLOW ANY BREAK KEY TO CLEAR LAST_CLICK_KEY
                        ;
0287  C6 06 00BA R 00      MOV     LAST_CLICK_KEY,0    ; YES, CLEAR LAST_CLICK_KEY
028C  EB 23                JMP     SHORT PREP29        ; GO TO BUFFER PROCESSING

028E                    PREP20:
028E  38 06 00BA R         CMP     LAST_CLICK_KEY,AL   ; T-MATIC KEY IF = TO LAST KEY MAKE
0292  74 1D                JE      PREP29              ; IF TYPAMATIC SKIP CLICK
                        ;
                        ; FIRST KEY HIT - NOT TYPAMATIC
                        ;
0294  A2 00BA R            MOV     LAST_CLICK_KEY,AL   ; THIS KEY IS NOW LAST_CLICK_KEY

0297  F6 06 00B4 R 08      TEST    KB_NMI_CNTL,CLICK_ON ; IS AUDIO FEEDBACK ENABLED ?
029C  74 13                JZ      PREP29              ; NO, GO TO BUFFER PROCESSING
029E  BB 0001              MOV     BX,1H               ; DURATION OF CLICK
02A1  B9 0020              MOV     CX,20H              ; FREQUENCY OF CLICK
02A4  8A D0                MOV     DL,AL               ; SAVE KEYSTROKE
02A6  E8 0759 R            CALL    SPKR_ON             ; FORCE SPEAKER ON
02A9  E8 0000 E            CALL    KB_NOISE            ; OUTPUT KEY STROKE CLICK
02AC  E8 0762 R            CALL    SPKR_RESTORE        ; RESTORE SPEAKER STATE
02AF  8A C2                MOV     AL,DL               ; RESTORE AL
                        ;
                        ; PROCESS KEY
                        ;
02B1                    PREP29:
02B1  8A 2E 00BB R         MOV     CH,KB_NMI_HEAD      ; BUFFER HEAD TO CH
02B5  8A 0E 00BC R         MOV     CL,KB_NMI_TAIL      ; BUFFER TAIL TO CL
02B9  FE C1                INC     CL                  ; INCREMENT TAIL POINTER
02BB  80 F9 10             CMP     CL,KB_NMI_BLTH      ; IS CL = END OF BUFFER +1
02BE  72 02                JB      PREP50              ; JUMP IF NOT PAST END OF BUFFER
02C0  B1 00                MOV     CL,0                ; PAST END - SET TO START

02C2                    PREP50:
02C2  3A E9                CMP     CH,CL               ; HAS TAIL REACHED HEAD - OVERFLOW
02C4  75 12                JNE     PREP60              ; JUMP IF NOT OVERFLOW
                        ;
                        ; KEYBOARD NMI BUFFER OVERFLOW
                        ;
02C6  C6 06 00BB R 00      MOV     KB_NMI_HEAD,0       ; BUFFER OVERFLOW - PURGE BUFFER
02CB  C6 06 00BC R 01      MOV     KB_NMI_TAIL,1       ; POSITION FOR OVERFLOW INDICATOR
02D0  C6 06 00BD R FF      MOV     KB_NMI_BUFFER,0FFH  ; OVERFLOW INDICATOR TO BUFFER HEAD
02D5  EB 0F 90             JMP     PREP70
                        ;
```

```
               ; NO OVERFLOW
               ;
02D8                     PREP60:
02D8  B7 00                 MOV     BH,0                 ; CLEAR BH
02DA  8A 1E 00BC R          MOV     BL,KB_NMI_TAIL       ; MOV ORIGINAL TAIL POINTER TO BL
02DE  88 87 00BD R          MOV     KB_NMI_BUFFER[BX],AL ; BUFFER S/C AT ORIGINAL TAIL POINTER
02E2  88 0E 00BC R          MOV     KB_NMI_TAIL,CL       ; UPDATE TAIL POINTER
               ;
               ; ATTEMPT TO DEQUEUE KEYSTROKE AND TRANSLATE
               ;
02E6                     PREP70:
02E6  F6 06 00B4 R 10       TEST    KB_NMI_CNTL,XLATE_BUSY ; IS A XLATION ALREADY IN PROGRESS?
02EB  75 06                 JNZ     PREP90               ; JUMP IF ACTIVE
02ED  E8 0319 R             CALL    KYBD_XLT             ; XLATE AND WRITE TO PORT 60
02F0  E9 0050 R             JMP     NMIH_EXIT            ; EXIT KEYBOARD DATA ROUTINE

02F3                     PREP90:
02F3  E9 0050 R             JMP     NMIH_EXIT            ; GO TO 1ST LEVEL INTERRUPT HANDLER

02F6                     KYBD_PREP ENDP                  ; PROCEDURE END
```

Keyboard Clear NMI (KYBD_CLR)

```
               ;*****************************************************************
               ; KYBD_CLR - KEYBOARD CLEAR NMI PROCESSING
               ;
               ; DESCRIPTION:
               ;
               ; THIS ROUTINE IS JUMPED TO FROM THE FIRST LEVEL NMI INTERRUPT HANDLER
               ; WHEN THE INTERRUPT IS THE RESULT OF PORT 60 BEING CLEARED.
               ;
               ;   A BUFFER EMPTY TEST IS MADE TO DETERMINE IF THERE ARE MORE KEY
               ; SCAN CODES TO BE XLATED. IF THE BUFFER IS NOT EMPTY A CALL IS MADE TO
               ; THE XLATE ROUTINE TO PROCESS THE KEY AT THE BUFFER HEAD.
               ;
               ;   THE FIRST LEVEL NMI INTERRUPT HANDLER SAVES REGISTERS AND MOVES THE
               ; ADDRESS OF "DATA" TO THE DS REGISTER BEFORE IT JUMPS TO THIS ROUTINE.
               ; ON EXIT THE FIRST LEVEL INTERRUPT HANDLER RESTORES REGISTERS.
               ;
               ; INPUT:  P60_HOLD_BYTE
               ;
               ; OUTPUT:  P60_HOLD_BYTE RESET
               ;          P60_LOADED RESET
               ;          CLR_KEYBD OF PORT 61H RESET
               ;
               ; INTERNAL ROUTINES:  KYBD_XLT
               ; EXTERNAL ROUTINES:  NONE
               ;
               ; REGISTERS MODIFIED: N/A
               ;
               ; INTERRUPTS:  LEFT AS ARE
               ;
               ; NORMAL EXIT:  JMP NMIH_EXIT
               ;*****************************************************************
02F6                     KYBD_CLR PROC    NEAR

                            ASSUME  DS:DATA
                            ASSUME  ES:DATA
02F6  1E                    PUSH    DS
02F7  07                    POP     ES                   ; DATA POINTER TO ES
```

```
              ;----------------------------------------------------------------
              ;
              ; THIS NMI IS A RESULT OF INT9 CLEARING THE KEYBOARD INDICATING THAT
              ; THE PORT 60 SCAN CODE HAS BEEN READ AND IS READY TO ACCEPT ANOTHER.
              ;
              ;----------------------------------------------------------------
              ;
              ; KEYPAD /, * AND RIGHT-ALT NMI SCAN CODES GENERATE TWO CODES FOR PORT 60.
              ; KYBD_XLT SENDS THE FIRST (HIDDEN CODE) DIRECTLY AND QUEUES THE SECOND
              ; (P60 SCAN CODE) IN P60_HOLD_BYTE.  IF P60_HOLD_BYTE IS HOLDING, IT MUST
              ; BE SENT TO P60 AS SOON AS P60 IS EMPTY.
              ;
02F8  A0 00B9 R            MOV     AL,P60_HOLD_BYTE     ; GET HOLD BYTE
02FB  3C 00                CMP     AL,NO_HOLD           ; IS A SCAN CODE QUEUED?
02FD  74 09                JE      KC10                 ; NO, JUMP.
02FF  E6 60                OUT     KB_DATA,AL           ; HELD SCAN CODE TO P60
0301  C6 06 00B9 R 00      MOV     P60_HOLD_BYTE,NO_HOLD ; RESET HOLD BYTE
0306  EB 05                JMP     SHORT KC20           ; LEAVE P60_LOADED SET
0308  80 26 00B4 R 7F   KC10:   AND     KB_NMI_CNTL,NOT P60_LOADED ; SET OFF PRT 60 ACT
              ;
              ; RESET CLEAR KEYBOARD NMI SOURCE
              ;
030D  E4 61             KC20:   IN      AL,NMI_CNTL     ; GET NMI CONTROL PORT
030F  24 7F                AND     AL,NOT CLR_KEYBD     ; SET CLEAR KYBD BIT TO 0
0311  E6 61                OUT     NMI_CNTL,AL
              ;
              ; CALL TRANSLATE TO SEE IF ANYTHING IN QUEUE AND TRANSLATE IF SO
              ;
0313  E8 0319 R            CALL    KYBD_XLT             ; XLATE AND WRITE TO PORT 60
0316  E9 0050 R            JMP     NMIH_EXIT            ; GO TO 1ST LEVEL INTERRUPT HANDLER

0319                    KYBD_CLR ENDP                   ; PROCEDURE END
```

Translate Scan Code (KYBD_XLT)

```
              ;****************************************************************
              ; KYBD_XLT - KEYBOARD SCAN CODE XLATE ROUTINE
              ;
              ; DESCRIPTION:
              ;
              ;   IF THE SCAN CODE BUFFER IS EMPTY OR PORT 60H IS LOADED AT ENTRY
              ; THEN THIS ROUTINE ENABLES KEYBOARD NMI AND EXITS.
              ; IF THE BUFFER IS NOT EMPTY THE NEXT ENTRY IN THE
              ; SCAN CODE BUFFER IS PROCESSED. THE SCAN CODE IS TRANSLATED TO THE
              ; PC1 EQUIVALENT AND THE PC1 SCAN CODE IS WRITTEN TO PORT 60. HARDWARE
              ; WILL GENERATE A SET HARDWARE INTERRUPT 1 REQUEST WHEN PORT 60 IS
              ; WRITTEN. IF THE TRANSLATION DOES NOT RESULT IN A PORT 60 LOAD THEN
              ; OTHER KEYS IN THE QUEUE ARE PROCESSED IF ANY.
              ;
              ; INPUT:  KB_NMI_BUFFER
              ;
              ; OUTPUT:  WRITE TO PORT 60H
              ;
              ; INTERNAL ROUTINES:  NUM_STATE_FIX
              ; EXTERNAL ROUTINES:  NMI_CYCLE
              ;
              ; REGISTERS MODIFIED: N/A
              ;
              ; INTERRUPTS: I FLAG RESTORED TO PRE-NMI STATE
              ;   KYBD_NMI ENABLED
              ;
              ; NORMAL EXIT:  RET
              ;****************************************************************
0319                    KYBD_XLT PROC    NEAR
```

```
                         ASSUME  DS:DATA
                         ASSUME  ES:DATA
                ;-----------------------------------------------------------------
                ;
                ;    SCAN CODE BUFFER PROCESSING
                ;
                ;-----------------------------------------------------------------
0319                    XLT10:
0319  E8 0000 E            CALL    DISABLE_NMI         ; DISABLE ALL INTERRUPTS

031C  F6 06 00B4 R 80      TEST    KB_NMI_CNTL,P60_LOADED ; IS PORT 60 CURRENTLY LOADED?
0321  75 0D                JNZ     XLT13               ; YES THEN EXIT THE ROUTINE
                ;
                ;  PORT 60 IS NOT LOADED SO CHECK QUEUE FOR ANY KEYS
                ;
0323  8A 1E 00BB R         MOV     BL,KB_NMI_HEAD      ; BUFFER HEAD DISPLACEMENT
0327  3A 1E 00BC R         CMP     BL,KB_NMI_TAIL      ; IS BUFFER EMPTY
032B  75 06                JNE     XLT14               ; YES, GO TO BUFFER EMPTY ENTRY
                ;
                ; PORT 60 IS NOT LOADED AND NO KEYS IN QUEUE SO FIX NUM_LOCK STATUS
                ; AND EXIT
                ;
032D  E8 0524 R            CALL    NUM_STATE_FIX       ; MAKE NUM_STATE = KEYPAD_STATE
                ;
                ; RE-ENABLE KEYBOARD NMIS AND EXIT THE ROUTINE
                ;
0330  E9 0523 R         XLT13:  JMP     XLT100
                ;
                ; DEQUEUE THE SCAN CODE FROM THE NMI QUEUE
                ;
0333  80 0E 00B4 R 10   XLT14:  OR      KB_NMI_CNTL,XLATE_BUSY ; SET KEY XLATE IN PROCESS
0338  B7 00                MOV     BH,0                ; CLEAR BH - BL IS BUFFER HEAD DISP
033A  8A 87 00BD R         MOV     AL,KB_NMI_BUFFER[BX] ; GET SCAN CODE FROM BUFFER
033E  FE C3                INC     BL                  ; INCREMENT HEAD POINTER
0340  80 FB 10             CMP     BL,KB_NMI_BLTH      ; IS BL = END OF BUFFER +1
0343  72 02                JB      XLT15               ; JUMP IF NOT PAST END OF BUFFER
0345  B3 00                MOV     BL,0                ; PAST END - SET TO START

0347                    XLT15:
0347  88 1E 00BB R         MOV     KB_NMI_HEAD,BL      ; UPDATE BUFFER HEAD
                ;-----------------------------------------------------------------
                ;
                ;  START OF SCAN CODE TRANSLATION
                ;    ENABLE NMIS AND RESTORE INTERRUPT STATE TO PRE-NMI CONDITION
                ;-----------------------------------------------------------------
034B  E8 076E R            CALL    NMI_CYCLE           ; CYCLE NMI MASK, RESTORE INT FLAGS

034E  8A E0                MOV     AH,AL               ; AH <=== ORIGINAL NMI SC
0350  3C FF                CMP     AL,0FFH             ; KEYBOARD OVERRUN ?
0352  75 14                JNE     XLT18               ; NO, GO LOOK FOR FUNCTION KEY
                ;
                ; KEYBOARD OVERRUN DETECTED
                ;
0354  C7 06 00B5 R 0000    MOV     B_PEND1,0           ; RESET BREAK PENDING FLAGS
035A  C7 06 00B7 R 0000    MOV     B_PEND2,0           ; RESET BREAK PENDING FLAGS
0360  80 26 00B4 R BF      AND     KB_NMI_CNTL,NOT FUNC_STATE ; CLEAR FUNCTION STATE
0365  E9 050C R            JMP     XLT90               ; OVERRUN, GO WRITE PORT 60
                ;
                ; CHECK IF SC IS FOR A BASE KEY ONLY
0368                    XLT18:
0368  80 E4 80             AND     AH,BREAK_BIT        ; AH <=== BREAK BIT OF NMI SC
036B  8A D0                MOV     DL,AL               ; DL <=== ORIGINAL NMI SC
036D  24 7F                AND     AL,NOT BREAK_BIT    ; AL <=== NMI SC W/O BREAK BIT
036F  8D 1E 0000 E         LEA     BX,KBNMI_TBL        ; SCAN CODE TABLE - NMI TO PC1
0373  2E: D7               XLAT    KBNMI_TBL           ; XLATE NMI SC TO PC1 SC
0375  A8 80                TEST    AL,80H              ; BASE KEY ONLY?
0377  75 03                JNZ     XLT19               ; NO, JUMP AROUND
0379  E9 04DE R            JMP     XLT60_1             ; YES, JUMP TO BASE KEY PROCESS
                ;-----------------------------------------------------------------
                ;
                ;   IF THIS IS THE FUNCTION KEY, SET/RESET FUNCTION STATE
                ;
                ;-----------------------------------------------------------------
```

```
                   ;
037C  8A C2             XLT19:  MOV     AL,DL               ; AL <=== ORIGINAL NMI SC
037E  24 7F                AND     AL,NOT BREAK_BIT    ; AL <=== NMI SC W/O BREAK BIT
0380  3C 52                CMP     AL,FN_KEY           ; FUNCTION KEY ?
0382  75 15                JNE     XLT23               ; NO, GO PROCESS FUNCTION + KEYS
0384  F6 C4 80             TEST    AH,BREAK_BIT        ; FUNCTION KEY MAKE ?
0387  74 08                JZ      XLT20               ; JUMP IF MAKE
0389  80 26 00B4 R BF      AND     KB_NMI_CNTL,NOT FUNC_STATE ; BREAK, CLEAR FUNCTION STATE
038E  E9 051B R            JMP     XLT95               ; GO TO RETURN
0391                    XLT20:
0391  80 0E 00B4 R 40      OR      KB_NMI_CNTL,FUNC_STATE ; MAKE, SET FUNCTION STATE
0396  E9 051B R            JMP     XLT95               ; GO TO RETURN
                   ;-------------------------------------------------------------------
                   ;
                   ;   FUNCTION + KEYS PROCESSING
                   ;
                   ; IF FUNCTION STATE IS ACTIVE SEARCH THE FUNCTION TABLE FOR A MATCH.
                   ; IF THERE IS BREAK PENDING AND THIS KEY IS A BREAK KEY SEARCH THE
                   ; FUNCTION TABLE FOR A MATCH.
                   ;
                   ;-------------------------------------------------------------------
0399                    XLT23:
0399  F6 06 00B4 R 40      TEST    KB_NMI_CNTL,FUNC_STATE ; FUNCTION KEY STATE ?
039E  75 0C                JNZ     XLT24               ; YES, GO SEARCH TABLE
03A0  83 3E 00B7 R 00      CMP     B_PEND2,0           ; BREAK PENDING ?
03A5  74 1E                JE      XLT27               ; NO, GO CHECK FOR KEYPAD STATE
03A7  F6 C4 80             TEST    AH,BREAK_BIT        ; YES, TEST FOR BREAK KEY
03AA  74 19                JZ      XLT27               ; MAKE KEY - GO CHK KEYPAD STATE
03AC                    XLT24:
03AC  B9 0001              MOV     CX,01H              ; INIT KEYS BREAK PENDING FLAG
03AF  BE 0000 E            MOV     SI,OFFSET KBFUNL    ; TABLE LENGTH
03B2                    XLT25:
03B2  83 EE 02             SUB     SI,2                ; SEARCH FUNCTION TABLE FOR MATCH
03B5  2E: 8B 9C 0000 E     MOV     BX,KBFUN_TBL[SI]    ; BH <=== NMI SC ENTRY
                                                       ; BL <=== P60 EXTENDED SC
03BA  3A C7                CMP     AL,BH               ; CHECK FOR MATCH
03BC  74 0A                JE      XLT28               ; JMP IF MATCH FOUND
03BE  D1 E1                SHL     CX,1                ; CX <===  SEARCH KEY BRK PEND FLAG
03C0  83 FE 00             CMP     SI,0                ; IS SEARCH COMPLETE ?
03C3  75 ED                JNE     XLT25               ; NO, GO CHECK NEXT ENTRY
                   ;
                   ; THIS KEY IS NOT A FUNCTION + KEY.
                   ;
03C5                    XLT27:
03C5  E9 0463 R            JMP     XLT40               ; GO TO KEYPAD SEARCH

                   ;-------------------------------------------------------------------
                   ;
                   ; A MATCH HAS BEEN FOUND IN THE FUNCTION STATE TABLE. THE TABLE CONTAINS
                   ; FOUR, FN + CURSOR KEYS; FN + F1 OR F2; AND FOUR, STATE TOGGLE KEYS.
                   ; THE STATE, CURSOR, AND F1, F2 KEYS ARE:
                   ; STATE KEYS                             CURSOR, F1, F2 KEYS
                   ;  ---------------------------------              -----------------
                   ;  FN  +  SCRL LOCK  = SPEAKER ON/OFF             FN  +  ←  = HOME
                   ;  FN  +  CAPS LOCK  = CLICKER ON/OFF             FN  +  ↑  = PGUP
                   ;  FN  +  NUM LOCK   = KEYPAD ON/OFF              FN  +  →  = END
                   ;  FN  +  ESC        = SYS REQ                    FN  +  ↓  = PGDN
                   ;                                FN  +  F1   = F11
                   ;                                FN  +  F2   = F12
                   ;
                   ;  THE FUNCTION STATE TABLE MAPS FUNCTION     THE FUNCTION STATE TABLE MAPS
                   ;  + STATE KEYS TO A BIT.                     FUNCTION + CURSOR KEYS TO
                   ;                               EXTENDED SCAN CODES.
                   ;  01H = SPEAKER ON/OFF
                   ;  02H = CLICKER ON/OFF                       IT ALSO MAPS FUNCTION + F1,F2
                   ;  04H = KEYPAD ON/OFF                        TO EXTENDED SCAN CODES FOR
                   ;  08H = SYS REQ                              F11,F12.
                   ;-------------------------------------------------------------------
03C8                    XLT28:
03C8  F6 C4 80             TEST    AH,BREAK_BIT        ; PROCESSING BREAK SC?
03CB  74 09                JZ      XLT28A              ; NO, JMP
03CD  85 0E 00B7 R         TEST    CX,B_PEND2          ; BREAK PENDING FOR THIS SPECIFIC SC
03D1  75 03                JNZ     XLT28A              ; YES, JMP TO RESET
03D3  E9 04D6 R            JMP     XLT60               ; NO, PROCESS AS BASE KEY
```

```
03D6                        XLT28A:                             ;
03D6  80 FB 08                 CMP     BL,08H                   ; IS THIS A STATE CHANGE KEY ?
03D9  77 57                    JA      XLT36                    ; JUMP TO CUR KEYS IF NOT STATE CHG
03DB  80 FB 08                 CMP     BL,08H                   ; SYSTEM REQUEST KEY ?
03DE  74 3B                    JE      XLT33                    ; YES, GO PROCESSES SYSTEM REQUEST
03E0  F6 C4 80                 TEST    AH,BREAK_BIT             ; MAKE KEY ?
03E3  74 07                    JZ      XLT29                    ; YES, GO SET BREAK PENDING
03E5  31 0E 00B7 R             XOR     B_PEND2,CX               ; BREAK KEY - RESET BRK PEND FLAG
03E9  E9 051B R                JMP     XLT95                    ; GO TO RETURN
03EC                        XLT29:
03EC  85 0E 00B7 R             TEST    B_PEND2,CX               ; IS BREAK ALREADY PENDING
03F0  75 26                    JNZ     XLT32                    ; TYPAMATIC INVALID FOR STATE KEY
03F2  09 0E 00B7 R             OR      B_PEND2,CX               ; SET KEYS BREAK PENDING FLAG
03F6  80 FB 01                 CMP     BL,01H                   ; SPEAKER STATE KEY ?
03F9  75 09                    JNE     XLT30                    ; NO, GO TEST FOR CLICK STATE
03FB  E4 61                    IN      AL,NMI_CNTL              ; READ SPEAKER CONTROL PORT
03FD  34 04                    XOR     AL,EN_SPKR               ; TOGGLE ENABLE SPEAKER STATE
03FF  E6 61                    OUT     NMI_CNTL,AL              ; WRITE SPEAKER CONTROL PORT
0401  E9 051B R                JMP     XLT95                    ; GO TO RETURN
0404                        XLT30:
0404  80 FB 02                 CMP     BL,02H                   ; CLICK STATE KEY ?
0407  75 08                    JNE     XLT31                    ; NO, GO TOGGLE KEYPAD STATE
0409  80 36 00B4 R 08          XOR     KB_NMI_CNTL,CLICK_ON ; TOGGLE CLICK STATE
040E  E9 051B R                JMP     XLT95                    ; GO TO RETURN
0411                        XLT31:
0411  30 1E 00B4 R             XOR     KB_NMI_CNTL,BL           ; TOGGLE KEYPAD STATE
0415  E8 0524 R                CALL    NUM_STATE_FIX            ; MAKE NUM_STATE RFCT KEYPAD_STATE
0418  E9 051B R             XLT32:  JMP     XLT95               ; GO TO RETURN
                      ;
                      ;  SYSTEM REQUEST KEY PROCESSING
                      ;
041B                        XLT33:
041B  F6 C4 80                 TEST    AH,BREAK_BIT             ; IS THIS A REQUEST KEY MAKE ?
041E  75 09                    JNZ     XLT34                    ; NO, GO RESET BREAK PENDING FLAG
0420  09 0E 00B7 R             OR      B_PEND2,CX               ; YES, SET KEYS BREAK PENDING FLAG
0424  B0 54                    MOV     AL,SYSREQ_MAKE           ; SET AL TO SYSTEM REQUEST MAKE
0426  E9 050C R                JMP     XLT90                    ; GO WRITE TO PORT 60
0429                        XLT34:
0429  31 0E 00B7 R             XOR     B_PEND2,CX               ; BREAK KEY - RESET BRK PEND FLAG
042D  B0 D4                    MOV     AL,SYSREQ_BREAK          ; SET AL TO SYSTEM REQUEST BREAK
042F  E9 050C R                JMP     XLT90                    ; GO WRITE TO PORT 60
                      ;
                      ; IF HERE, MUST BE FN + CURSOR KEY, OR  FN + F1, F2
0432                        XLT36:
0432  8A C3                    MOV     AL,BL                    ; AL <=== PC1 EXTENDED SC FROM TBL
0434  80 FB 57                 CMP     BL,P60_F11_SC            ; FN + F1, F2?
0437  73 15                    JAE     XLT38                    ; YES, JUMP
                      ;
                      ; FUNCTION + CURSOR KEYS  -  HOME, PGUP, END, PGDN
                      ;
0439  F6 C4 80                 TEST    AH,BREAK_BIT             ; CURSOR KEY BREAK ?
043C  74 09                    JZ      XLT37                    ; NO, GO SET BREAK PENDING FLAG
043E  31 0E 00B7 R             XOR     B_PEND2,CX               ; BREAK KEY - RESET BREAK PEND FLAG
0442  0C 80                    OR      AL,BREAK_BIT             ; SET BREAK IN PC1 SCAN CODE
0444  E9 04FB R                JMP     XLT80                    ; GO RESET PORT 60 KEYPAD STATE
0447                        XLT37:
0447  09 0E 00B7 R             OR      B_PEND2,CX               ; SET KEYS BREAK PENDING FLAG
044B  E9 04FB R                JMP     XLT80                    ; GO RESET PORT 60 KEYPAD STATE
                      ;
                      ; FUNCTION + F1  (F11)     P60-SC:       F11 MAKE = 57H   F11 BREAK = D7H
                      ; FUNCTION + F2  (F12)                   F12 MAKE = 58H   F12 BREAK = D8H
                      ;
044E                        XLT38:                              ;
044E  F6 C4 80                 TEST    AH,BREAK_BIT             ; MAKE KEY?
0451  74 09                    JZ      XLT38_1                  ; YES, JUMP
0453  0A C4                    OR      AL,AH                    ; CHANGE TO BREAK SC
0455  31 0E 00B7 R             XOR     B_PEND2,CX               ; BREAK KEY - RESET BREAK PEND FLAG
0459  E9 050C R                JMP     XLT90                    ; WRITE TO P60
045C                        XLT38_1:                            ;
045C  09 0E 00B7 R             OR      B_PEND2,CX               ; MAKE KEY - SET BREAK PEND FLAG
0460  E9 050C R                JMP     XLT90                    ; WRITE TO P60
```

```
              ;-------------------------------------------------------------------
              ;
              ;     KEYPAD KEYS PROCESSING
              ;
              ; DETERMINE IF THE KEYPAD TABLE IS TO BE SEARCHED. THE KEYPAD TABLE
              ; SEARCH/NO SEARCH DECISION IS MADE USING THE FOLLOWING RULES:
              ; THE SEARCH IS ALWAYS MADE IF THERE ARE ANY BREAK PENDING FLAGS AND
              ; THIS KEY IS A BREAK KEY. THE FUNCTION AND KEYPAD STATES ARE MUTUALLY
              ; EXCLUSIVE. THE SEARCH IS PERFORMED IF EITHER STATE IS ACTIVE BUT
              ; NOT BOTH STATES ACTIVE.
              ;
              ;-------------------------------------------------------------------
0463                        XLT40:
0463  F6 06 00B4 R 44          TEST    KB_NMI_CNTL,FUNC_STATE+KEYPAD_STATE
0468  74 0E                    JZ      XLT41               ; BOTH OFF - GO TEST BREAK PENDING
046A  F6 06 00B4 R 40          TEST    KB_NMI_CNTL,FUNC_STATE ; ONE OR BOTH ARE ON
046F  74 13                    JZ      XLT42               ; OFF - ONLY ONE ON - GO SEARCH
0471  F6 06 00B4 R 04          TEST    KB_NMI_CNTL,KEYPAD_STATE ; ONE OR BOTH ARE ON
0476  74 0C                    JZ      XLT42               ; OFF - ONLY ONE ON - GO SEARCH
              ;
              ; BOTH FUNC_STATE & KEYPAD_STATE ARE OFF OR ON....KEYPAD INACTIVE
0478                        XLT41:
0478  83 3E 00B5 R 00          CMP     B_PEND1,0           ; BREAK PENDING ?
047D  74 57                    JE      XLT60               ; NO, SKIP KEYPAD SEARCH
047F  F6 C4 80                 TEST    AH,BREAK_BIT        ; YES, TEST FOR BREAK KEY
0482  74 52                    JZ      XLT60               ; MAKE KEY - SKIP KEYPAD SEARCH
0484                        XLT42:
0484  B9 0001                  MOV     CX,0001H            ; CX <=== SEARCH KEY BRK PEND FLAG
0487  BE 0000 E                MOV     SI,OFFSET KBPADL    ; TABLE LENGTH
048A                        XLT44:
048A  83 EE 02                 SUB     SI,2                ; SEARCH KEYPAD TABLE FOR MATCH
048D  2E: 8B 9C 0000 E         MOV     BX,KBPAD_TBL[SI]    ; GET TABLE ENTRY
0492  3A C7                    CMP     AL,BH               ; CHECK FOR MATCH
0494  74 09                    JE      XLT50               ; JMP IF MATCH FOUND
0496  D1 E1                    SHL     CX,1                ; CX <=== SEARCH KEY BRK PEND FLAG
0498  83 FE 00                 CMP     SI,0                ; IS SEARCH COMPLETE ?
049B  75 ED                    JNE     XLT44               ; NO, GO CHECK NEXT ENTRY
              ;
              ; THIS KEY IS NOT A KEYPAD KEY.
              ;
049D  EB 37                    JMP     SHORT XLT60         ; GO XLATE NORMAL

              ;
              ; A MATCH HAS BEEN FOUND IN THE KEYPAD TABLE. SET/RESET THE KEYS BREAK
              ; PENDING FLAG THEN WRITE TO PORT 60.  SEND HIDDEN CODE FIRST FOR / OR *.
              ;
049F                        XLT50:
049F  F6 C4 80                 TEST    AH,BREAK_BIT        ; BREAK KEY ?
04A2  75 06                    JNZ     XLT52               ; YES, GO RESET BREAK PENDING
04A4  09 0E 00B5 R             OR      B_PEND1,CX          ; SET KEYS BREAK PENDING FLAG
04A8  EB 0A                    JMP     SHORT XLT54
04AA                        XLT52:
04AA  85 0E 00B5 R             TEST    CX,B_PEND1          ; BREAK PENDING FOR THIS SPECIFIC SC
04AE  74 26                    JZ      XLT60               ; NO, JMP TO PROCESS AS BASE KEY
04B0  31 0E 00B5 R             XOR     B_PEND1,CX          ; RESET KEYS BREAK PENDING FLAG
04B4  0A DC                 XLT54:  OR      BL,AH          ; USE BREAK BIT FROM NMI SC
04B6  8A C3                    MOV     AL,BL               ; AL <=== PC1 EXTENDED SC FROM TBL
04B8  80 FF 4B                 CMP     BH,NMI_SLASH_SC     ; / OR * ?
04BB  72 05                    JB      XLT56               ; NO, JUMP OUT
04BD  A2 00B9 R                MOV     P60_HOLD_BYTE,AL    ; QUEUE P60 SC
04C0  B0 E0                    MOV     AL,HIDN_CODE_E0     ; SEND HIDDEN CODE TO P60
              ;-------------------------------------------------------------------
              ;
              ; THE PORT 60 KEYPAD STATE IS SET OR RESET USING THE FOLLOWING RULES:
              ; THIS KEY HAS BEEN TRANSLATED ASSUMING THE KEYPAD STATE IS ACTIVE.
              ; A PORT 60 SHIFT STATE TEMPORARILY TOGGLES THE KEYPAD STATE. IF THE
              ; PORT 60 SHIFT STATE IS NOT SET THE P60 KEYPAD STATE IS SET AND THE
              ; PC1 KEYPAD SCAN CODE IS WRITTEN TO PORT 60. IF THE PORT 60 SHIFT STATE
              ; IS SET THE P60 KEYPAD STATE IS RESET (IT WILL REVERT TO THE SET STATE IN
              ; THE PORT 60 PROCESSING) AND THE PC1 SCAN CODE IS WRITTEN TO PORT 60.
              ;
              ;-------------------------------------------------------------------
04C2                        XLT56:
04C2  80 0E 0017 R 20          OR      KB_FLAG,NUM_STATE   ; SET KEYPAD STATE
```

```
04C7  F6 06 0017 R 03      TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; PORT 60 SHIFT STATE ?
04CC  74 3E                JZ      XLT90                ; NO, GO WRITE TO PORT 60
04CE  80 36 0017 R 20      XOR     KB_FLAG,NUM_STATE    ; RESET KEYPAD STATE
04D3  EB 37 90             JMP     XLT90                ; GO WRITE TO PORT 60

                ;-----------------------------------------------------------------
                ;
                ;    BASE KEYS PROCESSING
                ;
                ; SCAN CODE IS NOT IN SPECIAL TABLES. XLATE TO PC1 SCAN CODES.
                ; AFTER TRANSLATION THE CURSOR, INSERT, AND DELETE KEYS ARE DETECTED
                ; AND ROUTED TO THE ROUTINE THAT GUARANTEES THE PORT 60 KEYPAD STATE
                ; IS INACTIVE. ALL OTHER KEYS ARE WRITTEN TO PORT 60.
                ;
                ;-----------------------------------------------------------------
                ;
                ; NOT A FN+KEY OR KEYPAD KEY -- GET KEY'S P60 SC
04D6                    XLT60:
                                                        ; AL <=== NMI SC W/O BREAK BIT
04D6  8D 1E 0000 E         LEA     BX,KBNMI_TBL         ; SCAN CODE TABLE - NMI TO PC1
04DA  2E: D7               XLAT    KBNMI_TBL            ; XLATE NMI SCAN CD TO PC1 SCAN CD
                                                        ; AL <=== PC1 SC FROM TBL
04DC  24 7F                AND     AL,7FH               ; CLEAR TYPE BIT
04DE                    XLT60_1:
04DE  3C 48                CMP     AL,48H               ; CURSOR, INSERT, DELETE  ?
04E0  73 17                JAE     XLT65                ; YES, JUMP TO PROCESS THESE KEYS
04E2  3C 38                CMP     AL,P60_ALT_SC        ; ALT KEY?
04E4  75 0C                JNE     XLT62                ; NO, JUMP AROUND
                ;
                ; ALT KEY
04E6  F6 C2 08             TEST    DL,NMI_R_ALT_BIT3    ; R_ALT KEY?
04E9  74 07                JZ      XLT62                ; NO, JUMP OUT
04EB  0A C4                OR      AL,AH                ; USE NMI SC BREAK BIT
04ED  A2 00B9 R            MOV     P60_HOLD_BYTE,AL     ; QUEUE P60 ALT SC
04F0  B0 E0                MOV     AL,HIDN_CODE_E0      ; SEND HIDDEN CODE TO P60
                ;
                ; NOT CURSOR, INSERT, DELETE
04F2                    XLT62:
04F2  0A C4                OR      AL,AH                ; USE NMI SC BREAK BIT
04F4  E8 0524 R            CALL    NUM_STATE_FIX        ; MAKE NUM_STATE RLCT KEYPAD_STATE
04F7  EB 13                JMP     SHORT XLT90          ; GO WRITE TO PORT 60
                ;
                ; CURSOR, INSERT, OR DELETE KEY
04F9                    XLT65:
04F9  0A C4                OR      AL,AH                ; USE NMI SC BREAK BIT

                ;-----------------------------------------------------------------
                ;
                ; THE CURSOR, FUNCTION+CURSOR, INSERT, AND DELETE KEYS ARE NO LONGER IN
                ; THE KEYPAD AREA OF THE KEYBOARD. WHEN THEIR SCAN CODES ARE WRITTEN TO
                ; PORT 60, THE PORT 60 KEYPAD STATE MUST BE INACTIVE.
                ; A PORT 60 SHIFT STATE TEMPORARILY TOGGLES THE KEYPAD STATE. IF THE
                ; PORT 60 SHIFT STATE IS NOT SET THE P60 KEYPAD STATE IS RESET AND THE
                ; PC1 SCAN CODE IS WRITTEN TO PORT 60. IF THE PORT 60 SHIFT STATE
                ; IS SET THE P60 KEYPAD STATE IS SET (IT WILL REVERT TO THE RESET STATE IN
                ; THE PORT 60 PROCESSING) AND THE PC1 SCAN CODE IS WRITTEN TO PORT 60.
                ;-----------------------------------------------------------------
04FB                    XLT80:
04FB  80 0E 0017 R 20      OR      KB_FLAG,NUM_STATE    ; SET P60 KEYPAD STATE
0500  F6 06 0017 R 03      TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; PORT 60 SHIFT STATE ?
0505  75 05                JNZ     XLT90                ; YES, GO WRITE TO PORT 60
0507  80 36 0017 R 20      XOR     KB_FLAG,NUM_STATE    ; RESET P60 KEYPAD STATE
                ;
                ; THE PC1 SCAN CODE IN AL IS WRITTEN TO PORT 60
                ;
050C                    XLT90:
050C  FA                   CLI                          ; DISABLE INTERRUPTS
050D  80 0E 00B4 R 80      OR      KB_NMI_CNTL,P60_LOADED ; SET ON PORT 60 ACTIVE STATUS
0512  E6 60                OUT     KB_DATA,AL           ; WRITE PC1 SCAN CODE TO PORT 60
0514  80 26 00B4 R EF      AND     KB_NMI_CNTL,NOT XLATE_BUSY ; RESET BUSY FLAG
0519  EB 08                JMP     SHORT XLT100
051B                    XLT95:
051B  80 26 00B4 R EF      AND     KB_NMI_CNTL,NOT XLATE_BUSY ; RESET BUSY FLAG
0520  E9 0319 R            JMP     XLT10                ; REDRIVE XLATE ROUTINE
```

```
              ;                                      ; IN QUEUE
              ; RETURN TO CALLER
              ;
0523                 XLT100:
0523  C3                RET                          ; RETURN

0524                 KYBD_XLT ENDP                   ; PROCEDURE END

              ;-----------------------------------------------------------------------
              ;
              ; SUBROUTINE  NUM_STATE_FIX
              ;   THIS ROUTINE MAKES THE NUM_STATE FLAG REFLECT THE KEYPAD_STATE FLAG.
              ;
              ;   INPUT:  NONE     OUTPUT:  NUM_STATE SET OR RESET
              ;-----------------------------------------------------------------------
0524                 NUM_STATE_FIX  PROC  NEAR
0524  80 26 0017 R DF   AND     KB_FLAG,NOT NUM_STATE ; TURN OFF NUM_STATE
0529  F6 06 00B4 R 04   TEST    KB_NMI_CNTL,KEYPAD_STATE ; IS KEYPAD IN ACTIVE STATE?
052E  74 05             JZ      NSF_RET              ; NO, LEAVE NUM_STATE OFF
0530  80 0E 0017 R 20   OR      KB_FLAG,NUM_STATE    ; YES, TURN NUM_STATE ON
0535                 NSF_RET:
0535  C3                RET
0536                 NUM_STATE_FIX  ENDP
              ;
              ; THIS ROUTINE IS ACTIVATED WHEN THE NMI_FLIH DETECTES AN I/O CHECK
              ; IT CAUSES A PARITY CHECK ICON TO BE DISPLAYED ON THE ACTIVE DISPLAY AFTER
              ; THE DISPLAY SCREEN IS CLEARED. SUSPEND IS DISABLED AND THE KEYBOARD IS
              ; WAITING FOR A FUNCTION CONTROL DELETE KEY SEQUENCE.
              ;
0536                 CHAN_CHK        PROC    NEAR
0536  B4 00           MOV     AH,0                   ; INIT AND SET MODE FOR VIDEO
0538  A0 0049 R       MOV     AL,CRT_MODE
053B  CD 10           INT     10H                    ; CALL VIDEO_IO PROCEDURE
053D  2B D2           SUB     DX,DX                  ; SET ROW/COLUMN FOR ICON
053F  BD 0000 E       MOV     BP,OFFSET PAR_CHK      ; GET ADDRESS OF PARITY CHECK
0542  0E              PUSH    CS
0543  07              POP     ES                     ; ES:BP CONTAIN ICON ADDRESS
0544  E8 0000 E       CALL    ICON_PR                ; DISPLAY CHECK ICON
0547  E4 7F           IN      AL,PWR_STAT            ; DISABLE SUSPEND NMI
0549  24 F3           AND     AL,NOT EN_SUS_NMI+HDWR_RESET ;
054B  E6 7F           OUT     PWR_STAT,AL
054D                 CHAN_STOP:
054D  E4 7D           IN      AL,7DH                 ; KEEP READING KEYBOARD
054F  EB FC           JMP     SHORT CHAN_STOP        ; FOR RESET SEQUENCE

0551                 CHAN_CHK        ENDP
```

Real-Time Clock NMI (RTC_ALARM_NMI)

```
              ;**************************************************************
              ; RTC_ALARM_NMI -- REAL TIME CLOCK INTERRUPT HANDLER
              ; THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM
              ; THE NON-VOLATILE TIMER.  INPUT FREQUENCY IS 1.024 KHZ
              ; OR APPROXIMATELY 1024 INTERRUPTS EVERY SECOND FOR THE
              ; PERIODIC INTERRUPT.  FOR THE ALARM FUNCTION, AN INTERRUPT WILL
              ; OCCUR AT THE DESIGNATED TIME.
              ;
```

```
                ; THE INTERRUPT IS ENABLED ONLY WHEN EVENT OR ALARM FUNCTIONS
                ; ARE ACTIVE OR WHEN SYSTEM IS SLEEPING.
                ; FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE
                ; WAIT COUNTER AND WHEN IT EXPIRES WILL TURN ON THE HIGH ORDER
                ; BIT OF THE DESIGNATED FLAG.
                ; FOR THE ALARM INTERRUPT, THE USER ROUTINE WILL BE INVOKED
                ; ON THE NEXT TIMER 0 INTERRUPT THROUGH INT 4AH.
                ;
                ; ON ENTRY THE BP REGISTER POINTS TO THE LAST STACK POSITION
                ;--------------------------------------------------------------
0551                    RTC_ALARM_NMI   PROC    NEAR
0551  B4 0B                MOV     AH,RTC_MODE         ; GET INTERRUPT MODE REGISTER
0553  E8 0747 R            CALL    GET_RTC_NMI         ;
0556  8A F8                MOV     BH,AL               ; SAVE MODE
0558  FE C4                INC     AH                  ; GET INTERRUPT STATUS REGISTER
055A  E8 0747 R            CALL    GET_RTC_NMI         ;
055D  22 C7                AND     AL,BH               ; MASK SOURCE WITH ENABLES
055F  50                   PUSH    AX                  ; SAVE INTERRUPT CONDITIONS
0560  A8 40                TEST    AL,PI_INT           ; CHECK FOR PERIODIC INTERRUPT
0562  74 25                JZ      RTC_INT_9           ; NO - GO AROUND
                ;
                ; PERIODIC INTERRUPT HAS BEEN RECEIVED
                ;
0564  81 2E 009C R 03D0    SUB     RTC_LOW,0976        ; DECREMENT ELAPSED TIME COUNT
056A  73 1D                JNC     RTC_INT_9           ; SKIP HIGH BYTE
056C  83 1E 009E R 00      SBB     RTC_HIGH,0
0571  73 16                JAE     RTC_INT_9           ; WAIT TELL ROLLS FROM 0
                ;
                ; USERS ELAPSED TIME EVENT HAS OCCURRED
                ;
0573  B4 0B                MOV     AH,RTC_MODE         ; GET INTERRUPT MODE CONTROL
0575  E8 0747 R            CALL    GET_RTC_NMI
0578  24 BF                AND     AL,NOT PIE_ENABLE   ; RESET PERIODIC INTERRUPT
057A  E8 0750 R            CALL    PUT_RTC_NMI
                ;
                ; CLEAR EVENT_WAIT ACTIVE FLAG AND SET USERS EVENT COMPLETE FLAG
                ;
057D  80 26 00A0 R FE      AND     RTC_WAIT_FLAG,NOT INTERVAL_WAIT ; RESET INT WAIT FLAG
0582  C5 3E 0098 R         LDS     DI,DWORD PTR USER_FLAG ; DS:DI <-- USERS_FLAG ADDRESS
0586  C6 05 80             MOV     BYTE PTR[DI],POSTED ; SET USERS FLAG

0589                    RTC_INT_9:
0589  58                   POP     AX                  ; RETRIEVE INTERRUPT SOURCE
058A  A8 20                TEST    AL,AL_INT           ; TEST FOR ALARM INTERRUPT
058C  74 05                JZ      RTC_INT_10          ; NO - GO AROUND
                ;
                ; RTC ALARM TIME HAS BEEN REACHED - SET BIOS_STATUS FLAG TO ALARM_PEND
                ; THIS WILL CAUSE AN INT 4AH TO BE EXECUTED ON THE NEXT TIMERO INTERRUPT
                ;
058E  80 0E 00A0 R 02      OR      RTC_WAIT_FLAG,ALARM_PEND ; SET INT 4AH CALL PENDING
                ;
                ; CHECK FOR UPDATE IN PROGRESS INTERRUPT (EVERY 1 SECOND)
                ;
0593                    RTC_INT_10:
0593  A8 10                TEST    AL,UE_INT           ; UPDATE ENDED INTERRUPT?
                           JFZ     RTC_INT_14          ; JUMP IF NOT
0595  75 03                JNZ     $+5                  ; IF NOT ZERO JUMP AROUND JUMP
0597  E9 0647 R            JMP     RTC_INT_14           ; ELSE TAKE A LONG JUMP

059A  E8 064A R            CALL    LOW_BAT_CHK         ; CHECK AND DISPLAY LOW BATTERY
                                                       ; MESSAGE IF NECESSARY
059D  E4 7F                IN      AL,PWR_STAT         ; GET POWER STATUS
059F  A8 40                TEST    AL,EXT_PWR          ; ARE WE ON EXTERNAL PWR?
                           JFNZ    RTC_INT_14          ; JUMP IF SO
05A1  74 03                JZ      $+5                  ; IF NOT NOT ZERO JUMP AROUND JUMP
05A3  E9 0647 R            JMP     RTC_INT_14           ; ELSE TAKE A LONG JUMP
```

```
05A6  F6 06 0016 R 20      TEST    BIOS_STATUS,KYBD_ACTIVE ; HAS KEYBOARD BEEN ACTIVE?
05AB  74 1A                JZ      RTC_INT_12          ; JUMP IF KEYBOARD NOT ACTIVE
                  ;
                  ; RELOAD INACTIVITY COUNTS FROM RTC RAM TO SYSTEM RAM
                  ;
05AD  80 26 0016 R DF      AND     BIOS_STATUS,NOT KYBD_ACTIVE ; RESET KBD ACTIVE FLAG
05B2  1E                   PUSH    DS                  ; SET ES TO DATA
05B3  07                   POP     ES
05B4  BF 0067 R            MOV     DI,OFFSET DSP_BLANK_CTR
05B7  B4 19                MOV     AH,RTC_LCD_INACT    ; START WITH LCD INACTIVITY COUNT
05B9  B9 0004              MOV     CX,4                ; TRANSFER 4 BYTES

05BC                    RTC_INT_11:
05BC  E8 0747 R            CALL    GET_RTC_NMI
05BF  AA                   STOSB                       ; STORE IN COUNTER SAVE AREA
05C0  FE C4                INC     AH                  ; GET NEXT BYTE
05C2  E2 F8                LOOP    RTC_INT_11
05C4  E9 0647 R            JMP     RTC_INT_14          ; EXIT LEVEL
                  ;
                  ; CHECK FOR DISKETTE MOTORS ON AND IF SO THEN RELOAD COUNT
                  ;
05C7                    RTC_INT_12:
05C7  BA 03F7              MOV     DX,DRIVE_SENSE      ; CHECK DISKETTE MOTOR STATUS
05CA  EC                   IN      AL,DX               ;
05CB  A8 78                TEST    AL,DR0_SEL_SENSE+DR1_SEL_SENSE+DR0_MOT_SENSE+DR1_MOT_SENSE
05CD  74 07                JZ      RTC_INT_12_1        ; JUMP IF NOT

05CF  80 0E 0016 R 20      OR      BIOS_STATUS,KYBD_ACTIVE ; CAUSE RELOAD OF COUNTERS
05D4  EB 71                JMP     SHORT RTC_INT_14    ; EXIT
                  ;
                  ; CHECK DISPLAY BLANK COUNT
                  ;
05D6                    RTC_INT_12_1:
05D6  83 3E 0067 R 00      CMP     DSP_BLANK_CTR,0
05DB  74 06                JE      RTC_INT_13

05DD  FF 0E 0067 R         DEC     DSP_BLANK_CTR
05E1  74 36                JZ      DSP_BLANK
                  ;
                  ; CHECK SYSTEM POWER OFF COUNT
                  ;
05E3                    RTC_INT_13:
05E3  83 3E 0069 R 00      CMP     SYS_OFF_CTR,0
05E8  74 5D                JE      RTC_INT_14
05EA  FF 0E 0069 R         DEC     SYS_OFF_CTR
05EE  74 46                JZ      DEACT_SYSTEM
05F0  83 3E 0069 R 1E      CMP     SYS_OFF_CTR,30      ; AT THE 30 SECOND MARK?
05F5  75 50                JNE     RTC_INT_14          ; NO THEN EXIT
                  ;
                  ; 30 SECONDS TO POWER OFF SO RING ALARM
                  ;
05F7  E8 076E R            CALL    NMI_CYCLE           ; CYCLE NMI AND RESTORE INT FLAGS
05FA  E8 0759 R            CALL    SPKR_ON             ; TURN ON SPEAKER
05FD  BB 003C              MOV     BX,60               ; TONE LENGTH (60 MSECS)
0600  B9 0081              MOV     CX,129              ; 1/2 CYCLE FREQUENCY FOR 1KHZ TONE
0603  E8 0000 E            CALL    KB_NOISE            ; SOUND BEEPER
0606  B9 050A              MOV     CX,5*MS_DELAY       ; DELAY BETWEEN SOUNDS
0609  E2 FE                LOOP    $
060B  BB 0030              MOV     BX,48               ; TONE LENGTH (60 MSECS)
060E  B9 00A1              MOV     CX,161              ; 1/2 CYCLE FREQ FOR 800 HZ TONE
0611  E8 0000 E            CALL    KB_NOISE            ; SOUND BEEPER
0614  E8 0762 R            CALL    SPKR_RESTORE        ; RESTORE SPEAKER STATE
0617  EB 2E                JMP     SHORT RTC_INT_14    ; EXIT
                  ;
                  ; LCD/CRT MUST BE BLANKED
                  ;
0619                    DSP_BLANK:
0619  B0 00                MOV     AL,0                ;
061B  E6 74                OUT     LCD_INDX,AL         ;
061D  E4 75                IN      AL,LCD_DATA         ;
061F  24 BF                AND     AL,NOT PANEL_ENABLE ; TURN OFF PANEL
0621  E6 75                OUT     LCD_DATA,AL         ;
```

```
0623  E8 076E R             CALL    NMI_CYCLE           ; CYCLE NMI , RESTORE INT FLAGS
0626  B9 64C8               MOV     CX,100*MS_DELAY     ;
0629  E2 FE                 LOOP    $                   ;
062B  24 DF                 AND     AL,NOT SYNC_ENABLE  ; TURN OFF SYNCS FOR POWER
062D  E6 75                 OUT     LCD_DATA,AL         ;
062F  80 0E 0016 R 80       OR      BIOS_STATUS,DSP_BLANKED ; SET DISPLAY BLANKED STATUS
0634  EB 11                 JMP     SHORT RTC_INT_14    ; EXIT
                   ;
                   ; TURN OFF SYSTEM
                   ;
0636                     DEACT_SYSTEM:
0636  80 0E 0016 R 40       OR      BIOS_STATUS,F_RESUME ; INDICATE FORCED RESUME MODE
063B  E4 7F                 IN      AL,PWR_STAT         ; GET POWER STAT/CNTL REG
063D  24 F7                 AND     AL,NOT HDWR_RESET   ; TURN OFF RESET FLAG
063F  0C 04                 OR      AL,EN_SUS_NMI       ; ENABLE SYSTEM SUSPEND NMI
0641  E6 7F                 OUT     PWR_STAT,AL
0643  0C 02                 OR      AL,REQ_POFF         ; REQUEST POWER OFF
0645  E6 7F                 OUT     PWR_STAT,AL
                   ;
                   ; EXIT FROM RTC INTERRUPT
                   ;
0647                     RTC_INT_14:
0647  E9 0050 R             JMP     NMIH_EXIT           ; RETURN TO FIRST LEVEL HANDLER

064A                     RTC_ALARM_NMI   ENDP
```

Low Battery Check (LOW_BAT_CHK)

```
                   ;**************************************************************************
                   ;       LOW BATTERY CHECK
                   ; THIS ROUTINE CHECKS FOR A LOW BATTERY CONDITION AND IF DETECTED WILL ISSUE
                   ; A WARNING. THE LOW BATTERY WARNING WILL OCCUR AFTER TWO CONSECUTIVE LOW
                   ; BATTERY SENSES, NO EXTERNAL POWER SUPPLIED AND THE LOW BATTERY WARNING
                   ; FLAG IS ENABLED IN SYSTEM PROFILE. THE WARNING ISSUED WILL SOUND THREE
                   ; SHORT BEEPS, FLASH THE SCREEN ON AND OFF AT AN ONE SECOND INTERVAL AND
                   ; EXECUTE A PAUSE. THE PAUSE WILL HOLD UP ALL MAIN LEVEL PROCESSING AND
                   ; LOCK OUT NON-INTERRUPT DRIVEN PROCESSING. ONCE THE USER ACKNOWLEDGES THE
                   ; WARNING, EITHER BY PRESSING A KEY OR APPLYING EXTERNAL POWER, THE SCREEN
                   ; FLASHING WILL STOP AND MAIN LEVEL PROCESSING WILL CONTINUE.
                   ; AT SUCCESSIVE TWO MINUTE INTERVALS, THE LOW BATTERY WARNING WILL BE
                   ; REISSUED IF THE LOW BATTERY CONDITION STILL EXISTS OR EXTERNAL POWER HAS
                   ; NOT BEEN APPLIED. IF TWO MINUTES HAVE ELAPSED AND THE WARNING HAS NOT BEEN
                   ; ACKNOWLEDGED, THE APPLICATION WILL BE SUSPENDED AND THE SYSTEM POWERED OFF.
                   ;
                   ; THIS ROUTINE IS CALLED BY THE RTC_ALARM_NMI ROUTINE EVERY ONE SECOND TO CHECK
                   ; ON THE BATTERY CONDITION.  IF LOW BATTERY IS DETECTED, THIS ROUTINE  WILL
                   ; CALL WAIT ON EXTERNAL EVENT. THIS WILL ALLOW THE SYSTEM TO SLEEP WHILE
                   ; WAITING FOR A KEY TO BE PRESSED. WHILE WAITING, THE ONE SECOND INTERRUPT CAN
                   ; OCCUR AGAIN CAUSING THE LOW BATTERY CHECK ROUTINE TO BE CALLED AGAIN. SO
                   ; THIS ROUTINE IS RECURSIVE.
                   ;
                   ; LOW BATTERY CHECK IS DISABLED DURING POST
                   ;**************************************************************************
064A                     LOW_BAT_CHK     PROC    NEAR
064A  F6 06 0012 R 01       TEST    POST_STATUS,POST_ACTIVE ; IS POST CURRENTLY ACTIVE?
                            JFNZ    LOWBEND             ; YES, JUMP TO END
064F  74 03                 JZ      $+5                 ; IF NOT NOT ZERO JUMP AROUND JUMP
0651  E9 0746 R             JMP     LOWBEND             ; ELSE TAKE A LONG JUMP
0654  E4 7F                 IN      AL,PWR_STAT         ; GET POWER INTERFACE
0656  A8 80                 TEST    AL,LOW_BAT          ; LOW BATTERY SIGNAL ON?
0658  74 1C                 JZ      LO                  ; NO LOW BATTERY SIGNAL - JUMP TO END
065A  A8 40                 TEST    AL,EXT_PWR          ; IS EXTERNAL POWER SUPPLIED?
065C  75 18                 JNZ     LO                  ; YES EXTERNAL POWER  - JUMP TO END
065E  B4 17                 MOV     AH,RTC_SYS_PROF1    ;
0660  E8 0747 R             CALL    GET_RTC_NMI         ; GET SYSTEM PROFILE
```

```
0663  A8 40                  TEST    AL,LOWBAT_ENABLE    ; LOW BATTERY MESSAGE WANTED
0665  74 0F                  JZ      L0                  ; NO MESSAGE WANTED  - JUMP TO END
0667  F6 06 0015 R 20        TEST    BAT_STATUS,LOW_BAT_PEND ; FIRST LOW BAT SIGNAL?
066C  75 23                  JNZ     L1                  ; NO, ALREADY 2 CONSECUTIVE SIGNALS
066E  80 0E 0015 R 20        OR      BAT_STATUS,LOW_BAT_PEND ; YES, SET WAITING FOR 2ND SIG
0673  E9 0746 R              JMP     LOWBEND             ; EXIT
0676                      L0:
               ;
               ; BATTERY NOT LOW OR EXTERNAL POWER IS APPLIED
               ;
0676  F6 06 0015 R 80        TEST    BAT_STATUS,LOW_BAT_SIG ; WAS LOW BTRY ALREADY SIGNALLED
067B  75 08                  JNZ     L0_A                ; YES, JUMP
067D  80 26 0015 R DF        AND     BAT_STATUS,NOT LOW_BAT_PEND ; TURN OFF LOW BTRY PENDING
0682  E9 0746 R              JMP     LOWBEND             ; JUMP TO EXIT
0685                      L0_A:
               ;
               ; LOW BATTERY HAS BEEN SIGNALLED. IF EXTERNAL PWR APPLIED RESET THE WARNING.
               ;
0685  A8 40                  TEST    AL,EXT_PWR          ; IS EXTERANL POWER APPLIED?
0687  74 08                  JZ      L1                  ; NO, JUMP TO WARNING
               ;
               ; BATTERY WAS LOW BUT EXTERNAL POWER APPLIED.
               ; TURN OFF WAITING FOR KEY AND LOW BATTERY WARNING SIGNALED FLAGS.
               ; JUMP TO TURN ON PANEL POWER.
               ;
0689  80 26 0015 R 1F        AND     BAT_STATUS,NOT LOW_BAT_SIG+LOW_BAT_HOLD+LOW_BAT_PEND;
068E  E9 073C R              JMP     L_PANEL_ON          ; ENABLE PANEL BEFORE EXITING
               ;
               ; BATTERY IS LOW WITH NO EXTERNAL POWER AND WARNING ENABLED
               ;
0691                      L1:
0691  F6 06 0015 R 80        TEST    BAT_STATUS,LOW_BAT_SIG ; LOW BTRY ALREADY SIGNALLED?
0696  74 2B                  JZ      L3                  ; NO, GO SIGNAL LOW BATTERY
               ;
               ; LOW BATTERY HAS ALREADY BEEN SIGNALLED
               ;
0698  F6 06 0015 R 40        TEST    BAT_STATUS,LOW_BAT_HOLD ; ARE WE IN HOLD STATE ?
069D  74 0C                  JZ      L2                  ; NO, JUMP AROUND SCREEN TOGGLING
               ;
               ; TOGGLE SCREEN ON AND OFF WHILE WAITING FOR A KEY TO BE PRESSED(HOLD STATE).
               ;
069F  B0 00                  MOV     AL,LCD_FUNCT        ;
06A1  E6 74                  OUT     LCD_INDX,AL         ; ACCESS LCDC CONTROL REGISTER
06A3  E4 75                  IN      AL,LCD_DATA         ; READ LCDC CONTROL REGISTER
06A5  34 40                  XOR     AL,PANEL_ENABLE     ; TOGGLE PANEL ON AND OFF
06A7  0C 20                  OR      AL,SYNC_ENABLE      ; FORCE SYNCS ON
06A9  E6 75                  OUT     LCD_DATA,AL         ; ISSUE LCDC CONTROL REG COMMAND
               ;
               ; DECREMENT 2 MINUTE COUNTER. IF COUNTER GOES TO 0 SIGNAL LOW BATTERY
               ; WARNING AGAIN IF KEY HAS BEEN PRESSED. IF COUNTER = 0 AND A KEY HAS NOT
               ; BEEN PRESSED, TURN OFF WAITING FOR KEY FLAG. THIS WILL CAUSE THE WAITING
               ; FOR KEY LOOP TO BE EXITED AND THEN SET UP THE SYSTEM TO SUSPEND.
               ;
06AB                      L2:
06AB  FE 0E 0093 R           DEC     LOW_BAT_CTR         ; COUNT DOWN  2 MINUTE COUNTER
                             JFNZ    LOWBEND             ; NOT 0, SO EXIT
06AF  74 03                  JZ      $+5                 ; IF NOT NOT ZERO JUMP AROUND JUMP
06B1  E9 0746 R              JMP     LOWBEND             ; ELSE TAKE A LONG JUMP
06B4  F6 06 0015 R 40        TEST    BAT_STATUS,LOW_BAT_HOLD ; HAS KEY BEEN PRESSED?
06B9  74 08                  JZ      L3                  ; YES, SIGNAL WARNING AGAIN
06BB  80 26 0015 R 3F        AND     BAT_STATUS,NOT LOW_BAT_SIG + LOW_BAT_HOLD ; NO KEY,
06C0  E9 0746 R              JMP     LOWBEND             ; TURN OFF FLAG AND EXIT
               ;
               ; SIGNAL LOW BATTERY WARNING
               ; CHECK FOR ANY INTERRUPTS IN SERVICE BEFORE SIGNALING LOW BATTERY
               ;
06C3                      L3:
06C3  B0 0B                  MOV     AL,0BH              ; GET INTERRUPT IN SERVICE REG
06C5  E6 20                  OUT     INTA00,AL           ; INTERRUPT CONTROLLER PORT
06C7  E4 20                  IN      AL,INTA00           ; READ INTERRUPT IN SERVICE REG
06C9  0A C0                  OR      AL,AL               ; ANY INTERRUPTS IN SERVICE
06CB  74 06                  JZ      L4                  ; IF ZERO - NONE IN SERVICE
```

```
06CD  FE 06 0093 R          INC     LOW_BAT_CTR          ; PROCESS INTERRUPT
06D1  EB 73                 JMP     SHORT  LOWBEND
                    ;
                    ; NO INTERRUPTS IN SERVICE.  SIGNAL LOW BATTERY.
                    ;
06D3                       L4:
06D3  80 0E 0015 R 80       OR      BAT_STATUS,LOW_BAT_SIG ; SET LOW BTRY SGNL FLAG ON


06D8  E8 076E R             CALL    NMI_CYCLE            ; CYCLE NMI AND RESTORE INT FLAGS
                    ;
                    ; SOUND THE 3 BEEPS
                    ;
06DB  C6 06 0093 R 78       MOV     LOW_BAT_CTR,120      ; SET 2 MINUTE COUNTER
06E0  B9 0003               MOV     CX,3                 ; 3 BEEPS - LOOP COUNT
06E3  E8 0759 R             CALL    SPKR_ON              ; FORCE SPEAKER ON
06E6  BB 00DF               MOV     BX,223               ; SHORT BEEP (.25 SECONDS)
06E9                       LOW_BEEP_LOOP:
06E9  51                    PUSH    CX                   ; SAVE LOOP COUNT
06EA  B9 0090               MOV     CX,144               ; 1/2 CYCLE FOR 890 HZ TONE
06ED  E8 0000 E             CALL    KB_NOISE             ; SOUND SPEAKER

06F0  2B C9                 SUB     CX,CX
06F2  E2 FE                 LOOP    $                    ; 250 MS DELAY BETWEEN BEEPS
06F4  59                    POP     CX                   ; RESTORE LOOP COUNT
06F5  E2 F2                 LOOP    LOW_BEEP_LOOP
06F7  E8 0762 R             CALL    SPKR_RESTORE         ; RESTORE SPEAKER ENABLE
06FA  E4 61                 IN      AL,NMI_CNTL          ;
06FC  24 F7                 AND     AL,NOT DIS_ALARM     ; RE-ENABLE ALARM NMI
06FE  E6 61                 OUT     NMI_CNTL,AL          ;
0700  80 0E 0015 R 40       OR      BAT_STATUS,LOW_BAT_HOLD ; SET BIT FOR WAITING FOR KEY
                    ;
                    ; THIS LOOP WAITS FOR A KEY TO BE PRESSED. WHILE WAITING THE SYSTEM SLEEPS.
                    ; ALSO, WHILE WAITING IN THIS LOOP, THE INTERRUPTS WILL BE PROCESSED. THE
                    ; UPDATE ENDED INTERRUPT WHICH CALLS THE LOW BATTERY ROUTINE WILL INTERRUPT
                    ; OUT EVERY SECOND. TO EXIT THIS LOOP, EITHER A KEY IS PRESSED, EXTERNAL
                    ; POWER IS APPLIED OR THE TWO MINUTE COUNTER GOES TO 0. BY APPLYING EXTERNAL
                    ; POWER (SEE LO_A) OR BY THE TWO MINUTE COUNTER GOING TO 0 (SEE L2) WILL
                    ; FORCE OFF THE WAITING FOR KEY FLAG. IF THE TWO MINUTE COUNTER GOES TO 0,
                    ; THE SYSTEM WILL SET UP TO SUSPEND.
                    ;
0705                       LOW_KB_LOOP:                  ; LOOP UNTIL KEY PRESSED

0705  B8 4104               MOV     AX,4104H             ; FUNCTION 41H, AL=04=RETURN IF ZERO
0708  BB 4000               MOV     BX,LOW_BAT_HOLD*100H ; BH=LOW_BAT_HOLD, BL=0=NO TIME OUT
070B  1E                    PUSH    DS                   ; MAKE ES:DI POINT TO BAT_STATUS
070C  07                    POP     ES                   ;
070D  BF 0015 R             MOV     DI,OFFSET BAT_STATUS ;
0710  CD 15                 INT     15H                  ; SLEEP UNTIL KEY HIT

0712  F6 06 0015 R 40       TEST    BAT_STATUS,LOW_BAT_HOLD ; BIT IS OFF IF KEY WAS PRESSED
0717  74 02                 JZ      L5                   ; KEY PRESSED EXIT LOOP
0719  EB EA                 JMP     LOW_KB_LOOP          ; KEEP LOOPING
071B                       L5:                           ;
071B  80 3E 0093 R 00       CMP     LOW_BAT_CTR,0        ; CTR = 0 FORCE SUSPEND SET UP
0720  75 12                 JNE     L6                   ; NO, JUMP ON
0722  BA 03F2               MOV     DX,DRIVE_CNTL        ; TURN OFF DISKETTE MOTORS
0725  2A C0                 SUB     AL,AL                ;
0727  EE                    OUT     DX,AL                ;
                    ;
                    ; THESE VARIABLES ARE INITIALIZED TO VALUES THAT CAUSES THE SYSTEM TO TURN
                    ; OFF WHEN RETURNING TO CALLER(RTC_ALARM_NMI).
                    ;
0728  C7 06 0067 R 0000     MOV     DSP_BLANK_CTR,0      ;
072E  C7 06 0069 R 0001     MOV     SYS_OFF_CTR,1        ;
0734                       L6:                           ; KEY PRESSED OR CTR=1
0734  E8 0000 E             CALL    DISABLE_NMI          ; DISABLE NMI
0737  80 26 0015 R DF       AND     BAT_STATUS,NOT LOW_BAT_PEND ; TURN OFF LOW BAT PEND FLAG

073C                       L_PANEL_ON:                   ; TURN PANEL ON BEFORE EXITING
073C  B0 00                 MOV     AL,LCD_FUNCT         ;
073E  E6 74                 OUT     LCD_INDX,AL          ; ACCESS LCDC CONTROL REGISTER
0740  E4 75                 IN      AL,LCD_DATA          ; READ LCDC CONTROL REGISTER
0742  0C 60                 OR      AL,PANEL_ENABLE+SYNC_ENABLE ; FORCE PANEL ON
```

```
0744  E6 75                OUT    LCD_DATA,AL         ; ISSUE LCDC CONTROL REG COMMAND
0746                     LOWBEND:
0746  C3                   RET                        ; RETURN TO CALLER
0747                     LOW_BAT_CHK     ENDP

               ;**************************************************
               ; GET RTC REGISTER WHEN ON NMI LEVEL
               ; INPUT AH= REGISTER #   OUTPUT AL = REGISTER DATA
               ;**************************************************
0747                     GET_RTC_NMI PROC NEAR
0747  86 E0                XCHG   AH,AL
0749  E6 70                OUT    RTCR_PORT,AL        ; OUTPUT REGISTER #
074B  86 C4                XCHG   AL,AH
074D  E4 71                IN     AL,RTCD_PORT        ; GET DATA
074F  C3                   RET
0750                     GET_RTC_NMI ENDP

               ;**************************************************
               ; PUT RTC REGISTER WHEN ON NMI LEVEL
               ; INPUT AH= REGISTER #, AL = REGISTER DATA OUTPUT: RTC RAM MODIFIED
               ;**************************************************
0750                     PUT_RTC_NMI  PROC NEAR
0750  86 E0                XCHG   AH,AL
0752  E6 70                OUT    RTCR_PORT,AL
0754  86 C4                XCHG   AL,AH
0756  E6 71                OUT    RTCD_PORT,AL
0758  C3                   RET
0759                     PUT_RTC_NMI ENDP
               ;********************************
               ; FORCE SPEAKER ENABLE ON
               ; ON EXIT AH HAS OLD SPEAKER CONDITION
               ;********************************
0759                     SPKR_ON PROC   NEAR
0759  E4 61                IN     AL,NMI_CNTL         ; FORCE SPEAKER ON
075B  8A E0                MOV    AH,AL
075D  0C 04                OR     AL,EN_SPKR
075F  E6 61                OUT    NMI_CNTL,AL
0761  C3                   RET
0762                     SPKR_ON ENDP

               ;********************************
               ; RESTORE SPEAKER TO PREVIOUS STATE
               ; ON INPUT AH HAS OLD SPEAKER CONDITION
               ;********************************
0762                     SPKR_RESTORE PROC NEAR
0762  80 E4 04             AND    AH,EN_SPKR
0765  E4 61                IN     AL,NMI_CNTL         ; RESTORE SPEAKER CONTROL
0767  24 FB                AND    AL,NOT EN_SPKR
0769  0A C4                OR     AL,AH
076B  E6 61                OUT    NMI_CNTL,AL
076D  C3                   RET
076E                     SPKR_RESTORE ENDP

               ;******************************************
               ; NMI_CYCLE: THIS CODE DISABLES
               ; AND RE-ENABLES THE NMI BEFORE RESTORING
               ; THE INTERRUPT FLAGS TO THEIR PREVIOUS
               ; STATE.
               ; ALL REGISTERS PRESERVED EXCEPT FLAGS
               ;******************************************
076E                     NMI_CYCLE PROC NEAR
076E  50                   PUSH   AX                  ;
076F  B0 07                MOV    AL,DISABLE_SLEEP+CLOCK_RUN ; DISABLE GLOBAL NMI
0771  E6 72                OUT    CLOCK_CTL,AL        ;
0773  EB 00                JMP    $+2                 ; DELAY
0775  0C 20                OR     AL,GLOBAL_NMI       ; RE-ENABLE NMI'S
0777  F7 46 16 0200        TEST   FLGSAVE[BP],I_FLAG  ; CHECK FOR INTERRUPTS ON
077C  74 01                JZ     NMI_C1              ; IF NOT THEN DON'T ENABLE
077E  FB                   STI                        ; ALLOW INTERRUPTS AFTER
077F                     NMI_C1:
077F  E6 72                OUT    CLOCK_CTL,AL        ; ENABLE NMI INSTRUCTION
0781  58                   POP    AX                  ;
0782  C3                   RET
0783                     NMI_CYCLE ENDP
```

```
                                INCLUDE SUSPEND.INC
                                SUBTTL  SUSPEND SYSTEM STATE
               ;************************************************************************
               ; SUSPEND
               ; THIS ROUTINE IS ACTIVATED WHEN THE NMI_FLIH DETECTES A SYSTEM SUSPEND NMI.
               ;
               ; DATE LAST MODIFIED: 09/12/85
               ;
               ;************************************************************************

               ;************************************************************************
               ;L O C A L   E Q U A T E S
               ;************************************************************************
                              PUBLIC  RESUME
                              EXTRN   COM_POWER:NEAR
                              EXTRN   MODEM_CONFIG:NEAR

= B9C0                       SUSPEND_COLOR    EQU     0B9C0H ; SUSPEND SAVE AREA IN FONT
= B1C0                       SUSPEND_MONO     EQU     0B1C0H ; SAVE FOR WHEN IN MONO
= 0030                       INIT_DISP        EQU        30H ; INITIAL VIDEO BITS OF EQUIP_FLAG
               ;
               ; TABLE OF LCD REGISTER ADDRESSES THAT MUST BE SAVED AND RESTORED
               ;
0783                         LCDR_TABLE       LABEL   BYTE
0783  01 06 09 0A 0B 0C  DB       1,6,9,0AH,0BH,0CH,0DH,0EH,0FH
      0D 0E 0F
078C  12 14 15 16 17 18  DB       12H,14H,15H,16H,17H,18H,19H,1AH,1BH,1CH,1DH
      19 1A 1B 1C 1D
= 0014                       LCDR_LENGTH      EQU     $-LCDR_TABLE

                                ASSUME  DS:DATA
```

Suspend NMI (SUSPEND)

```
0797                         SUSPEND PROC    NEAR
0797  C7 06 0072 R 0000         MOV     RESET_FLAG,0          ; CLEAR RESET_FLAG
079D  8B 16 0063 R              MOV     DX,ADDR_6845
07A1  83 C2 04                  ADD     DX,4                  ; OFFSET TO DISPLAY MODE ADDRESS
07A4  EC                        IN      AL,DX                 ; GET CURRENT MODE (LCD ONLY)
07A5  8A E8                     MOV     CH,AL                 ; CH <-- CURRENT DISPLAY MODE
07A7  B0 00                     MOV     AL,LCD_FUNCT
07A9  E6 74                     OUT     LCD_INDX,AL           ; SELECT LCD CONTROL REGISTER
07AB  E4 75                     IN      AL,LCD_DATA
07AD  8A F8                     MOV     BH,AL                 ; BH <--- CURRENT LCD CONTROL
07AF  24 BF                     AND     AL,NOT PANEL_ENABLE   ; TURN OFF LCD PANEL
07B1  E6 75                     OUT     LCD_DATA,AL
07B3  2A DB                     SUB     BL,BL                 ; BL <--- SUSPEND ERROR FLAGS
               ;
               ; TEST FOR LCD OPERABLE I.E. CAN BE USED AS ACTIVE OR ALTERNATE DISPLAY
               ;
07B5  B4 20                     MOV     AH,RTC_DSP_CON        ; GET LCD CONFIG FLAG
07B7  E8 0747 R                 CALL    GET_RTC_NMI
07BA  8A C8                     MOV     CL,AL                 ; CL <--- RTC_DSP_CON FLAGS
07BC  F6 C1 01                  TEST    CL,DSP_CLCD           ; TEST LCD STATE
07BF  75 15                     JNZ     SUS_000               ; JUMP IF LCD IS CGA
07C1  BA 03B4                   MOV     DX,MONO_CNTL-4        ; ASSUME LCD IS MONO
07C4  BF B1C0                   MOV     DI,SUSPEND_MONO       ; DI <-- SAVE AREA SEGMENT
07C7  F6 C1 02                  TEST    CL,DSP_MLCD           ; TEST LCD STATE
07CA  75 10                     JNZ     SUS_001               ; JUMP IF LCD IS MONO
               ;
               ; LCD NOT ACTIVE
               ;

07CC  24 0C                     AND     AL,DSP_MONO+DSP_CGA   ; SAVE ONLY OTHER DISPLAY INFO
07CE  3C 0C                     CMP     AL,DSP_MONO+DSP_CGA   ; ARE BOTH DISPLAYS ATTACHED?
07D0  74 33                     JE      SUS_002               ; IS SO THEN CANNOT SUSPEND
```

```
07D2  A8 04                TEST    AL,DSP_CGA           ; CGA DISPLAY ONLY?
07D4  75 06                JNZ     SUS_001              ; IF YES LEAVE LCD AS MONO
                    ;
                    ; MONO DISPLAY OR LCD AS CGA SO SET CGA ACCESS
                    ;
07D6                    SUS_000:
07D6  BA 03D4              MOV     DX,CGA_CNTL-4
07D9  BF B9C0              MOV     DI,SUSPEND_COLOR     ; DI <-- SAVE AREA SEGMENT

07DC                    SUS_001:
07DC  39 16 0063 R         CMP     ADDR_6845,DX         ; IS LCD THE ACTIVE DISPLAY?
07E0  74 25                JE      SUS_003              ; IF SO THEN OKAY

07E2  8A 2E 0065 R         MOV     CH,CRT_MODE_SET      ; CH <--- MODE SET FOR CRT
07E6  B4 22                MOV     AH,RTC_DSP_STAT      ; GET DISPLAY STATUS
07E8  E8 0747 R            CALL    GET_RTC_NMI
07EB  8B F0                MOV     SI,AX                ; SI LOW <--- RTC_DSP_STAT
07ED  A8 80                TEST    AL,DIAG_FORCE_SUS    ; CHECK RTC_DSP_STAT FLAGS FOR DIAG
                                                        ; RESUME
07EF  74 14                JZ      SUS_002              ; IF NOT THEN SUSPEND ERROR
                    ;
                    ; SETUP LCD CONTROLLER FOR ACCESS
                    ;
07F1  F6 C7 08             TEST    BH,LCD_ENAB          ; CHECK FOR LCD ENABLED
07F4  75 11                JNZ     SUS_003              ; IF SO THEN ALREADY ACCESSED
                    ;
                    ; LCD NOT ACTIVE SO SET ACTIVE FOR ACCESS
                    ;
07F6  B0 08                MOV     AL,LCD_ENAB          ; SET ENABLE
07F8  81 FF B1C0           CMP     DI,SUSPEND_MONO      ; CHECK FOR LCD AS MONO
07FC  74 02                JE      SUS_001A             ; JUMP IF SO
07FE  0C 02                OR      AL,LCD_CGA           ; OTHERWISE SET LCD AS CGA

0800                    SUS_001A:
0800  E6 75                OUT     LCD_DATA,AL          ; WRITE TO LCD CONTROL
0802  EB 03 90             JMP     SUS_003
                    ;
                    ; SET SUSPEND UNSUCCESSFUL DUE TO LCD INACCESSABILITY
                    ;
0805                    SUS_002:
0805  B3 02                MOV     BL,LCD_NOT_ACTIVE    ; SET FLAG - FOR LCD INOPERABLE, THIS
                                                        ; WILL NOT BECOME AN ERROR
                    ;
                    ; CHECK FOR DISKETTE MOTORS OFF
                    ;
0807                    SUS_003:
0807  52                   PUSH    DX                   ; SAVE LCD ADDRESS
0808  BA 03F7              MOV     DX,DRIVE_SENSE       ; READ DRIVE STATUS
080B  EC                   IN      AL,DX
080C  A8 78                TEST    AL,DR0_SEL_SENSE+DR1_SEL_SENSE+DR0_MOT_SENSE+DR1_MOT_SENSE
080E  74 09                JZ      SUS_004
                    ;
                    ;  DISKETTE MOTORS NOT OFF/ TURN THEM OFF AND SET ERROR FLAG
                    ;
0810  BA 03F2              MOV     DX,DRIVE_CNTL        ; DISKETTE CONTROL
0813  2A C0                SUB     AL,AL
0815  EE                   OUT     DX,AL
0816  80 CB 01             OR      BL,DSKT_ACTIVE       ; SET ERROR FLAG
                    ;
                    ; TURN OFF CGA DISPLAY VIDEO
                    ;
0819                    SUS_004:
0819  2A C0                SUB     AL,AL
081B  BA 03D8              MOV     DX,CGA_CNTL          ; DISABLE VIDEO COLOR
081E  EE                   OUT     DX,AL
                    ;
                    ; TURN OFF MONO DISPLAY VIDEO
                    ;
081F  FE C0                INC     AL
0821  BA 03B8              MOV     DX,MONO_CNTL         ; DISABLE VIDEO ON MONO
0824  EE                   OUT     DX,AL
0825  5A                   POP     DX                   ; RESTORE LCD ADDRESS
                    ;
                    ; SAVE REAL TIME CLOCK INTERRUPT MODE AND TURN OFF ALL BUT ALARM INTERRUPTS
```

```
                  ;
0826  B4 0B                MOV    AH,RTC_MODE
0828  E8 0747 R            CALL   GET_RTC_NMI
082B  8A F0                MOV    DH,AL                ; DH <-- RTC INTERRUPT MODE
                                                       ; DL <-- LOW ORDER LCD ADDRESS
082D  24 AF                AND    AL,NOT PIE_ENABLE+UIE_ENABLE ; DISABLE INTS
082F  E8 0750 R            CALL   PUT_RTC_NMI
0832  FE C4                INC    AH                   ; READ LAST INTERRUPT STATUS
0834  E8 0747 R            CALL   GET_RTC_NMI          ; TO CLEAR IT

0837  F6 06 00A0 R 04      TEST   RTC_WAIT_FLAG,PON_ALRM_PEND ; IS POWER ON PENDING?
083C  74 0D                JZ     SUS04A

083E  80 26 00A0 R FB      AND    RTC_WAIT_FLAG,NOT PON_ALRM_PEND ; RESET FLAG
0843  E4 7F                IN     AL,PWR_STAT          ;
0845  24 F7                AND    AL,NOT HDWR_RESET    ; TURN OFF RESET FLAG
0847  0C 01                OR     AL,EN_PON_ALRM       ; ENABLE POWER ON BY ALARM
0849  E6 7F                OUT    PWR_STAT,AL
                  ;
                  ; CHECK SYSTEM PROFILE FOR RESUME OPTION ENABLE OR FORCED RESUME
                  ;
084B                    SUS04A:
084B  F6 06 0016 R 40      TEST   BIOS_STATUS,F_RESUME ; IS A FORCED RESUME REQUESTED?
0850  75 09                JNZ    SUS_04B              ; JUMP IF YES
0852  B4 17                MOV    AH,RTC_SYS_PROF1     ; GET SYSTEM PROFILE
0854  E8 0747 R            CALL   GET_RTC_NMI
0857  A8 80                TEST   AL,RESUME_ENABLE     ; SYSTEM TO BE RESUMED?
0859  74 1B                JZ     SUS_04D              ; JUMP IF YES
                  ;
                  ; SET DIAGNOSTIC FLAGS IN RTC AREA
                  ;
085B                    SUS_04B:
085B  B4 0E                MOV    AH,RTC_DIAG_STAT     ; UPDATE DIAGNOSTIC STATUS
085D  E8 0747 R            CALL   GET_RTC_NMI
0860  0A C3                OR     AL,BL                ; SET FLAGS
0862  0A DB                OR     BL,BL                ; ANY ERRORS?
0864  74 13                JZ     SUS_04E              ; NO THEN SKIP SAVE
0866  F6 C1 03             TEST   CL,DSP_CLCD+DSP_MLCD ; IF LCD INOPERABLE, THEN RESET
0869  75 08                JNZ    SUS_04C              ; ERROR FLAG
086B  F7 C6 0080           TEST   SI,DIAG_FORCE_SUS    ; CHECK FOR FORCE SUSPEND
086F  74 05                JZ     SUS_04D              ; IF NO LCD AND NO FORCE SUSPEND
                                                       ; THEN DO NOT LOG ANY ERRORS
0871  24 FD                AND    AL,NOT LCD_NOT_ACTIVE ; RESET LCD NOT ACTIVE FLAG

0873                    SUS_04C:
0873  E8 0750 R            CALL   PUT_RTC_NMI

0876                    SUS_04D:                       ;
0876  E9 092C R            JMP    SUSP_HLT             ; YES THEN DO NOT SUSPEND
                  ;
                  ; CH CONTAINS LCD CONTROL SAVE, BX CONTAINS SUSPEND SEGMENT ADDRESS
                  ;
0879                    SUS_04E:
0879  8A CF                MOV    CL,BH                ; CL <--- CURRENT LCD CONTROL
087B  8E C7                MOV    ES,DI                ; SET SEGMENT
087D  2B FF                SUB    DI,DI                ; CLEAR DESTINATION OFFSET

087F  E4 75                IN     AL,LCD_DATA          ; GET LCD CONTROL
0881  0C 10                OR     AL,LCD_FONT          ; SET FONT ACCESS FLAG
0883  E6 75                OUT    LCD_DATA,AL
                  ;
                  ; SAVE STACK SEGMENT AND POINTER
                  ;
0885  8C D0                MOV    AX,SS                ; SAVE STACK SEGMENT
0887  AB                   STOSW
0888  8B C4                MOV    AX,SP                ; SAVE STACK POINTER
088A  AB                   STOSW
                  ;
                  ; SAVE REAL TIME CLOCK INTERRUPT MODE
                  ;
088B  8A C6                MOV    AL,DH                ; SAVE RTC INTERRUPT MODE
088D  AA                   STOSB
                  ;
                  ; SAVE LCD SYSTEM CONTROL,MODE CONTROL, AND PARAMETER REGISTERS
```

```
                    ;
088E  8B C1                  MOV     AX,CX               ; SAVE LCD MODE CONTROL
0890  AB                     STOSW
0891  BE 0783 R              MOV     SI,OFFSET LCDR_TABLE ; SAVE LCD REGISTERS
0894  B9 0014                MOV     CX,LCDR_LENGTH
0897  B6 03                  MOV     DH,03               ; SET HIGH ORDER LCD ADDRESS
                                                         ; LOW ORDER ALREADY SET
0899                      SUS_005:
0899  2E: 8A 04              MOV     AL,CS:[SI]            ; GET REGISTER #
089C  46                     INC     SI                  ; BUMP POINTER
089D  EE                     OUT     DX,AL               ; SET REGISTER #
089E  42                     INC     DX                  ; GET DATA PORT
089F  EC                     IN      AL,DX
08A0  AA                     STOSB                       ; SAVE
08A1  4A                     DEC     DX
08A2  E2 F5                  LOOP    SUS_005             ; CONTINUE
                    ;
                    ; SAVE TIMER 0 AND TIMER 2 INFORMATION
                    ;
08A4  B0 50                  MOV     AL,50H              ; SELECT TIMER 0 AND LSB
08A6  B9 0002                MOV     CX,2                ; LOOP NUMBER
08A9                      SUS_006:
08A9  E6 43                  OUT     TIMER_CTL,AL
08AB  E4 43                  IN      AL,TIMER_CTL        ; GET TIMER 0/2 MODE
08AD  AA                     STOSB
08AE  E4 40                  IN      AL,TIMER0           ; GET TIMER 0 LSB/MSB
08B0  AA                     STOSB
08B1  E4 42                  IN      AL,TIMER2           ; GET TIMER 2 LSB/MSB
08B3  AA                     STOSB
08B4  B0 D0                  MOV     AL,0D0H             ; SELECT TIMER 2 AND MSB
08B6  E2 F1                  LOOP    SUS_006
                    ;
                    ; SAVE INTERRUPT CONTROLLER STATE
                    ;
08B8  E4 21                  IN      AL,INTA01           ; GET INTERRUPT MASK
08BA  AA                     STOSB
08BB  B0 04                  MOV     AL,04               ; SELECT INTERRUPT BYTE 0
08BD  E6 72                  OUT     CLOCK_CTL,AL
08BF  E4 63                  IN      AL,63H              ; SAVE BYTE 0
08C1  AA                     STOSB
08C2  B0 44                  MOV     AL,44H              ; SELECT INTERRUPT BYTE 1
08C4  E6 72                  OUT     CLOCK_CTL,AL
08C6  E4 63                  IN      AL,63H              ; SAVE BYTE 1
08C8  AA                     STOSB
                    ;
                    ; SAVE INTERRUPT 0-32 VECTORS
                    ;
08C9  B8 0000                MOV     AX,0
08CC  8E D8                  MOV     DS,AX               ; SET DS SEGMENT 0
08CE  BE 0000                MOV     SI,0
08D1  B9 0040                MOV     CX,64               ; SAVE 32 VECTORS
08D4  F3/ A5                 REP     MOVSW
                    ;
                    ; SAVE INTERRUPT VECTOR 44H
                    ;
08D6  BE 0110                MOV     SI,44H*4            ; SAVE VECTOR 44H
08D9  A5                     MOVSW
08DA  A5                     MOVSW
                    ;
                    ; SAVE DATA AREA FROM 0300-053A
                    ;
08DB  B9 011E                MOV     CX,11EH             ; SET MOVE LENGTH
08DE  BE 0300                MOV     SI,0300H            ; SET SOURCE ADDRESS
08E1  F3/ A5                 REP     MOVSW
                    ;
                    ; SAVE KEYBOARD AND FEATURE CONTROL REGISTER
                    ;
08E3  E4 7C                  IN      AL,KYBD_CNTL
08E5  AA                     STOSB
                    ;
                    ;  SAVE I/O CHANNEL CHECK FLAG
                    ;
08E6  E4 A0                  IN      AL,IONMI_CNTL
08E8  AA                     STOSB
```

```
                ;
                ; SAVE NMI AND SPEAKER CONTROL REGISTER
                ;
08E9  E4 61                  IN      AL,NMI_CNTL
08EB  AA                     STOSB
08EC  B0 00                  MOV     AL,0
08EE  E6 61                  OUT     NMI_CNTL,AL          ; DISABLE SPEAKER
                ;
                ;  SAVE COMMUNICAYTIONS CONTROLLER(S) CURRENT STATE
                ;
08F0  B3 01                  MOV     BL,01                ; SET SUSPEND PARAMETER FOR
08F2  E8 0B04 R              CALL    ASYNC_SUSPEND        ; SAVING ASYNC DEVICES
                ;
                ; CHECKSUM LOWER 128K OF STORAGE
                ;
08F5  2B DB                  SUB     BX,BX                ; SET ACCUMULATOR
08F7  BE 053A                MOV     SI,53AH              ; STARTING OFFSET
08FA  B9 7D63                MOV     CX,7D63H             ; CHECKSUM 32K WORDS
08FD                      SUS_010:
08FD  AD                     LODSW
08FE  03 D8                  ADD     BX,AX
0900  E2 FB                  LOOP    SUS_010
0902  B9 8000                MOV     CX,8000H             ; CHECKSUM NEXT 32K WORDS
0905  8C D8                  MOV     AX,DS
0907  80 C4 10               ADD     AH,10H               ; OFFSET TO NEXT SEGMENT
090A  8E D8                  MOV     DS,AX
090C                      SUS_011:
090C  AD                     LODSW
090D  03 D8                  ADD     BX,AX
090F  E2 FB                  LOOP    SUS_011
                ;
                ; CANNOT USE SYSTEM STACK FROM HERE ON
                ;
0911  B0 23                  MOV     AL,RTC_BMEM_CKSL     ; ADDRESS CHECKSUM LOW SAVE AREA
0913  E6 70                  OUT     RTCR_PORT,AL         ; OUTPUT ADDRESS
0915  8A C3                  MOV     AL,BL
0917  E6 71                  OUT     RTCD_PORT,AL         ; OUTPUT DATA

0919  B0 24                  MOV     AL,RTC_BMEM_CKSH     ; ADDRESS CHECKSUM HI SAVE AREA
091B  E6 70                  OUT     RTCR_PORT,AL         ; OUTPUT ADDRESS
091D  8A C7                  MOV     AL,BH
091F  E6 71                  OUT     RTCD_PORT,AL         ; OUTPUT DATA
                ;
                ; SET SUSPEND FLAG
                ;
                             ASSUME  DS:DATA
0921  B8 ---- R              MOV     AX,DATA              ; SET DS BACK TO DATA AREA
0924  8E D8                  MOV     DS,AX
0926  C7 06 0072 R 5678      MOV     RESET_FLAG,SYS_SUSPEND ; SET SUSPEND SUCCESSFUL

092C                      SUSP_HLT:
092C  B0 00                  MOV     AL,CLOCK_STOP        ; STOP SYSTEM CLOCKS
092E  E6 72                  OUT     CLOCK_CTL,AL
0930  EB FA                  JMP     SUSP_HLT
0932                      SUSPEND        ENDP
```

Resume (RESUME)

```
                             SUBTTL  RESUME SYSTEM STATE
                ;****************************************************************
                ; RESUME: SUBROUTINE TO RESTORE SYSTEM FOR APPLICATION RESUME
                ;
                ;****************************************************************
0932                      RESUME         PROC     NEAR
                             ASSUME  DS:DATA
0932  E8 0000 E              CALL    DISABLE_NMI          ; DISABLE ALL INTERRUPTS
0935  B4 20                  MOV     AH,RTC_DSP_CON       ; GET LCD TYPE
0937  E8 0747 R              CALL    GET_RTC_NMI
```

```
093A  BB B1C0               MOV     BX,SUSPEND_MONO      ; DEFAULT TO LCD AS MONO
093D  BA 03B4               MOV     DX,MONO_CNTL-4       ;
                ;
                ; SET RESUME MODE ACCORDING TO DISPLAY CONFIGURATION
                ;
0940  A8 06                 TEST    AL,DSP_MLCD+DSP_CGA  ; IF LCD AS MONO OR CGA INSTALLED
0942  75 06                 JNZ     RES_001              ; JMP - RESUME IN MONO MODE

0944  BB B9C0               MOV     BX,SUSPEND_COLOR     ; LCD MUST BE COLOR
0947  BA 03D4               MOV     DX,CGA_CNTL-4        ;

094A                    RES_001:
094A  8E DB                 MOV     DS,BX                ; GET SAVE AREA SEGMENT
094C  BE 0007               MOV     SI,7                 ; RETRIEVE LCD INFO
                ;
                ; ACCESS LCD FONT AREA WHERE SYSTEM SUSPEND SAVE AREA IS LOCATED
                ;
094F  B0 00                 MOV     AL,LCD_FUNCT
0951  E6 74                 OUT     LCD_INDX,AL          ; SELECT LCD REG 0
0953  B0 18                 MOV     AL,LCD_ENAB+LCD_FONT ; SET ENABLE WITH FONT ACCESS
0955  81 FA 03B4            CMP     DX,MONO_CNTL-4       ; IS LCD SET IN MONO MODE?
0959  74 02                 JE      RES_001A             ; JUMP IF YES
095B  0C 02                 OR      AL,LCD_CGA           ; NO,  USE LCDC AS CGA

095D                    RES_001A:                        ;
095D  E6 75                 OUT     LCD_DATA,AL          ; SET LCD MODE FOR RESUME

                ;
                ; RESTORE LCD CONTROL REGISTERS
                ;
095F  BF 0783 R             MOV     DI,OFFSET LCDR_TABLE ; POINT TO LCD REG TABLE
0962  B9 0014               MOV     CX,LCDR_LENGTH       ; GET LENGTH OF TABLE
0965                    RES_002:
0965  2E: 8A 05             MOV     AL,CS:[DI]             ; GET REGISTER ADDRESS
0968  47                    INC     DI
0969  EE                    OUT     DX,AL                ; OUTPUT REG NUMBER
096A  42                    INC     DX                   ; GET DATA PORT
096B  AC                    LODSB                        ; RETRIEVE REGISTER VALUE
096C  EE                    OUT     DX,AL
096D  4A                    DEC     DX                   ; SET DX BACK TO INDEX PORT
096E  E2 F5                 LOOP    RES_002
                ;
                ; RESTORE SYSTEM TIMERS
                ;
0970  AC                    LODSB                        ; GET TIMER 0 MODE
0971  24 3F                 AND     AL,3FH               ; TURN OFF UPPER TWO BITS
0973  8A E0                 MOV     AH,AL                ; AH HAS TIMER 0 MODE
0975  E6 43                 OUT     TIMER_CTL,AL         ; WRITE TIMER 0 MODE
0977  AC                    LODSB                        ;
0978  8A D8                 MOV     BL,AL                ; BL HAS TIMER 0 LSB
097A  AC                    LODSB                        ;
097B  8A C8                 MOV     CL,AL                ; CL HAS TIMER 2 LSB
097D  AC                    LODSB                        ;
097E  8A F0                 MOV     DH,AL                ; DH HAS TIMER 2 MODE
0980  AC                    LODSB                        ;
0981  8A F8                 MOV     BH,AL                ; BH HAS TIMER 0 MSB
0983  AC                    LODSB
0984  8A E8                 MOV     CH,AL                ; CH HAS TIMER 2 MSB
                ;
                ; WRITE TIMER 0 COUNTER
                ;
0986  80 E4 30              AND     AH,30H               ; SAVE ONLY READ/WRITE TYPE
0989  80 FC 10              CMP     AH,10H               ; IS IT LSB ONLY?
098C  74 0F                 JE      RES_003
098E  80 FC 20              CMP     AH,20H               ; IS IT MSB ONLY?
0991  74 10                 JE      RES_004
0993  8A C3                 MOV     AL,BL                ; OUT LSB FIRST THEN MSB
0995  E6 40                 OUT     TIMER0,AL
0997  8A C7                 MOV     AL,BH
0999  E6 40                 OUT     TIMER0,AL
099B  EB 0A                 JMP     SHORT RES_005
099D                    RES_003:
099D  8A C3                 MOV     AL,BL                ; WRITE LSB ONLY
099F  E6 40                 OUT     TIMER0,AL
```

```
09A1  EB 04               JMP     SHORT RES_005
09A3                    RES_004:
09A3  8A C7               MOV     AL,BH                 ; WRITE MSB ONLY
09A5  E6 40               OUT     TIMER0,AL
                 ;
                 ; RESTORE TIMER 2
                 ;
09A7                    RES_005:
09A7  8A C6               MOV     AL,DH                 ; SET TIMER 2 MODE
09A9  24 3F               AND     AL,3FH                ; TURN OFF UPPER TWO BITS
09AB  0C 80               OR      AL,80H                ; SET TIMER 2
09AD  E6 43               OUT     TIMER_CTL,AL
09AF  24 30               AND     AL,30H                ; LEAVE ONLY READ/WRITE TYPE
09B1  3C 10               CMP     AL,10H                ; IS IT LSB ONLY?
09B3  74 0E               JE      RES_006
09B5  3C 20               CMP     AL,20H                ; IS IT MSB ONLY?
09B7  74 10               JE      RES_007
09B9  8A C1               MOV     AL,CL                 ; OUT LSB FIRST THEN MSB
09BB  E6 42               OUT     TIMER2,AL
09BD  8A C5               MOV     AL,CH
09BF  E6 42               OUT     TIMER2,AL
09C1  EB 0A               JMP     SHORT RES_008
09C3                    RES_006:
09C3  8A C1               MOV     AL,CL                 ; WRITE LSB ONLY
09C5  E6 42               OUT     TIMER2,AL
09C7  EB 04               JMP     SHORT RES_008
09C9                    RES_007:
09C9  8A C5               MOV     AL,CH                 ; WRITE MSB ONLY
09CB  E6 42               OUT     TIMER2,AL
                 ;
                 ; INITIALIZE INTERRUPT CONTROLLER
                 ;
09CD                    RES_008:
09CD  AC                  LODSB                         ; GET INTERRUPT MASK
09CE  8A E0               MOV     AH,AL                 ; AH HAS INTERRUPT MASK
09D0  AC                  LODSB                         ; GET INTERRUPT BYTE 0
09D1  8A F8               MOV     BH,AL
09D3  AC                  LODSB                         ; GET INTERRUPT BYTE 1
09D4  8A D8               MOV     BL,AL
                 ;
                 ; BUILD AND RESTORE ICW 1
                 ;
09D6  24 08               AND     AL,08H                ; AND OFF ALL BUT LEVEL/EDGE
09D8  0C 10               OR      AL,10H                ; BIT 4 MUST BE 1
09DA  E6 20               OUT     INTA00,AL             ; OUTPUT ICW1
                 ;
                 ; BUILD AND RESTORE ICW 2
                 ;
09DC  8A C7               MOV     AL,BH                 ; GET INTERRUPT TYPE ASSIGN
09DE  24 F8               AND     AL,0F8H               ; ONLY SAVE ICW2 INFO
09E0  E6 21               OUT     INTA01,AL             ; SEND ICW 2
                 ;
                 ; BUILD AND RESTORE ICW 4
                 ;
09E2  8A C3               MOV     AL,BL                 ; GET INTERRUPT BYTE 1
09E4  D0 E8               SHR     AL,1                  ; GET AUTO EOI BIT CORRECT
09E6  24 02               AND     AL,02                 ; SAVE ONLY AUTO EOI BIT
09E8  E6 21               OUT     INTA01,AL             ; SEND ICW 4
                 ;
                 ; RESTORE INTERRUPT MASK REGISTER
                 ;
09EA  8A C4               MOV     AL,AH                 ; GET INTERRUPT MASK
09EC  E6 21               OUT     INTA01,AL
                 ;
                 ; RESTORE INTERRUPT VECTORS 0 -32 TO RAM
                 ;
09EE  B8 0000             MOV     AX,0
09F1  8E C0               MOV     ES,AX                 ; SET SEGMENT 0
09F3  BF 0000             MOV     DI,0                  ; START AT VECTOR 0
09F6  B9 0040             MOV     CX,64                 ; RESTORE 32 VECTORS
09F9  F3/ A5              REP     MOVSW
                 ;
                 ; RESTORE VECTOR 44H
```

```
                       ;
09FB  BF 0110                  MOV     DI,44H*4            ; RESTORE VECTOR 44H
09FE  A5                       MOVSW
09FF  A5                       MOVSW
                       ;
                       ; RESTORE DATA AREA FROM 0300-3FFH
                       ; WARNING: CANNOT DO ANY STACK OPERATIONS FROM NOW UNTIL SS AND SP RESTORED
                       ;
0A00  BF 0300                  MOV     DI,0300H            ; SET DESTINATION
0A03  B9 0080                  MOV     CX,128              ; 128 WORDS
0A06  F3/ A5                   REP     MOVSW
0A08  83 C7 12                 ADD     DI,18               ; SKIP FROM 400H-410H
0A0B  83 C6 10                 ADD     SI,16               ; SOURCE ADDRESSES OLD EQUIP_FLAG
0A0E  AD                       LODSW                       ; AX=PRE SUSPEND EQUIP_FLAG
0A0F  8B E8                    MOV     BP,AX               ; SAVE OLD EQUIP_FLAG
                       ;
                       ; RESTORE DATA AREA FROM 0412-053A
                       ;
0A11  B9 0095                  MOV     CX,149              ; RESTORE 149 WORDS
0A14  F3/ A5                   REP     MOVSW
                       ;
                       ;
                       ; CLEAR KEYBOARD BREAK PENDING FLAGS
                       ;
0A16  8C DB                    MOV     BX,DS               ; SAVE RESTORE SEGMENT
0A18  B8 ---- R                MOV     AX,DATA
0A1B  8E D8                    MOV     DS,AX               ; RESTORE DATA SEGMENT
0A1D  2B C0                    SUB     AX,AX
0A1F  A3 00B5 R                MOV     B_PEND1,AX
0A22  A3 00B7 R                MOV     B_PEND2,AX
0A25  A2 0018 R                MOV     KB_FLAG_1,AL        ; CLEAR KEY DEPRESSED BITS
0A28  A3 0072 R                MOV     RESET_FLAG,AX       ; CLEAR RESET FLAG
0A2B  A2 00BA R                MOV     LAST_CLICK_KEY,AL   ; CLEAR KEY CLICK TRACKING
0A2E  A2 0015 R                MOV     BAT_STATUS,AL       ; CLEAR BATTERY STATUS FLAG
0A31  A2 00B9 R                MOV     P60_HOLD_BYTE,AL    ; CLEAR PORT 60 HOLDING REG
                       ;
                       ; CLEAR BIOS STATUS FLAGS , CAUSE TIMEOUT COUNTERS TO BE RELOADED AND
                       ; A CHECK FOR DISKETTE CHANGE TO BE PERFORMED
                       ;
0A34  80 26 0016 R 04          AND     BIOS_STATUS,DCL_SUPPORTED ; SAVE DCL SUPPORT FLAG
0A39  80 0E 0016 R 22          OR      BIOS_STATUS,KYBD_ACTIVE+FORCE_DCL
0A3E  8B C5                    MOV     AX,BP               ; GET OLD EQUIPMENT INFO
0A40  80 26 0010 R CF          AND     BYTE PTR EQUIP_FLAG,NOT INIT_DISP  ; CLR NEW VIDEO FLAGS
0A45  24 30                    AND     AL,INIT_DISP        ; ONLY SAVE OLD VIDEO FLAGS
0A47  08 06 0010 R             OR      BYTE PTR EQUIP_FLAG,AL ; MOV OLD VIDEO FLAGS TO EQUIP
0A4B  8B 0E 0010 R             MOV     CX,EQUIP_FLAG       ; CX <-- EQUIPMENT WORD
                       ;
                       ; RESTORE APPLICATION PROGRAMS STACK POINTER
                       ;
0A4F  8E DB                    MOV     DS,BX               ; RESTORE SAVE AREA SEGMENT
0A51  8B FE                    MOV     DI,SI               ; SAVE CURRENT PLACE
0A53  BE 0000                  MOV     SI,0
0A56  AD                       LODSW                       ; GET SS VALUE
0A57  8B D8                    MOV     BX,AX               ; SAVE IN BX
0A59  AD                       LODSW                       ; GET SP VALUE
0A5A  8E D3                    MOV     SS,BX               ; RESTORE APPLICATION STACK
0A5C  8B E0                    MOV     SP,AX
0A5E  AC                       LODSB                       ; GET RTC MODE
0A5F  50                       PUSH    AX                  ; SAVE ON STACK
0A60  AD                       LODSW                       ; GET LCD AND VIDEO CONTROL
0A61  50                       PUSH    AX                  ; SAVE ON STACK
0A62  8B F7                    MOV     SI,DI               ; RESTORE POINTER
                       ;
                       ; RESTORE EQUIP WORD IN RTC RAM FROM SAVED WORD IN REGISTER CX
                       ;
0A64  B4 13                    MOV     AH,RTC_EQUIP_LO     ; SAVE IN RTC EQUIPMENT AREA
0A66  8A C1                    MOV     AL,CL
0A68  E8 0750 R                CALL    PUT_RTC_NMI         ; SAVE EQUIPMENT INFO IN RTC
0A6B  FE C4                    INC     AH
0A6D  8A C5                    MOV     AL,CH
0A6F  E8 0750 R                CALL    PUT_RTC_NMI
                       ;
                       ; RELOAD KEYBOARD NMI CONTROL STATE
```

```
                ;
0A72  AC                  LODSB                      ; RETRIEVE FEATURE CONTROL
0A73  24 80               AND    AL,EN_KYBD_NMI      ; LEAVE ONLY KEYBOARD STATE
0A75  8A D8               MOV    BL,AL               ; SAVE IN BL
0A77  E4 7C               IN     AL,KYBD_CNTL
0A79  24 7F               AND    AL,NOT EN_KYBD_NMI  ; MASK CURRENT NMI STATE
0A7B  0A C3               OR     AL,BL               ; RESTORE KEYBOARD NMI STATE
0A7D  E6 7C               OUT    KYBD_CNTL,AL        ; OUTPUT FEATURE CONTROL
                ;
                ; RESTORE I/O CHANNEL CHECK FLAG
                ;
0A7F  AC                  LODSB
0A80  E6 A0               OUT    IONMI_CNTL,AL
                ;
                ; RESTORE NMI AND SPEAKER CONTROL REGISTER
                ;
0A82  AC                  LODSB
0A83  E6 61               OUT    NMI_CNTL,AL
                ;
                ; RESTORE POWER ENABLE TO  MODEM
                ;
0A85  E4 7F               IN     AL,PWR_STAT         ; GET CURRENT POWER STATUS
0A87  A8 40               TEST   AL,EXT_PWR          ; ON EXTERNAL POWER?
0A89  75 13               JNZ    RES_009             ; JUMP IF ON EXTERNAL POWER
                ;
                ; CURRENTLY ON BATTERY POWER SO CHECK USER PROFILE FOR MODEM OPTIONS
                ;
0A8B  B4 17               MOV    AH,RTC_SYS_PROF1
0A8D  E8 0747 R           CALL   GET_RTC_NMI
0A90  A8 02               TEST   AL,MODEM_BATT       ; OPERATE COM1 ON BATTERY?
0A92  75 0A               JNZ    RES_009             ; YES THEN JUMP
                ;
                ; ON BATTERY POWER AND MODEM PROFILE INDICATES NO BATTERY OPERATION
                ;
0A94  B3 02               MOV    BL,ACT_MODEM        ; SPECIFY MODEM OFF
0A96  2A FF               SUB    BH,BH               ; INDICATE POWER OFF REQUEST
0A98  E8 0000 E           CALL   COM_POWER           ; TURN OFF PRIMARY COM POWER
0A9B  EB 09 90            JMP    RES_010             ; SKIP MODEM _CONFIG
                ;
                ; MODEM IS POWERED ON SO RESUME CONFIGURATION
                ;
0A9E                  RES_009:
0A9E  B4 1D               MOV    AH,RTC_MOD_PROF1    ; GET MODEM PROFILE
0AA0  E8 0747 R           CALL   GET_RTC_NMI         ;
0AA3  E8 0000 E           CALL   MODEM_CONFIG        ; GO SETUP MODEM CONFIGURATION
                ;
                ; RESTORE COMMUNICATION CONTROLLERS STATE AT POWER OFF TIME
                ;
0AA6                  RES_010:
0AA6  2A DB               SUB    BL,BL               ; SET RESUME CODE FOR RESTORING
0AA8  E8 0B04 R           CALL   ASYNC_SUSPEND       ; OF COMMUNICATION STATE
                ;
                ; TURN OFF FONT ACCESS AND ENABLE PANEL
                ;
0AAB  B8 ---- R           MOV    AX,DATA             ; SET DATA SEGMENT ADDRESS
0AAE  8E D8               MOV    DS,AX
0AB0  58                  POP    AX                  ; RETRIEVE VIDEO FLAGS
0AB1  24 BF               AND    AL,NOT PANEL_ENABLE ; FORCE PANEL OFF
0AB3  E6 75               OUT    LCD_DATA,AL         ; RESTORE LCD CONTROL
0AB5  A8 08               TEST   AL,LCD_ENAB         ; CHECK FOR LCD ENABLED
0AB7  74 04               JZ     RES010A             ; ONLY ENABLE PANEL IF LCD ENABLED
                ;
                ; ENABLE PANEL IF LCD WAS ENABLED
                ;
0AB9  0C 60               OR     AL,SYNC_ENABLE+PANEL_ENABLE ; ENABLE PANEL
0ABB  E6 75               OUT    LCD_DATA,AL         ;
                ;
                ; RESTORE ACTIVE VIDEO MODE CONTROL REGISTER
                ;
0ABD                  RES010A:
0ABD  8B 16 0063 R        MOV    DX,ADDR_6845
0AC1  83 C2 04            ADD    DX,4
0AC4  8A C4               MOV    AL,AH               ; GET RESTORE VIDEO MODE
0AC6  EE                  OUT    DX,AL               ; RESTORE VIDEO MODE
```

```
                ;
                ; ENABLE DISKETTE NMI'S
                ;
0AC7  E4 77                IN      AL,DSKT_CNTL
0AC9  0C 80                OR      AL,DSKT_NMI          ; ENABLE DISKETTE POWER ON NMI
0ACB  E6 77                OUT     DSKT_CNTL,AL
                ;
                ; SET REAL TIME CLOCK ALARM FLAG IF POWERED ON BY ALARM
                ;
0ACD  B4 21                MOV     AH,RTC_SYS_STAT      ; GET SYSTEM STATUS
0ACF  E8 0747 R            CALL    GET_RTC_NMI
0AD2  8A F8                MOV     BH,AL                ; BH <-- SYS_STATUS
0AD4  58                   POP     AX                   ; RETRIEVE RTC MODE
0AD5  F6 C7 10             TEST    BH,PON_ALRM          ; POWERED ON BY ALARM?
0AD8  74 07                JZ      RES_011

0ADA  80 0E 00A0 R 02      OR      RTC_WAIT_FLAG,ALARM_PEND ; SET ALARM PENDING
0ADF  24 DF                AND     AL,NOT AIE_ENABLE    ; TURN OFF ALARM IF POWERED
                                                        ; ON BY IT
0AE1                    RES_011:
                ;
                ; RESTORE REAL TIME CLOCK INTERRUPT MODE
                ;
0AE1  B4 0B                MOV     AH,RTC_MODE
0AE3  E8 0750 R            CALL    PUT_RTC_NMI
                ;
                ; ENABLE PRINTER PORTS
                ;
0AE6  BA 027A              MOV     DX,27AH              ; START WITH SECONDARY
0AE9  B0 0C                MOV     AL,0CH               ;
0AEB  EE                   OUT     DX,AL                ;
0AEC  80 C6 01             ADD     DH,1                 ;
0AEF  EE                   OUT     DX,AL                ; DO PRIMARY
0AF0  BA 03BE              MOV     DX,03BEH             ;
0AF3  EE                   OUT     DX,AL                ; DO MONO PRINTER PORT
                ;
                ; ENABLE SYSTEM SUSPEND NMI
                ;
0AF4  E4 7F                IN      AL,PWR_STAT
0AF6  0C 04                OR      AL,EN_SUS_NMI        ; ENABLE SUSPEND NMI
0AF8  24 F7                AND     AL,NOT HDWR_RESET    ; RESET FLAG
0AFA  E6 7F                OUT     PWR_STAT,AL

                ;
                ; ISSUE RESUME VECTOR CALL
                ;
0AFC                    RES_012:                        ;
0AFC  E8 0000 E            CALL    ENABLE_NMI           ; ENABLE NMI'S
                ;
                ; ISSUE RESUME VECTOR CALL
                ;
0AFF  CD 6C                INT     6CH                  ; ALLOW OP/SYS TO CORRECT
                                                        ; REAL TIME INFORMATION
0B01  E9 0050 R            JMP     NMIH_EXIT            ; EXIT TO USERS PROGRAM
0B04                    RESUME      ENDP

                           SUBTTL  ASYNC SAVE/RESTORE
                ;-----------------------------------------------------------------
                ; ASYNC_SUSPEND        THIS SAVES OR RESTORES THE REGISTERS OF THE
                ;       8250 ASYNC CHIP.  USED IN SUSPEND AND RESUME.
                ;       ES:DI MUST BE SET TO FIRST BYTE ON ENTRY FOR
                ;       SUSPEND, DS:SI MUST BE SET UP TO FIRST BYTE
                ;       FOR RESUME. SI OR DI IS INCREASED BY 14 ON EXIT.
                ;
                ;       THE DATA STATUS BYTE AND THE 8250 REGISTERS FOR
                ;       THE ASYNC PORT ARE STORED IN RAM AS FOLLOWS:
                ;               BYTE 1 - DATA STATUS (0AAH = GOOD)
                ;               BYTE 2 - 8250 LINE CONTROL REG
                ;               BYTE 3 - 8250 MODEM CONTROL REG
                ;               BYTE 4 - 8250 SCRATCH REG
                ;               BYTE 5 - 8250 INTERRUPT CTL REG
                ;               BYTE 6 - 8250 DIVISOR LATCH (MSB)
                ;               BYTE 7 - 8250 DIVISOR LATCH (LSB)
                ;       THE MODEM 8250 REGISTERS AND DATA STATUS BYTE
                ;       ARE STORED IN THE SAME FORMAT IN THE NEXT 7 BYTES.
                ;
                ; INPUT                BL=0 : RESTORES REGISTERS (RESUME)
                ;                      BL=1 : SAVE REGISTERS (SUSPEND)
                ;                      DIRECTION FLAG=0
                ;                      DS:SI = BEGINNING OF DATA AREA FOR RESUME.
                ;                      ES:DI = BEGINNING OF DATA AREA FOR SUSPEND
                ;
                ; REGISTERS USED       BX DESTROYED
                ;                      14 ADDED TO SI (RESUME) OR DI (SUSPEND).
                ;-----------------------------------------------------------------
                ;LOCAL EQUATES
                ;
= 00AA                  GOOD_DATA       EQU     0AAH    ; THIS MEANS THAT THE FOLLOWING 6
                                                        ; BYTES ARE VALID SUSPEND DATA
                ;
0B04                    ASYNC_SUSPEND   PROC    NEAR
                ;
0B04  50                   PUSH    AX                   ; SAVE REGISTERS USED
0B05  52                   PUSH    DX
                ;
                ;-----GET RTC_FEAT_CON AND KYBD_CNTL TO SEE WHO IS INSTALLED
                ;
0B06  B4 1F                MOV     AH,RTC_FEAT_CON      ; GET ASYNC INSTALLATION INFO
0B08  E8 0747 R            CALL    GET_RTC_NMI          ; FROM REAL TIME CLOCK RAM
0B0B  8A E0                MOV     AH,AL                ; SAVE FEATURE REGISTER
0B0D  BA 007C              MOV     DX,KYBD_CNTL         ; ADDRESS THE POWER CONTROL REG
0B10  EC                   IN      AL,DX                ; AND READ IT
0B11  86 C4                XCHG    AL,AH                ; EXCHANGE BYTES
0B13  50                   PUSH    AX                   ; SAVE FOR LATER (REST_MODEM)
                ;
                ;-----GET ADDRESS OF ASYNC PORT, EVEN IF NOT PRESENT
                ;
0B14  BA 02F8              MOV     DX,2F8H              ; ADDRESS OF SECONDARY PORT
0B17  F6 C4 01             TEST    AH,SET_RS232_PRIM    ; TEST FOR ASYNC BEING PRIMARY
0B1A  74 04                JZ      SV1                  ; SKIP IF NOT
0B1C  81 C2 0100           ADD     DX,100H              ; CHANGE TO PRIMARY ADDRESS IF SO
                ;
                ;-----IS ASYNC INSTALLED AND ON?
                ;
0B20  A8 80             SV1:    TEST    AL,SERPLL_INST  ; TEST FOR SER/PAR INSTALLED
0B22  75 02                JNZ     SV2                  ; YES, GO TO NEXT TEST
0B24  2B C0                SUB     AX,AX                ; NO, FORCE TO FAIL NEXT TEST
                ;
0B26  F6 C4 04          SV2:    TEST    AH,ACT_RS232    ; TEST FOR ASYNC POWER ON
0B29  74 02                JZ      SV3                  ; NO, AL=0 TO INDICATE BAD
0B2B  B0 AA                MOV     AL,GOOD_DATA         ; YES, SET AL TO INDICATE GOOD
                ;
                ;-----DO THE SUSPEND OR RESUME
                ;
0B2D  E8 0B47 R         SV3:    CALL    ASY_RES_SUS     ; RESUME OR SUSPEND ASYNC
                ;
                ;-----IS MODEM INSTALLED AND ON?
                ;
0B30                    REST_MODEM:
0B30  58                   POP     AX                   ; RETRIEVE FEATURE AND KYBD REGS
0B31  A8 40                TEST    AL,INTMOD_INST       ; TEST FOR INTERNAL MODEM
```

```
0B33  75 02                 JNZ     SV4               ; YES, DO NEXT TEST
0B35  2B C0                 SUB     AX,AX             ; NO, FORCE FAILURE OF NEXT TEST
                  ;
0B37  F6 C4 02          SV4:    TEST    AH,ACT_MODEM    ; TEST FOR MODEM POWER ON
0B3A  74 02                 JZ      SV5               ; NO, LEAVE AL=0
0B3C  B0 AA                 MOV     AL,GOOD_DATA      ; YES, SET AL=GOOD_DATA
                  ;
                  ;-----GET ADDRESS OF MODEM EVEN IF NOT THERE
                  ;
0B3E  BA 03F8           SV5:    MOV     DX,3F8H         ; ADDRESS MODEM PORT
                  ;
                  ;-----CALL SUSPEND/RESUME
                  ;
0B41  E8 0B47 R             CALL    ASY_RES_SUS       ; RESUME OR SUSPEND MODEM
                  ;
                  ;-----FINISHED
                  ;
0B44  5A                    POP     DX                ; RESTORE REGISTERS
0B45  58                    POP     AX
0B46  C3                    RET
                  ;
0B47                    ASYNC_SUSPEND   ENDP

                  ;-------------------------------------------------------------------
                  ;ASY_RES_SUS            THIS PROCEDURE SAVES OR RESTORES THE ASYNC PORTS
                  ;                       REGISTERS.
                  ;
                  ; INPUT                 AL=0 IF PORT IS NOT POWERED ON.
                  ;                       AL=GOOD_DATA BYTE IF PORT IS WORKING.
                  ;                       BL=0 IF RESUME, BL!=0 IF SUSPEND.
                  ;                       DX=PORT BASE ADDRESS
                  ;
                  ; REGISTERS USED        AL,DX
                  ;--------------------------------------------------------------------------
0B47                    ASY_RES_SUS     PROC    NEAR
                  ;
0B47  02 DB                 ADD     BL,BL             ; ARE WE RESUMING OR SUSPENDING?
0B49  74 2B                 JZ      ASY_RESUME        ; RESUMING
                  ;
                  ;-----SUSPEND COMMAND
                  ;
0B4B                    ASY_SUSPEND:
0B4B  AA                    STOSB                     ; STORE FIRST BYTE (GOOD OR BAD)
0B4C  22 C0                 AND     AL,AL             ; IS PORT WORKING?
0B4E  74 22                 JZ      AS1               ; NO, DO NOT STORE REGISTERS
                  ;                               YES, STORE ALL REGISTERS
                  ;
0B50  42                    INC     DX                ; ADDRESS THE LINE CTL REG (XF9)
0B51  42                    INC     DX                ;                          (XFA)
0B52  42                    INC     DX                ;                          (XFB)
0B53  EC                    IN      AL,DX             ; READ FROM 8250
0B54  AA                    STOSB                     ; AND STORE IN DATA AREA
                  ;
0B55  24 7F                 AND     AL,7FH            ; ASSURE THAT DLAB BIT IS ZERO
0B57  EE                    OUT     DX,AL
                  ;
0B58  42                    INC     DX                ; ADDRESS MODEM CONTROL REG (XFC)
0B59  EC                    IN      AL,DX             ; READ
0B5A  AA                    STOSB                     ; STORE IN DATA AREA
                  ;
0B5B  42                    INC     DX                ; ADDRESS SCRATCH REG       (XFD)
0B5C  42                    INC     DX                ;                           (XFE)
0B5D  42                    INC     DX                ;                           (XFF)
0B5E  EC                    IN      AL,DX             ; READ FROM 8250
0B5F  AA                    STOSB                     ; STORE IN RAM

0B60  83 EA 06              SUB     DX,6              ; ADDRESS INTR ENABLE REG   (XF9)
0B63  EC                    IN      AL,DX             ; READ FROM 8250
0B64  AA                    STOSB                     ; STORE IN RAM

0B65  42                    INC     DX                ;                           (XFA)
0B66  42                    INC     DX                ; ADDRESS LINE CTL REG AGAIN (XFB)
0B67  B0 80                 MOV     AL,80H            ; SET DLAB BIT TO 1 TO READ BAUD
0B69  EE                    OUT     DX,AL             ;
```

```
0B6A  4A                DEC     DX              ;                          (XFA)
0B6B  4A                DEC     DX              ; ADDRESS BAUD MSB         (XF9)
0B6C  EC                IN      AL,DX           ; READ FROM 8250
0B6D  AA                STOSB                   ; SAVE
0B6E  4A                DEC     DX              ; ADDRESS OTHER BAUD BYTE  (XF8)
0B6F  EC                IN      AL,DX           ; READ FROM 8250
0B70  AA                STOSB                   ; SAVE
0B71  C3                RET                     ; EXIT
0B72  83 C7 06       AS1:    ADD     DI,6       ; SKIP OVER DATA AREA
0B75  C3                RET                     ; EXIT
                ;
                ;-----RESUME COMMAND
                ;
0B76                 ASY_RESUME:
0B76  22 C0             AND     AL,AL           ; IS PORT OPERATING?
0B78  74 2F             JZ      AS2             ; NO, DO NOT RESTORE
0B7A  AC                LODSB                   ; YES, LOOK AT STORED DATA
0B7B  3C AA             CMP     AL,GOOD_DATA    ; IS IT GOOD?
0B7D  75 2B             JNE     AS3             ; NO, DO NOT RESTORE
                                                ; YES, SO RESTORE 8250
0B7F  AC                LODSB                   ; GET LINE CONTROL REG FROM RAM
0B80  50                PUSH    AX              ; STORE ON STACK
                ;
0B81  42                INC     DX              ;                       (XF9)
0B82  42                INC     DX              ;                       (XFA)
0B83  42                INC     DX              ; ADDRESS LINE CTL REG (XFB)
0B84  2A C0             SUB     AL,AL           ; SET DLAB BIT 0
0B86  EE                OUT     DX,AL           ; TO ALLOW NORMAL ADDRESSING

0B87  42                INC     DX              ; ADDRESS MODEM CONTROL REG(XFC)
0B88  AC                LODSB                   ; GET FROM RAM
0B89  EE                OUT     DX,AL           ; AND RESTORE 8250

0B8A  42                INC     DX              ;                       (XFD)
0B8B  42                INC     DX              ;                       (XFE)
0B8C  42                INC     DX              ; ADDRESS SCRATCH REG   (XFF)
0B8D  AC                LODSB                   ; GET FROM RAM
0B8E  EE                OUT     DX,AL           ; AND WRITE TO 8250

0B8F  83 EA 07          SUB     DX,7            ; ADDRESS DATA REG      (XF8)
0B92  EC                IN      AL,DX           ; GET ANY TRASH CHRS
0B93  EC                IN      AL,DX

0B94  42                INC     DX              ; ADDRESS INTR CTL REG (XF9)
0B95  AC                LODSB                   ; GET FROM RAM
0B96  EE                OUT     DX,AL           ; AND WRITE TO 8250

0B97  42                INC     DX              ;                       (XFA)
0B98  42                INC     DX              ; ADDR LINE CONTROL REG(XFB)
0B99  B0 80             MOV     AL,80H          ; SET DLAB BIT = 1
0B9B  EE                OUT     DX,AL           ; IN 8250

0B9C  4A                DEC     DX              ;                       (XFA)
0B9D  4A                DEC     DX              ;                       (XF9)
0B9E  AC                LODSB                   ; GET BAUD MSB FROM RAM
0B9F  EE                OUT     DX,AL           ; AND INTO 8250
0BA0  4A                DEC     DX              ; ADDR BAUD LSB         (XF8)
0BA1  AC                LODSB                   ; GET FROM RAM
0BA2  EE                OUT     DX,AL           ; AND INTO 8250

0BA3  42                INC     DX              ;                       (XF9)
0BA4  42                INC     DX              ;                       (XFA)
0BA5  42                INC     DX              ; ADDRE LINE CONTROL AGAIN (XFB)
0BA6  58                POP     AX              ; PULL OFF STACK
0BA7  EE                OUT     DX,AL           ; AND PUT IN 8250

0BA8  C3                RET                     ;

0BA9  46             AS2:    INC     SI         ; CORRECT ADDRESS COUNT
0BAA  83 C6 06       AS3:    ADD     SI,6       ;
0BAD  C3                RET
```

```
0BAE                    ASY_RES_SUS     ENDP
                ;--------------------------------------------------------------------------------

0BAE                    ROMCODE         ENDS
                        END
```

Keyboard Services (B11KYBD)

```
                TITLE B11KYBD BIOS KEYBOARD ROUTINES

0000                   ROMCODE SEGMENT BYTE PUBLIC
                          IDENT   B11KYBD,I1,00

              ;***********************************************************************
              ;
              ; MODULE-NAME :     B11KYBD
              ;
              ; DATE LAST MODIFIED:  9/16/85
              ;
              ; DESCRIPTIVE-NAME : THIS MODULE CONTAINS THE BIOS INTERRUPT 9
              ;                    KEYBOARD HANDLER AND THE ASSOCIATED INTERRUPT 16
              ;                    KEYBOARD SERVICE ROUTINES.
              ;
              ; COPYRIGHT : 7396-917 (C) COPYRIGHT IBM CORP. 1985
              ;             REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
              ;
              ; CHANGE LEVEL: 0.0
              ;
              ; FUNCTION:   KYBD_INT9   - INTERRUPT 9 KEYBOARD HANDLER (HARDWARE INT 1)
              ;             ROUTINE TO READ SCAN CODES FROM PORT 60H AND
              ;             CONVERT THEM TO ASCII CODES AND QUEUE IN THE
              ;             BIOS KEYBOARD BUFFER.
              ;
              ;             KYBD_IO     - KEYBOARD I/O ROUTINES TO ACCESS THE ASCII
              ;             KEYBOARD BUFFER. (INT 16H)
              ;
              ; MODULE SIZE:   945 BYTES
              ;
              ;
              ; EXTERNALLY REFERENCED ROUTINES: REFER TO EXTRN LIST
              ;
              ; EXTERNALLY REFERENCED DATA AREAS: REFER TO EXTRN LIST
              ;
              ; CHANGE ACTIVITY:  NONE
              ;***********************************************************************

              ;*****************************************************************
              ;*    P U B L I C S                                             *
              ;*****************************************************************

                          PUBLIC  KYBD_IO
                          PUBLIC  KYBD_INT9

              ;*****************************************************************
              ;*    E X T E R N A L   R E F E R E N C E S                     *
              ;*****************************************************************
              ;
              ; ROUTINES
              ;
                          EXTRN   DCS:NEAR
                          EXTRN   START:NEAR
                          EXTRN   KB_NOISE:NEAR
              ;
              ; TABLES
              ;
                          EXTRN   K6:BYTE
                          EXTRN   K6L:ABS
                          EXTRN   K7:BYTE
                          EXTRN   K8:BYTE
                          EXTRN   K9:BYTE
                          EXTRN   K10:BYTE
```

```
                                EXTRN   K11:BYTE
                                EXTRN   K12:BYTE
                                EXTRN   K13:BYTE
                                EXTRN   K14:BYTE
                                EXTRN   K15:BYTE
                                EXTRN   K30:BYTE

                  ; LOCAL EQUATES

= 0035                  P60_SLASH_SC    EQU   35H      ; P60 MAKE SC FOR BASE /
= 0037                  P60_ASTRK_SC    EQU   37H      ; P60 MAKE SC FOR BASE *
= 0057                  F11_MAKE_SC     EQU  057H      ; F11 MAKE SC AT NMI & P60 LEVEL
= 0058                  F12_MAKE_SC     EQU  058H      ; F12 MAKE SC AT NMI & P60 LEVEL
= 0085                  F11_BASE_ASCII  EQU  085H      ; F11 EXTENDED ASCII (BASE)
```

Keyboard Interrupt Hex 16 (KYBD_IO)

```
;**** INT 16H **********************************************************
;
; ROUTINE NAME: KYBD_IO   (INT 16H)
;
; FUNCTION: PROVIDE ACCESS TO THE ASCII KEYBOARD BUFFER AND CLICKER.
;
; INPUT CONDITIONS:
;       (AH)=0  PROCESS SYSTEM REQUEST KEY IF THE KEY IS ACTIVE ELSE
;               READ THE NEXT ASCII CHARACTER STRUCK FROM THE KEYBOARD
;               RETURN THE RESULT IN (AL), SCAN CODE IN (AH)
;
;       (AH)=1  PROCESS SYSTEM REQUEST KEY IF THE KEY IS ACTIVE ELSE
;               RESET THE Z FLAG TO INDICATE IF AN ASCII 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 REGISTER
;               THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
;               THE EQUATES FOR KB_FLAG
;
;       (AH)=4  TURN ON/OFF KEYBOARD CLICK STATE 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.
;
; NOTE:  FUNCTION CALLS OTHER THAN THE ABOVE RESULT IN NO OPERATION
;
; EXIT CONDITIONS:  FUNCTIONS 0, 2, 4 - IRET BACK TO CALLER
;                   FUNCTION  1       - RET 2 BACK TO CALLER
;
; REGISTERS MODIFIED:  AX, Z FLAG
;
; INTERRUPTS:  FOR FUNCTIONS 2 AND 4, I FLAG IS LEFT AS IS
;              FOR FUNCTION 0, INTERRUPTS ARE FORCED OFF (CLI) AND FORCED 0
;                  (STI) BEFORE DOING IRET
;              FOR FUNCTION 1, INTERRUPTS ARE FORCED OFF (CLI) AND FORCED 0
;                  (STI) BEFORE DOING RET 2
;
; INTERNALLY REFERENCED ROUTINES:   PTR_INC
;
; EXTERNALLY REFERENCED ROUTINES:   DDS, INT 15H
;***********************************************************************
```

```
                          ASSUME  cs:ROMCODE,CS:DATA
0000                   KYBD_IO PROC    FAR
0000  1E                  PUSH    DS                  ; SAVE CURRENT DS
0001  53                  PUSH    BX                  ; SAVE BX TEMPORARILY
0002  E8 0000 E           CALL    DDS
0005  0A E4               OR      AH,AH               ; AH=0
0007  74 10               JZ      K1                  ; ASCII_READ
0009  FE CC               DEC     AH                  ; AH=1
000B  74 2B               JZ      K2                  ; ASCII_STATUS
000D  FE CC               DEC     AH                  ; AH=2
000F  74 38               JZ      K3                  ; SHIFT_STATUS
0011  80 EC 02            SUB     AH,2                ; AH=4
0014  74 39               JZ      K4                  ; CLICK_STATE
0016  EB 49 90            JMP     KYBD_END            ; EXIT

               ;
               ;----- READ THE KEY TO FIGURE OUT WHAT TO DO
               ;
0019                   K1:                            ; ASCII READ
0019  FA                  CLI                         ; INTERRUPTS OFF
001A  8B 1E 001A R        MOV     BX,BUFFER_HEAD      ; GET POINTER TO HEAD OF BUFFER
001E  3B 1E 001C R        CMP     BX,BUFFER_TAIL      ; CHECK FOR HEAD = TAIL
0022  75 08               JNE     K1_A                ; JUMP IF SOMETHING IN BUFFER
               ;
               ; ISSUE KEYBOARD BUSY WAIT
               ;
0024  B8 9002             MOV     AX,09002H           ; CALL KEYBOARD BUSY HANDLER
0027  CD 15               INT     15H
0029  FB                  STI                         ; INTERRUPTS ON
               ;
               ; BIOS INT 15H WILL RETURN HERE WHEN HEAD <> TAIL
               ;
002A  EB ED               JMP     K1                  ; LOOP BACK TO WAIT FOR BFR NOT EMPTY
               ;
               ; DATA IN KEYBOARD BUFFER
               ;
002C                   K1_A:
002C  8B 07               MOV     AX,[BX]             ; GET SCAN CODE AND ASCII CODE
002E  E8 03A4 R           CALL    PTR_INC             ; MOVE POINTER TO NEXT POSITION
0031  89 1E 001A R        MOV     BUFFER_HEAD,BX      ; STORE VALUE IN VARIABLE
0035  EB 2A 90            JMP     KYBD_END            ; RETURN
               ;
               ;----- ASCII STATUS
               ;
0038                   K2:
0038  FA                  CLI                         ; INTERRUPTS OFF
0039  8B 1E 001A R        MOV     BX,BUFFER_HEAD      ; GET HEAD POINTER
003D  3B 1E 001C R        CMP     BX,BUFFER_TAIL      ; IF EQUAL (Z=1) THEN NOTHING THERE
0041  8B 07               MOV     AX,[BX]
0043  FB                  STI                         ; RE-ENABLE INTERRUPTS
0044  5B                  POP     BX                  ; RECOVER REGISTER
0045  1F                  POP     DS                  ; RECOVER SEGMENT
0046  CA 0002             RET     2                   ; THROW AWAY FLAGS
               ;
               ;----- SHIFT STATUS
               ;
0049                   K3:
0049  A0 0017 R           MOV     AL,KB_FLAG          ; GET THE SHIFT STATUS FLAGS
004C  EB 13 90            JMP     KYBD_END            ; RETURN
               ;
               ;----- CLICK STATE
               ;
004F                   K4:
004F  0A C0               OR      AL,AL               ; TURN OFF KEYBOARD CLICK ?
0051  75 05               JNZ     K5                  ; JUMP FOR RANGE CHECK
0053  80 26 00B4 R F7     AND     KB_NMI_CNTL,NOT CLICK_ON ; TURN OFF CLICK
0058                   K5:
0058  3C 01               CMP     AL,1                ; RANGE CHECK
005A  75 05               JNE     KYBD_END            ; NOT IN RANGE, RETURN
005C  80 0E 00B4 R 08     OR      KB_NMI_CNTL,CLICK_ON ; TURN ON KEYBOARD CLICK

0061                   KYBD_END:
0061  5B                  POP     BX                  ; RECOVER REGISTER
```

```
0062  1F                POP     DS                    ; RECOVER REGISTERS
0063  CF                IRET                          ; RETURN TO CALLER

0064               KYBD_IO ENDP
```

Level 1 Interrupt Hex 9 (KYBD_INT9)

```
                ;** INT 9 ************************************************************
                ;
                ; ROUTINE-NAME :  KYBD_INT9
                ;
                ; FUNCTION:   THIS ROUTINE IS ACTIVATED BY A INTERRUPT 9 (HARDWARE INT 1)
                ;     IT READS PORT 60 AND PROCESSES THAT SCAN CODE BY EITHER SETTING/
                ;     RESETTING ITS FLAG OR TRANSLATING THE SCAN CODE INTO AN EXTENDED
                ;     ASCII CODE AND PLACING IT IN THE ASCII BUFFER.  THIS INTERRUPT
                ;     SERVICE ROUTINE TRIGGERS THE KYBD_CLR NMI.  THE NMI DOES NOT
                ;     GO INTO EFFECT UNTIL A NON-SPECIFIC EOI IS DONE NEAR THE EXIT OF
                ;     THIS ROUTINE.
                ;
                ; ENTRY CONDITIONS:
                ;      PURPOSE OF ENTRY: PROCESS THE SCAN CODE IN PORT 60H
                ;      INPUT CONDITIONS: cs: ROMCODE SEGMENT
                ;      RESTRICTIONS:
                ;
                ; EXIT CONDITIONS:
                ;     NORMAL EXIT CONDITIONS: ASCII BFR HAS 2 BYTE EXTENDED ASCII CODE
                ;     ERROR EXIT CONDITIONS:  N/A
                ;     REGISTERS MODIFIED:     NONE
                ;
                ; INTERRUPTS:  FORCED ON UPON ENTRY (STI)
                ;
                ; INTERNALLY REFERENCED ROUTINES:   PTR_INC
                ;
                ; EXTERNALLY REFERENCED ROUTINES:   DDS, INT 5H, KB_NOISE, INT 15H, INT 1BH,
                ;
                ;**********************************************************************

0064               KYBD_INT9 PROC    FAR
0064  FB                STI                           ; ALLOW FURTHER INTERRUPTS
0065  50                PUSH    AX
0066  53                PUSH    BX
0067  51                PUSH    CX
0068  52                PUSH    DX
0069  56                PUSH    SI
006A  57                PUSH    DI
006B  1E                PUSH    DS
006C  06                PUSH    ES
006D  FC                CLD                           ; FORWARD DIRECTION
006E  E8 0000 E         CALL    DDS
0071  E4 60             IN      AL,KB_DATA            ; READ IN THE CHARACTER
0073  50                PUSH    AX                    ; SAVE IT
0074  E4 61             IN      AL,KB_CTL             ; GET THE CONTROL PORT
0076  8A E0             MOV     AH,AL                 ; SAVE VALUE
0078  0C 80             OR      AL,80H                ; RESET BIT FOR KEYBOARD
007A  E6 61             OUT     KB_CTL,AL
007C  86 E0             XCHG    AH,AL                 ; GET BACK ORIGINAL CONTROL
007E  E6 61             OUT     KB_CTL,AL             ; KB HAS BEEN RESET
                                                      ; NOTE: THIS WILL GENERATE A
                                                      ; KB_CLR NMI UPON INT9 EOI.
0080  58                POP     AX                    ; RECOVER SCAN CODE
                ;
                ; ALLOW OPERATING SYSTEM INTERCEPT  (INT 15 FUNCTION 4FH)
                ;                     (AH = 4FH , AL = SCAN CODE)
                ;
0081  B4 4F             MOV     AH,4FH
```

```
0083  F9                    STC                          ; PRE SET CARRY FOR INTERCEPT CHECK
0084  CD 15                 INT     15H
0086  73 60                 JNC     K14_S4               ; IF NO CARRY THEN VECTOR INTERCEPT
                                                         ; ELSE PROCESS KEY HERE:
0088  8A E0                 MOV     AH,AL                ; SAVE SCAN CODE IN AH ALSO
                  ;----------------------
                  ; KEYPAD /,*  MAKES    -
                  ;----------------------
008A  F6 06 0096 R 02       TEST    KB_FLAG_3,LC_HC      ; HAS AN E0H BEEN PROCESSED?
008F  74 21                 JZ      K9_1                 ; NO, JUMP
0091  80 36 0096 R 02       XOR     KB_FLAG_3,LC_HC      ; YES, RESET FLAG
0096  80 FC 35              CMP     AH,P60_SLASH_SC      ; KEYPAD / KEY?
0099  74 05                 JE      K6_1                 ; YES, JUMP
009B  80 FC 37              CMP     AH,P60_ASTRK_SC      ; KEYPAD * KEY?
009E  75 12                 JNE     K9_1                 ; NO, JUMP
00A0                     K6_1:
00A0  F6 06 0018 R 08       TEST    KB_FLAG_1,HOLD_STATE ; IN HOLD STATE?
00A5  75 03                 JNZ     K6_5                 ; YES,AROUND
00A7  E9 032D R             JMP     K55                  ; NO,JUMP LEAVING /,* AS IS
00AA  80 26 0018 R F7    K6_5:  AND     KB_FLAG_1,NOT HOLD_STATE ; RST HLD ST, DISCARD KEY
00AF  E9 0396 R             JMP     KYBD9_EXIT           ; EXIT INT9 ROUTINE
                  ;----------------------
                  ; HIDDEN CODE  -  E0H  -
                  ;----------------------
00B2  3C E0              K9_1:  CMP     AL,HIDN_CODE_E0 ; IS P60 SC A HIDDEN CODE?
00B4  75 08                 JNE     K10_1                ; NO, JUMP
00B6  80 0E 0096 R 02       OR      KB_FLAG_3,LC_HC      ; SET FLAG
00BB  E9 0396 R             JMP     KYBD9_EXIT           ; EXIT ROUTINE
                  ;-----------------------
                  ; OVERRUN SC   -  FFH   -
                  ;-----------------------
00BE  3C FF              K10_1:  CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR
00C0  75 03                 JNZ     K14_S1               ; NO, TEST FOR SYS REQ KEY
00C2  E9 0386 R             JMP     K62                  ; BUFFER_FULL_BEEP
                  ;-------------------------------------------------
                  ; SYSTEM REQUEST MAKE/BREAK  (INT 15H  FUNCTION 85H) -
                  ;-------------------------------------------------
00C5                     K14_S1:
00C5  3C 54                 CMP     AL,SYSREQ_MAKE       ; SYSTEM REQUEST KEY MAKE ?
00C7  75 10                 JNE     K14_S2               ; NO, GO LOOK FOR BREAK
00C9  F6 06 0018 R 04       TEST    KB_FLAG_1,SYS_SHIFT  ; ALREADY DEPRESSED ?
00CE  75 18                 JNZ     K14_S4               ; IF SO THEN THROW AWAY MAKE
00D0  80 0E 0018 R 04       OR      KB_FLAG_1,SYS_SHIFT  ; SET SYS REQ DEPRESSED FLAG
00D5  B0 00                 MOV     AL,00                ; SET MAKE FLAG
00D7  EB 0B                 JMP     SHORT K14_S3         ; GO DO THE INT 15H
00D9                     K14_S2:
00D9  3C D4                 CMP     AL,SYSREQ_BREAK      ; SYSTEM REQUEST KEY BREAK ?
00DB  75 0E                 JNE     K16                  ; NO, GO TEST FOR SHIFT KEYS
00DD  80 26 0018 R FB       AND     KB_FLAG_1,NOT SYS_SHIFT ; RESET SYS REQ DEPRESSED FLAG
00E2  B0 01                 MOV     AL,01                ; SET REQUEST BREAK FLAG

00E4                     K14_S3:
00E4  B4 85                 MOV     AH,85H               ; SET SYSTEM REQUEST NOTIFICATION
00E6  CD 15                 INT     15H
00E8                     K14_S4:
00E8  E9 0396 R             JMP     KYBD9_EXIT           ; EXIT
                  ;
                  ;----- TEST FOR SHIFT KEYS
                  ;
00EB                     K16:                            ; TEST_SHIFT
00EB  24 7F                 AND     AL,07FH              ; TURN OFF THE BREAK BIT
00ED  0E                    PUSH    CS
00EE  07                    POP     ES                   ; ESTABLISH ADDRESS OF SHIFT TABLE
00EF  BF 0000 E             MOV     DI,OFFSET K6         ; SHIFT KEY TABLE
00F2  B9 0000 E             MOV     CX,OFFSET K6L        ; LENGTH
00F5  F2/ AE                REPNE   SCASB                ; LOOK THROUGH THE TABLE FOR A MATCH
00F7  8A C4                 MOV     AL,AH                ; RECOVER SCAN CODE
00F9  74 03                 JE      K17                  ; JUMP IF MATCH FOUND
00FB  E9 0187 R             JMP     K24_0                ; IF NO MATCH, THEN SHIFT NOT FOUND
                  ;
                  ;------ SHIFT KEY FOUND
                  ;
00FE  81 EF 0001 E       K17:   SUB     DI,OFFSET K6+1 ; ADJUST PTR TO SCAN CODE MTCH
0102  2E: 8A A5 0000 E      MOV     AH,cs:K7[DI]           ; GET MASK INTO AH
```

```
0107  A8 80                 TEST    AL,80H              ; TEST FOR BREAK KEY
0109  75 54                 JNZ     K23                 ; BREAK_SHIFT_FOUND
                ;------------------------------------------
                ; SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE   -
                ;------------------------------------------
010B  80 FC 10              CMP     AH,SCROLL_SHIFT
010E  73 0A                 JAE     K18                 ; IF SCRL SFT OR ABOVE, TOGGLE KEY
                ;
                ;----- PLAIN SHIFT KEY, SET SHIFT ON
                ;
0110  08 26 0017 R          OR      KB_FLAG,AH          ; TURN ON SHIFT BIT
0114  E9 0396 R             JMP     KYBD9_EXIT          ; INTERRUPT_RETURN
                ;
                ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
                ;
0117  E9 01C5 R         K25_JMP: JMP    K25             ; JUMP TO K25 FOR JNZ's BELOW
011A                    K18:                            ; SHIFT-TOGGLE
011A  F6 06 0017 R 04       TEST    KB_FLAG, CTL_SHIFT  ; CHECK CTL SHIFT STATE
011F  75 F6                 JNZ     K25_JMP             ; JUMP IF CTL STATE
0121  3C 52                 CMP     AL,INS_KEY          ; CHECK FOR INSERT KEY
0123  75 22                 JNE     K22                 ; JUMP IF NOT INSERT KEY
                                                        ; [[[ INSERT KEY HIT ]]]]
0125  F6 06 0017 R 08       TEST    KB_FLAG, ALT_SHIFT  ; CHECK FOR ALTERNATE SHIFT
012A  75 EB                 JNZ     K25_JMP             ; JUMP IF ALTERNATE SHIFT
012C  F6 06 0017 R 20   K19:    TEST    KB_FLAG, NUM_STATE ; CHECK FOR BASE STATE
0131  75 0D                 JNZ     K21                 ; JUMP IF NUM LOCK IS ON
0133  F6 06 0017 R 03       TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
0138  74 0D                 JZ      K22                 ; JUMP IF BASE STATE

013A                    K20:                            ; NUMERIC ZERO, NOT INSERT KEY
013A  B8 5230               MOV     AX, 5230H           ; PUT OUT AN ASCII ZERO
013D  E9 0334 R             JMP     K57                 ; BUFFER_FILL
0140                    K21:                            ; MIGHT BE NUMERIC
0140  F6 06 0017 R 03       TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
0145  74 F3                 JZ      K20                 ; JUMP NUMERIC, NOT INSERT

0147                    K22:                            ; SHFT TOGGLE KEY HIT; PROCESS
0147  84 26 0018 R          TEST    AH,KB_FLAG_1        ; IS KEY ALREADY DEPRESSED
014B  75 37                 JNZ     KYBD9_EXIT1         ; JUMP IF KEY ALREADY DEPRESSED
014D  08 26 0018 R          OR      KB_FLAG_1,AH        ; IND THAT THE KEY IS DEPRESSED
0151  30 26 0017 R          XOR     KB_FLAG,AH          ; TOGGLE THE SHIFT STATE
0155  3C 52                 CMP     AL,INS_KEY          ; TEST 1ST MAKE OF INSERT KEY
0157  75 2B                 JNE     KYBD9_EXIT1         ; JUMP IF NOT INSERT KEY
0159  B8 5200               MOV     AX,INS_KEY*256      ; SET CODE INTO AH, 0 INTO AL
015C  E9 0334 R             JMP     K57                 ; PUT INTO OUTPUT BUFFER
                ;------------------------------------------
                ; SHIFT BREAK FOUND                           -
                ;------------------------------------------
015F                    K23:                            ; BREAK-SHIFT-FOUND
015F  80 FC 10              CMP     AH,SCROLL_SHIFT     ; IS THIS A TOGGLE KEY
0162  73 1A                 JAE     K24                 ; YES, HANDLE BREAK TOGGLE
0164  F6 D4                 NOT     AH                  ; INVERT MASK
0166  20 26 0017 R          AND     KB_FLAG,AH          ; TURN OFF SHIFT BIT
016A  3C B8                 CMP     AL,ALT_KEY+80H      ; IS THIS ALT SHIFT RELEASE
016C  75 16                 JNE     KYBD9_EXIT1         ; GO TO EOI EXIT
                ;-----------------------------------------------------------
                ; ALT_SHIFT RELEASED,  PROCESS ALT + KEYPAD (0-9) IF PENDING   -
                ;-----------------------------------------------------------
016E  A0 0019 R             MOV     AL,ALT_INPUT
0171  B4 00                 MOV     AH,0                ; BUILD PSEUDO SC
0173  88 26 0019 R          MOV     ALT_INPUT,AH        ; CLEAR ALT_INPUT
0177  3C 00                 CMP     AL,0                ; WAS ALT_INPUT HOLDING?
0179  74 09                 JE      KYBD9_EXIT1         ; NO, EOI EXIT
017B  E9 033D R             JMP     K58                 ; YES, T, SO PUT IN BUFFER
                ;
                ; ----- SHIFT BREAK IS A STATE KEY - TOGGLE FLAG
                ;
017E                    K24:                            ; BREAK-TOGGLE
017E  F6 D4                 NOT     AH                  ; INVERT MASK
0180  20 26 0018 R          AND     KB_FLAG_1,AH        ; INDICATE STATE EXITED
                ;
                ; LINK TO KEYBOARD EXIT FOR SHORT JUMPS
```

```
                  ;
0184                         KYBD9_EXIT1:
0184  E9 0396 R                 JMP     KYBD9_EXIT          ; INTERRUPT_RETURN
                  ;
                  ;----- TEST FOR F11 OR F12 MAKE
                  ;
0187                         K24_0:
0187  B4 85                     MOV     AH,F11_BASE_ASCII   ; AH HAS BASE F11 EXT ASCII
0189  3C 57                     CMP     AL,F11_MAKE_SC      ; F11 MAKE?
018B  74 06                     JE      K24_1               ; YES, JUMP
018D  FE C4                     INC     AH                  ; AH HAS BASE F12 EXT ASCII
018F  3C 58                     CMP     AL,F12_MAKE_SC      ; F12 MAKE?
0191  75 32                     JNE     K25                 ; NO, JUMP
                  ;----------------------------------------------------------------
                  ; F11 / F12  FOUND  -   CHECK FOR   ALT+,  CTL+,  SHIFT+ F11/F12   -
                  ;----------------------------------------------------------------
0193                         K24_1:
0193  F6 06 0018 R 08           TEST    KB_FLAG_1,HOLD_STATE ; HOLD STATE?
0198  74 08                     JZ      K24_5               ; NO, AROUND
019A  80 26 0018 R F7           AND     KB_FLAG_1,NOT HOLD_STATE ; YES, ENTERED KEY ONLY
019F  E9 0396 R                 JMP     KYBD9_EXIT          ; RESETS HOLD STATE
01A2                         K24_5:
01A2  F6 06 0017 R 0F           TEST    KB_FLAG,ALT_SHIFT+CTL_SHIFT+LEFT_SHIFT+RIGHT_SHIFT
                                                            ; ANY SHIFT STATES?
01A7  74 17                     JZ      K24_7               ; NO, AROUND WITH BASE F11/F12
01A9  80 C4 06                  ADD     AH,6                ; AH HAS ALT+F11/F12 EXT ASCII
01AC  F6 06 0017 R 08           TEST    KB_FLAG,ALT_SHIFT   ; ALT SHIFT CASE?
01B1  75 0D                     JNZ     K24_7               ; YES, JUMP
01B3  80 EC 02                  SUB     AH,2                ; AH HAS CTL+F11/F12 EXT ASCII
01B6  F6 06 0017 R 04           TEST    KB_FLAG,CTL_SHIFT   ; CTL SHIFT CASE?
01BB  75 03                     JNZ     K24_7               ; YES, JUMP
01BD  80 EC 02                  SUB     AH,2                ; AH HAS SHIFT+F11/F12 EXT ASCII
01C0                         K24_7:
01C0  2A C0                     SUB     AL,AL               ; CLEAR AL TO MAKE AX EXTNDED SC
01C2  E9 0364 R                 JMP     K61                 ; GO BUFFER F11/F12 EXT ASCII
                  ;-------------------------
                  ; TEST FOR HOLD STATE     -
                  ;-------------------------
01C5                         K25:                           ; NO-SHIFT-FOUND
01C5  8A E0                     MOV     AH,AL               ; RESTORE AH AFTER F11/F12 TEST
01C7  3C 80                     CMP     AL,80H              ; BREAK SC?
01C9  73 B9                     JAE     KYBD9_EXIT1         ; YES, EOI EXIT
                                                            ; <<<<<<<<<          >>>>>>>
                                                            ; < NO BREAK CODES PAST HERE >>
                                                            ; <<<<<<<<<          >>>>>>>
01CB  F6 06 0018 R 08           TEST    KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE
01D0  74 0C                     JZ      K28                 ; BRANCH AROUND TEST IF NOT
01D2  3C 45                     CMP     AL,NUM_KEY
01D4  74 AE                     JE      KYBD9_EXIT1         ; CAN'T END HOLD ON NUM_LOCK
01D6  80 26 0018 R F7           AND     KB_FLAG_1,NOT HOLD_STATE ; TURN OFF HOLD STATE BIT
01DB  E9 0396 R                 JMP     KYBD9_EXIT
                  ;
                  ;----- NOT IN  HOLD STATE, TEST FOR SPECIAL CHARS
                  ;
01DE                         K28:                           ; NO-HOLD-STATE
01DE  F6 06 0017 R 08           TEST    KB_FLAG,ALT_SHIFT   ; ARE WE IN ALTERNATE SHIFT
01E3  75 03                     JNZ     K29                 ; JUMP IF ALTERNATE SHIFT
01E5  EB 6D 90                  JMP     K38                 ; JUMP IF NOT ALTERNATE
                  ;-------------------------------------------------------
                  ; TEST FOR RESET KEY SEQUENCE   (CTL+ALT+DEL)           -
                  ;-------------------------------------------------------
01E8                         K29:                           ; TEST-RESET
01E8  F6 06 0017 R 04           TEST    KB_FLAG,CTL_SHIFT   ; ARE WE IN CONTROL SHIFT ALSO
01ED  74 0D                     JZ      K31                 ; NO_RESET
01EF  3C 53                     CMP     AL,DEL_KEY          ; SHIFT STATE THERE, TEST KEY
01F1  75 09                     JNE     K31                 ; NO_RESET
                  ;
                  ; CTL-ALT-DEL  ENTERED, DO I/O CLEANUP FOR SOFT START
                  ;
01F3  C7 06 0072 R 1234         MOV     RESET_FLAG, 1234H   ; SET FLAG FOR RESET FUNCTION
01F9  E9 0000 E                 JMP     START               ; JUMP TO POWER ON DIAGNOSTICS
                  ;--------------------------------
                  ; IN  ALT_SHIFT  STATE  AT LEAST -
```

```
                ;-------------------------------
01FC                     K31:                              ; NO-RESET
01FC  3C 39                 CMP     AL,57                  ; TEST FOR SPACE KEY
01FE  75 05                 JNE     K32                    ; NOT THERE
0200  B0 20                 MOV     AL,' '                 ; SET SPACE CHAR
0202  E9 0334 R             JMP     K57                    ; BUFFER_FILL
                ;
                ;----- LOOK FOR KEY PAD ENTRY             --- ALT + (KEYPAD 0-9) MAKE --
                ;
0205                     K32:                              ; ALT-KEY-PAD
0205  BF 0000 E             MOV     DI,OFFSET K30          ; ALT-INPUT-TABLE
0208  B9 000A               MOV     CX,10                  ; LOOK FOR ENTRY USING KEYPAD
020B  F2/ AE                REPNE   SCASB                  ; LOOK FOR MATCH
020D  75 13                 JNE     K33                    ; NO_ALT_KEYPAD
020F  81 EF 0001 E          SUB     DI,OFFSET K30+1        ; DI NOW HAS ENTRY VALUE
0213  A0 0019 R             MOV     AL,ALT_INPUT           ; GET THE CURRENT BYTE
0216  B4 0A                 MOV     AH,10                  ; MULTIPLY BY 10
0218  F6 E4                 MUL     AH
021A  03 C7                 ADD     AX,DI                  ; ADD IN THE LATEST ENTRY
021C  A2 0019 R             MOV     ALT_INPUT,AL           ; STORE IT AWAY
021F  E9 0396 R             JMP     KYBD9_EXIT             ; THROW AWAY THAT KEYSTROKE
                ;
                ;----- LOOK FOR SUPERSHIFT ENTRY          -- ALT + A-Z TYPEWRITER MAKE -
                ;
0222                     K33:                              ; NO-ALT-KEYPAD
0222  C6 06 0019 R 00       MOV     ALT_INPUT,0            ; ZERO PREVIOUS ENTRY IN INPUT
0227  B9 001A               MOV     CX,26                  ; DI,ES ALREADY POINTING
022A  F2/ AE                REPNE   SCASB                  ; LOOK FOR MATCH IN ALPHABET
022C  75 05                 JNE     K34                    ; NOT FOUND, FN KEY OR OTHER
022E  B0 00                 MOV     AL,0                   ; ASCII CODE OF ZERO
0230  E9 0334 R             JMP     K57                    ; PUT IT IN THE BUFFER
                ;
                ;----- CHECK FOR TOP ROW KEYS              -- ALT + PC1 (1-9,-,=) MAKE -
                ;
0233                     K34:
0233  3C 02                 CMP     AL,2                   ; KEY IN TOP ROW?
0235  72 0C                 JB      K35                    ; NO, JMP
0237  3C 0E                 CMP     AL,14
0239  73 08                 JAE     K35                    ; NO, JMP
023B  80 C4 76              ADD     AH,118                 ; BUILD PSEUDO SC
023E  B0 00                 MOV     AL,0
0240  E9 0334 R             JMP     K57                    ; BUFFER_FILL
                ;
                ;----- CHECK FOR F1 - F10                  --- ALT + (F1 - F10) MAKE ---
                ;
0243                     K35:
0243  3C 3B                 CMP     AL,F1_KEY              ; SC POTENTIAL FUNCTION KEY?
0245  73 03                 JAE     K37                    ; YES, JMP
0247                     K36:                              ; NO.  THIS KEY NOT SUPPORTED
0247  E9 0396 R             JMP     KYBD9_EXIT             ; IN ALT_SHIFT.  EOI RETURN.
024A                     K37:
024A  3C 47                 CMP     AL,71                  ; SC IN F1-F10 REGION?
024C  73 F9                 JAE     K36                    ; NO, JMP TO EXIT
024E  BB 0000 E             MOV     BX,OFFSET K13          ; ALT+(F1-F10) EXT ASCII TABLE
0251  E9 037C R             JMP     K63                    ; GO XLATE & BUILD PSEUDO SC
                                                           ; <<<<<                 >>>>>
                                                           ; <<<<<NOT IN ALT_SHIFT  >>>>>
0254                     K38:
0254  F6 06 0017 R 04       TEST    KB_FLAG,CTL_SHIFT      ; IN CTL_SHIFT?
0259  74 65                 JZ      K44                    ; NO, JMP
                ;-------------------------------------------------------
                ; IN  CTL_SHIFT  STATE  W/O ALT_SHIFT                  -
                ;-------------------------------------------------------
025B  3C 46                 CMP     AL,SCROLL_KEY          ; TEST FOR BREAK
025D  75 18                 JNE     K39                    ; NO-BREAK
                ;
                ;------ CTL+BREAK ENTERED  -  CLEAN UP & DO INT 1BH - CTL + BREAK MAKE
                ;
025F  8B 1E 0080 R          MOV     BX,BUFFER_START        ; RESET BUFFER TO EMPTY
0263  89 1E 001A R          MOV     BUFFER_HEAD,BX
0267  89 1E 001C R          MOV     BUFFER_TAIL,BX
026B  C6 06 0071 R 80       MOV     BIOS_BREAK,80H         ; TURN ON BIOS_BREAK BIT
0270  CD 1B                 INT     1BH                    ; BREAK INTERRUPT VECTOR
0272  2B C0                 SUB     AX,AX                  ; PUT OUT DUMMY CHARACTER
```

```
0274  E9 0334 R           JMP    K57                  ; BUFFER_FILL
0277                    K39:                          ; NO-BREAK
0277  3C 45               CMP    AL,NUM_KEY           ; LOOK FOR PAUSE KEY
0279  75 2E               JNE    K41                  ; NO-PAUSE
               ;
               ;------ PAUSE ENTERED  -  EOI & SLEEP UNTIL UNPAUSE - CTL + NUM_LOCK MAKE
               ;
027B  80 0E 0018 R 08     OR     KB_FLAG_1,HOLD_STATE ; TURN ON THE HOLD FLAG
0280  B0 20               MOV    AL,EOI               ; EOI TO ALLOW MORE KEYSTROKES
0282  E6 20               OUT    020H,AL
               ;
               ;--------- DURING PAUSE INTERVAL, TURN CRT BACK ON
               ;
0284  80 3E 0049 R 07     CMP    CRT_MODE,7           ; CURRENTLY USING MONO?
0289  74 07               JE     K40                  ; YES, JMP TO SLEEP
028B  BA 03D8             MOV    DX,03D8H             ; PORT FOR COLOR CARD
028E  A0 0065 R           MOV    AL,CRT_MODE_SET      ; GET CURRENT MODE VAULE
0291  EE                  OUT    DX,AL                ; SET THE CRT MODE ON
               ;
               ;--------- ISSUE SYS SERVICES INT, WAIT ON EXTERNAL EVENT FUNCTION
               ;     (WAIT UNTIL HOLD_STATE BIT OF KB_FLAG_1 IS RESET)
0292                    K40:
0292  B8 4104             MOV    AX,4104H             ; FUNCTION 41H, AL=04=RETURN IF 0
0295  BB 0800             MOV    BX,HOLD_STATE*100H   ; BH=HOLD_STATE, BL=0=NO TIME OUT
0298  1E                  PUSH   DS                   ; MAKE ES:DI POINT TO KB_FLAG_1
0299  07                  POP    ES                   ; .
029A  BF 0018 R           MOV    DI,OFFSET KB_FLAG_1  ; .
029D  CD 15               INT    15H                  ; SLEEP UNTIL OUT OF HOLD
029F  F6 06 0018 R 08     TEST   KB_FLAG_1,HOLD_STATE ; DID INT 15H RESET HOLD_STATE?
02A4  75 EC               JNZ    K40                  ; NO, KEEP LOOPING
02A6  E9 039B R           JMP    KYBD9_RET            ; YES, GO TO NON_EOI EXIT
02A9                    K41:
               ;
               ;----- TEST FOR PRINT SWITCH TOGGLE CMD          [[[ CTL + */PRTSC MAKE ]]]
               ;
02A9  3C 37               CMP    AL,P60_ASTRK_SC      ; * KEY HIT?
02AB  75 06               JNE    K42                  ; NO, JMP
02AD  B8 7200             MOV    AX,114*256           ; YES, BUILD TOGGLE_PRT_SW PSEUDO SC
02B0  E9 0334 R           JMP    K57                  ; BUFFER_FILL
               ;
               ;----- CHECK FOR TYPEWRITER KEYS          [[[ CTL + TYPEWRITER KEY MAKE ]]]
               ;
02B3                    K42:
02B3  BB 0000 E           MOV    BX,OFFSET K8         ; BX <=== CTL + TYPEWRITER KEYS TABLE
02B6  3C 3B               CMP    AL,F1_KEY            ; SC BELOW TYPEWRITER REGION?
02B8  72 76               JB     K56                  ; YES, GO TRANSLATE TO ASCII CODE
               ;
               ;------ KEY IS IN FN OR KEYPAD REGION   [[[ CTL + (F1-F10) ]]]
               ;                             [[[ CTL + PAGE/CURSOR KEY]]]
               ;
02BA                    K43:
02BA  BB 0000 E           MOV    BX,OFFSET K9         ; CTL FN & CTL PAGE/CUR TBLS
02BD  E9 037C R           JMP    K63                  ; GO XLATE & BUILD PSEUDO SC
                                                      ; <<<<<                 >>>>>
                                                      ; <<<<< NOT IN CTL_SHIFT >>>>>
                                                      ; <<<<<    NOR ALT_SHIFT >>>>>
               ;
               ;------ CHECK IF KEY IN KEYPAD REGION (1-9,.,-,+)
               ;
02C0                    K44:
02C0  3C 47               CMP    AL,71                ; KEY IN KEYPAD REGION?
02C2  73 2C               JAE    K48                  ; YES, JMP
02C4  F6 06 0017 R 03     TEST   KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; IN SHIFT STATE?
02C9  74 5A               JZ     K54                             ; NO, JMP
               ;---------------------------------------------------------
               ; IN  SHIFT  STATE  ,BUT KEY IS NOT IN KEYPAD REGION    -
               ;---------------------------------------------------------
               ;
               ;------ CHECK FOR BACK TAB                [[[[ SHIFT + TAB ]]]
               ;
02CB  3C 0F               CMP    AL,TAB_KEY           ; TAB KEY?
02CD  75 05               JNE    K45                  ; NO ,JMP
02CF  B8 0F00             MOV    AX,TAB_KEY*256       ; BUILD BACK_TAB PSEUDO SC
02D2  EB 60               JMP    SHORT K57            ; BUFFER_FILL
```

```
                ;
                ;----- CHECK FOR PRINT SCREEN  -  EOI THEN INT 5H  - SHIFT + */PRTSC
                ;
02D4                  K45:
02D4  3C 37              CMP     AL,P60_ASTRK_SC      ; PRINT SCREEN KEY?
02D6  75 09              JNE     K46                  ; NO, JMP
02D8  B0 20              MOV     AL,EOI               ; EOI TO ALLOW MORE INTERRUPTS
02DA  E6 20              OUT     INTA00,AL
02DC  CD 05              INT     5H                   ; ISSUE PRINT SCREEN INTERRUPT
02DE  E9 039B R          JMP     KYBD9_RET            ; GO TO NON_EOI EXIT
                ;
                ;----- CHECK FOR FUNCTION KEY         [[[ SHIFT + (F1-F10) ]]]
                ;
02E1                  K46:
02E1  3C 3B              CMP     AL,F1_KEY            ; F1 - F10 KEY?
02E3  72 06              JB      K47                  ; NO, JMP
02E5  BB 0000 E          MOV     BX,OFFSET K12        ; BX SHIFT F1-F10 EXT ASCII TBL
02E8  E9 037C R          JMP     K63                  ; GO XLATE & BUILD PSEUDO SC
                ;
                ;----- KEY IS IN TYPEWRITER REGION    [[[ SHIFT + TYPEWRITER KEY ]]]
                ;
02EB                  K47:
02EB  BB 0000 E          MOV     BX,OFFSET K11        ; BX=UPPERCASE ASCII CODE TBL
02EE  EB 40              JMP     SHORT K56            ; GO TRANSLATE TO ASCII CODE
                ;--------------------------------------------------------------------
                ; KEYPAD KEY:  NOT IN ALT OR CTL SHIFT:DETERMINE IF KEYPAD OR PG/CUR
                ;--------------------------------------------------------------------
02F0                  K48:
02F0  F6 06 0017 R 20    TEST    KB_FLAG,NUM_STATE    ; IN NUM_STATE?
02F5  75 20              JNZ     K52                  ; YES, JMP
02F7  F6 06 0017 R 03    TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; NUM_STATE?
02FC  75 20              JNZ     K53                             ; YES, GO PROCESS
                ;
                ;----- BASE CASE KEYPAD (PAGE/CURSOR) KEY -PAGE/CURSOR, DEL, -, + ---
                ;
02FE                  K49:
02FE  3C 4A              CMP     AL,KYPD_MINUS        ; KEYPAD MINUS KEY?
0300  74 0B              JE      K50                  ; YES, JMP
0302  3C 4E              CMP     AL,KYPD_PLUS         ; KEYPAD PLUS KEY?
0304  74 0C              JE      K51                  ; YES, JMP
0306  2C 47              SUB     AL,71                ; ADUST SC FOR TABLE OFFSET
0308  BB 0000 E          MOV     BX,OFFSET K15        ; BX=KEYPAD AREA BASE CASE TBL
030B  EB 71              JMP     SHORT K64            ; GO TRANSLATE & BUILD PSEUDO SC
030D                  K50:
030D  B8 4A2D            MOV     AX,74*256+'-'        ; BLD EXT ASCII CODE FOR MINUS
0310  EB 22              JMP     SHORT K57            ; BUFFER_FILL
0312                  K51:
0312  B8 4E2B            MOV     AX,78*256+'+'        ; BLD EXT ASCII CODE FOR PLUS
0315  EB 1D              JMP     SHORT K57            ; BUFFER_FILL
                ;
                ;----- IN NUM_STATE  -  CHECK IF TEMPORARILY SHIFTED OUT
                ;
0317                  K52:
0317  F6 06 0017 R 03    TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; NOT NUM STATE?
031C  75 E0              JNZ     K49                             ; YES, JMP TO BASE
                                                                 ; KEYPAD XLAT
031E                  K53:                            ; [[[ KEYPAD 0-9, ., -, + ]]]
031E  2C 46              SUB     AL,70                ; ADJUST SC FOR TABLE OFFSET
0320  BB 0000 E          MOV     BX,OFFSET K14        ; BX=KEYPAD KEYS' ASCII CODES
0323  EB 0B              JMP     SHORT K56            ; GO XLATE & BUILD ASCII CODE
                ;---------------------------------------------------
                ; NOT IN ANY SHIFT STATE                            -
                ;  KEYPAD REGION HAS BEEN PROCESSED (SC 69 & UP)  -
                ;---------------------------------------------------
                ;
                ;----- CHECK FOR F1-F10               [[[ BASE F1 - F10 ]]]
                ;
0325                  K54:
0325  3C 3B              CMP     AL,F1_KEY            ; FUNCTION KEY?
0327  72 04              JB      K55                  ; NO, JMP
0329  B0 00              MOV     AL,0                 ; SCAN CODE IN AH ALREADY
032B  EB 37              JMP     SHORT K61            ; BUFFER_FILL
                ;
                ;----- KEY MUST BE FROM TYPEWRITER REGION -- BASE TYPEWRITER KEY ---
```

```
                      ;
032D                        K55:
032D  BB 0000 E                MOV     BX,OFFSET K10       ; BX=BASE CASE ASCII TBL
                      ;-----------------------------------------------------------------
                      ;
                      ; T R A N S L A T E   SC TO EXTENDTED ASCII CODE   &   B U F F E R
                      ;
                      ;-----------------------------------------------------------------
0330                        K56:
0330  FE C8                    DEC     AL                      ; ADJUST SC FOR TABLE OFFSET
0332  2E: D7                   XLAT    cs:K11          ; TRANSLATE FROM TABLE IN BX
                      ;
                      ;----- PUT CHARACTER INTO BUFFER
                      ;
0334                        K57:
0334  3C FF                    CMP     AL,-1                   ; IGNORE CODE XLATED FROM TABLE?
0336  74 1F                    JE      K59                     ; YES, GO TO EOI EXIT
0338  80 FC FF                 CMP     AH,-1                   ; IGNORE PSEUDO SCAN?
033B  74 1A                    JE      K59                     ; YES, GO TO EOI EXIT
                      ;
                      ;----- CHECK FOR CAPS_STATE
                      ;
033D                        K58:
033D  F6 06 0017 R 40          TEST    KB_FLAG,CAPS_STATE  ; ARE WE IN CAPS LOCK STATE
0342  74 20                    JZ      K61                 ; SKIP IF NOT
                      ;                                   <<<<<<             >>>>>>
                      ;----- DETERMINE WHICH WAY TO CONVERT  <<<<<< IN CAPS_STATE >>>>>>
                      ;
0344  F6 06 0017 R 03          TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT  ; IN SHIFT STATE?
0349  74 0F                    JZ      K60                             ; NO, JMP
                      ;
                      ;----- CONVERT UPPER CASE TO LOWER  -  ALPHABETIC CHARS ONLY
                      ;
034B  3C 41                    CMP     AL,'A'                  ; ALPHABETIC CHAR?
034D  72 15                    JB      K61                     ; NO, JMP
034F  3C 5A                    CMP     AL,'Z'
0351  77 11                    JA      K61                     ; NO, JMP
0353  04 20                    ADD     AL,'a'-'A'              ; YES, CONVERT TO LOWER CASE.
0355  EB 0D                    JMP     SHORT K61               ; GO BUFFER
0357                        K59:
0357  EB 3D 90                 JMP     KYBD9_EXIT              ; INTERRUPT_RETURN
                      ;
                      ;----- CONVERT LOWER CASE TO UPPER  -  ALPHABETIC CHARS ONLY
                      ;
035A                        K60:
035A  3C 61                    CMP     AL,'a'                  ; ALPHABETIC CHAR?
035C  72 06                    JB      K61                     ; NO, JMP
035E  3C 7A                    CMP     AL,'z'
0360  77 02                    JA      K61                     ; NO, JMP
0362  2C 20                    SUB     AL,'a'-'A'              ; YES, CONVERT TO UPPER CASE
                      ;---------------------------------------------------
                      ; BUFFER EXTENDED ASCII CODE IF NOT FULL          -
                      ;---------------------------------------------------
0364                        K61:
0364  8B 1E 001C R             MOV     BX,BUFFER_TAIL          ; GET BUFFER TAIL POINTER
0368  8B F3                    MOV     SI,BX                   ; SI <=== TAIL PTR
036A  E8 03A4 R                CALL    PTR_INC                 ; ADVANCE THE TAIL
036D  3B 1E 001A R             CMP     BX,BUFFER_HEAD          ; HAS BUFFER WRAPPED AROUND?
0371  74 13                    JE      K62                     ; YES, GO GIVE BUFFER_FULL_BEEP
0373  89 04                    MOV     [SI],AX                 ; NO, BUFFER THE CODE
0375  89 1E 001C R             MOV     BUFFER_TAIL,BX          ; UPDATE TAIL PTR
0379  EB 1B 90                 JMP     KYBD9_EXIT              ; GO TO EOI EXIT
                      ;----------------------------------------------------------
                      ; TRANSLATE F1-F10 SCAN CODES TO AN EXTENDED ASCII CODE  -
                      ;    (FOR  SHIFT,  ALT, &  CTL  STATES)                  -
                      ;----------------------------------------------------------
037C                        K63:
037C  2C 3B                    SUB     AL,F1_KEY               ; ADJUST SC FOR TABLE OFFSET
037E                        K64:
037E  2E: D7                   XLAT    cs:K9           ; XLAT FROM TABLE IN BX
                                                               ; (K9,K12,K13)
0380  8A E0                    MOV     AH,AL                   ; BUILD EXTENDED ASCII CODE
0382  B0 00                    MOV     AL,0
0384  EB AE                    JMP     K57                     ; GO BUFFER
```

```
                ;------------------------------------------
                ; BUFFER IS FULL, SOUND THE BEEPER
                ;------------------------------------------
0386                     K62:
0386  B0 20                 MOV     AL,EOI              ; END OF INTERRUPT COMMAND
0388  E6 20                 OUT     INTA00,AL           ; SEND CMD TO INT CONTROL PORT
038A  BB 0053               MOV     BX,083              ; NO. CYCLES FOR 83 MSEC TONE
038D  B9 0081               MOV     CX,081H             ; 1/2 CYCLE FOR 1KHZ TONE
0390  E8 0000 E             CALL    KB_NOISE
0393  EB 06 90              JMP     KYBD9_RET           ; RETURN WITHOUT EOI
                ;--------------------------
                ; ISSUE EOI & RETURN       -
                ;--------------------------
0396                     KYBD9_EXIT:
0396  FA                    CLI                         ; TURN OFF INTERRUPTS
0397  B0 20                 MOV     AL,EOI              ; ISSUE EOI
0399  E6 20                 OUT     INTA00,AL
                ;
                ;----- RETURN WITH EOI ALREADY ISSUED
                ;
039B                     KYBD9_RET:
039B  07                    POP     ES                  ; RESTORE REGS
039C  1F                    POP     DS
039D  5F                    POP     DI
039E  5E                    POP     SI
039F  5A                    POP     DX
03A0  59                    POP     CX
03A1  5B                    POP     BX
03A2  58                    POP     AX
03A3  CF                    IRET                        ; RETURN:  IRPTS SET BACK AS WERE
03A4                     KYBD_INT9 ENDP
                ;*****************************************************************
                ;
                ; ROUTINE-NAME :  PTR_INC
                ;
                ; FUNCTION:    INCREMENT THE KEYBOARD BUFFER POINTER AND WRAP THE BUFFER
                ;              IF NECESSARY.
                ;
                ; ENTRY CONDITIONS:     DS= DATA SEGMENT
                ;                       BX= POINTER TO INCREMENT
                ;
                ; EXIT CONDITIONS:
                ;           BX INCREMENTED BY 2 AND IF BUFFER_END EXCEECE
                ;           BX IS SET TO BUFFER_START.
                ;
                ; REGISTERS MODIFIED: BX
                ;*****************************************************************
03A4                     PTR_INC      PROC    NEAR
03A4  43                    INC     BX                  ; MOVE TO NEXT WORD IN LIST
03A5  43                    INC     BX
03A6  3B 1E 0082 R          CMP     BX,BUFFER_END       ; AT END OF BUFFER?
03AA  75 04                 JNE     PTR_01              ; NO, CONTINUE
03AC  8B 1E 0080 R          MOV     BX,BUFFER_START     ; YES, RESET TO BUFFER BEGINNING
03B0                     PTR_01:
03B0  C3                    RET
03B1                     PTR_INC ENDP

03B1                     ROMCODE ENDS
                            END
```

Video I/O and Print Screen (B12VIDEO)

```
0000                    ROMCODE SEGMENT BYTE PUBLIC
                           ASSUME  CS:ROMCODE
                           IDENT B12VIDEO,12,00

              ;*******************************************************************
              ;
              ; MODULE-NAME :      B12VIDEO
              ;
              ; DATE LAST MODIFIED:  09/12/85
              ;
              ; DESCRIPTIVE-NAME : VIDEO_IO AND PRINT SCREEN BIOS
              ;
              ; COPYRIGHT : 7396-917 (C) COPYRIGHT IBM CORP. 1985
              ;             REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
              ;
              ; CHANGE LEVEL: 0.0
              ;
              ; FUNCTION:   VIDEO_IO  -  HANDLES ALL BIOS VIDEO REQUESTS
              ;             PRINT_SCREEN  -  HANDLES THE PRINT SCREEN FUNCTION
              ;
              ; MODULE SIZE: 2829 BYTES
              ;
              ; ENTRY CONDITIONS:
              ;       REFER TO ROUTINE PROLOGUES
              ;
              ; EXIT CONDITIONS:
              ;       REFER TO ROUTINE PROLOGUES
              ;
              ; ROUTINES IN MODULE:
              ;       VIDEO_IO       -   INT 10H DISPLAY INTERFACE ROUTINES
              ;       PRINT_SCREEN   -   INT 5H PRINT SCREEN INTERRUPT HANDLER
              ;
              ; INTERNAL DATA AREAS / TABLES:   M1_1, REGS_PUSHD, V1
              ;
              ; EXTERNALLY REFERENCED ROUTINES: REFER TO EXTRN LIST
              ;
              ; EXTERNALLY REFERENCED DATA AREAS: DATA SEG, XXDATA SEG, VIDEO RAM
              ;
              ; CHANGE ACTIVITY:  NONE
              ;
              ;*******************************************************************

              ;****************************************************************
              ;*    E X T E R N A L   R E F E R E N C E S
              ;****************************************************************
              ;
              ; ROUTINES
              ;
                         EXTRN   DDS:NEAR
                         EXTRN   BEEP:NEAR
                         EXTRN   KB_NOISE:NEAR
                         EXTRN   GET_RTC_REG:NEAR
                         EXTRN   PUT_RTC_REG:NEAR
                         EXTRN   LCDINIT:NEAR
              ;
              ; TABLES
              ;
                         EXTRN   CHAR_GEN_LO:BYTE
                         EXTRN   CHAR_GEN_HI:BYTE
                         EXTRN   M4:ABS
                         EXTRN   M5:WORD
                         EXTRN   M6:BYTE
                         EXTRN   M7:BYTE
```

```
            EXTRN   MONO_TBL:WORD
            EXTRN   CGA_TBL:WORD
            EXTRN   LCD_MONO_TBL:WORD
            EXTRN   LCD_CGA_TBL:WORD
;*****************************************************************
;*    P U B L I C S  D E C L A R A T I O N
;*****************************************************************

            PUBLIC VIDEO_IO_1
            PUBLIC PRT_SCRN
            PUBLIC SET_MODE
            PUBLIC SET_CTYPE
            PUBLIC SET_CPOS
            PUBLIC READ_CURSOR
            PUBLIC READ_LPEN
            PUBLIC ACT_DISP_PAGE
            PUBLIC SCROLL_UP
            PUBLIC SCROLL_DOWN
            PUBLIC READ_AC_CURRENT
            PUBLIC WRITE_AC_CURRENT
            PUBLIC WRITE_C_CURRENT
            PUBLIC SET_COLOR
            PUBLIC WRITE_DOT
            PUBLIC READ_DOT
            PUBLIC WRITE_TTY

            SUBTTL  VIDEO BIOS  - INT 10H

;********************************************************************
;
;
; ROUTINE-NAME :  VIDEO_IO
;
; FUNCTION:   THIS ROUTINE HANDLES THE VIDEO BIOS REQUESTS - INT 10H
;
; ENTRY CONDITIONS:
;       PURPOSE OF ENTRY: SEE INT 10H DESCRIPTION
;       INPUT CONDITIONS: SEE INT 10H DESCRIPTION
;       RESTRICTIONS: NONE
;
; EXIT CONDITIONS:
;       NORMAL EXIT CONDITIONS: SEE INT 10 DESCRIPTION
;       ERROR EXIT CONDITIONS: NO ERROR REPORTING INTERFACE IS DEFINED
;       REGISTERS MODIFIED: ALL REGISTERS EXCEPT AX ARE SAVED UNLESS
;                           THEY ARE USED TO RETURN INFO AS DEFINE
;                           IN THE INT 10H DESCRIPTION
;       RETURN TYPE:  IRET (ALL FLAGS RESTORED)
;       INTERRUPTS:  ENABLED DURING PROCESSING
;
; INTERNALLY REFERENCED ROUTINES:
;                 ACT_DISP_PAGE,          SET_COLOR,
;                 LCD_REQUEST,            SET_CPOS,
;                 PHYS_DSP_DESCR_REQ,     SET_CTYPE,
;                 PRT_SCREEN,             SET_MODE,
;                 READ_AC_CURRENT,        VIDEO_STATE,
;                 READ_CURSOR,            WRITE_AC_CURRENT,
;                 READ_DOT,               WRITE_C_CURRENT,
;                 READ_LPEN,              WRITE_DOT,
;                 SCROLL_UP,              WRITE_STRING,
;                 SCROLL_DOWN,            WRITE_TTY
;
; EXTERNALLY REFERENCED ROUTINES:
;                 DDS,
;                 BEEP,
;                 KB_NOISE,
;                 GET_RTC_REG,
;                 PUT_RTC_REG,
;                 LCDINIT
;
;
;*********************************************************************
```

```
;--- INT 10H -----------------------------------------------------
; VIDEO_IO
;       THESE ROUTINES PROVIDE THE CRT INTERFACE
;       THE FOLLOWING FUNCTIONS ARE PROVIDED:
;
;       (AH)=00H  SET MODE (AL) CONTAINS MODE VALUE
;                     TYPE     RES/DIM   DISPLAY    MAX PAGES   NOTES
;          ---------------------------------------------------------
;           (AL)=0  ALPHA     40X25     COLOR - BW**  8
;           (AL)=1  ALPHA     40X25     COLOR         8
;           (AL)=2  ALPHA     80X25     COLOR - BW**  4
;           (AL)=3  ALPHA     80X25     COLOR         4    *  DEFAULT
;           (AL)=4  GRAPHICS  320X200   COLOR         1
;           (AL)=5  GRAPHICS  320X200   COLOR - BW    1
;           (AL)=6  GRAPHICS  640X200   COLOR - BW    1
;           (AL)=7  ALPHA     80X25     MONOCHROME   1/4   *  INTERNAL
;
;     NOTE:  IF HIGH BIT OF AL IS SET, THE REGEN BUFFER IS NOT CLEARED.
;
;     *    FOR MONOCHROME, CRT MODE WILL INTERNALLY DEFAULT TO 7.
;          FOR MONO ON LCD THE MAXIMUM PAGES ALLOWED IS 4, OTHERWISE
;          THE MAXIMUM PAGES ALLOWED IS 1.
;
;     *    FOR COLOR, AL=(7-255) WILL DEFAULT TO MODE 3
;
;     **   BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR BURST IS
;          NOT ENABLED
;
;     (AH)=01H  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
;          (CL) =  BITS 4-0 = END LINE FOR CURSOR
;
;     (AH)=02H  SET CURSOR POSITION
;          (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
;          (DH,DL) = ROW,COLUMN  (0,0) IS UPPER LEFT
;          ROW = (0 - 24), COL = (0 - (CRT COLUMNS-1))
;          CRT COLUMNS IS EITHER 80 OR 40
;
;      (AH)=03H  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)=04H  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)=05H  SELECT ACTIVE DISPLAY PAGE(VALID ONLY FOR ALPHA MODES)
;                (AL)=NEW PAGE VALUE
;                VALID PAGE VALUES: ALSO, SEE AH=00H FOR PAGE INFO
;                     MODES 0 & 1  -  (0 - 7)
;                     MODES 2 & 3  -  (0 - 3)
;                     MODE  7      -    0
;                     MODE  7 & LCD CONFIGURED AS MONO - (0 - 3)
;
;      (AH)=06H  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)=07H  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)=08H  READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
;                 (BH) = DISPLAY PAGE (FOR ALPHA MODES ONLY)
;                 ON EXIT:
;                      (AL) = CHAR READ
;                      (AH) = ATTRIBUTE OF CHAR READ (ALPHA MODES ONLY
;
;       (AH)=09H  WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
;                 (BH) = DISPLAY PAGE (FOR ALPHA MODES ONLY)
;                 (CX) = COUNT OF CHARACTERS TO WRITE
;                 (AL) = CHAR TO WRITE
;                 (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR
;                        (GRAPHICS)
;                        SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
;
;       (AH)=0AH  WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION
;                 **** NOT VALID FOR MEDIUM RESOLUTION GRAPHICS ****
;                 (BH) = DISPLAY PAGE (FOR ALPHA MODES ONLY)
;                 (CX) = COUNT OF CHARACTERS TO WRITE
;                 (AL) = CHAR TO WRITE
;       FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE
;                CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE
;                MAINTAINED IN THE SYSTEM 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)=0BH  SET COLOR PALETTE
;                 (BH) = PALETTE COLOR ID BEING SET (0-127)
;                 (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
;                 NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT
;                       HAS MEANING ONLY FOR 320X200 GRAPHICS.
;                       COLOR ID = 0 SELECTS THE BACKGND COLOR (0-15)
;                       COLOR ID = 1 SELECTS THE PALETTE TO BE USED:
;                           0 = GREEN(1)/RED(2)/YELLOW(3)
;                           1 = CYAN(1)/MAGENTA(2)/WHITE(3)
;                       IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET
;                       FOR PALETTE COLOR 0 INDICATES THE
;                       BORDER COLOR TO BE USED (VALUES 0-31,
;                       WHERE 16-31 SELECT THE HIGH INTENSITY
;                       BACKGROUND SET.
;
;       (AH)=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)=0DH  READ DOT
;                 (DX) = ROW NUMBER
;                 (CX) = COLUMN NUMBER
;                        ON EXIT:
;                            (AL) = THE DOT READ
;
```

```
; ASCII TELETYPE ROUTINE FOR OUTPUT
;
;     (AH)=0EH  WRITE TELETYPE TO ACTIVE PAGE
;               (AL) = CHAR TO WRITE
;               (BL) = FOREGROUND COLOR IN GRAPHICS MODE
;               NOTE --SCREEN WIDTH IS CTRLLED BY PREVIOUS MODE SET
;
;     (AH)=0FH  CURRENT VIDEO STATE
;               RETURNS THE CURRENT VIDEO STATE
;               ON EXIT:
;                   (AL) = MODE CURRENTLY SET ( SEE AH=0 FOR EXPL
;                   (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
;                   (BH) = CURRENT ACTIVE DISPLAY PAGE
;
;     (AH)=10H  RESERVED - NO OPERATION
;     (AH)=11H  RESERVED - NO OPERATION
;     (AH)=12H  RESERVED - NO OPERATION
;
;     (AH)=13H  WRITE STRING
;
;               (ES:BP) = POINTER TO STRING TO BE WRITTEN
;               (CX)    = LENGTH OF CHARACER STRING TO WRITTEN
;                         IF CX = 0 NO OPERATION
;               (DX)    = CURSOR POSITION FOR STRING TO BE WRITTEN
;               (BH)    = PAGE NUMBER
;
;               (AL) = 0
;                      WRITE CHARACTER STRING
;                           BL     -  ATTRIBUTE
;                           STRING IS  {CHAR,CHAR, ... ,CHAR}
;                           CURSOR NOT MOVED
;               (AL) = 1
;                      WRITE CHARACTER STRING AND MOVE CURSOR
;                            BL     -  ATTRIBUTE
;                            STRING IS  {CHAR,CHAR, ... ,CHAR}
;                            CURSOR IS MOVED
;              (AL) = 2
;                     WRITE CHARACTER AND ATTRIBUTE STRING
;                     STRING IS  {CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR}
;                     CURSOR IS NOT MOVED
;              (AL) = 3
;                     WRITE CHARACTER AND ATTR STRING AND MOVE CURSO
;                     STRING IS  {CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR}
;                     CURSOR IS MOVED
;              (AL) = 4 - 255
;                     NO OPERATION
;
;   NOTE:  CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE
;          TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS.
;
```

```
;    (AH)=14H  LCD REQUEST
;              LOAD LCD CHARACTER FONT / SET LCD HIGH INTENSITY
;              SUBSTITUTE
;
;              (AL) = 0  - LOAD USER SPECIFIED FONT
;                   (ES) - SEGMENT ADDRESS OF USER CHARACTER TABLE
;                   (DI) - OFFSET TO FIRST CHARACTER TO BE STORED
;                   (CX) - NUMBER OF CHARACTERS TO STORE
;                          (1 - 256) VALUE CHECKED
;                   (DL) - CHAR OFFSET INTO RAM FONT STORAGE
;                   (BH) - NUMBER OF BYTES PER CHARACTER
;                          (1 - 255) VALUE CHECKED
;                   (BL) - 0 = LOAD MAIN FONT (BLOCK 0)
;                          1 = LOAD ALTERNATE FONT (BLOCK 1)
;                          2 - 255 = NO OPERATION
;
;              (AL) = 1  - LOAD SYSTEM ROM DEFAULT FONT
;                   (BL) - 0 = LOAD MAIN FONT (BLOCK 0)
;                          1 = LOAD ALTERNATE FONT (BLOCK 1)
;                          2 - 255 = NO OPERATION
;
;              (AL) = 2  -SET MAPPING OF LCD HIGH INTENSITY ATTRIBUTE
;                   (BL) - INTENSIFY MAPPING INPUT REGISTER
;                          0 = IGNORE HIGH INTENSITY ATTRIBUTE
;                          1 = MAP HIGH INTENSITY TO UNDERSCORE
;                          2 = MAP HIGH INTENSITY TO REVERSE IMAGE
;                          3 = MAP INTENSITY TO SELECT ALTERNATE FONT
;                          4 - 255 = NO OPERATION
;             (AL) = 3 - 255
;                    NO OPERATION
;
;      (AH)=15H  PHYSICAL DISPLAY DESCRIPTION PARAMETER REQUEST
;                RETURNS THE ADDRESS OF A 7 WORD TABLE WHICH CONTAINS
;                THE DESCRIPTION PARAMETERS OF THE CURRENT DISPLAY.
;                IT ALSO RETURNS THE MONITOR NUMBER OF THE ALT DISPLAY
;            ON EXIT:
;                (ES:DI) = POINTS TO A 7 WORD PARAMETER TABLE
;                AX  = MONITOR NUMBER OF ALT DISPLAY. IF THERE IS
;                      NO ALTERNATE DISPLAY OR THE ALT DISPLAY IS
;                      INOPERATIVE THEN AX = 0.
;
;                THE 7 WORD TABLE CONTAINS THE FOLLOWING INFOMATION:
;                WORD                 INFORMATION
;                 1            MONITOR MODEL NUMBER  (5140,5153,5151)
;                 2            NUMBER OF VERTICAL PELS / METER
;                 3            NUMBER OF HORIZONTAL PELS / METER
;                 4            TOTAL NUMBER OF VERTICAL PELS
;                 5            TOTAL NUMBER OF HORIZONTAL PELS
;                 6            HORIZONTAL PEL SPACING IN MICROMETERS
;                              (CENTER TO CENTER)
;                 7            VERTICAL PEL SPACING IN MICROMETERS
;                             (CENTER TO CENTER)
;
;               DISPLAY TYPES AND TABLES
;               -----------------------
;
;        WORD   MONOCHROME     CGA      LCD AS CGA   LCD AS MONO
;       ---------------------------------------------------------
;         1     5151 HEX    5153 HEX    5140 HEX     5140 HEX
;         2        0        0498 HEX    08E1 HEX        0
;         3        0        0A15 HEX    0987 HEX        0
;         4        0        00C8 HEX    00C8 HEX        0
;         5        0        0280 HEX    0280 HEX        0
;         6        0        0352 HEX    01BB HEX        0
;         7        0        0184 HEX    019A HEX        0
;
;---------------------------------------------------------------

;****************************************************************
;*       S T A R T  O F  C O D E                              *
;****************************************************************
           ASSUME  CS:ROMCODE,DS:DATA,ES:VIDEO_RAM
```

```
0000                         M1_1    LABEL   WORD          ; TBL OF RTNS WITHIN VIDEO I/O
0000  0069 R                    DW      OFFSET  SET_MODE
0002  01BA R                    DW      OFFSET  SET_CTYPE
0004  0241 R                    DW      OFFSET  SET_CPOS
0006  0290 R                    DW      OFFSET  READ_CURSOR
0008  07D3 R                    DW      OFFSET  READ_LPEN
000A  026C R                    DW      OFFSET  ACT_DISP_PAGE
000C  02F7 R                    DW      OFFSET  SCROLL_UP
000E  037A R                    DW      OFFSET  SCROLL_DOWN
0010  03B6 R                    DW      OFFSET  READ_AC_CURRENT
0012  03FC R                    DW      OFFSET  WRITE_AC_CURRENT
0014  0430 R                    DW      OFFSET  WRITE_C_CURRENT
0016  02AB R                    DW      OFFSET  SET_COLOR
0018  0474 R                    DW      OFFSET  WRITE_DOT
001A  0463 R                    DW      OFFSET  READ_DOT
001C  074F R                    DW      OFFSET  WRITE_TTY
001E  02D1 R                    DW      OFFSET  VIDEO_STATE
0020  016F R                    DW      OFFSET  VIDEO_RETURN
0022  016F R                    DW      OFFSET  VIDEO_RETURN
0024  016F R                    DW      OFFSET  VIDEO_RETURN
0026  0879 R                    DW      OFFSET  WRITE_STRING
0028  0927 R                    DW      OFFSET  LCD_REQUEST
002A  09E6 R                    DW      OFFSET  PHYS_DSP_DESCR_REQ
= 002C                       M1L_1   EQU     $-M1_1

                             REGS_PUSHD      STRUC
0000  0000                   TEMPBP  DW      0
0002  0000                   BP_POS  DW      0
0004  0000                   DI_POS  DW      0
0006  0000                   SI_POS  DW      0
0008  0000                   BX_POS  DW      0
000A  0000                   CX_POS  DW      0
000C  0000                   DX_POS  DW      0
000E  0000                   DS_POS  DW      0
0010  0000                   ES_POS  DW      0
0012  0000                   IP_POS  DW      0
0014  0000                   CS_POS  DW      0
0016  0000                   FL_POS  DW      0
0018                         REGS_PUSHD      ENDS

002C                         VIDEO_IO_1      PROC    NEAR
002C  06                        PUSH    ES
002D  1E                        PUSH    DS                    ; SAVE REGISTERS
002E  52                        PUSH    DX
002F  51                        PUSH    CX
0030  53                        PUSH    BX
0031  56                        PUSH    SI
0032  57                        PUSH    DI
0033  55                        PUSH    BP

0034  FB                        STI
0035  FC                        CLD
0036  50                        PUSH    AX                    ; SAVE AX VALUE
0037  8A C4                     MOV     AL,AH                 ; GET INTO LOW BYTE
0039  32 E4                     XOR     AH,AH                 ; ZERO TO HIGH BYTE
003B  D1 E0                     SAL     AX,1                  ; *2 FOR TABLE LOOKUP
003D  8B F0                     MOV     SI,AX                 ; PUT INTO SI FOR BRANCH
003F  3D 002C                   CMP     AX,M1L_1              ; TEST FOR WITHIN RANGE
0042  72 04                     JB      M2                    ; BRANCH AROUND BRANCH
0044  58                        POP     AX                    ; THROW AWAY THE PARAMETER
0045  E9 016F R                 JMP     VIDEO_RETURN          ; DO NOTHING IF NOT IN RANGE
0048                         M2:
0048  E8 0000 E                 CALL    DDS
004B  B8 B800                   MOV     AX,0B800H             ; SEGMENT FOR CGA CARD
004E  8B 3E 0010 R              MOV     DI,EQUIP_FLAG         ; GET EQUIPMENT SETTING
0052  81 E7 0030                AND     DI,30H                ; ISOLATE CRT SWITCHES
0056  83 FF 30                  CMP     DI,30H                ; IS SETTING FOR MONO CARD?
0059  75 02                     JNE     M3
005B  B4 B0                     MOV     AH,0B0H               ; SEGMENT FOR MONO CARD
005D                         M3:
005D  8E C0                     MOV     ES,AX                 ; SET PTR TT VIDEO RAM AREAS
005F  58                        POP     AX                    ; RECOVER VALUE
0060  8A 26 0049 R              MOV     AH,CRT_MODE           ; GET CURRENT MODE INTO AH
```

```
0064  2E: FF A4 0000 R      JMP     WORD PTR CS:[SI+OFFSET M1_1]
0069                   VIDEO_IO_1      ENDP

              ;-----------------------------------------------------
              ; SET_MODE
              ;      THIS ROUTINE INITIALIZES THE ATTACHMENT TO
              ;      THE SELECTED MODE.  THE SCREEN IS BLANKED.
              ;      IF CGA, THE ONLY VALID MODES ALLOWED ARE 0-6.
              ;      ALL OTHER MODES ARE DEFAULTED TO MODE 3.
              ;      IF MONO, MODE IS DEFAULTED TO 7.
              ; INPUT
              ;      (AL) = MODE SELECTED
              ;
              ;      (DS) = DATA SEGMENT
              ;      (ES) = REGEN BUFFER SEGMENT
              ;      (DI) = VIDEO SWITCHES FROM EQUIPMENT FLAG
              ; OUTPUT
              ;      CURRENT DISPLAY INITIALIZED TO SELECTED MODE
              ;
              ; INTERRUPTS
              ;      DISABLED DURING THE INITIALIZATION OF THE
              ;      6845 REGISTERS
              ;-----------------------------------------------------

0069                   SET_MODE        PROC    NEAR
0069  50                  PUSH    AX                   ; SAVE CLEAR REGEN BIT
006A  24 7F               AND     AL,7FH               ; TURN OFF CLEAR REGEN BIT
006C  BA 03D4             MOV     DX,03D4H             ; ADDRESS OF CGA CARD
006F  B3 00               MOV     BL,0                 ; MODE SET FOR CGA CARD
0071  83 FF 30            CMP     DI,30H               ; IS MONO CARD INSTALLED
0074  74 08               JE      M8                   ; YES, JUMP TO MONO SET
0076  3C 07               CMP     AL,7                 ; IS MODE 7 OR GREATER FOR CGA
0078  72 0A               JB      M8A                  ; CARD THEN DEFAULT TO
007A  B0 03               MOV     AL,3                 ; MODE 3 (80 x 25 COLOR)
007C  EB 06               JMP     SHORT M8A
              ;
              ; MONOCHROME OPERATION SELECTED
              ;
007E                   M8:
007E  B0 07               MOV     AL,7                 ; INDICATE MONO CARD MODE
0080  B2 B4               MOV     DL,0B4H              ; ADDRESS OF MONO CARD (3B4)
0082  FE C3               INC     BL                   ; MODE SET FOR MONO CARD
0084                   M8A:
0084  E8 0178 R           CALL    LCD_MOVE             ; CK FOR LCD ADA CHANGE REQ
0087  8A E0               MOV     AH,AL                ; SAVE MODE IN AH
0089  A2 0049 R           MOV     CRT_MODE,AL          ; SAVE IN GLOBAL VARIABLE
008C  89 16 0063 R        MOV     ADDR_6845,DX         ; SAVE ADDRESS OF BASE
0090  1E                  PUSH    DS                   ; SAVE POINTER TO DATA SEGMENT
0091  50                  PUSH    AX                   ; SAVE MODE
0092  52                  PUSH    DX                   ; SAVE OUTPUT PORT VALUE
0093  83 C2 04            ADD     DX,4                 ; POINT TO CONTROL REGISTER
0096  8A C3               MOV     AL,BL                ; GET MODE SET FOR CARD
0098  EE                  OUT     DX,AL                ; RESET VIDEO
                                                       ; AND CHANGE CONFIG IF REQD
0099  5A                  POP     DX                   ; BACK TO BASE REGISTER
009A  B8 ---- R           MOV     AX,ABSO              ; SET UP FOR ABSO SEGMENT
009D  8E D8               MOV     DS,AX                ; ESTABLISH VECTOR TBL ADDR

                          ASSUME  DS:ABSO

009F  C5 1E 0074 R        LDS     BX,PARM_PTR          ; GET POINTER TO VIDEO PARMS
00A3  58                  POP     AX                   ; RECOVER PARMS
00A4  B9 0000 E           MOV     CX,OFFSET M4         ; LENGTH OF EACH ROW OF TABLE
00A7  80 FC 02            CMP     AH,2                 ; DETERMINE WHICH ONE TO USE
00AA  72 10               JC      M9                   ; MODE IS 0 OR 1
00AC  03 D9               ADD     BX,CX                ; MOVE TO NEXT ROW OF INIT TBL
00AE  80 FC 04            CMP     AH,4
00B1  72 09               JC      M9                   ; MODE IS 2 OR 3
00B3  03 D9               ADD     BX,CX                ; MOVE TO GRAPHICS ROW OF TBL
00B5  80 FC 07            CMP     AH,7
00B8  72 02               JC      M9                   ; MODE IS 4,5, OR 6
00BA  03 D9               ADD     BX,CX                ; MOVE TO MONO CARD ROW OF TBL
```

```
               ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE

00BC                       M9:                           ; OUT_INIT
00BC  50                      PUSH    AX                 ; SAVE MODE IN AH
00BD  32 E4                   XOR     AH,AH              ; AH WILL SERVE AS REGISTER

               ;----- LOOP THROUGH TABLE, OUTPUTTTING REG ADDR, THEN VALUE FROM TABLE

00BF  9C                      PUSHF                      ; SAVE CURRENT FLAGS
00C0  FA                      CLI                        ; INHIBIT INTERRUPTS
00C1                       M10:                          ; INIT LOOP
00C1  8A C4                   MOV     AL,AH              ; GET 6845 REGISTER NUMBER
00C3  EE                      OUT     DX,AL
00C4  42                      INC     DX                 ; POINT TO DATA PORT
00C5  FE C4                   INC     AH                 ; NEXT REGISTER VALUE
00C7  8A 07                   MOV     AL,[BX]            ; GET TABLE VALUE
00C9  EE                      OUT     DX,AL              ; OUT TO CHIP
00CA  43                      INC     BX                 ; NEXT IN TABLE
00CB  4A                      DEC     DX                 ; BACK TO POINTER REGISTER
00CC  E2 F3                   LOOP    M10                ; DO THE WHOLE TABLE
00CE  9D                      POPF                       ; RESTORE FLAGS
00CF  58                      POP     AX                 ; GET MODE BACK
00D0  1F                      POP     DS                 ; REC+VER SEGMENT VALUE
                              ASSUME  DS:DATA

               ;----- FILL REGEN AREA WITH BLANKS

00D1  33 FF                   XOR     DI,DI              ; SET UP POINTER FOR REGEN
00D3  89 3E 004E R            MOV     CRT_START,DI       ; START ADDR SAVED IN GLOBAL
00D7  C6 06 0062 R 00         MOV     ACTIVE_PAGE,0      ; SET PAGE VALUE
00DC  59                      POP     CX                 ; RESTORE CLEAR REGEN BIT
00DD  D0 E1                   SHL     CL,1               ; LOOK, DON'T CLR REGEN REQ
00DF  72 21                   JC      M13A               ; IF ON - DON'T CLEAR REGEN
00E1  B9 2000                 MOV     CX,8192            ; NUMBER OF WORDS IN CGA CARD
00E4  80 FC 04                CMP     AH,4               ; TEST FOR GRAPHICS
00E7  72 14                   JC      M12                ; NO_GRAPHICS_INIT
00E9  80 FC 07                CMP     AH,7               ; TEST FOR MONO CARD
00EC  74 04                   JE      M11                ; MONO_CARD_INIT
00EE  33 C0                   XOR     AX,AX              ; FILL FOR GRAPHICS MODE
00F0  EB 0E                   JMP     SHORT M13          ; CLEAR_BUFFER
00F2                       M11:                          ; MONO_CARD_INIT
00F2  B4 20                   MOV     AH,RTC_DSP_CON     ; GET DISPLAY CONFIGURATION
00F4  E8 0000 E               CALL    GET_RTC_REG
00F7  A8 02                   TEST    AL,DSP_MLCD        ; IS LCD CONFIGURED AS MONO
00F9  75 02                   JNZ     M12                ; YES, CLEAR ENTIRE BUFFER
00FB  B5 08                   MOV     CH,08H             ; BUFFER SIZE ON MONO CARD
00FD                       M12:                          ; NO_GRAPHICS_INIT
00FD  B8 0720                 MOV     AX,' '+7*256       ; FILL CHAR FOR ALPHA
0100                       M13:                          ; CLEAR_BUFFER
0100  F3/ AB                  REP     STOSW              ; FILL REGEN BFR WITH BLANKS

               ;----- ENABLE VIDEO AND CORRECT PORT SETTING

0102  A0 0049 R            M13A:   MOV     AL,CRT_MODE   ; GET THE MODE
0105  32 E4                   XOR     AH,AH              ; INTO AX REGISTER
0107  8B F0                   MOV     SI,AX              ; TBL POINTER, INDEXED BY MODE
0109  8B 16 0063 R            MOV     DX,ADDR_6845       ; PREPARE TO OUTPUT TO
                                                         ; VIDEO ENABLE PORT
010D  83 C2 04                ADD     DX,4
0110  2E: 8A 84 0000 E        MOV     AL,CS:[SI+OFFSET M7]
0115  EE                      OUT     DX,AL              ; SET VIDEO ENABLE PORT
0116  A2 0065 R               MOV     CRT_MODE_SET,AL    ; SAVE THAT VALUE

               ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
               ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE

0119  2E: 8A 84 0000 E        MOV     AL,CS:[SI + OFFSET M6]
011E  32 E4                   XOR     AH,AH
0120  A3 004A R               MOV     CRT_COLS,AX        ; NO. OF COLS IN THIS SCREEN

               ;----- SET CRT LENGTH

0123  81 E6 000E              AND     SI,0EH             ; WORD OFFSET IN CLR LEN TABLE
0127  2E: 8B 8C 0000 E        MOV     CX,CS:[SI + OFFSET M5] ; LENGTH TO CLEAR
```

```
012C  80 3E 0049 R 07       CMP      CRT_MODE,7           ; MONO MODE
0131  75 0C                 JNE      M13C
0133  B4 20                 MOV      AH,RTC_DSP_CON       ; GET DISPLAY CONFIGURATION
0135  E8 0000 E             CALL     GET_RTC_REG
0138  A8 02                 TEST     AL,DSP_MLCD          ; LCD CONFIGURED AS MONO
013A  74 03                 JZ       M13C
013C  B9 1000               MOV      CX,4096              ; SET PG LEN TO 4096 TO ALLOW
                                                          ; MULTI PG KEYS WHEN LCD MONO
013F  89 0E 004C R      M13C:   MOV      CRT_LEN,CX       ; SAVE LENGTH OF CRT

             ;----- SET CURSOR POSITIONS

0143  B9 0008               MOV      CX,8                 ; CLEAR ALL CURSOR POSITIONS
0146  BF 0050 R             MOV      DI,OFFSET CURSOR_POSN
0149  1E                    PUSH     DS                   ; ESTABLISH SEGMENT
014A  07                    POP      ES                   ; ADDRESSING
014B  33 C0                 XOR      AX,AX
014D  F3/ AB                REP      STOSW                ; FILL WITH ZEROES

             ;----- SET UP OVERSCAN REGISTER

014F  42                    INC      DX                   ; SET OVERSCAN PORT TO DEFAULT
0150  B0 30                 MOV      AL,30H               ; VALUE OF 30H FOR ALL MODES
                                                          ; EXCEPT 640X200
0152  80 3E 0049 R 06       CMP      CRT_MODE,6           ; SEE IF MODE IS 640X200 BW
0157  75 02                 JNZ      M14                  ; IF NOT 640X200, GOTO REGULAR
0159  B0 3F                 MOV      AL,3FH               ; IF IT IS 640X200, PUT IN 3FH
015B                    M14:
015B  EE                    OUT      DX,AL                ; SEND CORRECT VAL TO 3D9 PORT
015C  A2 0066 R             MOV      CRT_PALETTE,AL       ; SAVE THE VAL FOR FUTURE USE

             ;----- SET CURSOR TYPE TO DEFAULT VALUES.

015F  B9 0607               MOV      CX,0607H             ; CURSOR MODE FOR MODES 0-6
0162  80 3E 0049 R 07       CMP      CRT_MODE,7           ; IS IT MODE 7
0167  75 03                 JNE      M14A                 ; NO, JUMP ON
0169  B9 0B0C               MOV      CX,0B0CH             ; CURSOR MODE MONO (MODE 7)
016C  EB 4C 90          M14A:   JMP      SET_CTYPE        ; SET CUROSR TYPE

             ;----- NORMAL RETURN FROM ALL VIDEO RETURNS

016F                    VIDEO_RETURN:
016F  5D                    POP      BP
0170  5F                    POP      DI
0171  5E                    POP      SI
0172  5B                    POP      BX
0173  59                    POP      CX
0174  5A                    POP      DX
0175  1F                    POP      DS
0176  07                    POP      ES
0177  CF                    IRET                          ; ALL DONE
0178                    SET_MODE        ENDP
```

```
                ;------------------------------------------------------------
                ; LCD_MOVE
                ;      THIS ROUTINE CHECKS FOR MONOCHROME TO CGA OR VICE
                ;      VERSA CHANGE REQUESTED FOR THE LCD DISPLAY. IF LCD
                ;      IS THE ONLY DISPLAY, A CHANGE IS ALLOWED AND A CALL TO
                ;      LCD_INIT IS MADE TO SET THE CONTROL REGISTERS BEFORE
                ;          CONTINUING WITH THE MODE SET.
                ; INPUT
                ;          AL HAS MODE SET VALUE
                ; OUTPUT
                ;      IF LCD IS NOT THE ONLY DISPLAY THEN A RETURN IS MADE
                ;      WITH NO CHANGES. IF THE LCD IS THE ONLY DISPLAY AND A
                ;      CGA TO MONO OR MONO TO CGA CHANGE IS REQUESTED, THE
                ;      DISPLAY CONFIG BYTE (RTC_DSP_CON) IS UPDATED TO REFLECT
                ;      THE CHANGE. AND, A CALL TO LCD_INIT IS MADE TO UPDATE
                ;      THE LCD CONTROL REGISTERS FOR THE NEW LCD MODE.
                ;      A RETURN IS THEN MADE TO FINISH THE SET MODE. IF NO
                ;      MONO TO CGA OR CGA TO MONO CHANGE IS REQUESTED - NO
                ;      ACTION IS TAKEN.
                ;
                ; REGISTERS MODIFIED: SI
                ;
                ; INTERRUPTS:  DISABLED DURING PROCESSING
                ;------------------------------------------------------------
0178                    LCD_MOVE        PROC    NEAR
0178  50                   PUSH    AX
0179  53                   PUSH    BX
017A  52                   PUSH    DX
017B  9C                   PUSHF                       ; SAVE FLAGS
017C  8A D8                MOV     BL,AL               ; SAVE NEW MODE REQUEST
017E  FA                   CLI                         ; DISABLE INTERRUPTS
017F  B4 20                MOV     AH,RTC_DSP_CON      ; GET DISPLAY CONFIGURATION
0181  E8 0000 E            CALL    GET_RTC_REG         ; DATA RETURNED IN AL
0184  80 FB 07             CMP     BL,7                ; MONOCHROME MODE REQUESTED?
0187  74 0E                JE      LCD_M01
                ;
                ; CGA MODE BEING REQUESTED
                ;
0189  3C 82                CMP     AL,DSP_MLCD+DSP_LCD_PRES; IS LCD PRES & CONFIGURED
                                                       ; MONO & NO OTHER ADA PRESENT
018B  75 28                JNE     LCD_M03             ; JUMP IF CAN'T EFFECT CHANGE
018D  B0 81                MOV     AL,DSP_CLCD+DSP_LCD_PRES; SET LCD TO CGA MODE
018F  BA 03B8              MOV     DX,MONO_CNTL        ; GET MONO CONTROL ADDRESS
0192  B3 01                MOV     BL,01               ; DISABLE VIDEO FOR MONOCHROME
0194  EB 0C 90             JMP     LCD_M02             ; GO CHANGE CONFIGURATION
                ;
                ; MONOCHROME MODE IS BEING REQUESTED
                ;
0197                    LCD_M01:
0197  3C 81                CMP     AL,DSP_CLCD+DSP_LCD_PRES; IS LCD PRES & CONFIGURED
                                                       ; CGA & NO OTHER ADA PRESENT
0199  75 1A                JNE     LCD_M03             ; NO THEN EXIT
019B  B0 82                MOV     AL,DSP_MLCD+DSP_LCD_PRES; SET CONFIG LCD MONO MODE
019D  BA 03D8              MOV     DX,CGA_CNTL         ; MODE CTRL FOR PRESENT STATE
01A0  2A DB                SUB     BL,BL               ; DISABLE VIDEO FOR CGA
                ;
                ; LCD ADAPTER CHANGE CAN BE MADE
                ;
01A2                    LCD_M02:
01A2  E8 0000 E            CALL    PUT_RTC_REG         ; UPDATE DISPLAY CONFIGURATION
01A5  8A C3                MOV     AL,BL               ; DISABLE VIDEO IN CRNT STATE
01A7  EE                   OUT     DX,AL
01A8  E8 0000 E            CALL    LCDINIT             ; SET UP LCD
01AB  B0 00                MOV     AL,LCD_FUNCT        ; AL = 0
01AD  E6 74                OUT     LCD_INDX,AL         ; PORT 74
01AF  E4 75                IN      AL,LCD_DATA         ; GET LCDC REGS
01B1  0C 40                OR      AL,PANEL_ENABLE     ; TURN ON PANEL POWER
01B3  E6 75                OUT     LCD_DATA,AL

01B5                    LCD_M03:
01B5  9D                   POPF                        ; RESTORE INTERRUPT FLAGS
01B6  5A                   POP     DX                  ; RESTORE REGISTERS
01B7  5B                   POP     BX
01B8  58                   POP     AX
```

```
01B9  C3                     RET
01BA                    LCD_MOVE ENDP

              ;------------------------------------------------------------
              ; SET_CTYPE
              ;         THIS ROUTINE SETS THE CURSOR VALUE.
              ;         SPECIAL HANDLING OCCURS FOR THE LCD CURSOR.
              ;         IF THE LCD IS THE CURRENT DISPLAY THE FOLLOWING OCCURS:
              ;         FOR THE DISPLAY NO CURSOR MODE (BIT 5 = 1 & BIT 6 = 0 OF THE
              ;         CH REGISTER) IS CONVERTED TO THE LCD CONTROLLER DISPLAY NO
              ;         CURSOR MODE (CX=0808H). ALSO, THE BLINKING MODE BITS (BIT 5
              ;         AND BIT 6 OF THE CH REGISTER) ARE TURNED OFF.
              ;
              ;         IF LCD CONFIGURED AS MONO IS THE CURRENT DISPLAY THEN THE
              ;         LCDC REGISTERS FOR LCD CURSOR START AND LCD CURSOR END ARE
              ;         REGISTERS 23 AND 24 RESPECTIVELY. AND THE CURSOR VALUES ARE
              ;         RESCALED TO FIT IN AN 8x8 CHARACTER BOX. THIS IS BECAUSE
              ;         THE MONO CHARACTER BOX IS 9x14 BUT THE LCD HAS AN 8x8
              ;         CHARACTER BOX.
              ;
              ; INPUT
              ;         (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
              ;
              ;         (DS) = DATA SEGMENT
              ; OUTPUT
              ;         PHYSICAL CURSOR SET
              ;------------------------------------------------------------

01BA                    SET_CTYPE       PROC    NEAR

01BA  89 0E 0060 R         MOV     CURSOR_MODE,CX      ; SAVE CURSOR VAL IN DATA AREA
01BE  B4 20                MOV     AH,RTC_DSP_CON      ; GET LCD CONFIGURATION
01C0  E8 0000 E            CALL    GET_RTC_REG         ; FROM RTC.
01C3  B4 0A                MOV     AH,10               ; CURSOR START REGISTER
01C5  81 3E 0063 R 03D4    CMP     ADDR_6845,03D4H     ; ARE WE IN CGA MODE
01CB  74 33                JE      SET_2               ; JUMP TO CGA MODE TEST

              ; MONO IS CURRENT MODE

01CD  A8 02                TEST    AL,DSP_MLCD         ; IS LCD AS MONO CRNT DISPLAY?
01CF  74 45                JZ      SET_C               ; NO, JUMP ON IT IS MONO MNTR
01D1  B4 17                MOV     AH,017H             ; LCDC CUR ST REG AS MONO
01D3  F6 C5 40             TEST    CH,40H              ; IS BIT 6 ON
01D6  75 05                JNZ     SET_1               ; YES , JUMP ON
01D8  F6 C5 20             TEST    CH,20H              ; IS BIT 5 ON (NO CURSOR)
01DB  75 36                JNZ     SET_4               ; YES, JUMP TO RST TO LCDC VAL
01DD  80 E5 9F          SET_1:  AND     CH,09FH        ; TURN OFF BLINK (BIT 5 & 6)

              ; WHEN THE LCD AS MONO IS CURRENT MODE THE CURS IS RESCALED TO FIT IN
              ; A 8x8 CHARACTER BOX. THE CURSOR WILL ONLY BE RESCALED IF CURS START
              ; (CH) AND CURSOR END (CL) VALUES ARE BETWEEN VALUES 0 - 13.
01E0  80 FD 0D             CMP     CH,13               ; INVALID CURSOR START VALUE?
01E3  77 31                JA      SET_C               ; YES, PROCESS AS IS.
01E5  80 F9 0D             CMP     CL,13               ; INVALID CURSOR END VALUE?
01E8  77 2C                JA      SET_C               ; YES, PROCESS AS IS
01EA  50                   PUSH    AX                  ; SAVE LCDC CURSOR START REG
01EB  BB 080E              MOV     BX,080EH            ; BH - MULT, BL - DIV FACTOR
01EE  8A C5                MOV     AL,CH               ; RESCALE CURSOR START
01F0  E8 021C R            CALL    SET_RESCALE
01F3  8A E8                MOV     CH,AL               ; NEW CUROSR START

01F5  8A C1                MOV     AL,CL               ; RESCALE CURSOR END
01F7  E8 021C R            CALL    SET_RESCALE
01FA  8A C8                MOV     CL,AL               ; NEW CUROSR START
01FC  58                   POP     AX                  ; RESTORE LCDC CUSR START REG
01FD  EB 17 90             JMP     SET_C               ; CALL TO OUTPUT CX REG
              ;
              ; CGA IS CURRENT MODE.
              ;
0200                    SET_2:
0200  A8 01                TEST    AL,DSP_CLCD         ; IS LCD AS CGA CURRENT DSPLY?
0202  74 12                JZ      SET_C               ; NO, ITS CGA MONITOR- AS IS
0204  F6 C5 40             TEST    CH,40H              ; IS BIT 6 ON
0207  74 05                JZ      SET_3               ; NO, JUMP ON
```

```
0209  80 E5 9F              AND     CH,09FH             ; TURN OFF BLINK (BIT 5 & 6)
020C  EB 08                 JMP     SHORT SET_C
020E                     SET_3:
020E  F6 C5 20              TEST    CH,20H              ; IS BIT 5 ON (DSPLY NO CUSR)
0211  74 03                 JZ      SET_C               ; NO, JUMP ON
0213  B9 0808            SET_4:  MOV     CX,0808H       ; LCDC VAL FOR DSPLY NO CURSOR
0216                     SET_C:
0216  E8 0228 R             CALL    M16                 ; OUTPUT CX REG
0219  E9 016F R             JMP     VIDEO_RETURN

                 ; THE RESCALING FORMULA IS (( X * 8) / 14) + ROUND UP. X IS EITHER
                 ; CH REGISTER OR CL REGISTER, 8 IS FOR THE LCD CHAR BOX HEIGHT(8 x 8),
                 ; AND 14 IS FOR THE MONO CHARACTER BOX HEIGHT(9 x 14).  ROUND UP IS 1
                 ; IF THE REMAINDER IS GREATER THAN 6. AND, RND UP IS 0 IF REMNDR IS <
                 ; THAN 7. ON ENTRANCE AL WILL CONTAIN X, BH = 8 AND BL = 14.
021C                     SET_RESCALE:
021C  F6 E7                 MUL     BH                  ; MULT BY LCD CHAR BX LEN 8
021E  F6 F3                 DIV     BL                  ; DIV BY MONO CHAR BX LEN 14
0220  80 FC 07              CMP     AH,7                ; AH = RMNDR , AL = QUOTIENT
0223  72 02                 JB      SET_R1
0225  FE C0                 INC     AL                  ; ADD 1 TO QUOTIENT FOR RND UP
0227  C3                 SET_R1: RET                    ; RETURN TO CALLER


                 ;----- THIS ROUTINE OUTPUTS THE CX REGISTER TO THE REGS NAMED IN AH

0228                     M16:
0228  9C                    PUSHF                       ; SAVE CURRENT FLAGS
0229  FA                    CLI                         ; INHIBIT INTERRUPTS
022A  8B 16 0063 R          MOV     DX,ADDR_6845        ; ADDRESS REGISTER
022E  8A C4                 MOV     AL,AH               ; GET VALUE
0230  EE                    OUT     DX,AL               ; REGISTER SET
0231  42                    INC     DX                  ; DATA REGISTER
0232  8A C5                 MOV     AL,CH               ; DATA
0234  EE                    OUT     DX,AL
0235  4A                    DEC     DX
0236  8A C4                 MOV     AL,AH
0238  FE C0                 INC     AL                  ; POINT TO OTHER DATA REGISTER
023A  EE                    OUT     DX,AL               ; SET FOR SECOND REGISTER
023B  42                    INC     DX
023C  8A C1                 MOV     AL,CL               ; SECOND DATA VALUE
023E  EE                    OUT     DX,AL
023F  9D                    POPF                        ; RESTORE FLAGS
0240  C3                    RET                         ; ALL DONE
0241                     SET_CTYPE      ENDP

                 ;-----------------------------------------------
                 ; SET_CPOS
                 ;      THIS ROUTINE SETS THE CURRENT CURSOR
                 ;      POSITION TO THE NEW X-Y VALUES PASSED
                 ; INPUT
                 ;      DX - ROW,COLUMN OF NEW CURSOR
                 ;      BH - DISPLAY PAGE OF CURSOR
                 ;      *** BH = 0 FOR GRAPHICS  ***
                 ;
                 ;      DS - DATA SEGMENT
                 ; OUTPUT
                 ;      CURSOR IS SET AT 6845 IF DISPLAY PAGE
                 ;      IS CURRENT DISPLAY
                 ;-----------------------------------------------
0241                     SET_CPOS       PROC    NEAR
0241  8A CF                 MOV     CL,BH
0243  32 ED                 XOR     CH,CH               ; ESTABLISH LOOP COUNT
0245  D1 E1                 SAL     CX,1                ; WORD OFFSET
0247  8B F1                 MOV     SI,CX               ; USE INDEX REGISTER
0249  89 94 0050 R          MOV     [SI+OFFSET CURSOR_POSN],DX ; SAVE THE POINTER
024D  38 3E 0062 R          CMP     ACTIVE_PAGE,BH
0251  75 05                 JNZ     M17                 ; SET_CPOS_RETURN
0253  8B C2                 MOV     AX,DX               ; GET ROW/COLUMN TO AX
0255  E8 025B R             CALL    M18                 ; CURSOR_SET
0258                     M17:                           ; SET_CPOS_RETURN
0258  E9 016F R             JMP     VIDEO_RETURN
025B                     SET_CPOS       ENDP
```

```
                ;----- SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR

025B                     M18     PROC    NEAR
025B  E8 02E6 R             CALL    POSITION            ; FIND LOCATION IN REGEN BFR
025E  8B C8                 MOV     CX,AX
0260  03 0E 004E R          ADD     CX,CRT_START        ; ADD START ADDR FOR THIS PAGE
0264  D1 F9                 SAR     CX,1                ; DIVIDE BY 2 FOR CHAR COUNT
0266  B4 0E                 MOV     AH,14               ; REGISTER NUMBER FOR CURSOR
0268  E8 0228 R             CALL    M16                 ; OUTPUT THE VALUE TO THE 6845
026B  C3                    RET
026C                     M18     ENDP

                ;-------------------------------------------------------------
                ; ACT_DISP_PAGE
                ;           THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING THE
                ;           FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT.
                ;           **** VALID ONLY FOR ALPHA MODES   ****
                ; INPUT
                ;           AL HAS THE NEW ACTIVE DISPLAY PAGE
                ;               (0-7) FOR MODES 0&1, (0-3) FOR MODES 2&3, AND
                ;               (0)   FOR MODE 7 WHEN MONOCHROME MONITOR
                ;               (0-3) FOR MODE 7 WHEN THE LCD IS CONFIGURED AS MONO
                ;
                ;           (DS) = DATA SEGMENT
                ; OUTPUT
                ;           THE 6845 IS RESET TO DISPLAY THAT PAGE
                ;-------------------------------------------------------------
026C                     ACT_DISP_PAGE   PROC    NEAR
026C  A2 0062 R             MOV     ACTIVE_PAGE,AL      ; SAVE ACTIVE PAGE VALUE
026F  8B 0E 004C R          MOV     CX,CRT_LEN          ; GET SAVED LEN OF REGEN BFR
0273  98                    CBW                         ; CONVERT AL TO WORD
0274  50                    PUSH    AX                  ; SAVE PAGE VALUE
0275  F7 E1                 MUL     CX                  ; DISPLAY PAGE TIMES REGEN LEN
0277  A3 004E R             MOV     CRT_START,AX        ; SAVE START ADDRESS FOR
                                                        ; LATER REQUIREMENTS
027A  8B C8                 MOV     CX,AX               ; START ADDRESS TO CX
027C  D1 F9                 SAR     CX,1                ; DIVIDE BY 2 FOR 6845
027E  B4 0C                 MOV     AH,12               ; 6845 REGISTER FOR START ADDR
0280  E8 0228 R             CALL    M16
0283  5B                    POP     BX                  ; RECOVER PAGE VALUE
0284  D1 E3                 SAL     BX,1                ; *2 FOR WORD OFFSET
0286  8B 87 0050 R          MOV     AX,[BX + OFFSET CURSOR_POSN- ; GET CURSOR FOR PAGE
028A  E8 025B R             CALL    M18                 ; SET THE CURSOR POSITION
028D  E9 016F R             JMP     VIDEO_RETURN
0290                     ACT_DISP_PAGE   ENDP

                ;-------------------------------------------------------------
                ; READ_CURSOR
                ;           THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                ;           6845, AND SENDS IT BACK TO THE CALLER
                ; INPUT
                ;           BH - PAGE OF CURSOR, MUST BE 0 FOR GRAPHICS
                ;
                ;           DS - DATA SEGMENT
                ; OUTPUT
                ;           DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
                ;           CX - CURRENT CURSOR MODE
                ;-------------------------------------------------------------
0290                     READ_CURSOR     PROC    NEAR
0290  8A DF                 MOV     BL,BH
0292  32 FF                 XOR     BH,BH
0294  D1 E3                 SAL     BX,1                ; WORD OFFSET
0296  8B 97 0050 R          MOV     DX,[BX+OFFSET CURSOR_POSN]
029A  8B 0E 0060 R          MOV     CX,CURSOR_MODE
029E  55                    PUSH    BP
029F  8B EC                 MOV     BP,SP               ; GET PTR TO STACK SAVE AREA
02A1  89 56 0C              MOV     [BP].DX_POS,DX      ; SETUP RETURN VALUES IN STACK
02A4  89 4E 0A              MOV     [BP].CX_POS,CX      ; SAVE AREA
02A7  5D                    POP     BP
02A8  E9 016F R             JMP     VIDEO_RETURN
02AB                     READ_CURSOR     ENDP
```

```
               ;--------------------------------------------------------------------
               ; SET COLOR
               ;          THIS ROUTINE WILL ESTABLISH BACKGND COLOR, THE OVERSCAN
               ;          COLOR, AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION
               ;          GRAPHICS
               ; INPUT
               ;          (BH) HAS COLOR ID
               ;               IF BH=0, THE BACKGROUND COLOR VALUE IS SET
               ;                       FROM THE LOW BITS OF BL (0-31)
               ;               IF BH=1, THE PALETTE SELECTION IS MADE
               ;                       BASED ON THE LOW BIT OF BL:
               ;                             0=GREEN, RED, YELLOW FOR COLORS 1,2,3
               ;                             1=BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
               ;          (BL) HAS THE COLOR VALUE TO BE USED
               ;
               ;          (DS) - DATA SEGMENT
               ; OUTPUT
               ;          THE COLOR SELECTION IS UPDATED
               ;--------------------------------------------------------------------
02AB                    SET_COLOR       PROC    NEAR
02AB  8B 16 0063 R         MOV     DX,ADDR_6845        ; I/O PORT FOR PALETTE
02AF  83 C2 05             ADD     DX,5                ; OVERSCAN PORT
02B2  A0 0066 R            MOV     AL,CRT_PALETTE      ; GET THE CURRENT PALETTE VAL
02B5  0A FF                OR      BH,BH               ; IS THIS COLOR 0?
02B7  75 0E                JNZ     M20                 ; OUTPUT COLOR 1

               ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

02B9  24 E0                AND     AL,0E0H             ; TURN OFF LOW 5 BITS OF CRNT
02BB  80 E3 1F             AND     BL,01FH             ; TURN OFF H 3 BITS INPUT VAL
02BE  0A C3                OR      AL,BL               ; PUT VALUE INTO REGISTER
02C0                    M19:                           ; OUTPUT THE PALETTE
02C0  EE                   OUT     DX,AL               ; SEND COLOR TO 3D9 PORT
02C1  A2 0066 R            MOV     CRT_PALETTE,AL      ; SAVE THE COLOR VALUE
02C4  E9 016F R            JMP     VIDEO_RETURN

               ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED

02C7                    M20:
02C7  24 DF                AND     AL,0DFH             ; TURN OFF PALETTE SELECT BIT
02C9  D0 EB                SHR     BL,1                ; TEST THE LOW ORDER BIT OF BL
02CB  73 F3                JNC     M19                 ; ALREADY DONE
02CD  0C 20                OR      AL,20H              ; TURN ON PALETTE SELECT BIT
02CF  EB EF                JMP     M19                 ; GO DO IT
02D1                    SET_COLOR       ENDP

               ;-----------------------------------------------
               ; VIDEO STATE
               ;          RETURNS THE CURRENT VIDEO STATE INFORMATION
               ; INPUT
               ;          DS = DATA SEGMENT
               ; OUTPUT
               ;          AH = NUMBER OF COLUMNS ON THE SCREEN
               ;          AL = CURRENT VIDEO MODE
               ;          BH = CURRENT ACTIVE PAGE
               ;-----------------------------------------------
02D1                    VIDEO_STATE     PROC    NEAR
02D1  8A 26 004A R         MOV     AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
02D5  A0 0049 R            MOV     AL,CRT_MODE         ; CURRENT MODE
02D8  8A 3E 0062 R         MOV     BH,ACTIVE_PAGE      ; GET CURRENT ACTIVE PAGE
02DC  55                   PUSH    BP
02DD  8B EC                MOV     BP,SP               ; GET PTR TO STACK SAVE AREA
02DF  89 5E 08             MOV     [BP].BX_POS,BX      ; SETUP RETURN VALUES IN STACK
02E2  5D                   POP     BP
02E3  E9 016F R            JMP     VIDEO_RETURN
02E6                    VIDEO_STATE     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
                ;-------------------------------------------------------
02E6                    POSITION        PROC    NEAR
02E6  53                   PUSH    BX                  ; SAVE REGISTER
02E7  8B D8                MOV     BX,AX
02E9  8A C4                MOV     AL,AH               ; ROWS TO AL
02EB  F6 26 004A R         MUL     BYTE PTR CRT_COLS   ; DETERMINE BYTES TO ROW
02EF  32 FF                XOR     BH,BH
02F1  03 C3                ADD     AX,BX               ; ADD IN COLUMN VALUE
02F3  D1 E0                SAL     AX,1                ; * 2 FOR ATTRIBUTE BYTES
02F5  5B                   POP     BX
02F6  C3                   RET
02F7                    POSITION        ENDP

                ;-------------------------------------------------------
                ; SCROLL UP
                ;          THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                ;          ON THE SCREEN
                ; INPUT
                ;          (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
                ;          (AH) = CURRENT CRT MODE
                ;          (DS) = DATA SEGMENT
                ;          (ES) = REGEN BUFFER SEGMENT
                ; OUTPUT
                ;          NONE -- THE REGEN BUFFER IS MODIFIED
                ;-------------------------------------------------------
                           ASSUME  CS:ROMCODE,DS:DATA,ES:DATA
02F7                    SCROLL_UP       PROC    NEAR
02F7  8A D8                MOV     BL,AL               ; SAVE LINE COUNT IN BL
02F9  80 FC 04             CMP     AH,4                ; TEST FOR GRAPHICS MODE
02FC  72 08                JC      N1                  ; HANDLE SEPARATELY
02FE  80 FC 07             CMP     AH,7                ; TEST FOR MONO CARD
0301  74 03                JE      N1
0303  E9 04DA R            JMP     GRAPHICS_UP
0306                    N1:                            ; UP_CONTINUE
0306  53                   PUSH    BX                  ; SAVE FILL ATTRIBUTE IN BH
0307  8B C1                MOV     AX,CX               ; UPPER LEFT POSITION
0309  E8 0343 R            CALL    SCROLL_POSITION     ; DO SETUP FOR SCROLL
030C  74 31                JZ      N7                  ; BLANK_FIELD
030E  03 F0                ADD     SI,AX               ; FROM ADDRESS
0310  8A E6                MOV     AH,DH               ; # ROWS IN BLOCK
0312  2A E3                SUB     AH,BL               ; # ROWS TO BE MOVED
0314                    N2:                            ; ROW_LOOP
0314  E8 036A R            CALL    N10                 ; MOVE ONE ROW
0317  03 F5                ADD     SI,BP
0319  03 FD                ADD     DI,BP               ; POINT TO NEXT LINE IN BLOCK
031B  FE CC                DEC     AH                  ; COUNT OF LINES TO MOVE
031D  75 F5                JNZ     N2                  ; ROW_LOOP
031F                    N3:                            ; CLEAR_ENTRY
031F  58                   POP     AX                  ; RECOVER ATTRIBUTE IN AH
0320  B0 20                MOV     AL,' '              ; FILL WITH BLANKS
0322                    N4:                            ; CLEAR_LOOP
0322  E8 0373 R            CALL    N11                 ; CLEAR THE ROW
0325  03 FD                ADD     DI,BP               ; POINT TO NEXT LINE
0327  FE CB                DEC     BL                  ; COUNTER OF LINES TO SCROLL
0329  75 F7                JNZ     N4                  ; CLEAR_LOOP
032B                    N5:                            ; SCROLL_END
032B  E8 0000 E            CALL    DDS
032E  80 3E 0049 R 07      CMP     CRT_MODE,7          ; IS THIS THE MONO CARD
0333  74 07                JE      N6                  ; IF SO, SKIP THE MODE RESET
0335  A0 0065 R            MOV     AL,CRT_MODE_SET     ; GET VALUE OF THE MODE SET
0338  BA 03D8              MOV     DX,03D8H            ; ALWAYS SET CGA CARD PORT
033B  EE                   OUT     DX,AL
033C                    N6:                            ; VIDEO_RET_HERE
```

```
033C  E9 016F R            JMP     VIDEO_RETURN
033F                    N7:                              ; BLANK_FIELD
033F  8A DE                MOV     BL,DH                 ; GET ROW COUNT
0341  EB DC                JMP     N3                    ; GO CLEAR THAT AREA
0343                    SCROLL_UP       ENDP

               ;
               ;----- HANDLE COMMON SCROLL SET UP HERE
               ;

0343                    SCROLL_POSITION PROC    NEAR
0343  E8 02E6 R            CALL    POSITION              ; CONVERT TO REGEN POINTER
0346  03 06 004E R         ADD     AX,CRT_START          ; OFFSET OF ACTIVE PAGE
034A  8B F8                MOV     DI,AX                 ; TO ADDRESS FOR SCROLL
034C  8B F0                MOV     SI,AX                 ; FROM ADDRESS FOR SCROLL
034E  2B D1                SUB     DX,CX                 ; DX = #ROWS, #COLS IN BLOCK
0350  FE C6                INC     DH
0352  FE C2                INC     DL                    ; INCREMENT FOR 0 ORIGIN
0354  32 ED                XOR     CH,CH                 ; SET HIGH BYTE OF CNT TO ZERO
0356  8B 2E 004A R         MOV     BP,CRT_COLS           ; GET NO. OF COLS IN DISPLAY
035A  03 ED                ADD     BP,BP                 ; TIMES 2 FOR ATTRIBUTE BYTE
035C  8A C3                MOV     AL,BL                 ; GET LINE COUNT
035E  F6 26 004A R         MUL     BYTE PTR CRT_COLS     ; FIND OFFSET TO FROM ADDRESS
0362  03 C0                ADD     AX,AX                 ; *2 FOR ATTRIBUTE BYTE
0364  06                   PUSH    ES                    ; ESTAB ADDSSNG TO REGEN BFR
0365  1F                   POP     DS                    ; FOR BOTH POINTERS
0366  80 FB 00             CMP     BL,0                  ; 0 SCROLL MEANS BLANK FIELD
0369  C3                   RET                           ; RETURN WITH FLAGS SET
036A                    SCROLL_POSITION ENDP

               ;
               ;----- MOVE_ROW
               ;

036A                    N10     PROC    NEAR
036A  8A CA                MOV     CL,DL                 ; GET # OF COLS TO MOVE
036C  56                   PUSH    SI
036D  57                   PUSH    DI                    ; SAVE START ADDRESS
036E  F3/ A5               REP     MOVSW                 ; MOVE THAT LINE ON SCREEN
0370  5F                   POP     DI
0371  5E                   POP     SI                    ; RECOVER ADDRESSES
0372  C3                   RET
0373                    N10     ENDP

               ;
               ;----- CLEAR_ROW
               ;

0373                    N11     PROC    NEAR
0373  8A CA                MOV     CL,DL                 ; GET # COLUMNS TO CLEAR
0375  57                   PUSH    DI
0376  F3/ AB               REP     STOSW                 ; STORE THE FILL CHARACTER
0378  5F                   POP     DI
0379  C3                   RET
037A                    N11     ENDP

               ;--------------------------------------------------------
               ; SCROLL_DOWN
               ;          THIS ROUTINE MOVES THE CHARACTERS WITHIN A
               ;          DEFINED BLOCK DOWN ON THE SCREEN, FILLING THE
               ;          TOP LINES WITH A DEFINED CHARACTER
               ; INPUT
               ;          (AL) = NUMBER OF LINES TO SCROLL
               ;          (CX) = UPPER LEFT CORNER OF REGION
               ;          (DX) = LOWER RIGHT CORNER OF REGION
               ;          (BH) = FILL CHARACTER
               ;          (AH) = CURRENT CRT MODE
               ;          (DS) = DATA SEGMENT
               ;          (ES) = REGEN SEGMENT
               ; OUTPUT
               ;          NONE -- SCREEN IS SCROLLED
               ;--------------------------------------------------------
037A                    SCROLL_DOWN     PROC    NEAR
037A  FD                   STD                           ; DIRECTION FOR SCROLL DOWN
```

```
037B  8A D8                MOV    BL,AL              ; LINE COUNT TO BL
037D  80 FC 04             CMP    AH,4               ; TEST FOR GRAPHICS
0380  72 08                JC     N12
0382  80 FC 07             CMP    AH,7               ; TEST FOR MONO CARD
0385  74 03                JE     N12
0387  E9 0533 R            JMP    GRAPHICS_DOWN
038A                    N12:                         ; CONTINUE_DOWN
038A  53                   PUSH   BX                 ; SAVE ATTRIBUTE IN BH
038B  8B C2                MOV    AX,DX              ; LOWER RIGHT CORNER
038D  E8 0343 R            CALL   SCROLL_POSITION    ; GET REGEN LOCATION
0390  74 20                JZ     N16
0392  2B F0                SUB    SI,AX              ; SI IS FROM ADDRESS
0394  8A E6                MOV    AH,DH              ; GET TOTAL # ROWS
0396  2A E3                SUB    AH,BL              ; COUNT TO MOVE IN SCROLL
0398                    N13:
0398  E8 036A R            CALL   N10                ; MOVE ONE ROW
039B  2B F5                SUB    SI,BP
039D  2B FD                SUB    DI,BP
039F  FE CC                DEC    AH
03A1  75 F5                JNZ    N13
03A3                    N14:
03A3  58                   POP    AX                 ; RECOVER ATTRIBUTE IN AH
03A4  B0 20                MOV    AL,' '
03A6                    N15:
03A6  E8 0373 R            CALL   N11                ; CLEAR ONE ROW
03A9  2B FD                SUB    DI,BP              ; GO TO NEXT ROW
03AB  FE CB                DEC    BL
03AD  75 F7                JNZ    N15
03AF  E9 032B R            JMP    N5                 ; SCROLL_END
03B2                    N16:
03B2  8A DE                MOV    BL,DH
03B4  EB ED                JMP    N14
03B6                    SCROLL_DOWN     ENDP

                  ;--------------------------------------------------------
                  ; READ_AC_CURRENT
                  ;          THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER
                  ;          AT THE CURRENT CURSOR POSITION AND RETURNS THEM
                  ;          TO THE CALLER
                  ;INPUT
                  ;          (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
                  ;          (AH) = CURRENT CRT MODE
                  ;          (DS) = DATA SEGMENT
                  ;          (ES) = REGEN SEGMENT
                  ;OUTPUT
                  ;          (AL) = CHAR READ
                  ;          (AH) = ATTRIBUTE READ
                  ;
                  ;INTERRUPTS: DISABLED DURING THE READ
                  ;--------------------------------------------------------
                           ASSUME  CS:ROMCODE,DS:DATA,ES:DATA
03B6                    READ_AC_CURRENT PROC    NEAR
03B6  80 FC 04             CMP    AH,4               ; IS THIS GRAPHICS
03B9  72 08                JC     P1
03BB  80 FC 07             CMP    AH,7               ; IS THIS MONO CARD
03BE  74 03                JE     P1
03C0  E9 066F R            JMP    GRAPHICS_READ
03C3                    P1:                          ; READ_AC_CONTINUE
03C3  E8 03E0 R            CALL   FIND_POSITION
03C6  8B F3                MOV    SI,BX              ; ESTABLISH ADDRESSING IN SI

                  ;----- WAIT FOR HORIZONTAL RETRACE

03C8  8B 16 0063 R         MOV    DX,ADDR_6845       ; GET BASE ADDRESS
03CC  83 C2 06             ADD    DX,6               ; POINT AT STATUS PORT
03CF  06                   PUSH   ES
03D0  1F                   POP    DS                 ; GET SEGMENT FOR QUICK ACCESS
03D1                    P2:                          ; WAIT FOR RETRACE LOW
03D1  EC                   IN     AL,DX              ; GET STATUS
03D2  A8 01                TEST   AL,HORZ_RETRACE    ; IS HORZ RETRACE LOW
03D4  75 FB                JNZ    P2                 ; WAIT UNTIL IT IS
03D6  FA                   CLI                       ; NO MORE INTERRUPTS
03D7                    P3:                          ; WAIT FOR RETRACE HIGH
03D7  EC                   IN     AL,DX              ; GET STATUS
```

```
03D8  A8 01                 TEST    AL,HORZ_RETRACE      ; IS IT HIGH
03DA  74 FB                 JZ      P3                   ; WAIT UNTIL IT IS
03DC  AD                    LODSW                        ; GET THE CHAR/ATTR
03DD  E9 016F R             JMP     VIDEO_RETURN
03E0                    READ_AC_CURRENT ENDP

03E0                    FIND_POSITION   PROC    NEAR
03E0  8A CF                 MOV     CL,BH                ; DISPLAY PAGE TO CX
03E2  32 ED                 XOR     CH,CH
03E4  8B F1                 MOV     SI,CX                ; MOVE TO SI FOR INDEX
03E6  D1 E6                 SAL     SI,1                 ; * 2 FOR WORD OFFSET
03E8  8B 84 0050 R          MOV     AX,[SI+ OFFSET CURSOR_POSN- ; GET ROW/COL OF PAGE
03EC  33 DB                 XOR     BX,BX                ; SET START ADDRESS TO ZERO
03EE  E3 06                 JCXZ    P5                   ; NO_PAGE
03F0                    P4:                              ; PAGE_LOOP
03F0  03 1E 004C R          ADD     BX,CRT_LEN           ; LENGTH OF BUFFER
03F4  E2 FA                 LOOP    P4
03F6                    P5:                              ; NO_PAGE
03F6  E8 02E6 R             CALL    POSITION             ; DETERMINE LOCATION IN REGEN
03F9  03 D8                 ADD     BX,AX                ; ADD TO START OF REGEN
03FB  C3                    RET
03FC                    FIND_POSITION   ENDP

                ;------------------------------------------------
                ; WRITE_AC_CURRENT
                ;           THIS ROUTINE WRITES THE ATTRIBUTE
                ;           AND CHARACTER AT THE CURRENT CURSOR
                ;           POSITION
                ; INPUT
                ;           (AL) = CHAR TO WRITE
                ;           (BH) = DISPLAY PAGE
                ;           (BL) = ATTRIBUTE OF CHAR TO WRITE
                ;           (CX) = COUNT OF CHARACTERS TO WRITE
                ;           (AH) = CURRENT CRT MODE
                ;           (DS) = DATA SEGMENT
                ;           (ES) = REGEN SEGMENT
                ; OUTPUT
                ;           NONE
                ;
                ;INTERRUPTS: DISABLED DURING THE WRITE
                ;------------------------------------------------
03FC                    WRITE_AC_CURRENT        PROC    NEAR
03FC  80 FC 04              CMP     AH,4                 ; IS THIS GRAPHICS
03FF  72 08                 JC      P6
0401  80 FC 07              CMP     AH,7                 ; IS THIS MONO CARD
0404  74 03                 JE      P6
0406  E9 05BD R             JMP     GRAPHICS_WRITE
0409                    P6:                              ; WRITE_AC_CONTINUE
0409  8A E3                 MOV     AH,BL                ; GET ATTRIBUTE TO AH
040B  50                    PUSH    AX                   ; SAVE CHAR/ATTRIBUTE
040C  51                    PUSH    CX                   ; SAVE WRITE COUNT
040D  E8 03E0 R             CALL    FIND_POSITION
0410  8B FB                 MOV     DI,BX                ; ADDRESS TO DI REGISTER
0412  59                    POP     CX                   ; WRITE COUNT
0413  5B                    POP     BX                   ; CHARACTER/ATTR IN BX REG
0414                    P7:                              ; WRITE_LOOP

                ;----- WAIT FOR HORIZONTAL RETRACE

0414  8B 16 0063 R          MOV     DX,ADDR_6845         ; GET BASE ADDRESS
0418  83 C2 06              ADD     DX,6                 ; POINT AT STATUS PORT
041B  9C                    PUSHF                        ; SAVE CURRENT FLAGS
041C                    P8:
041C  EC                    IN      AL,DX                ; GET STATUS
041D  A8 01                 TEST    AL,HORZ_RETRACE      ; IS IT LOW
041F  75 FB                 JNZ     P8                   ; WAIT UNTIL IT IS
0421  FA                    CLI                          ; NO MORE INTERRUPTS
0422                    P9:
0422  EC                    IN      AL,DX                ; GET STATUS
0423  A8 01                 TEST    AL,HORZ_RETRACE      ; IS IT HIGH
0425  74 FB                 JZ      P9                   ; WAIT UNTIL IT IS
0427  8B C3                 MOV     AX,BX                ; RECOVER THE CHAR/ATTR
0429  AB                    STOSW                        ; PUT THE CHAR/ATTR
042A  9D                    POPF                         ; INTERRUPTS BACK ON - IF ON
```

```
042B  E2 E7               LOOP    P7                ; AS MANY TIMES AS REQUESTED
042D  E9 016F R           JMP     VIDEO_RETURN
0430                   WRITE_AC_CURRENT       ENDP

                ;--------------------------------------------
                ; WRITE_C_CURRENT
                ; ** NOT VALID FOR MEDIUM RESOLUTION GRAPHICS **
                ;          THIS ROUTINE WRITES THE CHARACTER AT
                ;          THE CURRENT CURSOR POSITION, ATTRIBUTE
                ;          UNCHANGED
                ; INPUT
                ;          (BH) = DISPLAY PAGE
                ;          (CX) = COUNT OF CHARACTERS TO WRITE
                ;          (AL) = CHAR TO WRITE
                ;          (AH) = CURRENT CRT MODE
                ;          (DS) = DATA SEGMENT
                ;          (ES) = REGEN SEGMENT
                ; OUTPUT
                ;          NONE
                ;
                ; INTERRUPTS: DISABLED DURING THE WRITE
                ;--------------------------------------------
0430                   WRITE_C_CURRENT PROC    NEAR
0430  80 FC 04            CMP     AH,4              ; IS THIS GRAPHICS
0433  72 08               JC      P10
0435  80 FC 07            CMP     AH,7              ; IS THIS MONO CARD
0438  74 03               JE      P10
043A  E9 05BD R           JMP     GRAPHICS_WRITE
043D                   P10:
043D  50                  PUSH    AX                ; SAVE CHAR ON STACK
043E  51                  PUSH    CX                ; SAVE WRITE COUNT
043F  E8 03E0 R           CALL    FIND_POSITION
0442  8B FB               MOV     DI,BX             ; ADDRESS TO DI
0444  59                  POP     CX                ; WRITE COUNT
0445  5B                  POP     BX                ; BL HAS CHAR TO WRITE
0446                   P11:                         ; WRITE_LOOP

                ;----- WAIT FOR HORIZONTAL RETRACE

0446  8B 16 0063 R        MOV     DX,ADDR_6845      ; GET BASE ADDRESS
044A  83 C2 06            ADD     DX,6              ; POINT AT STATUS PORT
044D  9C                  PUSHF                     ; SAVE CURRENT FLAGS
044E                   P12:
044E  EC                  IN      AL,DX             ; GET STATUS
044F  A8 01               TEST    AL,HORZ_RETRACE   ; IS IT LOW
0451  75 FB               JNZ     P12               ; WAIT UNTIL IT IS
0453  FA                  CLI                       ; NO MORE INTERRUPTS
0454                   P13:
0454  EC                  IN      AL,DX             ; GET STATUS
0455  A8 01               TEST    AL,HORZ_RETRACE   ; IS IT HIGH
0457  74 FB               JZ      P13               ; WAIT UNTIL IT IS
0459  8A C3               MOV     AL,BL             ; RECOVER CHAR
045B  AA                  STOSB                     ; WRITE CHAR
045C  9D                  POPF                      ; INTERRUPTS BACK ON - IF ON
045D  47                  INC     DI                ; BUMP POINTER PAST ATTRIBUTE
045E  E2 E6               LOOP    P11               ; AS MANY TIMES AS REQUESTED
0460  E9 016F R           JMP     VIDEO_RETURN
0463                   WRITE_C_CURRENT ENDP
```

```
              ;-----------------------------------------------------------------
              ; READ DOT  -- WRITE DOT
              ;           THESE ROUTINES WILL WRITE A DOT, OR READ THE DOT AT
              ;           THE INDICATED LOCATION
              ; ENTRY --
              ;            DX = ROW (0-199)    (THE ACTUAL VAL 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 JU1TIFIED)
              ;                 BIT 7 OF AL=1 INDICATES XOR THE VAL INTO LOCATION
              ;            DS = DATA SEGMENT
              ;            ES = REGEN SEGMENT
              ;
              ; EXIT
              ;            AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
              ;-----------------------------------------------------------------
                          ASSUME  CS:ROMCODE,DS:DATA,ES:DATA
0463                   READ_DOT         PROC    NEAR
0463  E8 0497 R           CALL    R3                    ; FIND BYTE POSITION OF DOT
0466  26: 8A 04           MOV     AL,ES:-SI-    ; GET THE BYTE
0469  22 C4               AND     AL,AH                 ; MASK OFF OTHER BITS IN BYTE
046B  D2 E0               SHL     AL,CL                 ; LEFT JUSTIFY THE VALUE
046D  8A CE               MOV     CL,DH                 ; GET NUMBER OF BITS IN RESULT
046F  D2 C0               ROL     AL,CL                 ; RIGHT JUSTIFY THE RESULT
0471  E9 016F R           JMP     VIDEO_RETURN          ; RETURN FROM VIDEO IO
0474                   READ_DOT         ENDP

0474                   WRITE_DOT        PROC    NEAR
0474  50                  PUSH    AX                    ; SAVE DOT VALUE
0475  50                  PUSH    AX                    ; TWICE
0476  E8 0497 R           CALL    R3                    ; FIND BYTE POSITION OF DOT
0479  D2 E8               SHR     AL,CL                 ; SHIFT TO SET UP OUT BITS
047B  22 C4               AND     AL,AH                 ; STRIP OFF THE OTHER BITS
047D  26: 8A 0C           MOV     CL,ES:-SI-    ; GET THE CURRENT BYTE
0480  5B                  POP     BX                    ; RECOVER XOR FLAG
0481  F6 C3 80            TEST    BL,80H                ; IS IT ON
0484  75 0D               JNZ     R2                    ; YES, XOR THE DOT
0486  F6 D4               NOT     AH                    ; SET THE MASK TO REMOVE THE
0488  22 CC               AND     CL,AH                 ; INDICATED BITS
048A  0A C1               OR      AL,CL                 ; OR IN THE NEW VALUE OF BITS
048C                   R1:                              ; FINISH_DOT
048C  26: 88 04           MOV     ES:-SI-,AL    ; RESTORE THE BYTE IN MEMORY
048F  58                  POP     AX
0490  E9 016F R           JMP     VIDEO_RETURN          ; RETURN FROM VIDEO IO
0493                   R2:                              ; XOR_DOT
0493  32 C1               XOR     AL,CL                 ; EXCLUSIVE OR THE DOTS
0495  EB F5               JMP     R1                    ; FINISH UP THE WRITING
0497                   WRITE_DOT        ENDP

              ;-----------------------------------------------------
              ; R3 PROCEDURE
              ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION
              ; OF THE INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
              ; ENTRY --
              ;           DX = ROW VALUE (0-199)
              ;           CX = COLUMN VALUE (0-639)
              ; EXIT --
              ;           SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
              ;           AH = MASK TO STRIP OFF THE BITS OF INTEREST
              ;           CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
              ;           DH = # BITS IN RESULT
              ;-----------------------------------------------------
0497                   R3 PROC    NEAR
0497  53                  PUSH    BX                    ; SAVE BX DURING OPERATION
0498  50                  PUSH    AX                    ; SAVE AL DURING OPERATION

              ;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLY ROW VALUE BY 40
              ;----- ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW

0499  B0 28               MOV     AL,40
049B  52                  PUSH    DX                    ; SAVE ROW VALUE
049C  80 E2 FE            AND     DL,OFEH               ; STRIP OFF ODD/EVEN BIT
049F  F6 E2               MUL     DL                    ; AX HAS ADDRESS OF 1ST BYTE
```

```
                                                          ; OF INDICATED ROW
04A1  5A                      POP     DX                  ; RECOVER IT
04A2  F6 C2 01                TEST    DL,1                ; TEST FOR EVEN/ODD
04A5  74 03                   JZ      R4                  ; JUMP IF EVEN ROW
04A7  05 2000                 ADD     AX,2000H            ; OFFSET TO LOC OF ODD ROWS
04AA                       R4:                            ; EVEN_ROW
04AA  8B F0                   MOV     SI,AX               ; MOVE POINTER TO SI
04AC  58                      POP     AX                  ; RECOVER AL VALUE
04AD  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 FOR HIGH/MED RES)
                ;     CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M)
                ;     BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/C0H FOR H/M)
                ;     BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M)
                ;-------------------------------------------------------------

04AF  BB 02C0                 MOV     BX,2C0H
04B2  B9 0302                 MOV     CX,302H             ; SET PARMS FOR MED RES
04B5  80 3E 0049 R 06         CMP     CRT_MODE,6
04BA  72 06                   JC      R5                  ; HANDLE IF MED ARES
04BC  BB 0180                 MOV     BX,180H
04BF  B9 0703                 MOV     CX,703H             ; SET PARMS FOR HIGH RES

                ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK

04C2                       R5:
04C2  22 EA                   AND     CH,DL               ; ADDR OF PEL IN BYTE TO CH

                ;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN

04C4  D3 EA                   SHR     DX,CL               ; SHIFT BY CORRECT AMOUNT
04C6  03 F2                   ADD     SI,DX               ; INCREMENT THE POINTER
04C8  8A F7                   MOV     DH,BH               ; GET # BITS IN RESULT TO DH

                ;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)

04CA  2A C9                   SUB     CL,CL               ; ZERO INTO STORAGE LOCATION
04CC                       R6:
04CC  D0 C8                   ROR     AL,1                ; LEFT JUSTIFY THE VALUE
                                                          ; IN AL (FOR WRITE)
04CE  02 CD                   ADD     CL,CH               ; ADD IN THE BIT OFFSET VALUE
04D0  FE CF                   DEC     BH                  ; LOOP CONTROL
04D2  75 F8                   JNZ     R6                  ; ON EXIT, CL HAS SHIFT COUNT
                                                          ; TO RESTORE BITS
04D4  8A E3                   MOV     AH,BL               ; GET MASK TO AH
04D6  D2 EC                   SHR     AH,CL               ; MOVE MASK TO CORRECT LOCAT
04D8  5B                      POP     BX                  ; RECOVER REG
04D9  C3                      RET                         ; RETURN WITH SET UP
04DA                       R3 ENDP

                ;-------------------------------------------------------------
                ; GRAPHICS UP
                ;           THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
                ; ENTRY
                ;           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
                ; EXIT
                ;           NOTHING, THE SCREEN IS SCROLLED
                ;-------------------------------------------------------------
04DA                       GRAPHICS_UP     PROC    NEAR
04DA  8A D8                   MOV     BL,AL               ; SAVE LINE COUNT IN BL
04DC  8B C1                   MOV     AX,CX               ; GET UPPER LEFT POS IN AX REG

                ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
```

```
                  ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

04DE  E8 073C R             CALL   GRAPH_POSN
04E1  8B F8                 MOV    DI,AX              ; SAVE RESULT AS DEST ADDRESS

                  ;----- DETERMINE SIZE OF WINDOW

04E3  2B D1                 SUB    DX,CX
04E5  81 C2 0101            ADD    DX,101H            ; ADJUST VALUES
04E9  D0 E6                 SAL    DH,1               ; MULTIPLY # ROWS BY 4
                                                      ; SINCE 8 VERT DOTS/CHAR
04EB  D0 E6                 SAL    DH,1               ; AND EVEN/ODD ROWS

                  ;----- DETERMINE CRT MODE

04ED  80 3E 0049 R 06       CMP    CRT_MODE,6         ; TEST FOR MEDIUM RES
04F2  73 04                 JNC    R7                 ; FIND_SOURCE

                  ;----- MEDIUM RES UP

04F4  D0 E2                 SAL    DL,1               ; # COL * 2 SINCE 2 BYTES/CHAR
04F6  D1 E7                 SAL    DI,1               ; OFFSET *2 SINCE 2 BYTES/CHAR

                  ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER

04F8                      R7:                         ; FIND_SOURCE
04F8  06                    PUSH   ES                 ; GET SEG BOTH POINTING REGEN
04F9  1F                    POP    DS
04FA  2A ED                 SUB    CH,CH              ; ZERO TO HIGH OF COUNT REG
04FC  D0 E3                 SAL    BL,1               ; MULTIPLY NO. OF LINES BY 4
04FE  D0 E3                 SAL    BL,1
0500  74 2D                 JZ     R11                ; IF ZERO, BLANK ENTIRE FIELD
0502  8A C3                 MOV    AL,BL              ; GET NUMBER OF LINES IN AL
0504  B4 50                 MOV    AH,80              ; 80 BYTES/ROW
0506  F6 E4                 MUL    AH                 ; DETERMINE OFFSET TO SOURCE
0508  8B F7                 MOV    SI,DI              ; SET UP SOURCE
050A  03 F0                 ADD    SI,AX              ; ADD IN OFFSET TO IT
050C  8A E6                 MOV    AH,DH              ; NUMBER OF ROWS IN FIELD
050E  2A E3                 SUB    AH,BL              ; DETERMINE NUMBER TO MOVE

                  ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN & ODD FIELDS

0510                      R8:                         ; ROW_LOOP
0510  E8 0593 R             CALL   R17                ; MOVE ONE ROW
0513  81 EE 1FB0            SUB    SI,2000H-80        ; MOVE TO NEXT ROW
0517  81 EF 1FB0            SUB    DI,2000H-80
051B  FE CC                 DEC    AH                 ; NUMBER OF ROWS TO MOVE
051D  75 F1                 JNZ    R8                 ; CONTINUE TILL ALL MOVED

                  ;----- FILL IN THE VACATED LINE(S)

051F                      R9:                         ; CLEAR_ENTRY
051F  8A C7                 MOV    AL,BH              ; ATTRIBUTE TO FILL WITH
0521                      R10:
0521  E8 05AC R             CALL   R18                ; CLEAR THAT ROW
0524  81 EF 1FB0            SUB    DI,2000H-80        ; POINT TO NEXT LINE
0528  FE CB                 DEC    BL                 ; NUMBER OF LINES TO FILL
052A  75 F5                 JNZ    R10                ; CLEAR_LOOP
052C  E9 016F R             JMP    VIDEO_RETURN       ; EVERYTHING DONE
052F                      R11:                        ; BLANK_FIELD
052F  8A DE                 MOV    BL,DH              ; SET BLANK COUNT TO
                                                      ; EVERYTHING IN FIELD
0531  EB EC                 JMP    R9                 ; CLEAR THE FIELD
0533                      GRAPHICS_UP     ENDP
```

```
                ;---------------------------------------------------------------
                ; GRAPHICS DOWN
                ;          THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                ; ENTRY
                ;          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
                ; EXIT
                ;          NOTHING, THE SCREEN IS SCROLLED
                ;---------------------------------------------------------------
0533                    GRAPHICS_DOWN   PROC    NEAR
0533  FD                   STD                          ; SET DIRECTION
0534  8A D8                MOV     BL,AL                ; SAVE LINE COUNT IN BL
0536  8B C2                MOV     AX,DX                ; GET LOWER RIGHT POS INTO AX

                ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
                ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

0538  E8 073C R            CALL    GRAPH_POSN
053B  8B F8                MOV     DI,AX                ; SAVE RESULT AS DEST ADDR

                ;----- DETERMINE SIZE OF WINDOW

053D  2B D1                SUB     DX,CX
053F  81 C2 0101           ADD     DX,101H              ; ADJUST VALUES
0543  D0 E6                SAL     DH,1                 ; MULTIPLY # ROWS BY 4
                                                        ; SINCE 8 VERT DOTS/CHAR
0545  D0 E6                SAL     DH,1                 ; AND EVEN/ODD ROWS

                ;----- DETERMINE CRT MODE

0547  80 3E 0049 R 06      CMP     CRT_MODE,6           ; TEST FOR MEDIUM RES
054C  73 05                JNC     R12                  ; FIND_SOURCE_DOWN

                ;----- MEDIUM RES DOWN

054E  D0 E2                SAL     DL,1                 ; # COLUMNS * 2, SINCE
                                                        ; 2 BYTES/CHAR (OFFSET OK)
0550  D1 E7                SAL     DI,1                 ; OFFSET *2 SINCE 2 BYTES/CHAR
0552  47                   INC     DI                   ; POINT TO LAST BYTE

                ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER

0553                    R12:                            ; FIND_SOURCE_DOWN
0553  06                   PUSH    ES                   ; BOTH SEGMENTS TO REGEN
0554  1F                   POP     DS
0555  2A ED                SUB     CH,CH                ; ZERO TO HIGH OF COUNT REG
0557  81 C7 00F0           ADD     DI,240               ; POINT TO LAST ROW OF PIXELS
055B  D0 E3                SAL     BL,1                 ; MULTIPLY NO. OF LINES BY 4
055D  D0 E3                SAL     BL,1
055F  74 2E                JZ      R16                  ; IF ZERO, BLANK ENTIRE FIELD
0561  8A C3                MOV     AL,BL                ; GET NUMBER OF LINES IN AL
0563  B4 50                MOV     AH,80                ; 80 BYTES/ROW
0565  F6 E4                MUL     AH                   ; DETERMINE OFFSET TO SOURCE
0567  8B F7                MOV     SI,DI                ; SET UP SOURCE
0569  2B F0                SUB     SI,AX                ; SUBTRACT THE OFFSET
056B  8A E6                MOV     AH,DH                ; NUMBER OF ROWS IN FIELD
056D  2A E3                SUB     AH,BL                ; DETERMINE NUMBER TO MOVE

                ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN & ODD FIELDS

056F                    R13:                            ; ROW_LOOP_DOWN
056F  E8 0593 R            CALL    R17                  ; MOVE ONE ROW
0572  81 EE 2050           SUB     SI,2000H+80          ; MOVE TO NEXT ROW
0576  81 EF 2050           SUB     DI,2000H+80
057A  FE CC                DEC     AH                   ; NUMBER OF ROWS TO MOVE
057C  75 F1                JNZ     R13                  ; CONTINUE TILL ALL MOVED

                ;----- FILL IN THE VACATED LINE(S)
```

```
057E                  R14:                             ; CLEAR_ENTRY_DOWN
057E  8A C7               MOV     AL,BH                ; ATTRIBUTE TO FILL WITH
0580                  R15:                             ; CLEAR_LOOP_DOWN
0580  E8 05AC R           CALL    R18                  ; CLEAR A ROW
0583  81 EF 2050          SUB     DI,2000H+80          ; POINT TO NEXT LINE
0587  FE CB               DEC     BL                   ; NUMBER OF LINES TO FILL
0589  75 F5               JNZ     R15                  ; CLEAR_LOOP_DOWN
058B  FC                  CLD                          ; RESET THE DIRECTION FLAG
058C  E9 016F R           JMP     VIDEO_RETURN         ; EVERYTHING DONE
058F                  R16:                             ; BLANK_FIELD_DOWN
058F  8A DE               MOV     BL,DH                ; SET BLANK COUNT TO EVERYTHNG
                                                       ; IN FIELD
0591  EB EB               JMP     R14                  ; CLEAR THE FIELD
0593                  GRAPHICS_DOWN   ENDP

                ;
                ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
                ;

0593                  R17     PROC    NEAR
0593  8A CA               MOV     CL,DL                ; NUMBER OF BYTES IN THE ROW
0595  56                  PUSH    SI
0596  57                  PUSH    DI                   ; SAVE POINTERS
0597  F3/ A4               REP     MOVSB                ; MOVE THE EVEN FIELD
0599  5F                  POP     DI
059A  5E                  POP     SI
059B  81 C6 2000          ADD     SI,2000H
059F  81 C7 2000          ADD     DI,2000H             ; POINT TO THE ODD FIELD
05A3  56                  PUSH    SI
05A4  57                  PUSH    DI                   ; SAVE THE POINTERS
05A5  8A CA               MOV     CL,DL                ; COUNT BACK
05A7  F3/ A4               REP     MOVSB                ; MOVE THE ODD FIELD
05A9  5F                  POP     DI
05AA  5E                  POP     SI                   ; POINTERS BACK
05AB  C3                  RET                          ; RETURN TO CALLER
05AC                  R17     ENDP

                ;----- CLEAR A SINGLE ROW

05AC                  R18     PROC    NEAR
05AC  8A CA               MOV     CL,DL                ; NUMBER OF BYTES IN FIELD
05AE  57                  PUSH    DI                   ; SAVE POINTER
05AF  F3/ AA               REP     STOSB                ; STORE THE NEW VALUE
05B1  5F                  POP     DI                   ; POINTER BACK
05B2  81 C7 2000          ADD     DI,2000H             ; POINT TO ODD FIELD
05B6  57                  PUSH    DI
05B7  8A CA               MOV     CL,DL
05B9  F3/ AA               REP     STOSB                ; FILL THE ODD FILELD
05BB  5F                  POP     DI
05BC  C3                  RET                          ; RETURN TO CALLER
05BD                  R18     ENDP

                ;--------------------------------------------------------------
                ; GRAPHICS WRITE
                ;           THIS ROUTINE WRITES THE ASCII CHARACTER TO THE
                ;           CURRENT POSITION ON THE SCREEN.
                ; ENTRY
                ;           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
                ;                BUFFER (0 IS USED FOR THE BACKGROUND COLOR)
                ;           CX = NUMBER OF CHARS TO WRITE
                ;           DS = DATA SEGMENT
                ;           ES = REGEN SEGMENT
                ; EXIT
                ;           NOTHING IS RETURNED
                ;
                ;
                ; FOR THIS ROUTINE, THE IMAGES USED TO FORM CHARS ARE CONTAINED
                ; IN ROM.
                ;--------------------------------------------------------------
```

```
                        ASSUME  CS:ROMCODE,DS:DATA,ES:DATA
05BD                  GRAPHICS_WRITE  PROC    NEAR
05BD  B4 00              MOV     AH,0                 ; ZERO TO HIGH OF CODE POINT
05BF  50                 PUSH    AX                   ; SAVE CODE POINT VALUE

              ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS

05C0  E8 0739 R          CALL    S26                  ; FIND LOC IN REGEN BUFFER
05C3  8B F8              MOV     DI,AX                ; REGEN POINTER IN DI

              ;----- DETERMINE REGION TO GET CODE POINTS FROM

05C5  58                 POP     AX                   ; RECOVER CODE POINT
05C6  1E                 PUSH    DS
05C7  2B F6              SUB     SI,SI
05C9  8E DE              MOV     DS,SI
                         ASSUME  DS:ABSO
05CB  3C 80              CMP     AL,80H               ; IF IT IS IN SECOND HALF
05CD  73 07              JAE     S0                   ; JUMP
05CF  C5 36 0110 R       LDS     SI,CSET_PTR          ; IT'S IN FIRST HALF
05D3  EB 07 90           JMP     S1
05D6  2C 80           S0:     SUB     AL,80H          ; ZERO ORIGIN FOR SECOND HALF
05D8  C5 36 007C R       LDS     SI,EXT_PTR           ; GET POINTER TO 2ND HALF
05DC  8C DA           S1:     MOV     DX,DS           ; SAVE THE SEGMENT OF THE TBL
                         ASSUME  DS:DATA
05DE  1F                 POP     DS                   ; RECOVER DATA SEGMENT
05DF  52                 PUSH    DX                   ; SAVE TABLE SEGMENT ON STACK

              ;----- DETERMINE GRAPHICS MODE IN OPERATION

05E0                  S2:                             ; DETERMINE_MODE
05E0  D1 E0              SAL     AX,1                 ; MULTIPLY CODE POINT
05E2  D1 E0              SAL     AX,1                 ; VALUE BY 8
05E4  D1 E0              SAL     AX,1
05E6  03 F0              ADD     SI,AX                ; SI HAS OFFSET OF CODES
05E8  80 3E 0049 R 06    CMP     CRT_MODE,6
05ED  1F                 POP     DS                   ; RECOVER TBL POINTER SEGMENT
05EE  72 2C              JC      S7                   ; TEST FOR MEDIUM RESOLUTION

              ;----- HIGH RESOLUTION MODE

05F0                  S3:                             ; HIGH_CHAR
05F0  57                 PUSH    DI                   ; SAVE REGEN POINTER
05F1  56                 PUSH    SI                   ; SAVE CODE POINTER
05F2  B6 04              MOV     DH,4                 ; NUMBER OF TIMES THROUGH LOOP
05F4                  S4:
05F4  AC                 LODSB                        ; GET BYTE FROM CODE POINTS
05F5  F6 C3 80           TEST    BL,80H               ; SHOULD WE USE THE FUNCTION
05F8  75 16              JNZ     S6                   ; TO PUT CHAR IN
05FA  AA                 STOSB                        ; STORE IN REGEN BUFFER
05FB  AC                 LODSB
05FC                  S5:
05FC  26: 88 85 1FFF     MOV     ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
0601  83 C7 4F           ADD     DI,79                ; MOVE TO NEXT ROW IN REGEN
0604  FE CE              DEC     DH                   ; DONE WITH LOOP
0606  75 EC              JNZ     S4
0608  5E                 POP     SI
0609  5F                 POP     DI                   ; RECOVER REGEN POINTER
060A  47                 INC     DI                   ; POINT TO NEXT CHAR POSITION
060B  E2 E3              LOOP    S3                   ; MORE CHARS TO WRITE
060D  E9 016F R          JMP     VIDEO_RETURN
0610                  S6:
0610  26: 32 05          XOR     AL,ES:[DI]    ; EXCLUSIVE OR WITH CURRENT
0613  AA                 STOSB                        ; STORE THE CODE POINT
0614  AC                 LODSB                        ; AGAIN FOR ODD FIELD
0615  26: 32 85 1FFF     XOR     AL,ES:[DI+2000H-1]
061A  EB E0              JMP     S5                   ; BACK TO MAINSTREAM

              ;----- MEDIUM RESOLUTION WRITE

061C                  S7:                             ; MED_RES_WRITE
061C  8A D3              MOV     DL,BL                ; SAVE HIGH COLOR BIT
061E  D1 E7              SAL     DI,1                 ; OFFSET*2 SINCE 2 BYTES/CHAR
0620  E8 06F7 R          CALL    S19                  ; EXPAND BL TO FULL WORD
```

```
0623                      S8:                           ; MED_CHAR
0623  57                     PUSH    DI                 ; SAVE REGEN POINTER
0624  56                     PUSH    SI                 ; SAVE THE CODE POINTER
0625  B6 04                  MOV     DH,4               ; NUMBER OF LOOPS
0627                      S9:
0627  AC                     LODSB                      ; GET CODE POINT
0628  E8 070C R              CALL    S21                ; DOUBLE UP ALL THE BITS
062B  23 C3                  AND     AX,BX              ; CONVERT THEM TO FOREGROUND
                                                        ; COLOR ( 0 BACK )
062D  F6 C2 80               TEST    DL,80H             ; IS THIS XOR FUNCTION
0630  74 07                  JZ      S10                ; NO, STORE IT IN AS IT IS
0632  26: 32 25              XOR     AH,ES:[DI]         ; DO FUNCTION WITH HALF
0635  26: 32 45 01           XOR     AL,ES:[DI+1]  ; AND WITH OTHER HALF
0639                      S10:
0639  26: 88 25              MOV     ES:-DI-,AH    ; STORE FIRST BYTE
063C  26: 88 45 01           MOV     ES:[DI+1],AL  ; STORE SECOND BYTE
0640  AC                     LODSB                      ; GET CODE POINT
0641  E8 070C R              CALL    S21
0644  23 C3                  AND     AX,BX              ; CONVERT TO COLOR
0646  F6 C2 80               TEST    DL,80H             ; AGAIN, IS THIS XOR FUNCTION
0649  74 0A                  JZ      S11                ; NO, JUST STORE THE VALUES
064B  26: 32 A5 2000         XOR     AH,ES:-DI+2000H-  ; FUNCTION WITH FIRST HALF
0650  26: 32 85 2001         XOR     AL,ES:-DI+2001H-  ; AND WITH SECOND HALF
0655                      S11:
0655  26: 88 A5 2000         MOV     ES:[DI+2000H],AH
065A  26: 88 85 2001         MOV     ES:-DI+2000H+1-,AL  ; PUT IN 2ND BUFR PART
065F  83 C7 50               ADD     DI,80              ; POINT TO NEXT LOCATION
0662  FE CE                  DEC     DH
0664  75 C1                  JNZ     S9                 ; KEEP GOING
0666  5E                     POP     SI                 ; RECOVER CODE PONTER
0667  5F                     POP     DI                 ; RECOVER REGEN POINTER
0668  47                     INC     DI                 ; POINT TO NEXT CHAR POSITION
0669  47                     INC     DI
066A  E2 B7                  LOOP    S8                 ; MORE TO WRITE
066C  E9 016F R              JMP     VIDEO_RETURN
066F                      GRAPHICS_WRITE  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
                  ; ENTRY
                  ;          ( 0 IS ASSUMED AS THE BACKGROUND COLOR )
                  ;          DS = DATA SEGMENT
                  ;          ES = REGEN SEGMENT
                  ; EXIT
                  ;          AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF
                  ;          NONE FOUND)
                  ;
                  ; FOR THIS ROUTINE, THE IMAGES USED TO FORM CHARS ARE CONTAINED
                  ; IN ROM.
                  ;-----------------------------------------------------------------
066F                      GRAPHICS_READ   PROC    NEAR
066F  E8 0739 R              CALL    S26                ; CONVERTED TO OFFSET IN REGEN
0672  8B F0                  MOV     SI,AX              ; SAVE IN SI
0674  83 EC 08               SUB     SP,8               ; ALLOCATE SPACE TO SAVE THE
                                                        ; READ CODE POINT
0677  8B EC                  MOV     BP,SP              ; POINTER TO SAVE AREA

                  ;----- DETERMINE GRAPHICS MODES

0679  80 3E 0049 R 06        CMP     CRT_MODE,6
067E  06                     PUSH    ES
067F  1F                     POP     DS                 ; POINT TO REGEN SEGMENT
0680  72 1A                  JC      S13                ; MEDIUM RESOLUTION

                  ;----- HIGH RESOLUTION READ

                  ;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT

0682  B6 04                  MOV     DH,4               ; NUMBER OF PASSES
0684                      S12:
0684  8A 04                  MOV     AL,[SI]            ; GET FIRST BYTE
```

```
06F4  E9 016F R             JMP     VIDEO_RETURN        ; ALL DONE
06F7                      GRAPHICS_READ   ENDP

                 ;----------------------------------------------------
                 ; EXPAND_MED_COLOR - S19
                 ;          THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO
                 ;          FILL THE ENTIRE BX REGISTER
                 ; ENTRY
                 ;          BL = COLOR TO BE USED ( LOW 2 BITS )
                 ; EXIT
                 ;          BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE
                 ;          2 COLOR BITS )
                 ;----------------------------------------------------
06F7                      S19     PROC    NEAR
06F7  80 E3 03              AND     BL,3                ; ISOLATE THE COLOR BITS
06FA  8A C3                 MOV     AL,BL               ; COPY TO AL
06FC  51                    PUSH    CX                  ; SAVE REGISTER
06FD  B9 0003               MOV     CX,3                ; NUMBER OF TIMES TO DO THIS
0700                      S20:
0700  D0 E0                 SAL     AL,1
0702  D0 E0                 SAL     AL,1                ; LEFT SHIFT BY 2
0704  0A D8                 OR      BL,AL               ; ANOTHER COLOR VERSION INTO BL
0706  E2 F8                 LOOP    S20                 ; FILL ALL OF BL
0708  8A FB                 MOV     BH,BL               ; FILL UPPER PORTION
070A  59                    POP     CX                  ; REGISTER BACK
070B  C3                    RET                         ; ALL DONE
070C                      S19     ENDP
                 ;----------------------------------------------------
                 ; EXPAND_BYTE  - S21
                 ;          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
                 ;----------------------------------------------------
070C                      S21     PROC    NEAR
070C  52                    PUSH    DX                  ; SAVE REGISTER
070D  BA 8000               MOV     DX,8000H            ; BIT TO TERMINATE LOOP
0710                      DBLBIT4:
0710  D0 E8                 SHR     AL,1                ; LOW ORDER BIT TO CARRY FLAG
0712  D1 DA                 RCR     DX,1                ; FROM CARRY TO HIGH OF RESULT
0714  D1 FA                 SAR     DX,1                ; DBL HIGH ORDER BIT -SIGN EXT
0716  73 F8                 JNB     DBLBIT4             ; TERM WHEN BIT SHIFTED OUT
0718  8B C2                 MOV     AX,DX               ; PUT RESULT INTO AX
071A  5A                    POP     DX                  ; RESTORE REGISTER
071B  C3                    RET                         ; RETURN
071C                      S21     ENDP
                 ;----------------------------------------------------
                 ; MED_READ_BYTE - S23
                 ;          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
                 ; ENTRY
                 ;          SI,DS = POINTER TO REGEN AREA OF INTEREST
                 ;          BX = EXPANDED FOREGROUND COLOR
                 ;          BP = POINTER TO SAVE AREA
                 ; EXIT
                 ;          BP IS INCREMENT AFTER SAVE
                 ;----------------------------------------------------
071C                      S23     PROC    NEAR
071C  8A 24                 MOV     AH,[SI]             ; GET FIRST BYTE
071E  8A 44 01              MOV     AL,[SI+1]           ; GET SECOND BYTE
0721  B9 C000               MOV     CX,0C000H           ; 2 BIT MASK TO TEST ENTRIES
0724  B2 00                 MOV     DL,0                ; RESULT REGISTER
0726                      S24:
0726  85 C1                 TEST    AX,CX               ; IS THIS SECTION BACKGROUND?
0728  F8                    CLC                         ; CLEAR CARRY
0729  74 01                 JZ      S25                 ; IF ZERO, IT IS BACKGROUND
072B  F9                    STC                         ; WASN'T, SO SET CARRY
072C  D0 D2               S25:    RCL     DL,1          ; MOVE THAT BIT INTO RESULT
072E  D1 E9                 SHR     CX,1
0730  D1 E9                 SHR     CX,1                ; MOVE MASK RIGHT BY 2 BITS
0732  73 F2                 JNC     S24                 ; AGAIN IF MASK NOT FALL OUT
0734  88 56 00              MOV     [BP],DL             ; STORE RESULT IN SAVE AREA
```

```
0686  88 46 00              MOV    [BP],AL              ; SAVE IN STORAGE AREA
0689  45                    INC    BP                   ; NEXT LOCATION
068A  8A 84 2000            MOV    AL,[SI+2000H]        ; GET LOWER REGION BYTE
068E  88 46 00              MOV    [BP],AL              ; ADJUST AND STORE
0691  45                    INC    BP
0692  83 C6 50              ADD    SI,80                ; POINTER INTO REGEN
0695  FE CE                 DEC    DH                   ; LOOP CONTROL
0697  75 EB                 JNZ    S12                  ; DO IT SOME MORE
0699  EB 17 90              JMP    S15                  ; GO MATCH SAVED CODE POINTS

               ;----- MEDIUM RESOLUTION READ

069C                     S13:                           ; MED_RES_READ
069C  D1 E6                 SAL    SI,1                 ; OFFSET*2 SINCE 2 BYTES/CHAR
069E  B6 04                 MOV    DH,4                 ; NUMBER OF PASSES
06A0                     S14:
06A0  E8 071C R             CALL   S23                  ; GET PAIR BYTES FROM REGEN
                                                        ; INTO SINGLE SAVE
06A3  81 C6 2000            ADD    SI,2000H             ; GO TO LOWER REGION
06A7  E8 071C R             CALL   S23                  ; GET THIS PAIR INTO SAVE
06AA  81 EE 1FB0            SUB    SI,2000H-80          ; ADJUST PTR BACK INTO UPPER
06AE  FE CE                 DEC    DH
06B0  75 EE                 JNZ    S14                  ; KEEP GOING UNTIL ALL 8 DONE

               ;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT

06B2                     S15:                           ; FIND_CHAR
06B2  2B FF                 SUB    DI,DI
06B4  8E C7                 MOV    ES,DI
                            ASSUME ES:ABSO
06B6  26: C4 3E 0110 R      LES    DI,CSET_PTR          ; ESTABLISH ADDRESSING
                            ASSUME ES:NOTHING
06BB  83 ED 08              SUB    BP,8                 ; ADJUST POINTER TO BEGINNING
                                                        ; OF SAVE AREA
06BE  8B F5                 MOV    SI,BP
06C0  FC                    CLD                         ; ENSURE DIRECTION
06C1  B0 00                 MOV    AL,0                 ; CURRENT CD PT BEING MATCHED
06C3                     S16:
06C3  16                    PUSH   SS                   ; ESTAB ADDRESSING TO STACK
06C4  1F                    POP    DS                   ; FOR THE STRING COMPARE
06C5  BA 0080               MOV    DX,128               ; NUMBER TO TEST AGAINST
06C8                     S17:
06C8  56                    PUSH   SI                   ; SAVE SAVE AREA POINTER
06C9  57                    PUSH   DI                   ; SAVE CODE POINTER
06CA  B9 0008               MOV    CX,8                 ; NUMBER OF BYTES TO MATCH
06CD  F3/ A6                REPE   CMPSB                ; COMPARE THE 8 BYTES
06CF  5F                    POP    DI                   ; RECOVER THE POINTERS
06D0  5E                    POP    SI
06D1  74 1E                 JZ     S18                  ; IF ZERO FLAG SET, MATCH
06D3  FE C0                 INC    AL                   ; NO MATCH, MOVE ON TO NEXT
06D5  83 C7 08              ADD    DI,8                 ; NEXT CODE POINT
06D8  4A                    DEC    DX                   ; LOOP CONTROL
06D9  75 ED                 JNZ    S17                  ; DO ALL OF THEM

               ;----- CHAR NOT MATCHED, MIGHT BE IN USER SUPPLIED SECOND HALF

06DB  3C 00                 CMP    AL,0                 ; AL <> 0 IF 1ST HALF SCANNED
06DD  74 12                 JE     S18                  ; IF = 0, IF ALL SCANNED
06DF  2B C0                 SUB    AX,AX
06E1  8E D8                 MOV    DS,AX                ; ESTAB ADDRESSING TO VECTOR
                            ASSUME DS:ABSO
06E3  C4 3E 007C R          LES    DI,EXT_PTR           ; GET POINTER
06E7  8C C0                 MOV    AX,ES                ; SEE IF POINTER REALLY EXISTS
06E9  0B C7                 OR     AX,DI                ; IF ALL 0, THEN DOESN'T EXIST
06EB  74 04                 JZ     S18                  ; NO SENSE LOOKING
06ED  B0 80                 MOV    AL,128               ; ORIGIN FOR SECOND HALF
06EF  EB D2                 JMP    S16                  ; GO BACK AND TRY FOR IT
                            ASSUME DS:DATA

               ;----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )

06F1                     S18:
06F1  83 C4 08              ADD    SP,8                 ; READJUST, THROW AWAY SAVE
```

```
0737  45                   INC     BP                  ; ADJUST POINTER
0738  C3                   RET                         ; ALL DONE
0739                    S23     ENDP
                 ;-------------------------------------------
                 ; V4_POSITION - S26
                 ;          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.
                 ; ENTRY
                 ;          NO REGISTERS, MEMORY LOCATION
                 ;          CURSOR_POSN IS USED
                 ; EXIT
                 ;          AX CONTAINS OFFSET INTO REGEN BUFFER
                 ;-------------------------------------------
0739                    S26     PROC    NEAR
0739  A1 0050 R            MOV     AX,CURSOR_POSN      ; GET CURRENT CURSOR
073C                    GRAPH_POSN      LABEL   NEAR
073C  53                   PUSH    BX                  ; SAVE REGISTER
073D  8B D8                MOV     BX,AX               ; SAVE COPY OF CURRENT CURSOR
073F  8A C4                MOV     AL,AH               ; GET ROWS TO AL
0741  F6 26 004A R         MUL     BYTE PTR CRT_COLS   ; MULTIPLY BY BYTES/COLUMN
0745  D1 E0                SHL     AX,1                ; MTPLY * 4 SINCE 4 ROWS/BYTE
0747  D1 E0                SHL     AX,1
0749  2A FF                SUB     BH,BH               ; ISOLATE COLUMN VALUE
074B  03 C3                ADD     AX,BX               ; DETERMINE OFFSET
074D  5B                   POP     BX                  ; RECOVER POINTER
074E  C3                   RET                         ; ALL DONE
074F                    S26     ENDP

                 ;------------------------------------------------------------------
                 ; WRITE_TTY
                 ;          THIS INTERFACE PROVIDES A TELETYPE LIKE INTERF TO VIDEO
                 ;          CARD. THE INPUT CHARACTER IS WRITTEN TO CURRENT CURSOR
                 ;          POSITION, AND THE CURSOR IS MOVED TO NEXT POSITION. IF
                 ;          CURSOR LEAVES THE LAST COLUMN OF THE FIELD, COLUMN IS SET
                 ;          TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF ROW VALUE
                 ;          LEAVES THE FIELD, CURSOR IS PLACED ON LAST ROW, FIRST
                 ;          COLUMN, AND ENTIRE SCREEN IS SCROLLED UP ONE LINE. WHEN
                 ;          THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING NEW
                 ;          BLANKED LINE IS READ FROM THE CURSOR POSITION ON PREVIOUS
                 ;          LINE BEFORE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
                 ;          THE 0 COLOR IS USED. FOR BKSP, THE CURSOR COLUMN POSITION
                 ;          IS DECREMENTED BY 1. IF THE CURSOR COLUMN IS 0 AND A BKSP
                 ;          OCCURS CURSOR COLUMN REMAINS 0. IT DOES NOT BACKSPACE TO
                 ;          THE PREVIOUS LINE.
                 ; ENTRY
                 ;          (AL) = CHARACTER TO BE WRITTEN
                 ;           NOTE:  BACK SPACE, CR, 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
                 ;          (DS) = DATA SEGMENT
                 ; EXIT
                 ;          ALL REGISTERS SAVED
                 ;-----------------------------------------------------------------
                           ASSUME  CS:ROMCODE,DS:DATA
074F                    WRITE_TTY       PROC    NEAR
074F  50                   PUSH    AX                  ; SAVE REGISTERS
0750  50                   PUSH    AX                  ; SAVE CHAR TO WRITE
0751  B4 03                MOV     AH,GET_CURSOR_INFO  ; READ CURRENT CURSOR POSITION
0753  8A 3E 0062 R         MOV     BH,ACTIVE_PAGE      ; SET CURRENT ACTIVE PAGE
0757  CD 10                INT     VIDEO_FN            ; INT 10H
0759  58                   POP     AX                  ; RECOVER CHAR

                 ;----- DX NOW HAS THE CURRENT CURSOR POSITION

075A  3C 08                CMP     AL,8                ; IS IT A BACKSPACE
075C  74 52                JE      U8                  ; BACK_SPACE
075E  3C 0D                CMP     AL,0DH              ; IS IT CARRIAGE RETURN
0760  74 57                JE      U9                  ; CAR_RET
0762  3C 0A                CMP     AL,0AH              ; IS IT A LINE FEED
```

```
0764  74 57                 JE      U10                 ; LINE_FEED
0766  3C 07                 CMP     AL,07H              ; IS IT A BELL
0768  74 5A                 JE      U11                 ; BELL

                ;----- WRITE THE CHAR TO THE SCREEN

076A  B4 0A                 MOV     AH,WRITE_CHAR_ONLY  ; WRITE CHAR ONLY TO ACTIVE PG
076C  B9 0001               MOV     CX,1                ; ONLY ONE CHARACTER
076F  CD 10                 INT     VIDEO_FN            ; INT 10H

                ;----- POSITION THE CURSOR FOR NEXT CHAR

0771  FE C2                 INC     DL
0773  3A 16 004A R          CMP     DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
0777  75 33                 JNZ     U7                  ; SET_CURSOR
0779  B2 00                 MOV     DL,0                ; COLUMN FOR CURSOR
077B  80 FE 18              CMP     DH,24
077E  75 2A                 JNZ     U6                  ; SET_CURSOR_INC

                ;----- SCROLL REQUIRED

0780                       U1:
0780  B4 02                 MOV     AH,SET_CURSOR_POS   ; SET THE CURSOR
0782  CD 10                 INT     VIDEO_FN

                ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL

0784  A0 0049 R             MOV     AL,CRT_MODE         ; GET THE CURRENT MODE
0787  3C 04                 CMP     AL,4
0789  72 06                 JC      U2                  ; READ-CURSOR
078B  3C 07                 CMP     AL,7
078D  B7 00                 MOV     BH,0                ; FILL WITH BACKGROUND
078F  75 06                 JNE     U3                  ; SCROLL-UP
0791                       U2:                          ; READ-CURSOR
0791  B4 08                 MOV     AH,READ_ATT_CHAR    ; READ CHAR/ATTR AT CURSOR
0793  CD 10                 INT     VIDEO_FN            ; INT 10H
0795  8A FC                 MOV     BH,AH               ; STORE IN BH
0797                       U3:                          ; SCROLL-UP
0797  B8 0601               MOV     AX,SCROLL_WINDOW_UP*256+01H ; SCROLL UP ONE LINE
079A  2B C9                 SUB     CX,CX               ; UPPER LEFT CORNER
079C  B6 18                 MOV     DH,24               ; LOWER RIGHT ROW
079E  8A 16 004A R          MOV     DL,BYTE PTR CRT_COLS ; LOWER RIGHT COLUMN
07A2  FE CA                 DEC     DL
07A4                       U4:                          ; VIDEO-CALL-RETURN
07A4  CD 10                 INT     VIDEO_FN            ; PROCESS VIDEO FUNCTION
07A6                       U5:                          ; TTY-RETURN
07A6  58                    POP     AX                  ; RESTORE THE CHARACTER
07A7  E9 016F R             JMP     VIDEO_RETURN        ; RETURN TO CALLER
07AA                       U6:                          ; SET-CURSOR-INC
07AA  FE C6                 INC     DH                  ; NEXT ROW
07AC                       U7:                          ; SET-CURSOR
07AC  B4 02                 MOV     AH,SET_CURSOR_POS   ; SET CURSOR POSITION
07AE  EB F4                 JMP     U4                  ; ESTABLISH THE NEW CURSOR

                ;----- BACK SPACE FOUND
                ; BACK SPACE DOES NOT BACK SPACE TO PREVIOUS LINE IF AT BEG OF LINE

07B0                       U8:
07B0  80 FA 00              CMP     DL,0                ; ALREADY AT BEGINNING OF LINE
07B3  74 F7                 JE      U7                  ; SET_CURSOR
07B5  FE CA                 DEC     DL                  ; NO -- JUST MOVE IT BACK
07B7  EB F3                 JMP     U7                  ; SET_CURSOR

                ;----- CARRIAGE RETURN FOUND

07B9                       U9:
07B9  B2 00                 MOV     DL,0                ; MOVE TO FIRST COLUMN
07BB  EB EF                 JMP     U7                  ; SET_CURSOR

                ;----- LINE FEED FOUND

07BD                       U10:
07BD  80 FE 18              CMP     DH,24               ; BOTTOM OF SCREEN
```

```
07C0  75 E8                JNE     U6                ; YES, SCROLL THE SCREEN
07C2  EB BC                JMP     U1                ; NO, JUST SET THE CURSOR

               ;----- BELL FOUND

07C4                    U11:
07C4  B3 02                MOV     BL,2              ; SET UP COUNT FOR BEEP
07C6  E8 0000 E            CALL    BEEP              ; SOUND THE POD BELL
07C9  EB DB                JMP     U5                ; TTY_RETURN
07CB                    WRITE_TTY       ENDP

               ;----------------------------------------------------------------
               ; LIGHT PEN
               ;            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.
               ; ON ENTRY
               ;            (DS) = DATA SEGMENT
               ; 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:ROMCODE,DS:DATA
               ;----- SUBTRACT_TABLE
07CB                    V1 LABEL   BYTE
07CB  03 03 05 05 03 03    DB      3,3,5,5,3,3,3,4
      03 04
07D3                    READ_LPEN       PROC    NEAR

               ;----- WAIT FOR LIGHT PEN TO BE DEPRESSED

07D3  B4 00                MOV     AH,0              ; SET NO LIGHT PEN RETURN CODE
07D5  8B 16 0063 R         MOV     DX,ADDR_6845      ; GET BASE ADDRESS OF 6845
07D9  83 C2 06             ADD     DX,6              ; POINT TO STATUS REGISTER
07DC  EC                   IN      AL,DX             ; GET STATUS REGISTER
07DD  A8 04                TEST    AL,LIGHT_PEN_SWITCH ; TEST LIGHT PEN SWITCH
07DF  75 7E                JNZ     V6                ; NOT SET, RETURN

               ;----- NOW TEST FOR LIGHT PEN TRIGGER

07E1  A8 02                TEST    AL,2              ; TEST LIGHT PEN TRIGGER
07E3  75 03                JNZ     V7A               ; RETURN W/O RESETTING TRIGGER
07E5  E9 0869 R            JMP     V7

               ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN

07E8                    V7A:
07E8  B4 10                MOV     AH,16             ; LIGHT PEN REGISTERS ON 6845

               ;----- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX

07EA  8B 16 0063 R         MOV     DX,ADDR_6845      ; ADDRESS REGISTER FOR 6845
07EE  8A C4                MOV     AL,AH             ; REGISTER TO READ
07F0  EE                   OUT     DX,AL             ; SET IT UP
07F1  42                   INC     DX                ; DATA REGISTER
07F2  EC                   IN      AL,DX             ; GET THE VALUE
07F3  8A E8                MOV     CH,AL             ; SAVE IN CX
07F5  4A                   DEC     DX                ; ADDRESS REGISTER
07F6  FE C4                INC     AH
07F8  8A C4                MOV     AL,AH             ; SECOND DATA REGISTER
07FA  EE                   OUT     DX,AL
07FB  42                   INC     DX                ; POINT TO DATA REGISTER
07FC  EC                   IN      AL,DX             ; GET SECOND DATA VALUE
07FD  8A E5                MOV     AH,CH             ; AX HAS INPUT VALUE

               ;----- AX HAS THE VALUE READ IN FROM THE 6845
```

```
07FF  8A 1E 0049 R          MOV     BL,CRT_MODE
0803  2A FF                 SUB     BH,BH               ; MODE VALUE TO BX
0805  2E: 8A 9F 07CB R      MOV     BL,CS:V1[BX]        ; DETERMINE AMOUNT TO SUBTRACT
080A  2B C3                 SUB     AX,BX               ; TAKE IT AWAY
080C  8B 1E 004E R          MOV     BX,CRT_START
0810  D1 EB                 SHR     BX,1
0812  2B C3                 SUB     AX,BX
0814  79 02                 JNS     V2                  ; IF POSITIVE, DETERMINE MODE
0816  2B C0                 SUB     AX,AX               ; <0 PLAYS AS 0

              ;----- DETERMINE MODE OF OPERATION

0818                     V2:                            ; DETERMINE_MODE
0818  B1 03                 MOV     CL,3                ; SET *8 SHIFT COUNT
081A  80 3E 0049 R 04       CMP     CRT_MODE,4          ; DETERMINE IF GRAPHICS OR ALPHA
081F  72 2A                 JB      V4                  ; ALPHA_PEN
0821  80 3E 0049 R 07       CMP     CRT_MODE,7
0826  74 23                 JE      V4                  ; ALPHA_PEN

              ;----- GRAPHICS MODE

0828  B2 28                 MOV     DL,40               ; DIVISOR FOR GRAPHICS
082A  F6 F2                 DIV     DL                  ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                        ; AL RANGE 0-99, AH RANGE 0-39

              ;----- DETERMINE GRAPHIC ROW POSITION

082C  8A E8                 MOV     CH,AL               ; SAVE ROW VALUE IN CH
082E  02 ED                 ADD     CH,CH               ; *2 FOR EVEN/ODD FIELD
0830  8A DC                 MOV     BL,AH               ; COLUMN VALUE TO BX
0832  2A FF                 SUB     BH,BH               ; MULTIPLY BY 8 FOR MEDIUM RES
0834  80 3E 0049 R 06       CMP     CRT_MODE,6          ; DETERMINE MEDIUM OR HIGH RES
0839  75 04                 JNE     V3                  ; NOT_HIGH_RES
083B  B1 04                 MOV     CL,4                ; SHIFT VALUE FOR HIGH RES
083D  D0 E4                 SAL     AH,1                ; COL VAL TIMES 2 FOR HIGH RES
083F                     V3:                            ; NOT_HIGH_RES
083F  D3 E3                 SHL     BX,CL               ; MULTIPLY *16 FOR HIGH RES

              ;----- DETERMINE ALPHA CHAR POSITION

0841  8A D4                 MOV     DL,AH               ; COLUMN VALUE FOR RETURN
0843  8A F0                 MOV     DH,AL               ; ROW VALUE
0845  D0 EE                 SHR     DH,1                ; DIVIDE BY 4
0847  D0 EE                 SHR     DH,1                ; FOR VALUE IN 0-24 RANGE
0849  EB 12                 JMP     SHORT V5            ; LIGHT_PEN_RETURN_SET

              ;----- ALPHA MODE ON LIGHT PEN

084B                     V4:                            ; ALPHA_PEN
084B  F6 36 004A R          DIV     BYTE PTR CRT_COLS   ; DETERMINE ROW,COLUMN VALUE
084F  8A F0                 MOV     DH,AL               ; ROWS TO DH
0851  8A D4                 MOV     DL,AH               ; COLS TO DL
0853  D2 E0                 SAL     AL,CL               ; MULTIPLY ROWS * 8
0855  8A E8                 MOV     CH,AL               ; GET RASTER VAL TO RETURN REG
0857  8A DC                 MOV     BL,AH               ; COLUMN VALUE
0859  32 FF                 XOR     BH,BH               ; TO BX
085B  D3 E3                 SAL     BX,CL
085D                     V5:                            ; LIGHT_PEN_RETURN_SET
085D  B4 01                 MOV     AH,1                ; INDICATE EVERTHING SET
085F                     V6:                            ; LIGHT_PEN_RETURN
085F  52                    PUSH    DX                  ; SAVE RETURN VALUE (IN CASE)
0860  8B 16 0063 R          MOV     DX,ADDR_6845        ; GET BASE ADDRESS
0864  83 C2 07              ADD     DX,7                ; POINT TO RESET PARM
0867  EE                    OUT     DX,AL               ; ADDR, NOT DATA, IS IMPORTANT
0868  5A                    POP     DX                  ; RECOVER VALUE
0869                     V7:                            ; RETURN_NO_RESET
0869  55                    PUSH    BP
086A  8B EC                 MOV     BP,SP               ; GET PTR TO STACK SAVE AREA
086C  89 5E 08              MOV     [BP].BX_POS,BX      ; SETUP RETURN VALUES IN STACK
086F  89 4E 0A              MOV     [BP].CX_POS,CX      ; SAVE AREA
0872  89 56 0C              MOV     [BP].DX_POS,DX
0875  5D                    POP     BP
0876  E9 016F R             JMP     VIDEO_RETURN
0879                     READ_LPEN       ENDP
```

```
               ;-----------------------------------------------------------------
               ; WRITE_STRING
               ;
               ;          THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT.
               ;
               ; INPUT
               ;          (AL) = WRITE STRING COMMAND  0 - 3
               ;          (BH) = DISPLAY PAGE
               ;          (BL) = ATTRIBUTE OF CHAR TO WRITE IF AL == 0 OR AL == 1
               ;          (CX) = COUNT OF CHARACTERS TO WRITE, IF CX=0 THEN RETURN
               ;          (DX) = CURSOR POSITION FOR STRING TO BE WRITTEN
               ;            ROW = (0-24), COL = (0-(CRT COLUMNS-1))
               ;            CRT COLUMN SIZE IS EITHER 80 OR 40
               ;          (ES) = STRING SEGMENT
               ;          (BP) = STRING OFFSET
               ;          (DS) = DATA SEGMENT
               ; OUTPUT
               ;          N/A
               ;-----------------------------------------------------------------

0879                    WRITE_STRING    PROC    NEAR
0879  55                   PUSH    BP
087A  8B EC                MOV     BP,SP
087C  8E 46 10             MOV     ES,[BP].ES_POS      ; RECOVER STRING SEG ADDRESS
087F  5D                   POP     BP
0880  3C 04                CMP     AL,04               ; TEST FOR INVAL STRING OPTION
0882  72 03                JB      W0                  ; IF OPTION INVAL THEN RETURN
0884  E9 0924 R            JMP     DONE
0887  0B C9             W0:     OR      CX,CX          ; TEST FOR 0 LENGTH STRING
0889  75 03                JNZ     W1
088B  E9 0924 R            JMP     DONE                ; IF 0 LENGTH STRING, RETURN
088E  53                W1:     PUSH    BX             ; SAVE PAGE AND POSSIBLE ATTR
088F  8A DF                MOV     BL,BH               ; GET CURRENT CURSOR POSITION
0891  32 FF                XOR     BH,BH
0893  D1 E3                SAL     BX,1
0895  8B B7 0050 R         MOV     SI,[BX+OFFSET CURSOR_POSN]
0899  5B                   POP     BX                  ; RESTORE BX
089A  56                   PUSH    SI                  ; SAVE CURRENT CURSOR POSITION

089B  50                   PUSH    AX                  ; SAVE WRITE STRING OPTION
089C  B4 02                MOV     AH,SET_CURSOR_POS   ; SET NEW CURSOR POSITION
089E  CD 10                INT     VIDEO_FN
08A0  58                   POP     AX                  ; RESTORE WRITE STRING OPTION

08A1                    WRITE_CHAR:
08A1  51                   PUSH    CX
08A2  53                   PUSH    BX
08A3  50                   PUSH    AX
08A4  06                   PUSH    ES

08A5  86 E0                XCHG    AH,AL               ; PUT WRT STRING OPTION IN AH
08A7  26: 8A 46 00         MOV     AL,ES:[BP]          ; GET CHAR FROM INPUT STRING
08AB  45                   INC     BP                  ; BUMP POINTER TO CHARACTER

               ;----- TEST FOR SPECIAL CHARACTER'S

08AC  3C 08                CMP     AL,8                ; IS IT A BACKSPACE
08AE  74 0C                JE      DO_TTY              ; BACK_SPACE
08B0  3C 0D                CMP     AL,0DH              ; IS IT CARRIAGE RETURN
08B2  74 08                JE      DO_TTY              ; CAR_RET
08B4  3C 0A                CMP     AL,0AH              ; IS IT A LINE FEED
08B6  74 04                JE      DO_TTY              ; LINE_FEED
08B8  3C 07                CMP     AL,07H              ; IS IT A BELL
08BA  75 1C                JNE     GET_ATTRIBUTE       ; IF NOT THEN DO WRITE CHAR
08BC                    DO_TTY:
```

```
                    ;
                    ;----- WRITE_TTY IS CALLED TO PROCESS THE SPECIAL CHAR. WRITE_TTY WRTS
                    ;      TO THE CURRENT ACTIVE PAGE DEFINED BY GLOBAL VAR ACTIVE_PAGE.
                    ;      WRITE_STRING WRITES TO THE PAGE DEFINED BY USER IN BH REGISTER.
                    ;      THE ACTIVE_PAGE VARIABLE MAY NOT BE SAME PG AS DEFINED BY USER
                    ;      IN BH REGISTER. THE ACTIVE_PAGE VAR MUST BE TEMP SET TO
                    ;      BH REGISTER BEFORE CALLING WRITE_TTY. AND, BOTH ACTIVE-PAGE AND
                    ;      BH REGISTER MUST BE RESTORED AFTER.

08BC  86 3E 0062 R           XCHG   ACTIVE_PAGE,BH      ; TEMPORARILY EXCHANGE VALUES
08C0  B4 0E                  MOV    AH,WRITE_TELETYPE   ; WRITE TTY CHAR TO THE ACT PG
08C2  CD 10                  INT    VIDEO_FN            ; INT 10H
08C4  86 3E 0062 R           XCHG   ACTIVE_PAGE,BH      ; RESOTRE VALUES
08C8  8A DF                  MOV    BL,BH               ; GET CURRENT CURSOR POSITION
08CA  32 FF                  XOR    BH,BH
08CC  D1 E3                  SAL    BX,1                ; INTO THE DX REGISTER
08CE  8B 97 0050 R           MOV    DX,[BX+OFFSET CURSOR_POSN]
08D2  07                     POP    ES
08D3  58                     POP    AX                  ; RESTORE REGISTERS
08D4  5B                     POP    BX
08D5  59                     POP    CX
08D6  EB 37                  JMP    SHORT ROWS_SET

08D8                       GET_ATTRIBUTE:
08D8  B9 0001                MOV    CX,1                ; SET CHAR WRITE AMOUNT TO ONE
08DB  80 FC 02               CMP    AH,2                ; IS ATTRIBUTE IN THE STRING
08DE  72 05                  JB     GOT_IT              ; IF NOT THEN JUMP
08E0  26: 8A 5E 00           MOV    BL,ES:[BP]          ; ELSE GET IT
08E4  45                     INC    BP                  ; BUMP STRING POINTER

08E5                       GOT_IT:
08E5  B4 09                  MOV    AH,WRITE_ATT_CHAR   ; WRITE CHAR AND ATTRIBUTE
08E7  CD 10                  INT    VIDEO_FN
08E9  07                     POP    ES
08EA  58                     POP    AX                  ; RESTORE REGISTERS
08EB  5B                     POP    BX
08EC  59                     POP    CX

08ED  FE C2                  INC    DL                  ; INCREMENT COLUMN COUNTER
08EF  3A 16 004A R           CMP    DL,BYTE PTR CRT_COLS ; IF COLS ARE IN RANGE FOR
                                                        ; THIS MODE THEN
08F3  72 1A                  JB     COLUMNS_SET         ; GOTO COLS SET
08F5  FE C6                  INC    DH                  ; BUMP ROW COUNTER BY ONE
08F7  2A D2                  SUB    DL,DL               ; SET COLUMN COUNTER TO ZERO
08F9  80 FE 19               CMP    DH,25               ; IF ROWS ARE < 25 THEN
08FC  72 11                  JB     ROWS_SET            ; GOTO ROWS_SET
                    ;
                    ;-- WRITE_TTY IS CALLED TO PROCESS SCROLL LINE CMD. WRITE_TTY WRITES
                    ;   TO THE CURRENT ACTIVE PAGE DEFINED BY GLOBAL VARIABLE ACTIVE_PAGE.
                    ;   WRITE_STRING WRITES TO THE PAGE DEFINED BY USER IN BH REGISTER.
                    ;   THE ACTIVE_PAGE VARIABLE MAY NOT BE THE SAME PAGE DEFINED BY USER
                    ;   IN BH REGISTER. ACTIVE_PAGE VARIABLE MUST BE TEMPORARILY SET TO
                    ;   BH REGISTER BEFORE CALLING WRITE_TTY. AND, BOTH ACTIVE-PAGE AND
                    ;   BH REGISTER MUST BE RESTORED AFTER.
                    ;
08FE  50                     PUSH   AX                  ; SAVE WRITE STRING PARM REGS
08FF  86 3E 0062 R           XCHG   ACTIVE_PAGE,BH      ; TEMPORARILY EXCHANGE VALUES
0903  B8 0E0A                MOV    AX,WRITE_TELETYPE*256+0AH ; SCROLL 1 LINE
0906  CD 10                  INT    VIDEO_FN            ; FEED COMMAND TO WRITE TTY FN
0908  86 3E 0062 R           XCHG   ACTIVE_PAGE,BH      ; RESTORE TO ORIG VALUES
090C  FE CE                  DEC    DH                  ; RESET ROW COUNTER TO 24
090E  58                     POP    AX                  ; RESTORE REG'S
090F                       ROWS_SET:
090F                       COLUMNS_SET:
090F  50                     PUSH   AX                  ; SAVE WRITE STRING OPTION
0910  B4 02                  MOV    AH,SET_CURSOR_POS   ; SET NEW CURSOR POSITION
0912  CD 10                  INT    VIDEO_FN            ; AND RESTORES VIDEO PAGE
0914  58                     POP    AX
0915  E2 8A                  LOOP   WRITE_CHAR          ; DO IT UNTIL CX = ZERO

0917  5A                     POP    DX                  ; RESTORE OLD CURSOR COORDIN
0918  3C 01                  CMP    AL,1                ; IF CURSOR WAS TO BE MOVED
091A  74 08                  JE     DONE                ; WE'RE DONE
091C  3C 03                  CMP    AL,3
```

```
091E  74 04                JE      DONE
0920  B4 02                MOV     AH,SET_CURSOR_POS   ; ELSE RESTORE OLD CURSOR POS
0922  CD 10                INT     VIDEO_FN
0924                    DONE:
0924  E9 016F R            JMP     VIDEO_RETURN        ; RETURN TO CALLER

0927                    WRITE_STRING    ENDP

              ;------------------------------------------------------------
              ; LCD_REQUEST
              ;           THESE ROUTINES PERFORM FUNCTIONS SPECIFIC TO THE LCD
              ;           CONTROLLER.
              ; INPUT
              ;           AL - LCD REQUEST NUMBER
              ;                0 = LOAD USER SPECIFIED FONT
              ;                1 = LOAD SYSTEM ROM DEFAULT FONT
              ;                2 = SET LCD HIGH INTENSITY ATTRIBUTE MAPPING
              ;                3 - 255 = NO OPERATION
              ;           (DS) = DATA SEGMENT
              ;           ADDITIONAL INPUTS ARE REQUIRED FOR EACH REQUEST
              ;           LCD MUST BE PRESENT. IF NOT, NO OPERATION IS PERFORMED
              ;
              ; OUTPUT
              ;           LCD SPECIFIC FUNCTION WILL BE PERFORMED IF A LCD
              ;           IS PRESENT
              ;------------------------------------------------------------
0927                    LCD_REQUEST     PROC    NEAR
0927  50                   PUSH    AX                  ; SAVE LCD_REQUEST
0928  B4 20                MOV     AH,RTC_DSP_CON
092A  E8 0000 E            CALL    GET_RTC_REG         ; GET DISPLAY CONFIGURATION
092D  A8 03                TEST    AL,DSP_CLCD+DSP_MLCD
092F  58                   POP     AX                  ; RESTORE LCD REQUEST
                           JFZ     VIDEO_RETURN        ; LCD IS NOT AVAILABLE - EXIT
0930  75 03                JNZ     $+5                 ; IF NOT ZERO JUMP AROUND JUMP
0932  E9 016F R            JMP     VIDEO_RETURN        ; ELSE TAKE A LONG JUMP
              ;
0935  3C 00                CMP     AL,0                ; IF LOAD USER FONT REQUEST
0937  75 5A                JNE     DEFONT
              ;------------------------------------------------------------
              ; LOAD_USER_FONT
              ;           THIS ROUTINE ACCESSES THE LCD FONT STORAGE AND ALTERS
              ;           ONE OR MORE CHARACTERS, ALLOWING THE USER TO CAUSE DIFF-
              ;           ERENT CHARACTERS TO BE DISPLAYED IN ALPHA/NUMERIC MODE
              ; INPUT
              ;           ES:DI - POINT TO CHARACTER FONT IN USER TABLE WHERE
              ;LOADING IS TO START FROM
              ;           CX - NUMBER OF CHARACTERS TO STORE (1-256) VALUE CHECKED
              ;           DL - CHAR OFFSET INTO RAM FONT AREA
              ;           BH - NUMBER OF BYTES PER CHARACTER (1-255) VALUE CHECKED
              ;           BL - 0 = LOAD MAIN FONT (BLOCK 0)
              ;                1 = LOAD ALTERNATE FONT (BLOCK 1)
              ;                2-255 = NO OPERATION
              ; OUTPUT
              ;           THE USER SPECIFIED FONT IS LOADED
              ;           ALTERED CHARACTERS WILL DISPLAY DIFFERENTLY IN A/N MODE
              ;
              ; INTERRUPTS:
              ;           DISABLED DURING THE LOADING OF THE FONT
              ;------------------------------------------------------------

0939  B4 20                MOV     AH,RTC_DSP_CON      ; GET DISPLAY CONFIGURATION
093B  E8 0000 E            CALL    GET_RTC_REG

093E  A8 01                TEST    AL,DSP_CLCD
0940  B8 B800              MOV     AX,CGA_RAM          ; USE CGA REGEN ADDRESS
0943  75 03                JNZ     LCD1                ; IS LCD CONFIGURED AS CGA ?
0945  B8 B000              MOV     AX,MONO_RAM         ; NO, USE MONO REGEN ADDRESS

0948                    LCD1:                          ; LCD REGEN ADDR IS FONT SEG =
0948  8E C0                MOV     ES,AX               ; DEST SEG FOR FONT MOVE

094A  0A FF                OR      BH,BH               ; IS # OF BYTES PER CHAR > 0
094C  74 43                JZ      LCD3                ; NO, IT IS 0 THEN END
094E  8A C7                MOV     AL,BH               ; BYTES PER CHARACTER TIMES
```

```
0950  F6 E2                MUL     DL                 ; CHARACTER OFFSET EQUALS
0952  8B F8                MOV     DI,AX              ; DESTINATION INDEX

0954  80 FB 00             CMP     BL,0               ; REQUEST TO LOAD MAIN FONT
0957  74 09                JE      LCD2               ; YES, JUMP TO LOAD
0959  80 FB 01             CMP     BL,1               ; REQUEST TO LOAD ALT FONT
095C  75 33                JNE     LCD3               ; NO, JUMP TO END
095E  81 C7 1000           ADD     DI,1000H           ; YES ADJ DEST. IX ALT FONT
0962                    LCD2:
0962  55                   PUSH    BP
0963  8B EC                MOV     BP,SP              ; RECOVER ES & DI VAL FROM STK
0965  8E 5E 10             MOV     DS,[BP].ES_POS     ; SOURCE SEGMENT FOR FONT MOVE
0968  8B 76 04             MOV     SI,[BP].DI_POS     ; SET UP SOURCE IX FOR MOVE
096B  5D                   POP     BP

096C  0B C9                OR      CX,CX              ; IS # OF CHARS TO STORE > 0
096E  74 21                JZ      LCD3               ; NO, IT IS 0 THEN EXIT
0970  81 F9 0100           CMP     CX,256             ; MAX CHARS ALLOWED IS 256
0974  77 1B                JA      LCD3               ; IF > 256 THEN EXIT

0976  8A C7                MOV     AL,BH              ; BYTES PER CHARACTER TIMES
0978  2A E4                SUB     AH,AH
097A  F7 E1                MUL     CX                 ; NUMBER OF CHARS TO STORE =
097C  8B C8                MOV     CX,AX              ; NUMBER OF BYTES TO STORE

097E  9C                   PUSHF
097E  FA                   CLI
0980  B0 00                MOV     AL,LCD_FUNCT
0982  E6 74                OUT     LCD_INDX,AL        ; ACCESS LCD FN CONTROL REG
0984  E4 75                IN      AL,LCD_DATA
0986  0C 10                OR      AL,LCD_FONT
0988  E6 75                OUT     LCD_DATA,AL        ; ACCESS DISPLAY FONT STORAGE
098A  F3/ A4              REP    MOVSB                ; MOVE THE FONT
098C  24 EF                AND     AL,OFFH-LCD_FONT
098E  E6 75                OUT     LCD_DATA,AL        ; RETURN FROM FONT TO REGEN
0990  9D                   POPF
0991                    LCD3:                         ; END LOAD USER SPECIFIED FONT
0991  EB 50                JMP     SHORT LCDXIT
0993                    DEFONT:
0993  3C 01                CMP     AL,1               ; IF LOAD DEFAULT FONT REQUEST
0995  75 1D                JNE     INTENS
                 ;-------------------------------------------------------------
                 ; LOAD_DEFAULT_FONT
                 ;           THIS ROUTINE CAUSES THE LCD FONT STORAGE TO BE REINIT-
                 ;           IALIZED WITH THE SYSTEM ROM DEFAULT FONT
                 ; INPUT
                 ;           BL - 0 = LOAD MAIN FONT (BLOCK 0)
                 ;                1 = LOAD ALTERNATE FONT (BLOCK 1)
                 ;                2 - 255 = NO OPERATION
                 ; OUTPUT
                 ;           DEFAULT CHARACTERS WILL BE DISPLAYED IN A/N MODE
                 ;-------------------------------------------------------------
                 ;
0997  0E                   PUSH    CS                 ; PREPARE TO ISSUE LOAD
0998  07                   POP     ES                 ; USER FONT = DEFLT BIOS FONT
0999  B9 0080              MOV     CX,128             ; NO. OF CHARS TO WRITE = 128
099C  B7 08                MOV     BH,8               ; 8 BYTES PER CHAR
099E  B8 1400              MOV     AX,256*LCD_REQ+LOAD_USER ; SETUP LD USR FONT REQ
09A1  BF 0000 E            MOV     DI,OFFSET CHAR_GEN_LO ; LOAD LOWER 128 CHARS
09A4  B2 00                MOV     DL,0               ; CHAR OFFSET IN FONT STORAGE
09A6  CD 10                INT     VIDEO_FN           ; IRPT TO VIDEO I/O TO LD FONT
09A8  B8 1400              MOV     AX,256*LCD_REQ+LOAD_USER ; SETUP LD USER FONT REQ
09AB  BF 0000 E            MOV     DI,OFFSET CHAR_GEN_HI ; LOAD UPPER 128 CHARACTERS
09AE  B2 80                MOV     DL,128             ; CHAR OFFSET IN FONT STORAGE
09B0  CD 10                INT     VIDEO_FN           ; IRPT TO VIDEO I/O TO LD FONT
09B2  EB 2F                JMP     SHORT LCDXIT
09B4                    INTENS:
09B4  3C 02                CMP     AL,2               ; SET HIGH INTEN MAP REQUEST
09B6  75 2B                JNE     LCDXIT
```

```
                ;------------------------------------------------------------
                ; SET_HIGH_INTENSITY_MAPPING
                ;           THIS ROUTINE MAPS THE LCD HIGH INTENSITY ATTRIBUTE.
                ;           THIS ROUTINE ACCESSES THE LCD INTENSIFY REGISTER AND
                ;           CAUSES CHARACTERS WITH THE INTENSIFIED ATTRIBUTE TO
                ;           BE DISPLAYED DIFFERENTLY
                ; INPUT
                ;           BL - INTENSIFY MAPPING INPUT REGISTER
                ;                0 = IGNORE HIGH INTENSITY ATTRIBUTE
                ;                1 = MAP INTENSITY TO UNDERSCORE
                ;                2 = MAP INTENSITY TO REVERSE IMAGE
                ;                3 = MAP INTENSITY TO SELECT ALTERNATE FONT
                ;                4 - 255 = NO OPERATION
                ; OUTPUT
                ;           INTENSIFIED CHARACTERS WILL DISPLAY DIFFERENTLY
                ;
                ; INTERRUPTS:
                ;           DISABLED DURING PROCESSING
                ;------------------------------------------------------------

09B8  80 FB 04              CMP     BL,04H              ; ONLY OPERATION 0 - 3 ALLOWED
09BB  73 26                 JAE     LCDXIT              ; ALL OTHERS EXIT
09BD  BA 03B4               MOV     DX,3B4H
09C0  B4 20                 MOV     AH,RTC_DSP_CON
09C2  E8 0000 E             CALL    GET_RTC_REG
09C5  A8 01                 TEST    AL,DSP_CLCD         ; GET LCD CTRL PORT ADDRESSES
09C7  74 02                 JZ      LCD4                ; IF EMULATING MONO = 3B4
09C9  B2 D4                 MOV     DL,0D4H             ; IF EMULATING COLOR = 3D4
09CB                      LCD4:
09CB  9C                    PUSHF
09CC  FA                    CLI
09CD  B0 14                 MOV     AL,14H
09CF  EE                    OUT     DX,AL               ; REQUEST TO WRT TO REG 20
09D0  B0 77                 MOV     AL,77H              ; FOREGND RGB ON, BKGND RGB ON
09D2  F6 C3 02              TEST    BL,02H
09D5  74 02                 JZ      LCD5
09D7  0C 80                 OR      AL,80H              ; SET UP H ORDER BIT OF INTEN
09D9  F6 C3 01            LCD5:  TEST    BL,01H
09DC  74 02                 JZ      LCD6
09DE  0C 08                 OR      AL,8                ; SET UP L ORDER BIT OF INTEN
09E0  42                  LCD6:  INC     DX
09E1  EE                    OUT     DX,AL               ; WRT INTENSIFY DATA TO REG 20
09E2  9D                    POPF
                ;
09E3  E9 016F R           LCDXIT: JMP    VIDEO_RETURN

09E6                      LCD_REQUEST    ENDP
                ;------------------------------------------------------------
                ; PHYS_DSP_DESCR_REQ
                ;          THIS ROUTINE RETURNS THE ADDRESS OF A 7 WORD TABLE
                ;          CONTAINING THE PHYSICAL DISPLAY DESCRIPTION PARAMETERS OF
                ;          CURRENT DISPLAY. IT ALSO RETURNS THE MONITOR NUMBER OF
                ;          ALTERNATE OPERATIONAL DISPLAY.
                ;          THE TABLE CONTAINS THE FOLLOWING:
                ;
                ;            WORD                 INFORMATION
                ;             1           MONITOR MODEL NUMBER  (5140,5153,5151)
                ;             2           NUMBER OF VERTICAL PELS / METER
                ;             3           NUMBER OF HORIZONTAL PELS / METER
                ;             4           TOTAL NUMBER OF VERTICAL PELS
                ;             5           TOTAL NUMBER OF HORIZONTAL PELS
                ;             6           HORIZONTAL PEL SPACING IN MICROMETERS
                ;                (CENTER TO CENTER)
                ;             7           VERTICAL PEL SPACING IN MICROMETERS
                ;                (CENTER TO CENTER)
                ; INPUT
                ;           NONE
                ; OUTPUT
                ;           ES:DI  -  POINTER TO DISPLAY DESCRIPTION TABLE
                ;           AX     -  CONTAINS THE MONITOR NUMBER OF THE ALTERNATE
                ;                     OPERATIONAL DISPLAY. IF THERE IS NO ALTERNATE
                ;                     DISPLAY OR ITS INOPERABLE, AX = 0
                ;------------------------------------------------------------
09E6                      PHYS_DSP_DESCR_REQ  PROC  NEAR
```

```
                 ; FIGURED OUT WHAT DISPLAY IS CURRENTLY ACTIVE
09E6  B4 20                  MOV    AH,RTC_DSP_CON
09E8  E8 0000 E              CALL   GET_RTC_REG        ; GET DISPLAY CONFIGURATION
09EB  8A D8                  MOV    BL,AL              ; SAVE DISPLAY CONFIG
09ED  0E                     PUSH   CS
09EE  07                     POP    ES                 ; ES = CS
09EF  81 3E 0063 R 03D4      CMP    ADDR_6845,03D4H    ; IS CGA CURRENT MODE?
09F5  75 14                  JNE    PHYS_MONO          ; NO, JUMP TO MONO TEST
                 ;
                 ; COLOR IS CURRENT MODE. SEE IF COLOR DISPLAY OR LCD CONFIG AS COLOR
                 ; IS THE CURRENT DISPLAY. SET ES:DI TO POINT TO TBL OF CRNT DSPLY.
                 ; ALSO TURN OFF CURRENT DISPLAY BIT IN THE DISPLAY CONFIG BYTE.
                 ;
09F7  A8 01                  TEST   AL,DSP_CLCD        ; IS LCD CONFIGURED AS CGA?
09F9  74 08                  JZ     PHYS_CGA1          ; NO, JUMP TO COLOR DISPLAY
09FB  BF 0000 E              MOV    DI,OFFSET LCD_CGA_TBL ; SET ADDR TO LCD AS CGA TBL
09FE  80 E3 7E               AND    BL,NOT DSP_CLCD+DSP_LCD_PRES ; TURN OFF LCD BITS
0A01  EB 1A                  JMP    SHORT PHYS_ALT     ; YES, JUMP TO TEST ALT DSPLY
0A03                      PHYS_CGA1:                   ; CURRENT DSPLY IS COLOR DSPLY
0A03  BF 0000 E              MOV    DI,OFFSET CGA_TBL  ; SET ADDRESS TO CGA TABLE
0A06  80 E3 FB               AND    BL,NOT DSP_CGA     ; TURN OFF CURRENT DISPLAY BIT
0A09  EB 12                  JMP    SHORT PHYS_ALT     ; YES, JUMP TO TEST ALT DSPLY
                 ;
                 ; MONO IS CURRENT MODE. SEE IF MONOCHROME DSPLY OR LCD CONFIG AS MONO
                 ; IS CURRENT DISPLAY. SET ES:DI TO POINT TO TABLE OF CURRENT DISPLAY.
                 ; ALSO TURN OFF CURRENT DISPLAY BIT IN THE DISPLAY CONFIG BYTE.
                 ;
0A0B                      PHYS_MONO:
0A0B  A8 02                  TEST   AL,DSP_MLCD        ; IS LCD CONFIGURED AS MONO?
0A0D  74 08                  JZ     PHYS_MONO1         ; NO, JUMP TO COLOR DISPLAY
0A0F  BF 0000 E              MOV    DI,OFFSET LCD_MONO_TBL ; DEFAULT TO LCD MONO TABLE
0A12  80 E3 7D               AND    BL,NOT DSP_MLCD+DSP_LCD_PRES ; TURN OFF LCD BITS
0A15  EB 06                  JMP    SHORT PHYS_ALT     ; YES, JUMP TO TEST ALT DSPLY
0A17                      PHYS_MONO1:                  ; CURRENT DSPLY MONO MONITOR
0A17  BF 0000 E              MOV    DI,OFFSET MONO_TBL ; NO, CURRENT DSPLY MONO MNTR
0A1A  80 E3 F7               AND    BL,NOT DSP_MONO    ; TURN OFF CURRENT DISPLAY BIT
                 ;
                 ; TEST FOR THE ALTERNATE DISPLAY AND SET AX TO THE MONITOR NUMBER OF
                 ; ALTERNATE DISPLAY.  BL SHOULD CONTAIN THE ALTERNATE DISPLAY CONFIG.
                 ; IF THERE IS NO ALTERNATE DISPLAY OR THE ALTERNATE IS INOPERATIVE
                 ; THE MONITOR NUMBER IS 0.
0A1D                      PHYS_ALT:                    ; TEST FOR ALTERNATE DISPLAY
0A1D  84 DB                  TEST   BL,BL              ; ANY ALTERNATE DSPS PRESENT?
0A1F  74 2E                  JZ     PHYS_ALT_NONE      ; NO, JUMP TO SET MNTR # TO 0
0A21  F6 C3 03               TEST   BL,DSP_CLCD+DSP_MLCD ; IS ALT MONITOR THE LCD
0A24  74 06                  JZ     PHYS_ALT1          ; NO, JUMP TO CHECK OTHER DSPLY
0A26  2E: A1 0000 E          MOV    AX,LCD_CGA_TBL[0]  ; YES, SET LCD MONITOR #
0A2A  EB 25                  JMP    SHORT PHYS_END     ; JUMP TO EXIT
0A2C                      PHYS_ALT1:                   ; TEST FOR THE COLOR DISPLAY
0A2C  B4 22                  MOV    AH,RTC_DSP_STAT    ; GET THE DISPLAY STATUS
0A2E  E8 0000 E              CALL   GET_RTC_REG
0A31  F6 C3 04               TEST   BL,DSP_CGA         ; IS ALT DSP COLOR MONITOR?
0A34  74 0A                  JZ     PHYS_ALT2          ; JUMP TO TEST MONO MONITOR
0A36  A8 02                  TEST   AL,CGA_BAD         ; IS THE CGA BAD?
0A38  75 15                  JNZ    PHYS_ALT_NONE      ; YES, SET FOR NO ALTERNATE
0A3A  2E: A1 0000 E          MOV    AX,CGA_TBL[0]      ; NO, SET CGA MONITOR #
0A3E  EB 11                  JMP    SHORT PHYS_END     ; JUMP TO EXIT
0A40                      PHYS_ALT2:                   ; TEST FOR MONOCHROME MONITOR
0A40  F6 C3 08               TEST   BL,DSP_MONO        ; IS ALT DSPAY MONO MONITOR?
0A43  74 0A                  JZ     PHYS_ALT_NONE      ; JUMP NO ALT DISPLAY
0A45  A8 04                  TEST   AL,MONO_BAD        ; IS THE MONOCHROME BAD?
0A47  75 06                  JNZ    PHYS_ALT_NONE      ; YES, SET FOR NO ALTERNATE
0A49  2E: A1 0000 E          MOV    AX,MONO_TBL[0]     ; NO, SET MONOCHROME MONITOR #
0A4D  EB 02                  JMP    SHORT PHYS_END     ; JUMP TO EXIT
0A4F                      PHYS_ALT_NONE:               ; ALT NOT THERE OR INOPERATIVE
0A4F  2B C0                  SUB    AX,AX              ; SET MONITOR # TO 0
0A51                      PHYS_END:
0A51  55                     PUSH   BP
0A52  8B EC                  MOV    BP,SP              ; GET PTR TO STACK SAVE AREA
0A54  8C 46 10               MOV    [BP].ES_POS,ES     ; SET UP RETURN VAL IN STACK
0A57  89 7E 04               MOV    [BP].DI_POS,DI
0A5A  5D                     POP    BP
```

```
0A5B  E9 016F R           JMP     VIDEO_RETURN
0A5E                     PHYS_DSP_DESCR_REQ  ENDP
```

Print Screen Interrupt Hex 05 (PRT_SCRN)

```
                          SUBTTL  PRINT SCREEN BIOS   -   INT 05H HANDLER
                ;********************************************************************
                ;
                ; ROUTINE-NAME :  PRT_SCRN
                ;
                ; FUNCTION:   THIS ROUTINE PRINTS THE SCREEN
                ;
                ; ENTRY CONDITIONS:
                ;       PURPOSE OF ENTRY: PRINT SCREEN
                ;       INPUT CONDITIONS: NONE
                ;       RESTRICTIONS: ONLY ONE PRINT SCREEN REQUEST AT A TIME
                ;
                ; EXIT CONDITIONS:
                ;       NORMAL EXIT CONDITIONS:
                ;            STATUS_BYTE (50:0) = 0
                ;                 THE SCREEN IS PRINTED OR IF CTRL BREAK IS
                ;                 PRESSED THE PRINT IS TERMINATED.
                ;       ERROR EXIT CONDITIONS:
                ;            STATUS_BYTE (50:0) = 0FFH  (PRINTER ERROR)
                ;                 THE PRINT IS TERMINATED.
                ;            STATUS_BYTE (50:0) = 01  (PRINT SCREEN IN PROG.)
                ;                 PRINT SCREEN IS ALREADY IN PROGRESS AND ANOTHER
                ;                 PRINT SCREEN IS REQUESTED THE LATEST PRINT
                ;                 SCREEN REQUEST IS IGNORED.
                ;
                ;       REGISTERS MODIFIED: ALL SAVED
                ;       RETURN TYPE:  IRET (ALL FLAGS RESTORED)
                ;       INTERRUPTS:  ENABLED DURING PROCESSING
                ;
                ; INTERNALLY REFERENCED ROUTINES: CRLF
                ;
                ; EXTERNALLY REFERENCED ROUTINES: INT 10H (VIDEO_FN)
                ;                                 INT 17H (PRINTER_FN)
                ;
                ;****************************************************************************
                ;-- INT 5H -------------------------------------------------------------------
                ;       THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT THE
                ;       SCREEN. CURSOR POSITION AT THE TIME THIS ROUTINE IS INVOKED
                ;       WILL BE SAVED AND RESTORED UPON COMPLETION. THE ROUTINE IS
                ;       INTENDED TO RUN WITH INTERRUPTS ENABLED. IF A SUBSEQUENT
                ;       'PRINT SCREEN' KEY IS DEPRESSED DURING THE TIME THIS ROUTINE
                ;       IS PRINTING IT WILL BE IGNORED.
                ;       IF THE CONTROL BREAK KEYS ARE PRESSED DURING THE PRINTING, THE
                ;       PRINT SCREEN REQUEST IS TERMINATED.
                ;       ERRORS ENCOUNTERED DURING PRINT (RETURN BY INT 17H) CAN BE
                ;       OUT OF PAPER, I/O ERROR AND TIME OUT.
                ;       STATUS BYTE IS UPDATED ON EACH CALL TO PRINT SCREEN.
                ;       ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:
                ;
                ;       50:0    =0      EITHER PRINT SCREEN HAS NOT BEEN CALLED
                ;                       OR UPON RETURN FROM A CALL THIS INDICATES
                ;                       A SUCCESSFUL OPERATION.
                ;               =1      PRINT SCREEN IS IN PROGRESS
                ;               =255    ERROR ENCOUNTERED DURING PRINTING
                ;----------------------------------------------------------------------------
                          ASSUME  CS:ROMCODE,DS:DATA,ES:XXDATA

0A5E                     PRT_SCRN  PROC  FAR
0A5E  FB                  STI                          ; MUST RUN WITH INTERRUPTS ENABLED
0A5F  06                  PUSH    ES                   ; MUST USE 50:0 FOR DATA AREA
```

```
0A60  1E                    PUSH    DS                    ; MUST USE 40:0 FOR DATA AREA
0A61  50                    PUSH    AX
0A62  53                    PUSH    BX
0A63  51                    PUSH    CX                    ; FOR CURSOR LIMITS
0A64  52                    PUSH    DX                    ; CURRENT CURSOR POSITION
0A65  B8 ---- R             MOV     AX,XXDATA             ; HEX 50
0A68  8E C0                 MOV     ES,AX
0A6A  26: 80 3E 0000 R 01   CMP     STATUS_BYTE,PRTSC_ACTIVE; PRINT IN PROGRESS?
                            JFZ     EXIT                  ; YES, JUMP TO END
0A70  75 03                 JNZ     $+5                   ; IF NOT ZERO JUMP AROUND JUMP
0A72  E9 0B07 R             JMP     EXIT                  ; ELSE TAKE A LONG JUMP
0A75  26: C6 06 0000 R 01   MOV     STATUS_BYTE,PRTSC_ACTIVE; SHOW PRNT IN PROGRESS
0A7B  B8 ---- R             MOV     AX,DATA               ; HEX 40
0A7E  8E D8                 MOV     DS,AX
0A80  83 3E 0008 R 00       CMP     PRINTER_BASE[0],0     ; IS PRINTER ATTACHED?
                            JFZ     ERR20                 ; NO, ERROR EXIT
0A85  75 03                 JNZ     $+5                   ; IF NOT ZERO JUMP AROUND JUMP
0A87  EB 6F 90              JMP     ERR20                 ; ELSE TAKE A LONG JUMP
0A8A  B4 0F                 MOV     AH,GET_VIDEO_STATE    ; REQUEST CURRENT SCREEN MODE
0A8C  CD 10                 INT     VIDEO_FN              ; ON RETURN   [AL] = MODE,
                                                          ; [AH]=NUMBER COLUMNS/LINE,
               ;-------------------------------------------------------------
               ;            AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
               ;            [AX] AND THE PAGE IF APPLICABLE IS IN -BH-. THE STACK
               ;            HAS DS,AX,BX,CX,DX PUSHED. -AL- HAS VIDEO MODE
               ;-------------------------------------------------------------
0A8E  8A CC                 MOV     CL,AH                 ; USE OF -CX- REGISTER TO
0A90  B5 19                 MOV     CH,25                 ; CONTROL ROW & COLUMNS
0A92  E8 0B0E R             CALL    CRLF                  ; CR LF ROUTINE
0A95  F6 C4 29              TEST    AH,29H                ; TEST FOR PRINTER ERRORS
                                                          ; OUT OF PAPER, I/O & TIME OUT
0A98  75 5E                 JNZ     ERR20                 ; JUMP IF ERROR DETECTED
0A9A  51                    PUSH    CX                    ; SAVE SCREEN BOUNDS
0A9B  B4 03                 MOV     AH,GET_CURSOR_INFO    ; WILL NOW READ THE CURSOR.
0A9D  CD 10                 INT     VIDEO_FN              ; ON RETURN: CX=CRNT CRSR MODE,
                                                          ; DX = ROW,COL OF CRNT CURSOR
0A9F  59                    POP     CX                    ; RECALL SCREEN BOUNDS
0AA0  52                    PUSH    DX                    ; SAVE CURRENT CURSOR POSITION
0AA1  33 D2                 XOR     DX,DX                 ; WILL SET CURSOR POS TO -0,0-
               ;-------------------------------------------------------------
               ;            THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20
               ;            IS THE LOOP TO READ EACH CURSOR POSITION FROM THE
               ;            SCREEN AND PRINT.
               ;-------------------------------------------------------------
0AA3                     PRI10:
0AA3  F6 06 0071 R 80       TEST    BIOS_BREAK,BREAK_HIT ; HAS BREAK KEY BEEN PRESSED?
0AA8  74 07                 JZ      PRI12                 ; NO KEY, JUMP AROUND
0AAA  80 26 0071 R 7F       AND     BIOS_BREAK,NOT BREAK_HIT; YES, TURN BREAK KEY OFF
0AAF  EB 35                 JMP     SHORT  PRI20          ; JUMP TO END
0AB1                     PRI12:
0AB1  B4 02                 MOV     AH,SET_CURSOR_POS     ; TO SHOW CURSOR SET REQUEST
0AB3  CD 10                 INT     VIDEO_FN              ; NEW CURSOR POSITION
0AB5  B4 08                 MOV     AH,READ_ATT_CHAR      ; TO INDICATE READ CHARACTER
0AB7  CD 10                 INT     VIDEO_FN              ; CHARACTER NOW IN [AL]
0AB9  0A C0                 OR      AL,AL                 ; SEE IF VALID CHAR
0ABB  75 02                 JNZ     PRI15                 ; JUMP IF VALID CHAR
0ABD  B0 20                 MOV     AL,' '                ; MAKE A BLANK
0ABF                     PRI15:
0ABF  52                    PUSH    DX                    ; SAVE CURSOR POSITION
0AC0  33 D2                 XOR     DX,DX                 ; INDICATE PRINTER 1
0AC2  32 E4                 XOR     AH,AH                 ; TO SHOW PRINT CHAR IN -AL-
0AC4  CD 17                 INT     PRINTER_FN            ; PRINT THE CHARACTER
0AC6  5A                    POP     DX                    ; RECALL CURSOR POSITION
0AC7  F6 C4 29              TEST    AH, 29H               ; TEST FOR PRINTER ERRORS
                                                          ; OUT OT PAPER, I/O & TIME OUT
0ACA  75 27                 JNZ     ERR10                 ; JUMP IF ERROR DETECTED
0ACC  FE C2                 INC     DL                    ; ADVANCE TO NEXT COLUMN
0ACE  3A CA                 CMP     CL,DL                 ; SEE IF AT END OF LINE
0AD0  75 D1                 JNE     PRI10                 ; IF NOT PROCEED
0AD2  32 D2                 XOR     DL,DL                 ; BACK TO COLUMN 0
0AD4  8A E2                 MOV     AH,DL                 ; [AH]=0
0AD6  52                    PUSH    DX                    ; SAVE NEW CURSOR POSITION
0AD7  E8 0B0E R             CALL    CRLF                  ; LINE FEED CARRIAGE RETURN
```

```
0ADA  5A                  POP     DX                  ; RECALL CURSOR POSITION
0ADB  F6 C4 29            TEST    AH,29H              ; TEST FOR PRINTER ERRORS
                                                      ; OUT OF PAPER, I/O & TIME OUT
0ADE  75 13               JNZ     ERR10               ; JUMP IF ERROR DETECTED
0AE0  FE C6               INC     DH                  ; ADVANCE TO NEXT LINE
0AE2  3A EE               CMP     CH,DH               ; FINISHED?
0AE4  75 BD               JNE     PRI10               ; IF NOT CONTINUE
0AE6                   PRI20:
0AE6  5A                  POP     DX                  ; RECALL CURSOR POSITION
0AE7  B4 02               MOV     AH,SET_CURSOR_POS   ; TO SHOW CURSOR SET REQUEST
0AE9  CD 10               INT     VIDEO_FN            ; CURSOR POSITION RESTORED
0AEB  26: C6 06 0000 R    MOV     STATUS_BYTE,0       ; INDICATE  FINISHED
00
0AF1  EB 14               JMP     SHORT EXIT          ; EXIT THE ROUTINE
0AF3                   ERR10:
0AF3  5A                  POP     DX                  ; GET CURSOR POSITION
0AF4  B4 02               MOV     AH,SET_CURSOR_POS   ; TO REQUEST CURSOR SET
0AF6  CD 10               INT     VIDEO_FN            ; CURSOR POSITION RESTORED
0AF8                   ERR20:
0AF8  BB 0053             MOV     BX,083              ; NO. CYCLES FOR 83MSEC TONE
0AFB  B9 0081             MOV     CX,081H             ; 1/2 CYCLE FOR 1KHZ TONE
0AFE  E8 0000 E           CALL    KB_NOISE            ; SOUND BEEP FOR ERROR
0B01  26: C6 06 0000 R    MOV     STATUS_BYTE,PRTSC_ERROR ; INDICATE   ERROR
FF
0B07                   EXIT:
0B07  5A                  POP     DX                  ; RESTORE ALL REGISTERS USED
0B08  59                  POP     CX
0B09  5B                  POP     BX
0B0A  58                  POP     AX
0B0B  1F                  POP     DS
0B0C  07                  POP     ES
0B0D  CF                  IRET
0B0E                   PRT_SCRN   ENDP

               ;------ CARRIAGE RETURN, LINE FEED SUBROUTINE

0B0E                   CRLF    PROC    NEAR
0B0E  33 D2               XOR     DX,DX               ; PRINTER 0
0B10  32 E4               XOR     AH,AH               ; WILL NOW SEND INITIAL LF,CR
                                                      ; TO PRINTER
0B12  B0 0A               MOV     AL,12Q              ; LF
0B14  CD 17               INT     PRINTER_FN          ; SEND THE LINE FEED
0B16  F6 C4 29            TEST    AH,29H              ; TEST FOR PRINTER ERRORS
                                                      ; OUT OF PAPER, I/O & TIME OUT
0B19  75 06               JNZ     CRLF1               ; EXIT IF ERROR
0B1B  32 E4               XOR     AH,AH               ; NOW FOR THE CR
0B1D  B0 0D               MOV     AL,15Q              ; CR
0B1F  CD 17               INT     PRINTER_FN          ; SEND THE CARRIAGE RETURN
0B21  C3               CRLF1:  RET
0B22                   CRLF    ENDP
0B22                   ROMCODE ENDS
                          END
```

Diskette Support (B13DSKT)

```
                 ;*********************************************************************
                 ;  P U B L I C S
                 ;*********************************************************************
                             PUBLIC  DSKT_IO
                             PUBLIC  SEEK
                             PUBLIC  DSKT_INTE
                             PUBLIC  SYS_BOOT
                             PUBLIC  DISK_RESET
                             PUBLIC  NEC_OUTPUT
                             PUBLIC  RESULTS
                             PUBLIC  SEEK
                             PUBLIC  GET_PARM

                 ;*********************************************************************
                 ;  E X T E R N A L      R E F E R E N C E S
                 ;*********************************************************************
                             EXTRN   DDS:NEAR
                             EXTRN   DSKT_BASE:BYTE
                             EXTRN   GET_RTC_REG:NEAR
                             EXTRN   PUT_RTC_REG:NEAR
                             EXTRN   PARMS_TPI48:BYTE
                             EXTRN   PARMS_TPI135:BYTE
                             EXTRN   DSP_FSETM:NEAR
                             EXTRN   DSP_INIT:NEAR
                             EXTRN   ICON_PR:NEAR
                             EXTRN   EXT_EVENT:NEAR
                             EXTRN   F1_ICON:BYTE
                             EXTRN   SYS_DSKT_ICON:BYTE
                             EXTRN   DSKT_ICON:BYTE
                             EXTRN   BAD_DSKT_ICON:BYTE
                             EXTRN   RES_ERR_CHK:NEAR
                             EXTRN   POST_LOOP:NEAR

0000                     ROMCODE SEGMENT BYTE PUBLIC
```

```
;*********************************************************************
;
; MODULE-NAME :      B13DSKT
;
; DATE LAST MODIFIED:  09/12/85
;
; DESCRIPTIVE-NAME : BIOS DISKETTE SERVICE ROUTINES
;
; COPYRIGHT : 7396-917 (C) COPYRIGHT IBM CORP. 1985
;             REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
;
; CHANGE LEVEL: 0.0
;
; FUNCTION:   THIS MODULE PROVIDES INPUT/ OUTPUT SERVICE FUNCTIONS TO THE
;             FLOPPY DISKETTE CONTROLLER / DISKETTE DRIVES AND ASSOCIATE
;             HARDWARE.
;
; MODULE SIZE: 1761 BYTES
;
; INPUT PARAMETERS: SEE LIST PROVIDED FOR EACH FUNCTION CALL BELOW
;
; OUTPUT PARAMETERS: SEE LIST PROVIDED FOR EACH FUNCTION CALL BELOW
;
; ROUTINES IN MODULE:   DSKT_IO - COMMON ENTRY FOR ALL OTHER ROUTINES
;                                 FOR DISKETTE SERVICES
;                       SYS_BOOT- ENTRY FOR LOADING
;                                 THE DISKETTE BOOT PROGRAM
;
; INTERNAL DATA AREAS / TABLES: BIOS DATA AREA AT SEGMENT 40H
;
; EXTERNALLY REFERENCED ROUTINES: REFER TO EXTRN LIST
;
; EXTERNALLY REFERENCED DATA AREAS:     REFER TO EXTRN LIST
;
; CHANGE ACTIVITY:
;
;*********************************************************************
```

Diskette I/O Interrupt Hex 13 (DSKT_IO)

```
;-- INT 13H ---------------------------------------------------------
; DISKETTE I/O - DSKT_IO
;--------------------------------------------------------------------
;  INPUT:
;       (AH)=0  RESET DISKETTE SYSTEM
;               HARD RESET TO NEC, PREPARE COMMAND, SET RECAL REQUIRED
;               ON ALL DRIVES
;--------------------------------------------------------------------
;       (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)
;               DISKETTE_STATUS FROM LAST OPERATION IS USED
;--------------------------------------------------------------------
;       (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, BUFFER POINTER (ES,BX)
;               MUST POINT TO COLLECTION OF DESIRED ADDR 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.
;-----------------------------------------------------------------------
;              REGISTERS FOR READ/WRITE/VERIFY/FORMAT
;       INPUT:
;            (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
;            (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
;            (CH) - TRACK NUMBER (0-4F, NOT VALUE CHECKED)
;            (CL) - SECTOR NO. (NOT VALUE CHKD, NOT USED FOR FORMAT)
;            (AL) - NO. OF SECTORS ( NOT VALUE CHKD, NOT USED FOR FMT)  T)
;            (ES:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY
;
;            DATA VARIABLE -- DISK_POINTER
;            DOUBLE WORD POINTER TO CURRENT SET OF DSKT PARAMETERS
;
;       OUTPUT:
;            AH = STATUS OF OPERATION
;                   GOOD RETURN      00H     NO ERROR DETECTED
;                   TIME_OUT         80H     ATTACHMENT FAILED TO RESPOND
;                   BAD_SEEK         40H     SEEK OPERATION FAILED
;                   BAD_NEC          20H     NEC CONTROLLER HAS FAILED
;                   BAD_CRC          10H     BAD CRC ON DISKETTE READ
;                   DMA_BOUNDARY     09H     ATMPT TO DMA CROSS 64K BNDRY
;                   BAD_DMA          08H     DMA OVERRUN ON OPERATION
;                   MEDIA_CHANGE     06H     MEDIA HAS BEEN CHANGED
;                   RECORD_NOT_FND   04H     REQUESTED SECTOR NOT FOUND
;                   WRITE_PROTECT    03H     WRT ATMPTD ON WRT PROT DISK
;                   BAD_ADDR_MARK    02H     ADDRESS MARK NOT FOUND
;                   BAD_CMD          01H     BAD CMD PASSED TO DSKT I/O
;
;            AL = NUMBER OF SECTORS ACTUALLY TRANSFERRED
;            DS,BX,DX,CX PRESERVED
;
;            CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)
;            CY = 1  FAILED OPERATION (AH HAS ERROR CODE)
;
;            ************   ERROR RETRY PROCEDURE   *************
;
;            NOTE:   IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE
;            APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN RETRY
;            THE OPERATION.  MEDIA CHANGE ERRORS NEED NOT BE RETRIED.
```

```
;-----------------------------------------------------------------------
;            (AH)=8  READ DRIVE PARAMETERS
;             DL = DRIVE NUMBER (0-3)
;
;       OUTPUT PARAMETERS:
;              IF DRIVE INSTALLED:
;                   ES:DI = POINTER TO DRIVE PARAMETERS TABLE
;                   CH = MAXIMUM TRACK NUMBER / SIDE (LOWER 8 BITS)
;                   CL (BITS 7-6) = TWO MOST SIGNIFICANT BITS OF
;                           10 BIT TRACK NUMBER
;                   CL (bits 5-0) = MAXIMUM 512 BYTE SECTOR NUMBER
;                           (6 BITS) PER TRACK
;                   DH = MAXIMUM HEAD NUMBER
;                   DL = NUMBER OF DISKETTE DRIVES INSTALLED ON
;                           SYSTEM  (1 - 2)
;                   AX = 0
;                   BH = 0
;                   BL = DRIVE TYPE (1 - 360K 40 TRACK DRIVE)
;                        (2 - 1.2 Meg 80 TRACK DRIVE)
;                        (3 - 720K 80 TRACK DRIVE)
;            DISKETTE_STATUS IS CLEARED, CARRY FLAG IS RESET.
;
;            IF DRIVE NOT INSTALLED:
;            ES,AX,BX,CX,DH,DI = 0
;                          DL = NUMBER OF DISKETTE DRIVES INSTALLED
;                 DISKETTE_STATUS = 0
;                 CARRY FLAG IS RESET.
;-----------------------------------------------------------------------
;            (AH) = 15H CHECK FOR CHANGE LINE SUPPORT (READ DASD)
;             DL = DRIVE NUMBER (0-3)
;       OUTPUT PARAMETERS:
;           AH = 00 - NO DRIVE PRESENT
;                01 - DSKT DRV WITH NO CHANGE LINE SUPPORT INSTALLED
;                02 - DSKT DRV WITH CHANGE LINE SUPPORT INSTALLED
;                03 - FIXED DISK
;          DISKETTE_STATUS = 0, CARRY FLAG CLEAR
;
;-----------------------------------------------------------------------
;       (AH)=16H READ DISKETTE CHANGE LINE STATUS
;        DL = DRIVE NUMBER (0-3)
;  OUTPUT PARAMETERS:
;      IF DRIVE INSTALLED:
;
;         DISKETTE_STATUS    = 00H- DISK CHANGE LINE NOT ACTIVE
;                                     - CARRY FLAG IS CLEARED
;                                06H- DISK CHANGE LINE ACTIVE
;                                     - CARRY FLAG IS SET
;                                     - DISKETTE HEAD STEPPED TO
;                                       TRACK 1 THEN 0 TO RESET CHANGE LINE
;                                80H- DISK CHANGE LINE ACTIVE AND
;                                       CANNOT BE RESET
;                                       (NO DISKETTE IN DRIVE)
;                                      - CARRY FLAG IS SET
;       IF DRIVE NOT INSTALLED:
;          DISKETTE_STATUS = TIMEOUT     (80H)
;                            CARRY FLAG IS CLEARED
;       IF DRIVE DOES NOT SUPPORT CHANGE LINE:
;          DISKETTE_STATUS = MEDIA CHANGE (06)
;                       CARRY FLAG IS CLEARED
;
;-----------------------------------------------------------------------
;       (AH)=17H   SET DASD TYPE FOR FORMAT
;        DL = DRIVE NUMBER (0-3)
;        AL = FORMAT TYPE
;             00 = NOT USED
;             01 = DISKETTE 320/360K IN 360K DRIVE
;             02 = DISKETTE 360K IN 1.2MEG DRIVE
;             03 = DISKETTE 1.2MEG IN 1.2MEG DRIVE
;             04 = DISKETTE 720K IN 720K DRIVE
;
;        OUTPUT PARAMETERS:
;        AH=DISKETTE_STATUS= 0   CARRY FLAG CLEAR
;        NO FUNCTION PERFORMED
;
```

```
                ;----------------------------------------------------------------
                ;       (AH) = ALL OTHER VALUES
                ;       OUTPUT:   AH=DISKETTE_STATUS = 01 (BAD COMMAND) CARRY FLAG SET
                ;----------------------------------------------------------------

                            ASSUME  CS:ROMCODE,DS:DATA

                ;
                ; STRUCTURE DEFINING SAVE AREA ON STACK
                ;
                        REGSAVE STRUC
0000  ????              DXSAVE  DW      ?               ; DX SAVE AREA ON STACK
0002  ????              BPSAVE  DW      ?               ; BP SAVE AREA ON STACK
0004  ????              DISAVE  DW      ?               ; DI SAVE AREA ON STACK
0006  ????              SISAVE  DW      ?               ; SI SAVE AREA ON STACK
0008  ????              DSSAVE  DW      ?               ; DS SAVE AREA ON STACK
000A  ????              CXSAVE  DW      ?               ; CX SAVE AREA ON STACK
000C  ????              BXSAVE  DW      ?               ; BX SAVE AREA ON STACK
000E  ????              AXSAVE  DW      ?               ; AX SAVE AREA ON STACK
0010                    REGSAVE ENDS

                        REGHSAV STRUC
0000  ??                DLSAVE  DB      ?               ; DL SAVE AREA ON STACK
0001  ??                DHSAVE  DB      ?               ; DH SAVE AREA ON STACK
0002     08 -                   DB      8 DUP(?)
              ??
                 -

243
000A  ??                CLSAVE  DB      ?               ; CL SAVE AREA ON STACK
000B  ??                CHSAVE  DB      ?               ; CH SAVE AREA ON STACK
000C  ??                BLSAVE  DB      ?               ; BL SAVE AREA ON STACK
000D  ??                BHSAVE  DB      ?               ; BH SAVE AREA ON STACK
000E  ??                ALSAVE  DB      ?               ; AL SAVE AREA ON STACK
000F  ??                AHSAVE  DB      ?               ; AH SAVE AREA ON STACK
0010                    REGHSAV ENDS

0000                    DSKT_TABLE      LABEL   WORD
0000  0082 R               DW      OFFSET DISK_RESET   ; AH=0    -- DISK RESET COMMAND  --
0002  00F6 R               DW      OFFSET DISK_STATUS  ; AH=1    -- DISK STATUS COMMAND --
0004  00FD R               DW      OFFSET DISK_READ    ; AH=2    -- DISK READ COMMAND   --
0006  0139 R               DW      OFFSET DISK_WRITE   ; AH=3    -- DISK WRITE COMMAND  --
0008  010F R               DW      OFFSET DISK_VERF    ; AH=4    -- DISK VERIFY COMMAND --
000A  0122 R               DW      OFFSET DISK_FORMAT  ; AH=5    -- DISK FORMAT COMMAND --
000C  007C R               DW      OFFSET DSKT_INV_CMD ; AH=6    -- INVALID COMMAND --
000E  007C R               DW      OFFSET DSKT_INV_CMD ; AH=7    -- INVALID COMMAND --
0010  0494 R               DW      OFFSET DSKT_RDPARM  ; AH=8    -- READ DISKETTE DRIVE PARMS --
                                                       ; AH=9 THRU 12 AND <17 ARE ALSO INVALID
0012  04E3 R               DW      OFFSET DSKT_RDDASD  ; AH=15   -- DETERMINE SUPPORT OF DCL  --
0014  050B R               DW      OFFSET DSKT_CHANGE  ; AH=16   -- DISKETTE CHANGE LINE STAT --
0016  0580 R               DW      OFFSET DSKT_SETDASD ; AH=17   -- DISKETTE SET DASD TYPE    --

0018                    DSKT_IO PROC    FAR
0018  FB                   STI                          ; INTERRUPTS BACK ON

0019  50                   PUSH    AX                   ; TEMPORARY SAVE AREA
001A  53                   PUSH    BX                   ; SAVE ADDRESS
001B  51                   PUSH    CX
001C  1E                   PUSH    DS                   ; SAVE SEGMENT REGISTER VALUE
001D  56                   PUSH    SI                   ; SAVE ALL REGISTERS DURING OPERATION
001E  57                   PUSH    DI
001F  55                   PUSH    BP
0020  52                   PUSH    DX
0021  8B EC                MOV     BP,SP                ; SET UP POINTER TO HEAD PARM
0023  E8 0000 E            CALL    DDS
0026  2A FF                SUB     BH,BH
0028  8A DC                MOV     BL,AH                ; SET FUNCTION CODE IN LOW BYTE

002A  80 FC 08             CMP     AH,8                 ; CHECK FUNCTION RANGE
002D  76 15                JBE     DIO_1                ; JUMP IF OKAY
002F  80 FC 15             CMP     AH,15H               ; CHECK FUNCTION RANGE
0032  72 05                JB      DIO_INV_CMD          ; JUMP IF ERROR
0034  80 FC 18             CMP     AH,18H               ; JUMP IF BELOW LIMIT
0037  72 08                JB      DIO_0                ; JUMP IF OKAY AND IN 15-17 RANGE
                ;
                ; FUNCTION CODE OUT OF RANGE
```

```
                 ;
0039                        DIO_INV_CMD:
0039  C6 06 0041 R 01          MOV     DISKETTE_STATUS,BAD_CMD ; INDICATE INVALID COMMAND
003E  EB 26 90                 JMP     DIO_2                ; GET STATUS AND EXIT

0041                        DIO_0:
0041  80 EB 0C                 SUB     BL,12                ; CORRECT FUNCTION CODE FOR 15,16
0044                        DIO_1:
0044  80 FC 01                 CMP     AH,1                 ; RESET COMMAND OR STATUS COMMAND?
0047  76 05                    JBE     DIO_11               ; JUMP AROUND DRIVE RANGE CHECK IF SO

                 ;
                 ; RANGE CHECK DRIVE NUMBER
                 ;
0049  80 FA 03                 CMP     DL,3
004C  77 EB                    JA      DIO_INV_CMD          ; JUMP IF INVALID

004E                        DIO_11:
004E  D1 E3                    SHL     BX,1                 ; DOUBLE LINK TABLE OFFSET
0050  80 26 003F R 7F          AND     MOTOR_STATUS,NOT WRITE_OP ; RESET WRITE OPERATION FLAG
0055  2E: FF 97 0000 R         CALL    DSKT_TABLE[BX]       ; CALL PROPER ROUTINE

005A  91                       XCHG    CX,AX                ; SAVE AX
005B  BB 0004                  MOV     BX,4                 ; GET THE MOTOR WAIT PARAMETER
005E  E8 0259 R                CALL    GET_PARM
0061  88 26 0040 R             MOV     MOTOR_COUNT,AH       ; SET THE TIMER COUNT FOR THE MOTOR
0065  91                       XCHG    AX,CX                ; RESTORE AL

0066                        DIO_2:
0066  8A 26 0041 R             MOV     AH,DISKETTE_STATUS   ; GET STATUS OF OPERATION
006A  88 66 0F                 MOV     AHSAVE[BP],AH        ; SAVE RETURN CODE ON STACK
006D  80 FC 01                 CMP     AH,1                 ; SET THE CARRY FLAG FOR FAILURE
0070  F5                       CMC
0071                        DIO_3:
0071  5A                       POP     DX                   ; RESTORE ALL REGISTERS
0072  5D                       POP     BP
0073  5F                       POP     DI
0074  5E                       POP     SI
0075  1F                       POP     DS
0076  59                       POP     CX
0077  5B                       POP     BX                   ; RECOVER ADDRESS
0078  58                       POP     AX                   ; RESTORE AX REGISTER
0079  CA 0002                  RET     2                    ; THROW AWAY SAVED FLAGS
007C                        DSKT_IO ENDP


                 ;*****************************
                 ; INVALID COMMAND RECEIVED
                 ;*****************************
007C                        DSKT_INV_CMD PROC NEAR
007C  C6 06 0041 R 01           MOV    DISKETTE_STATUS,BAD_CMD ; SV STAT & ST CRY FOR ERR
0081  C3                        RET
0082                        DSKT_INV_CMD ENDP

                 ;******************************************************
                 ;----- RESET THE DISKETTE SYSTEM
                 ;******************************************************
0082                        DISK_RESET      PROC    NEAR
0082  BA 03F2                  MOV     DX,DRIVE_CNTL        ; ADAPTER CONTROL PORT
0085  FA                       CLI                          ; NO INTERRUPTS
0086  C6 06 0040 R FF          MOV     MOTOR_COUNT,0FFH     ; SET LONG MOTOR ON TIME
008B  A0 003F R                MOV     AL,MOTOR_STATUS      ; WHICH MOTOR IS ON
008E  B1 04                    MOV     CL,4                 ; SHIFT COUNT
0090  D2 E0                    SAL     AL,CL                ; MOVE MOTOR VALUE TO HIGH NYBBLE
0092  A8 20                    TEST    AL, 20H              ; SELECT CORRESPONDING DRIVE
0094  75 0C                    JNZ     J5                   ; JUMP IF MOTOR ONE IS ON
0096  A8 40                    TEST    AL, 40H
0098  75 06                    JNZ     J4                   ; JUMP IF MOTOR TWO IS ON
009A  A8 80                    TEST    AL, 80H
009C  74 06                    JZ      J6                   ; JUMP IF MOTOR ZERO IS ON
009E  FE C0                    INC     AL
00A0                        J4:
00A0  FE C0                    INC     AL
00A2                        J5:
00A2  FE C0                    INC     AL
```

```
00A4                    J6:
00A4  0C 08                OR      AL,FDC_DMA_ENAB     ; TURN ON INTERRUPT ENABLE
00A6  EE                   OUT     DX,AL               ; RESET THE ADAPTER

00A7  C6 06 003E R 00      MOV     SEEK_STATUS,0       ; SET RECAL REQUIRED ON ALL DRIVES

00AC  C6 06 0041 R 00      MOV     DISKETTE_STATUS,0   ; SET OK STATUS FOR DISKETTE
00B1  0C 04                OR      AL,FDC_RUN          ; TURN OFF RESET
00B3  EE                   OUT     DX,AL               ; TURN OFF THE RESET
00B4  E8 0373 R            CALL    WAIT_INT            ; WAIT FOR INTERRUPT
00B7  72 3C                JC      J8                  ; IF NO INTERRUPT THEN EXIT
00B9  B8 00F5 R            MOV     AX,OFFSET J8        ; SET ERROR RETURN ADDRESS FOR NEC_OUT
00BC  50                   PUSH    AX
00BD  B9 0004              MOV     CX,4                ; READ RESET STATUS FOR ALL 4 DRIVES

00C0                    STAT_LOOP:
00C0  B4 08                MOV     AH,READ_INT_STATUS  ; SENSE INTERRUPT STATUS
00C2  E8 022C R            CALL    NEC_OUTPUT          ; COMMAND TO NEC
00C5  E8 03B3 R            CALL    RESULTS             ; GET THE RESULTS
00C8  A0 0042 R            MOV     AL,NEC_STATUS       ; IGNORE ERROR RETURN AND DO OWN TEST
00CB  24 F8                AND     AL,0F8H             ; IGNORE DRIVE SPECIFIER
00CD  3C C0                CMP     AL,0C0H             ; TEST FOR DRIVE READY TRANSITION
00CF  E1 EF                LOOPZ   STAT_LOOP           ; IF OKAY GET NEXT DRIVE READY INFO
00D1  0B C9                OR      CX,CX               ; ALL BYTES PICKED UP?
00D3  74 08                JZ      J7                  ; IF SO THEN OKAY
                  ;
                  ; NEC ERROR : ALL 4 DRIVE STATUSES NOT CORRECT AFTER RESET
                  ;
00D5  58                   POP     AX                  ; DISCARD ERROR ADDRESS ON STACK
00D6  80 0E 0041 R 20      OR      DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
00DB  EB 18                JMP     SHORT J8            ; EXIT
                  ;
                  ;----- SEND SPECIFY COMMAND TO NEC
                  ;
00DD                    J7:                            ; DRIVE_READY
00DD  B4 03                MOV     AH,SPECIFY          ; SPECIFY COMMAND
00DF  E8 022C R            CALL    NEC_OUTPUT          ; OUTPUT THE COMMAND
00E2  BB 0001              MOV     BX,1                ; FIRST BYTE PARM IN BLOCK
00E5  E8 0259 R            CALL    GET_PARM            ; TO THE NEC CONTROLLER
00E8  E8 022C R            CALL    NEC_OUTPUT
00EB  BB 0003              MOV     BX,3                ; SECOND BYTE PARM IN BLOCK
00EE  E8 0259 R            CALL    GET_PARM            ; TO THE NEC CONTROLLER
00F1  E8 022C R            CALL    NEC_OUTPUT
00F4  58                   POP     AX                  ; DISCARD ERROR RETURN ADDRESS

00F5                    J8:                            ; RESET_RET
00F5  C3                   RET                         ; RETURN TO CALLER
00F6                    DISK_RESET      ENDP

                  ;***********************************************************
                  ;----- DISKETTE STATUS ROUTINE
                  ;***********************************************************
00F6                    DISK_STATUS     PROC    NEAR
00F6  A0 0041 R            MOV     AL,DISKETTE_STATUS
00F9  88 46 0E             MOV     ALSAVE[BP],AL       ; MOVE STATUS IN AL SAVE AREA
00FC  C3                   RET
00FD                    DISK_STATUS     ENDP

                  ;***********************************************************
                  ;----- DISKETTE READ
                  ;***********************************************************
00FD                    DISK_READ       PROC    NEAR
00FD  E8 050B R            CALL    DSKT_CHANGE         ; CHECK FOR MEDIA CHANGE
0100  72 1F                JC      DSKT_ERR
0102  B0 46                MOV     AL,DMA_READ         ; READ COMMAND FOR DMA
0104  E8 02F1 R            CALL    DMA_SETUP           ; SET UP THE DMA
0107  72 18                JC      DSKT_ERR            ; JUMP IF ERROR
0109  C6 46 0F E6          MOV     AHSAVE[BP],READ_CMND ; RD COMMAND (AH SAVE)
010D  EB 41                JMP     SHORT RW_OPN        ; GO DO THE OPERATION
010F                    DISK_READ       ENDP

                  ;***********************************************************
                  ;----- DISKETTE VERIFY
```

```
               ;************************************************************
010F                    DISK_VERF       PROC    NEAR
010F  E8 050B R            CALL    DSKT_CHANGE          ; CHECK FOR MEDIA CHANGE
0112  72 0D                JC      DSKT_ERR
0114  B0 42                MOV     AL,DMA_VERIFY        ; VERIFY COMMAND FOR DMA
0116  E8 02F1 R            CALL    DMA_SETUP            ; SET UP THE DMA
0119  72 06                JC      DSKT_ERR
011B  C6 46 0F E6          MOV     AHSAVE[BP],READ_CMND ; RD COMMAND (AH SAVE)
011F  EB 2F                JMP     SHORT RW_OPN         ; GO DO THE OPERATION
0121                    DISK_VERF       ENDP

               ;************************************************************
               ; DISKETTE ERROR OCCURRED
               ;************************************************************
0121                    DSKT_ERR:
0121  C3                   RET                          ; RETURN TO MAIN ROUTINE

               ;************************************************************
               ;----- DISKETTE FORMAT
               ;************************************************************
0122                    DISK_FORMAT     PROC    NEAR
0122  E8 050B R            CALL    DSKT_CHANGE          ; CHECK FOR MEDIA CHANGE
0125  72 FA                JC      DSKT_ERR
0127  80 0E 003F R 80      OR      MOTOR_STATUS,WRITE_OP ; INDICATE WRITE OPERATION
012C  B0 4A                MOV     AL,DMA_WRITE         ; WILL WRITE TO THE DISKETTE
012E  E8 02F1 R            CALL    DMA_SETUP            ; SET UP THE DMA
0131  72 EE                JC      DSKT_ERR
0133  C6 46 0F 4D          MOV     AHSAVE[BP],FORMAT_CMND ; FORMAT COMMAND (AH SAVE)
0137  EB 17                JMP     SHORT RW_OPN         ; DO THE OPERATION
0139                    DISK_FORMAT     ENDP

               ;************************************************************
               ;----- DISKETTE WRITE ROUTINE
               ;************************************************************
0139                    DISK_WRITE      PROC    NEAR
0139  E8 050B R            CALL    DSKT_CHANGE          ; CHECK FOR MEDIA CHANGE
013C  72 E3                JC      DSKT_ERR
013E  80 0E 003F R 80      OR      MOTOR_STATUS,WRITE_OP ; INDICATE WRITE OPERATION
0143  B0 4A                MOV     AL,DMA_WRITE         ; DMA WRITE COMMAND
0145  E8 02F1 R            CALL    DMA_SETUP
0148  72 D7                JC      DSKT_ERR
014A  C6 46 0F C5          MOV     AHSAVE[BP],WRITE_CMND ; WRITE COMMAND (AH SAVE)
014E  EB 00                JMP     SHORT RW_OPN
0150                    DISK_WRITE      ENDP

               ;*****************************************************************
               ; RW_OPN
               ;       THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY
               ;       AND FORMAT OPERATIONS
               ;*****************************************************************

0150                    RW_OPN  PROC    NEAR
0150  E8 041B R            CALL    MOTOR_STARTUP        ; CHECK MOTOR STATE AND WAIT
                                                        ; FOR STARTUP IF NECESSARY
               ;----- DO THE SEEK OPERATION

0153  8B 4E 0A             MOV     CX,CXSAVE[BP]        ; GET TRACK/ SECTOR PARMS
0156  E8 0268 R            CALL    SEEK                 ; MOVE TO CORRECT TRACK

0159  C6 46 0E 00          MOV     ALSAVE[BP],0         ; SET NO SECTORS READ IN CASE OF ERROR
                                                        ; IN AL SAVE AREA
015D  72 79                JC      J17                  ; IF ERROR, THEN EXIT AFTER MOTOR OFF
               ;
               ; NEC_OUTPUT WILL POP RETURN ADDRESS AND RETURN TO THE 1ST LEVEL CALLER
               ;
015F  B8 0221 R            MOV     AX,OFFSET J20        ; DUMMY RETURN ON STACK FOR NEC_OUTPUT
0162  50                   PUSH    AX                   ; SO THAT IT WILL EXIT ROUTINE IF
                                                        ; ERROR

               ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER

0163  8A 66 0F             MOV     AH,AHSAVE[BP]        ; GET NEC COMMAND (AH SAVE)
0166  E8 022C R            CALL    NEC_OUTPUT           ; OUTPUT THE OPERATION COMMAND
0169  8A 66 01             MOV     AH,DHSAVE[BP]        ; GET THE CURRENT HEAD NUMBER
```

```
016C  D0 E4                 SAL    AH,1                 ; MOVE IT TO BIT 2
016E  D0 E4                 SAL    AH,1
0170  80 E4 04              AND    AH,4                 ; ISOLATE THAT BIT
0173  0A E2                 OR     AH,DL                ; OR IN THE DRIVE NUMBER
0175  E8 022C R             CALL   NEC_OUTPUT

             ;----- TEST FOR FORMAT COMMAND

0178  80 7E 0F 4D           CMP    AHSAVE[BP],FORMAT_CMND ; IS THIS A FORMAT OPERATION ?
017C  75 20                 JNE    J15                  ; NO. CONTINUE WITH R/W/V
             ;
             ; GET THE DATA FOR A FORMAT OPERATION
             ;
017E  BB 0007               MOV    BX,7                 ; GET THE
0181  E8 0259 R             CALL   GET_PARM             ; BYTES/SECTOR VALUE TO NEC
0184  E8 022C R             CALL   NEC_OUTPUT
0187  BB 0009               MOV    BX,9                 ; GET THE
018A  E8 0259 R             CALL   GET_PARM             ; SECTORS/TRACK VALUE TO NEC
018D  E8 022C R             CALL   NEC_OUTPUT
0190  BB 000F               MOV    BX,15                ; GET THE
0193  E8 0259 R             CALL   GET_PARM             ; GAP LENGTH VALUE TO NEC
0196  E8 022C R             CALL   NEC_OUTPUT
0199  BB 0011               MOV    BX,17                ; GET THE FILLER BYTE
019C  EB 30                 JMP    SHORT J16            ; TO THE CONTROLLER
             ;
             ; SEND THE DATA FOR A READ/WRITE/VERIFY OPERATION
             ;
019E                     J15:
019E  8A 66 0B              MOV    AH,CHSAVE[BP]        ; CYLINDER NUMBER (CH SAVE)
01A1  E8 022C R             CALL   NEC_OUTPUT
01A4  8A 66 01              MOV    AH,DHSAVE[BP]        ; HEAD NUMBER FROM STACK
01A7  E8 022C R             CALL   NEC_OUTPUT
01AA  8A 66 0A              MOV    AH,CLSAVE[BP]        ; SECTOR NUMBER   (CL SAVE)
01AD  E8 022C R             CALL   NEC_OUTPUT
01B0  BB 0007               MOV    BX,7                 ; BYTES/SECTOR PARM FROM BLOCK
01B3  E8 0259 R             CALL   GET_PARM             ; TO THE NEC
01B6  E8 022C R             CALL   NEC_OUTPUT
01B9  BB 0009               MOV    BX,9                 ; EOT PARM FROM BLOCK
01BC  E8 0259 R             CALL   GET_PARM             ; TO THE NEC
01BF  E8 022C R             CALL   NEC_OUTPUT
01C2  BB 000B               MOV    BX,11                ; GAP LENGTH PARM FROM BLOCK
01C5  E8 0259 R             CALL   GET_PARM             ; TO THE NEC
01C8  E8 022C R             CALL   NEC_OUTPUT
01CB  BB 000D               MOV    BX,13                ; DTL PARM FROM BLOCK
             ;
             ; COMPLETE SETUP TO NEC AND WAIT FOR INTERRUPT
             ;
01CE                     J16:                           ; RW_OPN_FINISH
01CE  E8 0259 R             CALL   GET_PARM             ; TO THE NEC
01D1  E8 022C R             CALL   NEC_OUTPUT
01D4  58                    POP    AX                   ; CAN NOW DISCARD THAT DUMMY
                                                        ; RETURN ADDRESS

             ;----- LET THE OPERATION HAPPEN

01D5  E8 0373 R             CALL   WAIT_INT             ; WAIT FOR THE INTERRUPT
01D8                     J17:                           ; MOTOR_OFF
01D8  72 49                 JC     J21                  ; LOOK FOR ERROR
01DA  E8 03B3 R             CALL   RESULTS              ; GET THE NEC STATUS
01DD  72 42                 JC     J20                  ; LOOK FOR ERROR

             ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER

01DF  FC                    CLD                         ; SET THE CORRECT DIRECTION
01E0  BE 0042 R             MOV    SI,OFFSET NEC_STATUS ; POINT TO STATUS FIELD
01E3  AC                    LODS   NEC_STATUS           ; GET ST0
01E4  24 C0                 AND    AL,0C0H              ; TEST FOR NORMAL TERMINATION
                            JFZ    J22                  ; OPN_OK
01E6  75 03                 JNZ    $+5                  ; IF NOT ZERO JUMP AROUND JUMP
01E8  EB 3E 90              JMP    J22                  ; ELSE TAKE A LONG JUMP
01EB  3C 40                 CMP    AL,040H              ; TEST FOR ABNORMAL TERMINATION
01ED  75 29                 JNZ    J18                  ; NOT ABNORMAL, BAD NEC

             ;----- ABNORMAL TERMINATION, FIND OUT WHY
```

```
01EF  AC                    LODS   NEC_STATUS          ; GET ST1
01F0  D0 E0                 SAL    AL,1                ; TEST FOR EOT FOUND
01F2  B4 04                 MOV    AH,RECORD_NOT_FND
01F4  72 24                 JC     J19                 ; RW_FAIL
01F6  D0 E0                 SAL    AL,1
01F8  D0 E0                 SAL    AL,1                ; TEST FOR CRC ERROR
01FA  B4 10                 MOV    AH,BAD_CRC
01FC  72 1C                 JC     J19                 ; RW_FAIL
01FE  D0 E0                 SAL    AL,1                ; TEST FOR DMA OVERRUN
0200  B4 08                 MOV    AH,BAD_DMA
0202  72 16                 JC     J19                 ; RW_FAIL
0204  D0 E0                 SAL    AL,1
0206  D0 E0                 SAL    AL,1                ; TEST FOR RECORD NOT FOUND
0208  B4 04                 MOV    AH,RECORD_NOT_FND
020A  72 0E                 JC     J19                 ; RW_FAIL
020C  D0 E0                 SAL    AL,1
020E  B4 03                 MOV    AH,WRITE_PROTECT    ; TEST FOR WRITE_PROTECT
0210  72 08                 JC     J19                 ; RW_FAIL
0212  D0 E0                 SAL    AL,1                ; TEST MISSING ADDRESS MARK
0214  B4 02                 MOV    AH,BAD_ADDR_MARK
0216  72 02                 JC     J19                 ; RW_FAIL

              ;----- NEC MUST HAVE FAILED

0218                      J18:                         ; RW-NEC-FAIL
0218  B4 20                 MOV    AH,BAD_NEC
021A                      J19:                         ; RW-FAIL
021A  08 26 0041 R          OR     DISKETTE_STATUS,AH
021E  E8 03F5 R             CALL   NUM_TRANS           ; HOW MANY WERE REALLY TRANSFERRED
0221                      J20:                         ; RW_ERR
0221  EB 08                 JMP    SHORT RW_EXIT       ; RETURN TO CALLER
0223                      J21:                         ; RW_ERR_RES
0223  E8 03B3 R             CALL   RESULTS             ; FLUSH THE RESULTS BUFFER
0226  EB 03                 JMP    SHORT RW_EXIT

              ;----- OPERATION WAS SUCCESSFUL

0228                      J22:                         ; OPN_OK
0228  E8 03F5 R             CALL   NUM_TRANS           ; HOW MANY GOT MOVED

022B                      RW_EXIT:
022B  C3                    RET
022C                      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
              ;------------------------------------------------------------------
022C                      NEC_OUTPUT      PROC    NEAR
022C  52                    PUSH   DX                  ; SAVE REGISTERS
022D  51                    PUSH   CX
022E  BA 03F4               MOV    DX,FDC_STATUS       ; STATUS PORT
0231  33 C9                 XOR    CX,CX               ; COUNT FOR TIME OUT
0233                      J23:
0233  EC                    IN     AL,DX               ; GET STATUS
0234  A8 40                 TEST   AL,DATA_READY       ; TEST DIRECTION BIT
0236  74 0E                 JZ     J25                 ; DIRECTION OK
0238  E2 F9                 LOOP   J23
023A                      J24:                         ; TIME_ERROR
```

```
023A  C6 06 0041 R 80       MOV     DISKETTE_STATUS,TIME_OUT
023F  59                    POP     CX
0240  5A                    POP     DX                  ; SET ERROR CODE AND RESTORE REGS
0241  83 C4 02              ADD     SP,2                ; DISCARD THE RETURN ADDRESS
0244  F9                    STC                         ; INDICATE ERROR TO CALLER
0245  C3                    RET
0246                     J25:
0246  33 C9                 XOR     CX,CX               ; RESET THE COUNT
0248                     J26:
0248  EC                    IN      AL,DX               ; GET THE STATUS
0249  A8 80                 TEST    AL,REQ_MASTER       ; IS IT READY
024B  75 04                 JNZ     J27                 ; YES, GO OUTPUT
024D  E2 F9                 LOOP    J26                 ; COUNT DOWN AND TRY AGAIN
024F  EB E9                 JMP     J24                 ; ERROR CONDITION
0251                     J27:                           ; OUTPUT
0251  8A C4                 MOV     AL,AH               ; GET BYTE TO OUTPUT
0253  B2 F5                 MOV     DL,0F5H             ; DATA PORT (3F5)
0255  EE                    OUT     DX,AL               ; OUTPUT THE BYTE
0256  59                    POP     CX                  ; RECOVER REGISTERS
0257  5A                    POP     DX
0258  C3                    RET                         ; CY = 0 FROM TEST INSTRUCTION
0259                     NEC_OUTPUT      ENDP
                ;----------------------------------------------------------------------
                ; GET_PARM
                ;       THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DSKT_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 --
                ;        BX = INDEX OF BYTE TO BE FETCHED * 2
                ; EXIT --
                ;        AH = THAT BYTE FROM BLOCK
                ;----------------------------------------------------------------------
0259                     GET_PARM        PROC    NEAR
0259  1E                    PUSH    DS                  ; SAVE SEGMENT
025A  2B C0                 SUB     AX,AX               ; ZERO TO AX
025C  8E D8                 MOV     DS,AX
                            ASSUME  DS:ABS0
025E  C5 36 0078 R          LDS     SI,DISK_POINTER     ; POINT TO BLOCK
0262  D1 EB                 SHR     BX,1                ; DIVIDE BX BY 2, AND SET FLAG
                                                        ; FOR EXIT
0264  8A 20                 MOV     AH,[SI+BX]          ; GET THE WORD
0266  1F                    POP     DS                  ; RESTORE SEGMENT
                            ASSUME  DS:DATA
0267  C3                    RET                         ; RETURN TO CALLER
0268                     GET_PARM        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 RECALIBRATED.
                ;       FOR DRIVES 0,1 NO SEEK PERFORMED IF ALREADY ON TRACK
                ; INPUT
                ;       (DL) = DRIVE TO SEEK ON
                ;       (CH) = TRACK TO SEEK TO
                ; OUTPUT
                ;       CY = 0 SUCCESS
                ;       CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
                ;       (AX,DI) DESTROYED
                ;----------------------------------------------------------------------
0268                     SEEK    PROC    NEAR
0268  B8 02EE R             MOV     AX,OFFSET J32       ; SET ERROR EXIT FOR NEC_OUTPUT
026B  50                    PUSH    AX
026C  B0 01                 MOV     AL,1                ; ESTABLISH MASK FOR RECAL TEST
026E  8B F9                 MOV     DI,CX               ; SAVE TRACK NUMBER (DI HIGH)
0270  8A CA                 MOV     CL,DL               ; GET DRIVE VALUE INTO CL
0272  D2 C0                 ROL     AL,CL               ; SHIFT IT BY THE DRIVE VALUE

0274  84 06 003E R          TEST    AL,SEEK_STATUS      ; TEST FOR RECAL REQUIRED
0278  75 1B                 JNZ     J28                 ; NO_RECAL
027A  08 06 003E R          OR      SEEK_STATUS,AL      ; TURN ON THE NO RECAL BIT IN FLAG
027E  B9 0002               MOV     CX,02H              ; # RECALS ATTEMPTS FOR 80 TRACKS
0281                     J27A:
0281  B4 07                 MOV     AH,RECALIBRATE      ; RECALIBRATE COMMAND
```

```
0283  E8 022C R           CALL    NEC_OUTPUT
0286  8A E2               MOV     AH,DL
0288  E8 022C R           CALL    NEC_OUTPUT            ; OUTPUT THE DRIVE NUMBER
028B  E8 034D R           CALL    CHK_STAT_2            ; GET THE INTERUPT AND SENSE INT STATU
028E  73 05               JNC     J28                   ; RECAL SUCCESSFUL
0290  E2 EF               LOOP    J27A                  ; RETRY IF MORE THAN 77 STEPS NEEDED
0292  58                  POP     AX                    ; DISCARD ERROR RETURN ADDRESS
0293  EB 59               JMP     SHORT J32             ; SEEK_ERROR

                ;----- DRIVE IS IN SYNC WITH CONTROLLER, SEEK TO TRACK

0295                    J28:
0295  C6 06 0041 R 00     MOV     DISKETTE_STATUS,0     ; CLEAR DISKETTE STATUS
029A  80 FA 01            CMP     DL,01                 ; CHECK FOR ABOVE DRIVE 1
029D  77 20               JA      J28_3
                ;
                ; CHECK FOR CURRENT TRACK = DESIRED TRACK
                ;
029F  E4 77               IN      AL,DSKT_CNTL
02A1  A8 80               TEST    AL,DSKT_NMI           ; IS DISKETTE CONTROL REG CORRECT?
02A3  74 1A               JZ      J28_3                 ; JUMP IF NOT
02A5  50                  PUSH    AX
02A6  24 E2               AND     AL,DSKT_NMI+FDC_PWR+DSKT_DEGATE+CNTL_SEL ; READ DR1 TRK CT
02A8  80 FA 00            CMP     DL,0
02AB  75 02               JNE     J28_2
02AD  0C 08               OR      AL,DR0_TRK_SEL        ; SET TRACK SENSE FOR DRIVE 0
02AF                    J28_2:
02AF  E6 77               OUT     DSKT_CNTL,AL          ; SELECT DRIVE # FOR TRACK SENSE
02B1  8B CF               MOV     CX,DI                 ; GET TRACK NUMBER IN CH
02B3  E4 77               IN      AL,DSKT_CNTL          ; READ TRACK POSITION
02B5  3A C5               CMP     AL,CH                 ; COMPARE TRACK ON WITH DESIRED TRACK
02B7  58                  POP     AX
02B8  E6 77               OUT     DSKT_CNTL,AL          ; RESTORE SENSE REGISTER
02BA  75 03               JNE     J28_3
                ;
                ;       ON SAME TRACK SO JUST EXIT WITH NO ERROR
                ;
02BC  F8                  CLC
02BD  EB 2F               JMP     SHORT   J32           ; EXIT

02BF  B4 0F             J28_3:  MOV     AH,SEEK_CMD     ; SEEK COMMAND TO NEC
02C1  E8 022C R           CALL    NEC_OUTPUT
02C4  8A E2               MOV     AH,DL                 ; DRIVE NUMBER
02C6  E8 022C R           CALL    NEC_OUTPUT
02C9  8B C7               MOV     AX,DI                 ; TRACK NUMBER
02CB  E8 022C R           CALL    NEC_OUTPUT
02CE  58                  POP     AX                    ; DISCARD ERROR RETURN ADDRESS
02CF  E8 034D R           CALL    CHK_STAT_2            ; GET ENDING INTERRUPT AND

                ;----- WAIT FOR HEAD SETTLE

02D2  9C                  PUSHF                         ; SAVE STATUS FLAGS
02D3  BB 0012             MOV     BX,18                 ; GET HEAD SETTLE PARAMETER
02D6  E8 0259 R           CALL    GET_PARM
02D9  80 FC 0F            CMP     AH,15                 ; CHECK FOR AT LEAST 15 MSEC
02DC  73 02               JAE     J29                   ; SET IT TO 15 IF NOT = OF ABOVE
02DE  B4 0F               MOV     AH,15
02E0                    J29:                            ; HEAD_SETTLE
02E0  0A E4               OR      AH,AH
02E2  74 09               JZ      J31                   ; EXIT LOOP IF NO WAIT
02E4  B9 0102             MOV     CX,MS_DELAY
02E7  E2 FE               LOOP    $
02E9  FE CC               DEC     AH
02EB  EB F3               JMP     SHORT J29             ; DO IT SOME MORE
02ED                    J31:
02ED  9D                  POPF

02EE                    J32:                            ; SEEK_ERROR
02EE  8B CF               MOV     CX,DI                 ; RESTORE CX
02F0  C3                  RET                           ; RETURN TO CALLER
02F1                    SEEK    ENDP
```

```
                 ;-----------------------------------------------------------------
                 ; DMA_SETUP
                 ;       THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
                 ; INPUT
                 ;       (AL) = MODE BYTE FOR THE DMA
                 ;       (ES) - SEGMENT TO READ/WRITE THE DATA
                 ;       (BP) - STACK POINTER (TO GET INPUT PARAMETERS)
                 ; OUTPUT
                 ;       (AX,CX) DESTROYED
                 ;-----------------------------------------------------------------
02F1                     DMA_SETUP       PROC    NEAR
02F1  FA                    CLI                         ; NO MORE INTERRUPTS
02F2  E6 0C                 OUT     DMA+12,AL           ; SET THE FIRST/LAST F/F
02F4  EB 00                 JMP     $+2                 ; DELAY FOR DMA CONTROLLER
02F6  E6 0B                 OUT     DMA+11,AL           ; OUTPUT THE MODE BYTE
02F8  50                    PUSH    AX                  ; SAVE COMMAND
02F9  8C C0                 MOV     AX,ES               ; GET THE ES VALUE
02FB  B1 04                 MOV     CL,4                ; SHIFT COUNT
02FD  D3 C0                 ROL     AX,CL               ; ROTATE LEFT
02FF  8A E8                 MOV     CH,AL               ; GET HIGHEST NYBLE OF ES TO CH
0301  24 F0                 AND     AL,0F0H             ; ZERO THE LOW NYBBLE FROM SEGMENT
0303  03 46 0C              ADD     AX,BXSAVE[BP]       ; ADD ADDRESS OFFSET
0306  73 02                 JNC     J33                 ; TEST FOR CARRY FROM ADDITION
0308  FE C5                 INC     CH                  ; CARRY MEANS HIGH 4 BITS MUST BE INC
030A                     J33:
030A  50                    PUSH    AX                  ; SAVE START ADDRESS
030B  E6 04                 OUT     DMA+4,AL            ; OUTPUT LOW ADDRESS
030D  8A C4                 MOV     AL,AH
030F  E6 04                 OUT     DMA+4,AL            ; OUTPUT HIGH ADDRESS
0311  8A C5                 MOV     AL,CH               ; GET HIGH 4 BITS
0313  24 0F                 AND     AL,0FH
0315  E6 81                 OUT     DMA_PAGE2,AL        ; OUTPUT THE HIGH 4 BITS TO

                 ;----- DETERMINE COUNT

0317  8A 66 0E              MOV     AH,ALSAVE[BP]       ; NUMBER OF SECTORS (AL SAVE)
031A  2A C0                 SUB     AL,AL               ; TIMES 256 INTO AX
031C  D1 E8                 SHR     AX,1                ; SECTORS * 128 INTO AX
031E  50                    PUSH    AX
031F  BB 0006               MOV     BX,6                ; GET THE BYTES/SECTOR PARM
0322  E8 0259 R             CALL    GET_PARM
0325  8A CC                 MOV     CL,AH               ; USE AS SHIFT COUNT (0=128, 1=256 ETC)
0327  58                    POP     AX
0328  D3 E0                 SHL     AX,CL               ; MULTIPLY BY CORRECT AMOUNT
032A  48                    DEC     AX                  ; -1 FOR DMA VALUE
032B  50                    PUSH    AX                  ; SAVE COUNT VALUE
032C  E6 05                 OUT     DMA+5,AL            ; LOW BYTE OF COUNT
032E  8A C4                 MOV     AL,AH
0330  E6 05                 OUT     DMA+5,AL            ; HIGH BYTE OF COUNT
0332  FB                    STI                         ; INTERRUPTS BACK ON
0333  59                    POP     CX                  ; RECOVER COUNT VALUE
0334  58                    POP     AX                  ; RECOVER ADDRESS VALUE
0335  03 C1                 ADD     AX,CX               ; ADD, TEST FOR 64K OVERFLOW
0337  B0 02                 MOV     AL,2                ; MODE FOR 8237
0339  E6 0A                 OUT     DMA+10,AL           ; INITIALIZE THE DISKETTE CHANNEL
033B  58                    POP     AX                  ; RESTORE COMMAND
033C  73 0E                 JNC     DMA_OUT

033E  3C 42                 CMP     AL,42H              ; NON-DMA OPERATION?
0340  74 0A                 JE      DMA_OUT             ; IF SO THEN NO BOUNDRY ERROR

0342  C6 06 0041 R 09       MOV     DISKETTE_STATUS,DMA_BOUNDARY ; SET DMA BOUNDRY ERROR
0347  C6 46 0E 00           MOV     BYTE PTR [BP+14],0  ; NO SECTORS TRANSFERRED (AL SAVE)
034B  F9                    STC                         ; SET CARRY TO INDICATE ERROR
034C                     DMA_OUT:
034C  C3                    RET                         ; RETURN TO CALLER,
                                                        ; CARRY SET BY ABOVE IF ERROR
034D                     DMA_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
               ;------------------------------------------------------------------
034D                   CHK_STAT_2      PROC    NEAR
034D  E8 0373 R           CALL    WAIT_INT            ; WAIT FOR THE INTERRUPT
0350  72 19               JC      J34                 ; IF ERROR, RETURN IT
0352  B8 036B R           MOV     AX,OFFSET J34
0355  50                  PUSH    AX                  ; SET ERROR RETURN ADDRESS

0356  B4 08               MOV     AH,READ_INT_STATUS  ; SENSE INTERRUPT STATUS COMMAND
0358  E8 022C R           CALL    NEC_OUTPUT
035B  E8 03B3 R           CALL    RESULTS             ; READ IN THE RESULTS
035E  72 0B               JC      J34                 ; CHK2_RETURN
0360  58                  POP     AX                  ; DISCARD ERROR RETURN ADDRESS
0361  A0 0042 R           MOV     AL,NEC_STATUS       ; GET THE FIRST STATUS BYTE
0364  24 60               AND     AL,060H             ; ISOLATE THE BITS
0366  3C 60               CMP     AL,060H             ; TEST FOR CORRECT VALUE
0368  74 02               JZ      J35                 ; IF ERROR, GO MARK IT
036A  F8                  CLC                         ; GOOD RETURN
036B                   J34:
036B  C3                  RET                         ; RETURN TO CALLER
036C                   J35:                           ; CHK2_ERROR
036C  80 0E 0041 R 40     OR      DISKETTE_STATUS,BAD_SEEK
0371  F9                  STC                         ; ERROR RETURN CODE
0372  C3                  RET
0373                   CHK_STAT_2      ENDP
               ;------------------------------------------------------------------
               ; WAIT_INT
               ;      THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR. A TIME OUT
               ;      ROUTINE TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE
               ;      RETURNED IF THE DRIVE IS NOT READY.
               ; INPUT
               ;      NONE
               ; OUTPUT
               ;      CY = 0 SUCCESS
               ;      CY = 1 FAILURE -- DISKETTE_STATUS IS SET ACCORDINGLY
               ;      (AX,BX) DESTROYED
               ;------------------------------------------------------------------
0373                   WAIT_INT        PROC    NEAR
0373  FB                  STI                         ; TURN ON INTERRUPTS, JUST IN CASE
0374  51                  PUSH    CX                  ; SAVE REGISTERS
0375  F8                  CLC
               ;
               ; CALL DEVICE BUSY SERVICE ROUTINE
               ;
0376  B8 9001              MOV     AX,09001H           ; CALL DEVICE BUSY (DISKETTE)
0379  CD 15               INT     15H
037B  72 11               JC      J36_A               ; JUMP IF TIMEOUT OCCURRED
               ;
               ; BIOS WILL PASS CONTROL HERE WITH CARRY FOR TIMEOUT OR WITH NO CARRY FOR
               ; OPERATION COMPLETE.
               ;
037D  B3 04               MOV     BL,4                ; WAIT FOR 2 SECONDS
037F  2B C9               SUB     CX,CX               ; CLEAR THE COUNTER
0381                   J36:
0381  F6 06 003E R 80     TEST    SEEK_STATUS,INT_FLAG ; TEST FOR INTERRUPT COMPLETE
0386  75 0C               JNZ     J37                 ; JUMP IF INTERRUPT COMPLETE
0388  E2 F7               LOOP    J36                 ; INNER LOOP COUNT
038A  FE CB               DEC     BL
038C  75 F3               JNZ     J36                 ; OUTER LOOP COUNT
               ;
               ; NO INTERRUPT RECEIVED TIMEOUT ERROR
               ;
038E                   J36_A:
```

```
038E  80 0E 0041 R 80       OR      DISKETTE_STATUS,TIME_OUT; NO INTERRUPT OCCURRED ERROR
0393  F9                    STC                         ; ERROR RETURN
0394                     J37:
0394  9C                    PUSHF                       ; SAVE CURRENT CARRY
0395  80 26 003E R 7F       AND     SEEK_STATUS,NOT INT_FLAG ; TURN OFF INTERRUPT FLAG
039A  9D                    POPF                        ; RECOVER CARRY
039B  59                    POP     CX
039C  C3                    RET                         ; GOOD RETURN CODE COMES
                                                        ; FROM TEST INST
039D                     WAIT_INT       ENDP

               ;---------------------------------------------------------
               ; DSKT_INTE
               ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT
               ;INPUT
               ;       NONE
               ; OUTPUT
               ;       THE INTERRUPT FLAG IS SET IS SEEK_STATUS
               ;---------------------------------------------------------

039D                     DSKT_INTE      PROC    FAR
039D  1E                    PUSH    DS
039E  50                    PUSH    AX
039F  E8 0000 E             CALL    DDS                 ; SET UP DATA SEGMENT
03A2  80 0E 003E R 80       OR      SEEK_STATUS,INT_FLAG
03A7  B0 20                 MOV     AL,EOI              ; END OF INTERRUPT MARKER
03A9  E6 20                 OUT     INTA00,AL           ; INTERRUPT CONTROL PORT
03AB  B8 9101               MOV     AX,09101H           ; SIGNAL DEVICE OPERATION COMPLETE
03AE  CD 15                 INT     15H
03B0  58                    POP     AX
03B1  1F                    POP     DS                  ; RECOVER SYSTEM
03B2  CF                    IRET                        ; RETURN FROM INTERRUPT
03B3                     DSKT_INTE      ENDP

               ;-------------------------------------------------------------------------
               ; RESULTS
               ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER HAS
               ;       TO SAY FOLLOWING AN INTERRUPT.
               ; 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,BX,SI) DESTROYED
               ;
               ;-------------------------------------------------------------------------
03B3                     RESULTS PROC    NEAR
03B3  FC                    CLD
03B4  BE 0042 R             MOV     SI,OFFSET NEC_STATUS ; POINTER TO DATA AREA
03B7  51                    PUSH    CX                  ; SAVE COUNTER
03B8  52                    PUSH    DX
03B9  B3 07                 MOV     BL,7                ; MAX STATUS BYTES

               ;----- WAIT FOR REQUEST FOR MASTER

03BB                     J38:                           ; INPUT_LOOP
03BB  33 C9                 XOR     CX,CX               ; COUNTER
03BD  BA 03F4               MOV     DX,FDC_STATUS       ; STATUS PORT
03C0                     J39:                           ; WAIT FOR MASTER
03C0  EC                    IN      AL,DX               ; GET STATUS
03C1  A8 80                 TEST    AL,REQ_MASTER       ; MASTER READY
03C3  75 0B                 JNZ     J40A                ; TEST_DIR
03C5  E2 F9                 LOOP    J39                 ; WAIT_MASTER
03C7  80 0E 0041 R 80       OR      DISKETTE_STATUS,TIME_OUT
03CC                     J40:                           ; RESULTS_ERROR
03CC  F9                    STC                         ; SET ERROR RETURN
03CD  5A                    POP     DX
03CE  59                    POP     CX
03CF  C3                    RET

               ;----- TEST THE DIRECTION BIT
```

```
03D0                    J40A:
03D0  EC                   IN      AL,DX              ; GET STATUS REG AGAIN
03D1  A8 40                TEST    AL,DATA_READY      ; TEST DIRECTION BIT
03D3  75 07                JNZ     J42                ; OK TO READ STATUS
03D5                    J41:                          ; NEC_FAIL
03D5  80 0E 0041 R 20      OR      DISKETTE_STATUS,BAD_NEC
03DA  EB F0                JMP     J40                ; RESULTS_ERROR

              ;----- READ IN THE STATUS

03DC                    J42:                          ; INPUT_STAT
03DC  42                   INC     DX                 ; POINT AT DATA PORT
03DD  EC                   IN      AL,DX              ; GET THE DATA
03DE  88 04                MOV     [SI],AL            ; STORE THE BYTE
03E0  46                   INC     SI                 ; INCREMENT THE POINTER
03E1  B9 000A              MOV     CX,10              ; LOOP TO KILL TIME FOR NEC
03E4  E2 FE             J43:    LOOP    J43
03E6  4A                   DEC     DX                 ; POINT AT STATUS PORT
03E7  EC                   IN      AL,DX              ; GET STATUS
03E8  A8 10                TEST    AL,FDC_BUSY        ; TEST FOR NEC STILL BUSY
03EA  74 06                JZ      J44                ; RESULTS DONE
03EC  FE CB                DEC     BL                 ; DECREMENT THE STATUS COUNTER
03EE  75 CB                JNZ     J38                ; GO BACK FOR MORE
03F0  EB E3                JMP     J41                ; CHIP HAS FAILED

              ;----- RESULT OPERATION IS DONE

03F2                    J44:
03F2  5A                   POP     DX
03F3  59                   POP     CX                 ; RECOVER REGISTERS
03F4  C3                   RET                        ; GOOD RETURN CODE FROM TEST INST
03F5                    RESULTS ENDP

              ;-------------------------------------------------------------
              ; NUM_TRANS
              ;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
              ;       WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE
              ; INPUT
              ;       (BP) = POINTER TO ORIGINAL STACK ENTRIES
              ; OUTPUT
              ;       (BP+14) = AL SAVE AREA ON STACK = # ACTUALLY TRANSFERRED
              ;       NO OTHER REGISTERS MODIFIED
              ;-------------------------------------------------------------
03F5                    NUM_TRANS       PROC    NEAR
03F5  BB 0008              MOV     BX,8               ; SECTORS/TRACK OFFSET TO DL
03F8  E8 0259 R            CALL    GET_PARM           ; AH = SECTORS/TRACK
03FB  A0 0047 R            MOV     AL,NEC_STATUS+5    ; GET ENDING SECTOR
03FE  8A 7E 01             MOV     BH,DHSAVE[BP]      ; BH = STARTING HEAD #
0401  3A 3E 0046 R         CMP     BH,NEC_STATUS+4    ; GET HEAD ENDED UP ON
0405  75 0B                JNZ     DIF_HD             ; IF ON SAME HEAD, THEN NO ADJUS
0407  8A 3E 0045 R         MOV     BH,NEC_STATUS+3    ; GET TRACK ENDED UP ON
040B  3A 7E 0B             CMP     BH,CHSAVE[BP]      ; SEE IF TRACK SWITCH
040E  74 04                JZ      SAME_TRK           ; IF SAME TRACK NO INCREASE
0410  02 C4                ADD     AL,AH              ; ADD SECTORS/TRACK
0412                    DIF_HD:
0412  02 C4                ADD     AL,AH              ; ADD SECTORS/TRACK
0414                    SAME_TRK:
0414  2A 46 0A             SUB     AL,CLSAVE[BP]      ; SUBTRACT START FROM END SECTORS
0417  88 46 0E             MOV     ALSAVE[BP],AL      ; SAVE RESULTS IN STACK SAVE AREA
041A  C3                   RET
041B                    NUM_TRANS       ENDP
```

```
                ;-----------------------------------------------------------------------
                ; MOTOR_STARTUP
                ;
                ; FUNCTION: TO CHECK FOR MOTOR STATE, TURN ON MOTOR OF REQUESTED DRIVE
                ;           AND WAIT THE NECESSARY STARTUP TIME BEFORE RETURNING
                ;           TO CALLER.
                ;
                ; INPUT PARAMETERS:
                ;          DL = 0-3 (DISKETTE DRIVE NUMBER)
                ; OUTPUT PARAMETERS:
                ;          MOTOR_STATUS FLAGS UPDATED
                ;          MOTOR OF DRIVE IS TURNED ON
                ; REGISTERS MODIFIED: AX,BX,CX,DH
                ;-----------------------------------------------------------------------
041B                    MOTOR_STARTUP   PROC    NEAR
                ;
                ;----- TURN ON THE MOTOR AND SELECT THE DRIVE
                ;
041B  B6 00                MOV     DH,0                 ; SET FIRST TIME THROUGH FLAG

041D                    MOT_01:
                ;
                ; GET MOTOR STARTUP TIME
                ;
041D  BB 0014              MOV     BX,20                ; GET THE MOTOR WAIT
0420  E8 0259 R            CALL    GET_PARM             ; CH HAS MOTOR START UP DELAY
0423  80 FC 04             CMP     AH,04                ; MUST BE MINIMUM OF 500 MSECS
0426  73 02                JAE     MOT_011              ; JUMP IF OKAY
0428  B4 04                MOV     AH,04                ; DEFAULT TIME TO 500 MSECS
042A                    MOT_011:
042A  8A EC                MOV     CH,AH                ; SAVE MOTOR START UP DELAY
042C  8A CA                MOV     CL,DL                ; GET DRIVE NUMBER AS SHIFT COUNT
042E  B0 01                MOV     AL,1                 ; MASK FOR DETERMINING MOTOR BIT
0430  D2 E0                SAL     AL,CL                ; SHIFT THE MASK BIT

0432  FA                   CLI                          ; NO INTERRUPTS WHILE DETERMINING
                                                        ; MOTOR STATUS
0433  84 06 003F R         TEST    AL,MOTOR_STATUS      ; TEST THAT MOTOR FOR OPERATING
0437  75 1B                JNZ     MOT_02               ; IF RUNNING GO CHECK TIME
                ;
                ; TURN ON MOTOR
                ;
0439  80 26 003F R F0      AND     MOTOR_STATUS,0F0H    ; TURN OFF ALL MOTOR BITS
043E  08 06 003F R         OR      MOTOR_STATUS,AL      ; TURN ON THE CURRENT MOTOR
0442  B0 10                MOV     AL,10H               ; MASK BIT
0444  8A CA                MOV     CL,DL                ; GET DRIVE NUMBER AS SHIFT COUNT
0446  D2 E0                SAL     AL,CL                ; DEVELOP BIT MASK FOR MOTOR ENABLE
0448  0A C2                OR      AL,DL                ; GET DRIVE SELECT BITS IN
044A  0C 0C                OR      AL,FDC_DMA_ENAB+FDC_RUN ; NO RESET, ENABLE DMA/INT
044C  52                   PUSH    DX                   ; SAVE REG
044D  BA 03F2              MOV     DX,DRIVE_CNTL        ; CONTROL PORT ADDRESS
0450  EE                   OUT     DX,AL
0451  5A                   POP     DX                   ; RECOVER REGISTERS
0452  EB 07                JMP     SHORT MOT_04         ; GO DELAY FOR STARTUP
                ;
                ; CHECK TO SEE IF MOTOR ON LONG ENOUGH
                ;
0454                    MOT_02:
0454  F6 06 003F R 20      TEST    MOTOR_STATUS,MOTOR_OK ; CHECK FOR MOTOR RUNNING LONG ENOUGH
0459  75 24                JNZ     MOT_08               ; IF MOTOR OKAY THEN EXIT
                ;
                ; CH HAS TIME TO WAIT IN 1/8 SECONDS
                ;
045B                    MOT_04:
045B  C6 06 0040 R FF      MOV     MOTOR_COUNT,0FFH     ; SET LONG MOTOR DELAY
0460  FB                   STI                          ; INTERRUPTS BACK ON
0461  0A ED                OR      CH,CH                ; DON'T WAIT IF NO WAIT SET
0463  74 1A                JZ      MOT_08
0465  0A F6                OR      DH,DH                ; CHECK FOR FIRST TIME THROUGH WAIT
0467  75 08                JNZ     MOT_06               ; IF NOT FIRST TIME BYPASS OP_SYS HOOK
                ;
                ; NOTIFY OPERATING SYSTEM OF WAIT FOR MOTOR STARTUP
                ;
0469  F8                   CLC                          ; RESET TIMEOUT INDICATOR
```

```
046A  B8 90FD               MOV     AX,90FDH           ; SET WAIT ON DISKETTE MOTOR
046D  CD 15                 INT     15H                ; SLEEP OR DO OTHER WORK
046F  72 0E                 JC      MOT_08             ; BYPASS TIME DELAY IF TIMEOUT
               ;
               ; TIME DELAY LOOP TO WAIT FOR MOTOR STARTUP
               ;
0471                     MOT_06:
0471  8A E5                 MOV     AH,CH              ; MOVE TIME IN AH
0473  0A E4                 OR      AH,AH              ; TEST FOR NO WAIT
0475                     MOT_07:                       ; TEST_WAIT_TIME
0475  74 08                 JZ      MOT_08             ; EXIT WITH TIME EXPIRED
0477  2B C9                 SUB     CX,CX              ; SET UP 1/8 SECOND LOOP TIME
0479  E2 FE                 LOOP    $                  ; WAIT FOR THE REQUIRED TIME
047B  FE CC                 DEC     AH                 ; DECREMENT TIME VALUE
047D  EB F6                 JMP     SHORT MOT_07       ; ARE WE DONE YET
               ;
               ; MOTOR IS RUNNING SO CONTINUE WITH OPERATION
               ;
047F                     MOT_08:
047F  C6 06 0040 R FF       MOV     MOTOR_COUNT,0FFH   ; SET LARGE COUNT AFTER WAIT
0484  B6 01                 MOV     DH,01              ; SET 2ND TIME THROUGH FLAG
0486  F6 06 003F R 0F       TEST    MOTOR_STATUS,0FH   ; TEST FOR MOTORS STILL ON
048B  74 90                 JZ      MOT_01             ; IF NOT OKAY THEN GO BACK AND RESTART
               ;
               ; MOTORS STILL ON AFTER WAIT
               ;
048D  80 0E 003F R 20       OR      MOTOR_STATUS,MOTOR_OK ; SET MOTOR ON LONG ENOUGH FLAG
0492  FB                    STI                        ; ALLOW INTERRUPTS
0493  C3                    RET
0494                     MOTOR_STARTUP ENDP

               ;-----------------------------------------------------------------
               ; R E A D  D I S K E T T E  D R I V E  P A R A M E T E R S  R O U T I N E
               ;-----------------------------------------------------------------
0494                     DSKT_RDPARM     PROC    NEAR
0494  E8 0586 R            CALL    GET_DRV_CONFIG      ; GET DRIVE CONFIGURATION IN AL
0497  2B FF                SUB     DI,DI               ; CLEAR REGISTERS
0499  2B C9                SUB     CX,CX
049B  2B D2                SUB     DX,DX
049D  8E C2                MOV     ES,DX               ; CLEAR ES
049F  8A E6                MOV     AH,DH               ; CLEAR AH
04A1  3C 00                CMP     AL,NO_DRIVE         ; DRIVE PRESENT?
04A3  74 1D                JE      RDPARM_02
04A5  BE 0000 E            MOV     SI,OFFSET PARMS_TPI135 ; SET  720K  DRIVE
04A8  3C 03                CMP     AL,TPI_135
04AA  74 07                JE      RDPARM_01
04AC  BE 0000 E            MOV     SI,OFFSET PARMS_TPI48 ; SET 360K DRIVE
04AF  3C 01                CMP     AL,TPI_48
04B1  75 0F                JNE     RDPARM_02           ; IF NOT VALID SET NO DRIVE
04B3                     RDPARM_01:
04B3  2E: 8E 04            MOV     ES,CS:[SI]          ; SETUP SEG TO DSKT_PARMS
04B6  2E: 8B 7C 02         MOV     DI,CS:[SI+2]        ; POINTER TO PARMS
04BA  2E: 8B 4C 04         MOV     CX,CS:[SI+4]        ; TRACKS AND SECTORS
04BE  2E: 8B 54 06         MOV     DX,CS:[SI+6]        ; GET HEADS
04C2                     RDPARM_02:
04C2  89 7E 04             MOV     DISAVE[BP],DI       ; MODIFY REGISTERS ON STACK
04C5  89 4E 0A             MOV     CXSAVE[BP],CX
04C8  89 46 0C             MOV     BXSAVE[BP],AX       ; SET DRIVE TYPE IN BL
04CB  2B C0                SUB     AX,AX
04CD  89 46 0E             MOV     AXSAVE[BP],AX       ; CLEAR AL
04D0  8A 16 0010 R         MOV     DL,BYTE PTR EQUIP_FLAG ; GET LOW BYTE OF EQUIPMENT
04D4  B1 06                MOV     CL,6                ; GET # DISKETTES IN LOW
04D6  D2 EA                SHR     DL,CL               ; BITS
04D8  FE C2                INC     DL                  ; CORRECT FOR AT LEAST 1 DRIVE
04DA  89 56 00             MOV     DXSAVE[BP],DX       ; SAVE # HEADS AND # DRIVES
04DD  C6 06 0041 R 00      MOV     DISKETTE_STATUS,0   ; GET RETURN CODE
04E2  C3                   RET
04E3                     DSKT_RDPARM     ENDP
```

```
               ;------------------------------------------------------------------
               ; DSKT_READDASD
               ;------------------------------------------------------------------
               ;
               ; INPUT PARAMETERS:
               ;       AH = 15H
               ;       DL = DRIVE NUMBER (0-3)
               ; OUTPUT PARAMETERS:
               ;      AH = 00 - NO DRIVE PRESENT
               ;           01 - DISKETTE DRVIE WITH NO CHANGE LINE SUPPORT INSTALLED
               ;           02 - DISKETTE DRIVE WITH CHANGE LINE SUPPORT INSTALLED
               ;           03 - FIXED DISK
               ;           DISKETTE_STATUS = 0
               ;------------------------------------------------------------------
04E3                    DSKT_RDDASD     PROC    NEAR
04E3  80 0E 0016 R 04    OR      BIOS_STATUS,DCL_SUPPORTED ; SET CHANGE LINE
                                                      ; SUPPORTED FLAG
04E8  C6 06 0041 R 00    MOV     DISKETTE_STATUS,0     ; SET GOOD RETURN CODE
04ED  E8 0586 R          CALL    GET_DRV_CONFIG        ; GET DRIVE CONFIGURATION
04F0  B4 00              MOV     AH,0                  ; CLEAR RETURN CODE
04F2  3C 00              CMP     AL,NO_DRIVE           ; CHECK FOR DRIVE PRESENT
04F4  74 0B              JE      DASD_EXIT             ; JUMP IF NOT PRESENT
04F6  3C 03              CMP     AL,TPI_135            ; 3 1/2 DRIVES?
04F8  74 05              JE      DASD_01               ; YES THEN JUMP
               ;
               ; SET NO CHANGE LINE AVAILABLE
               ;
04FA  B4 01              MOV     AH,01                 ; SET NO CHANGE LINE AVAIL
04FC  EB 03 90           JMP     DASD_EXIT
               ;
               ; SET CHANGE LINE AVAILABLE
               ;
04FF                    DASD_01:
04FF  B4 02              MOV     AH,02                 ; CHANGE LINE AVAILABLE
0501                    DASD_EXIT:
0501  88 66 0F           MOV     AHSAVE[BP],AH         ; SAVE VALUE IN AH SAVE AREA
0504  83 C4 02           ADD     SP,2                  ; THROW AWAY RETURN ADDRESS
0507  F8                 CLC
0508  E9 0071 R          JMP     DIO_3                 ; GO BACK TO MAIN ROUTINE
050B                    DSKT_RDDASD     ENDP

               ;*************************************************************
               ; D I S K E T T E   C H A N G E   L I N E   S T A T U S   R O U T I N E
               ;*************************************************************
050B                    DSKT_CHANGE     PROC    NEAR
050B  E8 0586 R            CALL    GET_DRV_CONFIG      ; GET DRIVE CONFIGURATION
050E  0A C0                OR      AL,AL               ; CHECK FOR DRIVE NO THERE
0510  74 41                JZ      DCL_06              ; IF SO THEN SET TIMEOUT STATUS
0512  3C 03                CMP     AL,TPI_135          ; CHECK FOR 3.5
0514  75 63                JNE     DCL_10              ; JUMP IF NO CHANGE LINE
0516  E8 041B R            CALL    MOTOR_STARTUP       ; TURN ON MOTOR AND SELECT
               ;
               ; CHECK CHANGE LINE
               ;
0519                    DCL_01:
0519  52                   PUSH    DX
051A  BA 03F4              MOV     DX,FDC_STATUS       ; ENSURE CONTROLLER IS ON
051D  EC                   IN      AL,DX
051E  BA 03F7              MOV     DX,DRIVE_SENSE      ; DIGITAL INPUT PORT
0521  EC                   IN      AL,DX
0522  5A                   POP     DX                  ; RESTORE DRIVE NUMBER
0523  A8 80                TEST    AL,CHG_LINE         ; TEST FOR DISKETTE CHANGE
0525  75 0F                JNZ     DCL_03              ; JUMP IF ACTIVE
0527  F6 06 0016 R 02      TEST    BIOS_STATUS,FORCE_DCL ; FORCE CHANGE ERROR
052C  75 2C                JNZ     DCL_07              ; AFTER RESUME? - JUMP IF YES
               ;
               ; CHANGE LINE NOT ACTIVE
               ;
052E                    DCL_02:
052E  C6 06 0041 R 00      MOV     DISKETTE_STATUS,0   ; SET GOOD RETURN
0533  F8                   CLC                         ; RESET ERROR FLAG
0534  EB 49                JMP     SHORT DCL_EXIT      ; RETURN
               ;
               ; CHANGE LINE ACTIVE
               ; SEEK TO 1 THEN TO 0 TO RESET CHANGE LINE
               ;
0536                    DCL_03:
0536  B5 01                MOV     CH,01H              ; SET UP TO SEEK TO TRACK 1
0538  E8 0268 R            CALL    SEEK                ; TO RESET CHANGE LINE
053B  72 07                JC      DCL_04              ; JUMP IF ERROR ON THE SEEK
053D  B5 00                MOV     CH,00H              ; NOW SET TO SEEK TO ZERO
053F  E8 0268 R            CALL    SEEK                ; DO THE SEEK
0542  73 05                JNC     DCL_05
               ;
               ; SEEKS FAILED SO SET RECAL REQUIRED
               ;
0544                    DCL_04:
0544  C6 06 003E R 00      MOV     SEEK_STATUS,0       ; SEEK FAILED RECAL REQUIRED
               ;
               ; CHECK TO SEE IF CHANGE LINE WAS SUCCESSFULLY RESET
               ;
0549                    DCL_05:
0549  52                   PUSH    DX
054A  BA 03F7              MOV     DX,DRIVE_SENSE      ; DIGITAL INPUT PORT
054D  EC                   IN      AL,DX
054E  5A                   POP     DX                  ; RESTORE DRIVE NUMBER

054F  A8 80                TEST    AL,CHG_LINE         ; TEST FOR DISKETTE CHANGE
0551  74 07                JZ      DCL_07              ; IF RESET THEN MEDIA CHANGE
               ;
               ; CHANGE LINE DID NOT RESET SO SET TIMEOUT ERROR
               ;
0553                    DCL_06:
0553  C6 06 0041 R 80      MOV     DISKETTE_STATUS,TIME_OUT ; NO DISKETTE IN DRIVE
0558  EB 17                JMP     SHORT DCL_09        ; EXIT WITH TIMEOUT ERROR
               ;
               ; MEDIA CHANGE WAS ACTIVE, AND RESET SUCCESSFULLY, IF CHANGE LINE NOT
               ; SUPPORTED AND NO READ DCL STATUS DON'T REPORT THE ERROR
               ;
055A                    DCL_07:
055A  F6 06 0016 R 04      TEST    BIOS_STATUS,DCL_SUPPORTED ; CHANGE LINE SUPPORTED
055F  75 06                JNZ     DCL_08              ; JUMP IF YES
0561  80 7E 0F 16          CMP     AHSAVE[BP],16H      ; FUNCTION = READ DCL STATUS?
0565  75 C7                JNE     DCL_02              ; NO THEN INDICATE INACTIVE

0567                    DCL_08:
0567  80 26 0016 R FD      AND     BIOS_STATUS,NOT FORCE_DCL ; RESET RESUME FLAG
056C  C6 06 0041 R 06      MOV     DISKETTE_STATUS,MEDIA_CHANGE ; SET RETURN CODE

0571                    DCL_09:
0571  80 26 003F R DF      AND     MOTOR_STATUS,NOT MOTOR_OK ; ACTIVATE STARTUP DELAY
                                                       ; ON NEXT OPERATION
0576  F9                   STC                         ; SET INTERNAL ERROR FLAG
0577  EB 06                JMP     SHORT DCL_EXIT      ; RETURN TO CALLER
               ;
               ; NO CHANGE LINE AVAILABLE
               ; (NOT AN INTERNAL FLAGGED ERROR)
               ;
0579                    DCL_10:
0579  C6 06 0041 R 06      MOV     DISKETTE_STATUS,MEDIA_CHANGE ; DEFAULT TO DISKETTE CHG
057E  F8                   CLC                         ; RESET INTERNAL ERROR FLAG

057F                    DCL_EXIT:
057F  C3                   RET                         ; RETURN TO CALLER
0580                    DSKT_CHANGE     ENDP

               ;***********************************
               ; SET DASD TYPE FOR FORMAT
               ;***********************************

0580                    DSKT_SETDASD    PROC    NEAR
0580  C6 06 0041 R 00      MOV     DISKETTE_STATUS,0   ; SET OKAY RETURN CODE
0585  C3                   RET
0586                    DSKT_SETDASD    ENDP
```

```
              ;*******************************************
              ; GET_DRV_CONFIG
              ; GET DRIVE INFORMATION SUBROUTINE
              ;
              ; INPUT CONDITIONS: DL = DRIVE NUMBER 0-3
              ; OUTPUT CONDITIONS: AL (LOW NIBBLE CONTAINS DRIVE TYPE)
              ;          CARRY FLAG SET IF DRIVE # OUT OF RANGE
              ;
              ; REGISTERS MODIFIED: AX,BX,CX
              ;
              ;*******************************************

0586                   GET_DRV_CONFIG PROC NEAR

0586  B4 10               MOV     AH,RTC_DSKT_CON
0588  E8 0000 E           CALL    GET_RTC_REG        ; GET DRIVE INFORMATION
058B  8A F8               MOV     BH,AL              ; DRIVE 0,1 IN BH
058D  FE C4               INC     AH
058F  E8 0000 E           CALL    GET_RTC_REG        ; DRIVE 2,3 IN AL
0592  8A E7               MOV     AH,BH              ; DRIVE 0-3 INFO IN AX (NIBBLES)
0594  B1 03               MOV     CL,3               ; SET SHIFT COUNT
0596  2A CA               SUB     CL,DL              ; SUBTRACT DRIVE NUMBER
0598  74 06               JZ      GET_DRV01
059A  D0 E1               SHL     CL,1               ; MULTIPLY BY 4 (BITS/ DRIVE)
059C  D0 E1               SHL     CL,1
059E  D3 E8               SHR     AX,CL              ; SHIFT TO GET DRIVE INFO

05A0                   GET_DRV01:
05A0  24 0F               AND     AL,0FH             ; SAVE ONLY LOW NIBBLE
05A2  C3                  RET
05A3                   GET_DRV_CONFIG ENDP
```

Bootstrap Loader Interrupt Hex 19 (SYS_BOOT)

```
              ;--- INT 19H ------------------------------------------
              ; BOOT STRAP LOADER
              ;       TRACK 0, SECTOR 1 IS READ INTO THE
              ;       BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
              ;       AND CONTROL IS TRANSFERRED THERE.
              ;
              ;       IF THERE IS A HARDWARE ERROR CONTROL IS
              ;       TRANSFERRED TO THE ROM BASIC ENTRY POINT.
              ;-----------------------------------------------------
                          ASSUME  CS:ROMCODE,DS:DATA,ES:ABS0
05A3                   SYS_BOOT PROC    NEAR
05A3  FC                  CLD                        ; SET FORWARD DIRECTION
              ;
              ; FORCE DCL SUPPORT FOR BOOTSTRAP
              ;
05A4  B8 ---- R           MOV     AX,DATA
05A7  8E D8               MOV     DS,AX

05A9  E8 0000 E           CALL    DSP_INIT           ; CLEAR THE SCREEN

05AC  E8 0000 E           CALL    RES_ERR_CHK        ; CHECK AND DISPLAY RESUME ERRORS
05AF  75 4C               JNZ     H1_3               ; JUMP IF ANY RESUME ERRORS

05B1                   RETRY_BOOT:
05B1  FB                  STI                        ; ENABLE INTERRUPTS
05B2  2B C0               SUB     AX,AX              ; ESTABLISH ADDRESSING
05B4  8E C0               MOV     ES,AX

              ;----- RESET THE DSKT PARAMETER TABLE VECTOR
```

```
05B6  26: C7 06 0078 R 0000 E         MOV     WORD PTR DISK_POINTER, OFFSET DSKT_BASE
05BD  26: 8C 0E 007A R     MOV     WORD PTR DISK_POINTER+2,CS
                    ;
                    ; CLEAR THE BOOT LOCATION (256 WORDS)
                    ;
05C2  B9 0100              MOV     CX,256
05C5  BF 7C00 R            MOV     DI,OFFSET BOOT_LOCN
05C8  2B C0                SUB     AX,AX
05CA  F3/ AB               REP     STOSW

                    ;       LOAD SYSTEM FROM DISKETTE

05CC  B9 0002              MOV     CX,2                 ; RETRY COUNTER
05CF                    H0:
05CF  51                   PUSH    CX
05D0                    H1:
05D0  B4 00                MOV     AH,0                 ; RESET THE DISKETTE SYSTEM
05D2  CD 13                INT     13H                  ; DISKETTE_IO
05D4  73 06                JNC     H1_1                 ; IF ERROR, TRY AGAIN
05D6  59                   POP     CX
05D7  E2 F6                LOOP    H0                   ; IF RETRY EXCEEDS TWO  THEN BASIC

05D9  EB 7C 90             JMP     HBASIC
05DC                    H1_1:
05DC  B8 0201              MOV     AX,201H              ; READ IN THE SINGLE SECTOR
05DF  2B D2                SUB     DX,DX                ; TO THE BOOT LOCATION
05E1  BB 7C00 R            MOV     BX,OFFSET BOOT_LOCN
                                                        ; DRIVE 0, HEAD 0
05E4  B9 0001              MOV     CX,1                 ; SECTOR 1, TRACK 0
05E7  CD 13                INT     13H                  ; DISKETTE_IO
05E9  59                   POP     CX                   ; GET RETRY COUNT
05EA  72 03                JC      H1_2                 ; GO CHECK BOOT IF READ OKAY
05EC  EB 76 90             JMP     H4
05EF                    H1_2:
05EF  B4 16                MOV     AH,16H               ; CHECK FOR NO MEDIA IN DRIVE BY
05F1  CD 13                INT     13H                  ; CHECKING CHANGE LINE
05F3  F6 C4 80             TEST    AH,TIME_OUT          ; CHECK FOR NO MEDIA IN DRIVE
05F6  75 05                JNZ     H1_3                 ; IF NO MEDIA THEN SHOW ICON
05F8  E2 D5                LOOP    H0                   ; ELSE GO RETRY ERROR
05FA  E9 068C R            JMP     H10                  ; RETRY EXCEEDED GO SHOW BAD DISKETTE
05FD                    H1_3:
05FD  F6 06 0016 R 10      TEST    BIOS_STATUS,BOOT_F1HIT ; DID THE USER INDICATE BASIC OPTIO
0602  74 03                JZ      H2_1                 ; JUMP IF NOT
0604  EB 51 90             JMP     HBASIC               ; EXIT TO BASIC
                    ;
                    ; SHOW DISKETTE
                    ;
0607                    H2_1:
0607  80 26 0016 R EF      AND     BIOS_STATUS,NOT BOOT_F1HIT ; RESET F1 KEY HIT FLAG
060C                    H3:
060C  BA 1200              MOV     DX,1200H             ; DX POINTS TO DISPLAY POSITION
060F  BD 0000 E            MOV     BP,OFFSET F1_ICON    ; POINT THE F1 ICON
0612  E8 06BE R            CALL    DSPY_ICON
0615                    H3_1:
0615  BA 011F              MOV     DX,011FH             ; DX POINTS TO DISPLAY POSITION
0618  BD 0000 E            MOV     BP,OFFSET SYS_DSKT_ICON ; POINT TO THE DISKETTE ICON
061B  E8 06BE R            CALL    DSPY_ICON
061E  BA 0E21              MOV     DX,0E21H             ; DX POINTS TO DISPLAY POSITION
0621  BD 0000 E            MOV     BP,OFFSET DSKT_ICON ; POINT TO THE DISKETTE ICON
0624  E8 06BE R            CALL    DSPY_ICON
0627                    H3_2:
0627  B3 05                MOV     BL,05                ; SET TIMEOUT TO 275 MSECS
0629  E8 06C4 R            CALL    KEY_WAIT             ; WAIT FOR F1 KEY OR TIME
                    ;
                    ; WAIT TIMEOUT OR F1 KEY OCCURRED
                    ;
062C                    T0:
062C  52                   PUSH    DX                   ; SAVE DISKTETTE POINTER
062D  BA 011F              MOV     DX,011FH             ; DX POINTS TO DISPLAY POSITION
0630  BD 0000 E            MOV     BP,OFFSET SYS_DSKT_ICON ; POINT TO THE DISKETTE ICON
0633  E8 06BE R            CALL    DSPY_ICON
0636  5A                   POP     DX
0637  FE CE                DEC     DH
0639  80 FE 06             CMP     DH,06H
```

```
063C  74 08                JE      TO_1                ; YES THEN REDISPLAY ICONS
063E  BD 0000 E            MOV     BP,OFFSET DSKT_ICON ; OTHERWISE JUST MOVE DISKETTE
0641  E8 06BE R            CALL    DSPY_ICON
0644  EB E1                JMP     H3_2                ; WAIT AGAIN
                     ;
                     ; CHECK FOR F1 HIT AND IF SO THEN RETRY DISKETTE
                     ;
0646                     TO_1:
0646  F6 06 0016 R 10      TEST    BIOS_STATUS,BOOT_F1HIT ; F1 KEY WAS HIT?
064B  75 07                JNZ     TO_3                ; NO THEN KEEP DISPLAYING ICONS
064D  B3 14                MOV     BL,20               ; DELAY LONGER FOR NO DISKETTE
064F  E8 06C4 R            CALL    KEY_WAIT            ; WAIT TIME
0652  EB C1                JMP     SHORT H3_1          ; GO RE DISPLAY ICONS

0654  E9 05B1 R          TO_3:  JMP    RETRY_BOOT      ; GO RETRY BOOT
                     ;
                     ; USER WANTS BASIC OR CONTROLLER FAILURE
                     ;
0657                     HBASIC:
0657  80 26 0016 R EF      AND     BIOS_STATUS,NOT BOOT_F1HIT ; RESET F1 HIT FLAG
065C  E8 0000 E            CALL    DSP_INIT            ; CLEAR THE DISPLAY
065F  E8 0000 E            CALL    DSP_FSETM           ; SET MODE TARGET DISPLAY
0662  CD 18                INT     18H                 ; GO TO RESIDENT BASIC

                     ;----- IPL WAS SUCCESSFUL
                     ; NOW CHECK BOOT_LOCN FOR NON ZERO DATA

0664                     H4:
0664  BF 7C00 R            MOV     DI,OFFSET BOOT_LOCN
0667  B9 000A              MOV     CX,10
066A  26: A1 7C00 R        MOV     AX,WORD PTR BOOT_LOCN
066E  0B C0                OR      AX,AX
0670  74 1A                JE      H10                 ; IF ZERO BAD BOOT RECORD
                     ;
                     ; NOW CHECK NEXT 10 WORDS FOR NOT EQUAL
                     ;
0672  83 C7 02           H4_LP:  ADD    DI,2
0675  26: 3B 05            CMP     AX,ES:[DI]
0678  E1 F8                LOOPZ   H4_LP               ; LOOP IF DATA SAME
067A  74 10                JZ      H10                 ; BAD BOOT RECORD IF ALL DATA SAME

067C  80 26 0016 R EB      AND     BIOS_STATUS,NOT BOOT_F1HIT+DCL_SUPPORTED ; RESET BIOS FLAGS
0681  E8 0000 E            CALL    DSP_INIT            ; CLEAR THE DISPLAY
0684  E8 0000 E            CALL    DSP_FSETM           ; CLEAR THE TARGET DISPLAY
0687  EA 7C00 ---- R       JMP     BOOT_LOCN
                     ;
                     ; SHOW DISKETTE EXITING DRIVE AND SPLITTING APART
                     ;

068C  80 26 0016 R EF    H10:   AND     BIOS_STATUS,NOT BOOT_F1HIT ; RESET KEY HIT FLAG

0691  BA 0621              MOV     DX,0621H            ; START INITIAL DISPLAY
0694                     H10_2:
0694  52                   PUSH    DX                  ; SAVE DISKTETTE POINTER
0695  BA 011F              MOV     DX,011FH            ; DX POINTS TO DISPLAY POSITION
0698  BD 0000 E            MOV     BP,OFFSET SYS_DSKT_ICON ; POINT TO THE DISKETTE ICON
069B  E8 06BE R            CALL    DSPY_ICON
069E  5A                   POP     DX
069F  FE C6                INC     DH
06A1  B3 05                MOV     BL,05               ; SET TIMEOUT TO 275 MSECS
06A3  BD 0000 E            MOV     BP,OFFSET DSKT_ICON ; WITH GOOD DISKETTE ICON
                     ;
                     ; CHECK FOR LAST POSITION FOR DISKETTE BEFORE BROKEN DISKETTE IS SHOWN
                     ;
06A6  80 FE 0E             CMP     DH,0EH
06A9  75 05                JNE     H10_3               ; YES THEN DISPLAY BROKEN ICON
06AB  BD 0000 E            MOV     BP,OFFSET BAD_DSKT_ICON ; OTHERWISE SHOW BAD DISKETTE
06AE  B3 28                MOV     BL,40               ; TIME DELAY FOR BROKEN DISKETTE
06B0                     H10_3:
06B0  E8 06BE R            CALL    DSPY_ICON
06B3  E8 06C4 R            CALL    KEY_WAIT            ; WAIT FOR TIME OR F1 KEY
06B6  80 FE 0E             CMP     DH,0EH              ; AT LAST ICON
06B9  75 D9                JNE     H10_2               ; NO THEN KEEP MOVING DISKETTE
06BB  E9 060C R            JMP     H3                  ; YES THEN SHOW DISKETTE GOING IN
```

```
              ;
              ; DISPLAY BROKEN ICON
              ;

06BE                    SYS_BOOT ENDP

06BE                    DSPY_ICON PROC    NEAR
06BE  0E                   PUSH    CS
06BF  07                   POP     ES                 ; POINT ES TO CODE SEGMENT
06C0  E8 0000 E            CALL    ICON_PR            ; DISPLAY THE INSERT DISKETTE ICON
06C3  C3                   RET
06C4                    DSPY_ICON ENDP
              ;
              ; THIS CODE WAITS FOR THE F1 KEY TO BE DEPRESSED OR THE TIME LIMIT EXPIRED
              ; WHICH EVER COMES FIRST
              ;
06C4                    KEY_WAIT PROC NEAR
06C4  1E                   PUSH    DS
06C5  07                   POP     ES                 ; POINT ES TO DATA SEGMENT
06C6  BF 001C R            MOV     DI,OFFSET BUFFER_TAIL ; GET ADDRESS OF BUFFER_TAIL
06C9  8A 3E 001A R         MOV     BH,BYTE PTR BUFFER_HEAD ; GET DATA IN BUFFER HRAD
06CD  B0 02                MOV     AL,02H             ; COMPARE RETURN NOT EQUAL
                                                      ; USING BH AND DATA AT DI
06CF  E8 0000 E            CALL    EXT_EVENT          ; WILL RETURN AFTER TIME LIMIT OR WHEN
                                                      ; THERE IS DATA IN KEYBOARD BUFFER
06D2  72 22                JC      KW_EXIT            ; IF TIMEOUT THEN JUMP
              ;
              ; A KEY WAS HIT
              ;
06D4  B4 00                MOV     AH,0
06D6  CD 16                INT     16H                ; OTHERWISE PURGE THE KEY
06D8  80 FC 3B             CMP     AH,3BH             ; F1 KEY?
06DB  74 0D                JE      F1BRK              ; YES THEN WAIT FOR BREAK
              ;
              ; CHECK FOR POWER ON SELF TEST LOOP MODE REQUEST
              ;
06DD  3C 0C                CMP     AL,0CH             ; CNTL + L KEY?
06DF  75 E3                JNE     KEY_WAIT           ; NO THEN WAIT SOME MORE
06E1  C7 06 0072 R ABCD    MOV     RESET_FLAG,LOOP_MODE ; SET POST LOOP MODE ACTIVE
06E7  E9 0000 E            JMP     POST_LOOP          ; EXIT BACK TO POST
              ;
              ; WAIT FOR BREAK OF F1 KEY
              ;
06EA  FB                F1BRK:  STI                   ; ENABLE INTERRUPTS
06EB  E4 60                IN      AL,KB_DATA         ; READ KEYBOARD PORT
06ED  3C 3B                CMP     AL,03BH            ; CHECK FOR STILL F1 KEY
06EF  74 F9                JE      F1BRK              ; WAIT UNTIL NOT F1 KEY MAKE
              ;
              ; F1 KEY WAS DEPRESSED THEN RELEASED
              ;
06F1  80 0E 0016 R 10      OR      BIOS_STATUS,BOOT_F1HIT ; SET F1 HIT FLAG
              ;
              ; RETURN TO CALLER
              ;
06F6                    KW_EXIT:
06F6  C3                   RET
06F7                    KEY_WAIT ENDP

06F7                    ROMCODE ENDS
                           END
```

Communications and Printer BIOS (B14COMMO)

```
0000                     ROMCODE SEGMENT BYTE PUBLIC
                            ASSUME  CS:ROMCODE,DS:DATA
                            IDENT B14COM,14,00

               ;*********************************************************************
               ;
               ; ROUTINE-NAME :      B14COMMO
               ;
               ; DATE LAST MODIFIED:   09/12/85
               ;
               ; DESCRIPTIVE-NAME : INT 14H CALLS COMMUNICATION BIOS FUNCTIONS.
               ;
               ;                    INT 17H CALLS PRINTER BIOS ROUTINES.
               ;
               ; COPYRIGHT : 7396-917 (C) COPYRIGHT IBM CORP. 1985
               ;             REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
               ;
               ; CHANGE LEVEL: 0.0
               ;
               ; FUNCTIONS:
               ;       COMMO_IO                              COMMUNICATIONS BIOS
               ;       PRT_IO                                PRINTER BIOS
               ;       COM_POWER                             POWER ON/OFF COM DEVICES
               ;
               ; MODULE SIZE:  550 BYTES (DECIMAL)
               ;
               ; ENTRY CONDITIONS:  REFER TO PROLOGUES
               ;
               ; EXIT CONDITIONS:  REFER TO PROLOGUES
               ;
               ; ROUTINES IN MODULE:
               ;       COMMO_IO                              COMMUNICATIONS BIOS
               ;       PRT_IO                                PRINTER BIOS
               ;       COM_POWER                             POWER ON/OFF COM DEVICES
               ;
               ; INTERNAL DATA AREAS / TABLES:  NONE
               ;
               ; EXTERNALLY REFERENCED ROUTINES:  REFER TO EXTRN LIST
               ;
               ; EXTERNALLY REFERENCED DATA AREAS:  BIOS DATA SEGMENT
               ;
               ; CHANGE ACTIVITY:  NONE
               ;*********************************************************************
```

Communications Interrupt Hex 14 (COMMO_IO)

```
;-----INT 14H ---------------------------------------------------------
; COMMO_IO
;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
;       PORT ACCORDING TO THE PARAMETERS:
;       (AH)=0  INITIALIZE THE COMMUNICATIONS PORT
;               (AL) HAS PARAMETERS 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 - 9600
;
;       ON RETURN, CONDITIONS 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, REFLECTING 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
;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY BITS
;                       LEFT ON ARE THE ERROR BITS (7,4,3,2,1)
;                       IF AH HAS BIT 7 ON (TIME OUT) THE REMAINING
;                       BITS ARE NOT PREDICTABLE.
;                       THUS, AH IS NON ZERO ONLY WHEN AN ERROR
;                       OCCURRED.
;       (AH)=3  RETURN THE COMMO PORT STATUS IN (AX)
;               AH CONTAINS THE LINE 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 = RECEIVED 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
;
;       (AH)=OTHER
;               NO ACTION TAKEN.
;
;       (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 LABEL RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
;       VALUE FOR TIMEOUT (DEFAULT=1).
;
; REGISTERS MODIFIED
;       AX MODIFIED ACCORDING TO PARMS OF CALL
;       ALL OTHERS UNCHANGED
;
```

```
                ; INTERRUPTS:
                ;       INTERRUPTS ARE ENABLE UPON ROUTINE ENTRY.
                ;----------------------------------------------------------------------

                ;*********************************
                ;         P U B L I C S
                ;*********************************
                           PUBLIC  COMMO_IO
                           PUBLIC  PRT_IO
                           PUBLIC  COM_POWER

                ;*********************************
                ;         E X T E R N A L S
                ;*********************************
                           EXTRN   DDS:NEAR
                           EXTRN   BAUD_TABLE:BYTE
                           EXTRN   GET_RTC_REG:NEAR
                           EXTRN   GET_VECTOR@:NEAR
                           EXTRN   D11:NEAR
                           EXTRN   CHECK_FOR_8250:NEAR

                ;*********************************
                ;         E Q U A T E S
                ;*********************************
= 0020                  DSR_BIT EQU     20H
= 0010                  CTS_BIT EQU     10H
= 0001                  DTR_BIT EQU     01H
= 0002                  RTS_BIT EQU     02H
= 001E                  RCV_ERR EQU     00011110B       ; RECV ERROR BITS
= 0080                  DLAB    EQU     80H             ; DIVISOR LATCH ACCESS BIT
                ;*********************************
                ;         S T A R T   O F   C O D E
                ;*********************************

0000                    COMMO_IO        PROC    FAR

                ;----- VECTOR TO APPROPRIATE ROUTINE

0000  1E                   PUSH    DS                   ; SAVE SEGMENT
0001  52                   PUSH    DX                   ; AND REGISTERS USED
0002  56                   PUSH    SI
0003  57                   PUSH    DI
0004  51                   PUSH    CX
0005  53                   PUSH    BX
0006  8B F2                MOV     SI,DX                ; PORT NUMBER TO SI
0008  8B FA                MOV     DI,DX                ; AND DI
000A  D1 E6                SHL     SI,1                 ; MAKE WORD OFFSET
000C  E8 0000 E            CALL    DDS                  ; BIOS DATA SEG INTO DS
000F  8B 94 0000 R         MOV     DX,RS232_BASE[SI]    ; GET PORT BASE ADDRESS
0013  0B D2                OR      DX,DX                ; TEST FOR 0 (NO PORT)
0015  74 28                JZ      A3                   ; RETURN
                ;
0017  8A D8                MOV     BL,AL                ; SAVE SEND CHAR IF IS ONE
                ;
                ;-----CHECK FOR PRESENSE OF 8250 AT THIS PORT
0019  83 C2 03             ADD     DX,3                 ; ADDRESS 8250 LINE CTRL REG
001C  E8 0000 E            CALL    CHECK_FOR_8250       ; CARRY SET IF PRESENT
001F  FB                   STI                          ; SHIELDS DOWN
0020  72 05                JC      A0                   ; YES, GO ON IF PRESENT
                ;
0022  80 CC 80             OR      AH,80H               ; NO, SET TIMEOUT ERROR
0025  EB 18                JMP     SHORT   A3           ; EXIT
                ;
0027  83 EA 03          A0:     SUB     DX,3            ; RESTORE ADDRESS
002A  8A C3             A1:     MOV     AL,BL           ; RESTORE SEND CHAR IF IS ONE
002C  0A E4                OR      AH,AH                ; TEST FOR AH=0
002E  74 16                JZ      A4                   ; YES, COMMO INIT
0030  FE CC                DEC     AH                   ; TEST FOR AH=1
0032  74 47                JZ      A5                   ; YES, SEND CHAR
0034  FE CC                DEC     AH                   ; TEST FOR AH=2
0036  74 6C                JZ      A12                  ; YES, RECEIVE CHAR
0038  FE CC                DEC     AH                   ; TEST FOR AH=3
003A  75 03                JNZ     A3                   ; NO, EXIT
003C  E9 00C6 R            JMP     A18                  ; YES, GET PORT STATUS

003F                    A3:                             ; RETURN FROM RS232

003F  5B                   POP     BX                   ; RESTORE REGISTERS
0040  59                   POP     CX
```

```
0041  5F                    POP    DI
0042  5E                    POP    SI
0043  5A                    POP    DX
0044  1F                    POP    DS
0045  CF                    IRET                        ; RETURN TO CALLER

              ;----- INITIALIZE THE COMMUNICATIONS PORT

0046                     A4:
0046  8A E0                 MOV    AH,AL                ; SAVE INIT PARMS IN AH
0048  83 C2 03              ADD    DX,3                 ; POINT TO 8250 CONTROL REG
004B  B0 80                 MOV    AL,DLAB              ; ACCESS BAUD RATE BY
004D  EE                    OUT    DX,AL                ; SETTING DLAB=1

              ;----- DETERMINE BAUD RATE DIVISOR

004E  8A D4                 MOV    DL,AH                ; GET PARMS TO DL
0050  B1 04                 MOV    CL,4
0052  D2 C2                 ROL    DL,CL
0054  81 E2 000E            AND    DX,0EH               ; ISOLATE THEM
0058  BF 0000 E             MOV    DI,OFFSET BAUD_TABLE ; BASE OF TABLE
005B  03 FA                 ADD    DI,DX                ; PUT INTO INDEX REGISTER
005D  8B 94 0000 R          MOV    DX,RS232_BASE[SI]    ; POINT TO HIGH ORDER OF DIV
0061  42                    INC    DX
0062  2E: 8A 45 01          MOV    AL,CS:[DI]+1         ; GET HIGH ORDER OF DIVISOR
0066  EE                    OUT    DX,AL                ; SET HIGH OF DIVISOR
0067  4A                    DEC    DX
0068  2E: 8A 05             MOV    AL,CS:[DI]           ; GET LOW ORDER OF DIVISOR
006B  EE                    OUT    DX,AL                ; SET LOW OF DIVISOR
006C  83 C2 03              ADD    DX,3                 ; ADDRESS LINE CTR REG
006F  8A C4                 MOV    AL,AH                ; GET PARMS BACK
0071  24 1F                 AND    AL,01FH              ; STRIP OFF THE BAUD BITS
0073  EE                    OUT    DX,AL                ; SET LINE CTRL TO WORD LENGTH
0074  4A                    DEC    DX                   ; PARITY AND # STOP BITS
0075  4A                    DEC    DX                   ; ADDRESS INTR ENABLE REG
0076  B0 00                 MOV    AL,0
0078  EE                    OUT    DX,AL                ; DISABLE INTERRUPTS (IER)
0079  EB 4B                 JMP    SHORT A18            ; COM_STATUS

              ;----- SEND CHARACTER IN (AL) OVER COMMO LINE

007B                     A5:
007B  50                    PUSH   AX                   ; SAVE CHAR TO SEND
007C  83 C2 04              ADD    DX,4                 ; ADDRESS MODEM CONTROL REG
007F  B0 03                 MOV    AL,DTR_BIT OR RTS_BIT ; SET DTR AND RTS
0081  EE                    OUT    DX,AL                ; DATA TERM RDY, REQ TO SEND
0082  42                    INC    DX                   ; MODEM STATUS REGISTER
0083  42                    INC    DX
0084  B7 30                 MOV    BH,DSR_BIT OR CTS_BIT ; DATA SET RDY & CLR TO SEND
0086  E8 0158 R             CALL   WAIT_FOR_STATUS      ; ARE BOTH TRUE
0089  74 08                 JZ     A9                   ; YES, READY TO TRANSMIT CHAR
008B                     A7:
008B  59                    POP    CX
008C  8A C1                 MOV    AL,CL                ; RELOAD DATA BYTE
008E                     A8:
008E  80 CC 80              OR     AH,80H               ; INDICATE TIME OUT
0091  EB AC                 JMP    A3                   ; RETURN
0093                     A9:                            ; CLEAR_TO_SEND
0093  4A                    DEC    DX                   ; ADDRESS LINE STATUS REGISTER
0094  B7 20                 MOV    BH,DSR_BIT           ; IS TRANSMITTER READY
0096  E8 0158 R             CALL   WAIT_FOR_STATUS      ; TEST FOR TRANSMITTER READY
0099  75 F0                 JNZ    A7                   ; NO, RETURN WITH TIME OUT SET
009B  83 EA 05              SUB    DX,5                 ; ADDRESS DATA PORT
009E  59                    POP    CX                   ; RECOVER IN CX TEMPORARILY
009F  8A C1                 MOV    AL,CL                ; MOVE CHAR TO AL FOR OUT,
                                                        ; STATUS IN AH
00A1  EE                    OUT    DX,AL                ; OUTPUT CHARACTER
00A2  EB 9B                 JMP    A3                   ; RETURN

              ;----- RECEIVE CHARACTER FROM COMMO LINE

00A4                     A12:
00A4  83 C2 04              ADD    DX,4                 ; MODEM CONTROL REGISTER
00A7  B0 01                 MOV    AL,DTR_BIT           ; SET DATA TERMINAL READY
```

```
00A9  EE                    OUT     DX,AL
00AA  42                    INC     DX                    ; MODEM STATUS REGISTER
00AB  42                    INC     DX
00AC  B7 20                 MOV     BH,DSR_BIT            ; DATA SET READY
00AE  E8 0158 R             CALL    WAIT_FOR_STATUS       ; TEST FOR DSR
00B1  75 DB                 JNZ     A8                    ; NO, RETURN WITH ERROR
00B3  4A                    DEC     DX                    ; LINE STATUS REGISTER
00B4  B7 01                 MOV     BH,1                  ; RECEIVE BUFFER FULL
00B6  E8 0158 R             CALL    WAIT_FOR_STATUS       ; TEST FOR REC. BUFF. FULL
00B9  75 D3                 JNZ     A8                    ; NO, SET TIME OUT ERROR
00BB  80 E4 1E              AND     AH,RCV_ERR            ; SAVE ERR ON RECV CHAR
00BE  8B 94 0000 R          MOV     DX,RS232_BASE[SI]     ; DATA PORT
00C2  EC                    IN      AL,DX                 ; GET CHARACTER FROM LINE
00C3  E9 003F R             JMP     A3                    ; RETURN

               ;----- COMMO PORT STATUS ROUTINE

00C6                     A18:
00C6  8B 94 0000 R          MOV     DX,RS232_BASE[SI]
00CA  83 C2 05              ADD     DX,5                  ; LINE STATUS REGISTER
00CD  EC                    IN      AL,DX                 ; GET LINE CONTROL STATUS
00CE  8A E0                 MOV     AH,AL                 ; PUT IN AH FOR RETURN
00D0  42                    INC     DX                    ; POINT TO MODEM STATUS REG
00D1  EC                    IN      AL,DX                 ; GET MODEM CONTROL STATUS
00D2  E9 003F R             JMP     A3                    ; RETURN

               ;--------------------------------------------------------------------
               ; COM_POWER
               ;
               ; TURN POWER ON/OFF TO COMMO CARD
               ;
               ; ENTRY:
               ;       BL = MODEM/ASYNC POWER BIT IN FEATURE CONFIGURATION REGISTER
               ;            ( =02 FOR MODEM, =04 FOR ASYNC SEC, =05 ASYNC PRIMARY
               ;       BH = 0 - TURN POWER OFF
               ;          = 1 - TURN POWER ON
               ;
               ; EXIT: AH = 0 AND CF = 0  IF OPERATION OKAY
               ;       AH = 80 AND CF = 1  IF POWER ON REQSTED AND NO RESP FROM 8250
               ;                           WITHIN 500 MSEC
               ; REGISTERS USED:
               ;       AX IS DESTROYED.  ALL OTHERS ARE NOT MODIFIED.
               ;
               ; INTERRUPTS:
               ;       INTERRUPTS ARE ENABLED WITHIN ROUTINE, RESTORED UPON EXIT.
               ;---------------------------------------------------------------------
               ;LOCAL EQUATES
               ;
= 03FB                   LINE_CTL_REG    EQU     3FBH
               ;---------------------------------------------------------------------

00D5                     COM_POWER       PROC    NEAR

00D5  53                    PUSH    BX                    ; SAVE REGISTERS
00D6  51                    PUSH    CX
00D7  52                    PUSH    DX
00D8  E4 7C                 IN      AL,KYBD_CNTL          ; GET INITIAL REG CONTENTS
00DA  BA 03FB               MOV     DX,LINE_CTL_REG       ; PRI PORT LINE CONTROL REG
00DD  B4 10                 MOV     AH,10H                ; SET FOR LVL 4 SYSTEM IRPT
00DF  B1 04                 MOV     CL,4                  ; PARM FOR GET_VECTOR CALL
00E1  F6 C3 03              TEST    BL,SET_RS232_PRIM+ACT_MODEM ; PRIMARY ASYNC?
00E4  75 06                 JNZ     CP0
00E6  FE CE                 DEC     DH                    ; NO - POINT TO SEC INT ID REG
00E8  D0 EC                 SHR     AH,1                  ; SET FOR LEVEL 3 SYSTEM IRPT
00EA  FE C9                 DEC     CL                    ; ADJUST PARM FOR LEVEL 3 IRPT

00EC  0A FF              CP0:    OR      BH,BH            ; CHECK REQUEST CODE
00EE  74 06                 JZ      CP1                   ; JUMP IF A POWER OFF REQUEST
00F0  0A C3                 OR      AL,BL                 ; TURN POWER ON
00F2  E6 7C                 OUT     KYBD_CNTL,AL
00F4  EB 47                 JMP     SHORT    CP2

               ; BE ABLE TO HANDLE ANY PENDING INTERRUPT WHEN POWER IS TURNED OFF
```

```
00F6  9C                CP1:   PUSHF                  ; SAVE FLAGS
00F7  FA                   CLI                        ; SHIELDS UP
00F8  F6 D3                NOT     BL                 ; TURN POWER OFF
00FA  22 C3                AND     AL,BL
00FC  E6 7C                OUT     KYBD_CNTL,AL
00FE  56                   PUSH    SI                 ; SAVE SI
00FF  E8 0000 E            CALL    GET_VECTOR@        ; SI CONTAINS VECTOR OFFSET
0102  1E                   PUSH    DS                 ; SAVE DS

0103  B9 ---- R            MOV     CX,ABS0
0106  8E D9                MOV     DS,CX              ; SEGMENT THAT CONTAINS VECTOR
0108  8B 0C                MOV     CX,[SI]
010A  51                   PUSH    CX                 ; SAVE OFFSET
010B  8B 4C 02             MOV     CX,[SI+2]
010E  51                   PUSH    CX                 ; SAVE SEGMENT
010F  C7 04 0000 E         MOV     WORD PTR[SI],OFFSET D11 ; SET UP DUMMY IRPT HNDLER
0113  8C 4C 02             MOV     [SI+2],CS

            ; UNMASK INTERRUPT LEVEL 3/4

0116  F6 D4                NOT     AH
0118  E4 21                IN      AL,INTA01          ; GET CURRENT SYSTEM IRPTS
011A  8A D8                MOV     BL,AL              ; SAVE CURRENT SYSTEM IRPTS
011C  22 C4                AND     AL,AH              ; LEVEL 3/4 INTERRUPT
011E  E6 21                OUT     INTA01,AL          ; UNMASK INTERRUPT

            ; ENABLE SYSTEM INTERRUPTS

0120  FB                   STI                        ; SHIELDS DOWN
0121  B9 64C8              MOV     CX,100*MS_DELAY    ; DELAY 100ms
0124  E2 FE                LOOP    $

            ; RESTORE INTERRUPT MASK

0126  FA                   CLI                        ; SHIELDS UP
0127  E4 21                IN      AL,INTA01          ; CURRENT INTERRUPTS
0129  F6 D4                NOT     AH                 ; MASK INTERRUPT LEVEL 3/4 IF
012B  22 DC                AND     BL,AH              ; UNMASKED DURING THIS ROUTINE
012D  0A C3                OR      AL,BL
012F  E6 21                OUT     INTA01,AL          ; MASK INTERRUPT

            ; RESTORE INTERRUPT VECTOR

0131  59                   POP     CX                 ; RESTORE IRPT VECTOR SEGMENT
0132  89 4C 02             MOV     [SI+2],CX
0135  59                   POP     CX                 ; RESTORE IRPT VECTOR OFFSET
0136  89 0C                MOV     [SI],CX

0138  1F                   POP     DS
0139  5E                   POP     SI
013A  9D                   POPF
013B  EB 15                JMP     SHORT    CP3       ; EXIT GOOD

            ; WAIT FOR RESPONSE FROM 8250 UP TO 500 msec

013D  B3 05             CP2:    MOV     BL,5          ; LOOP COUNT FOR 500 MSEC
            ;
013F  B9 64C8           CP2_1:  MOV     CX,100*MS_DELAY ; INNER LOOP COUNT
0142  E2 FE                LOOP    $

            ; CHECK FOR RESPONSE FROM 8250

0144  E8 0000 E            CALL    CHECK_FOR_8250     ; SETS CARRY IF PRESENT
0147  72 09                JC      CP3                ; YES, GO ON.
0149  FE CB                DEC     BL
014B  75 F2                JNZ     CP2_1              ; NO, TRY TIL TIMES UP

            ; NO RESPONSE FROM 8250 - TIMEOUT

014D  B4 80                MOV     AH,80H             ; SET TIMEOUT
014F  F9                   STC                        ; SET CARRY FLAG
0150  EB 02                JMP     SHORT CP4

            ; RESET ASYNC
```

```
0152  2A E4            CP3:    SUB     AH,AH          ; GOOD RETURN

0154  5A               CP4:    POP     DX             ; RESTORE REGISTERS
0155  59                  POP     CX
0156  5B                  POP     BX
0157  C3                  RET

0158                   COM_POWER       ENDP

             ;--------------------------------------
             ; 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
             ;--------------------------------------
0158                   WAIT_FOR_STATUS PROC    NEAR
0158  8A 9D 007C R        MOV     BL,RS232_TIM_OUT[DI] ; LOAD OUTER LOOP COUNT
015C                   WFS0:
015C  2B C9               SUB     CX,CX
015E                   WFS1:
015E  EC                  IN      AL,DX          ; GET STATUS
015F  8A E0               MOV     AH,AL          ; MOVE TO AH
0161  22 C7               AND     AL,BH          ; ISOLATE BITS TO TEST
0163  3A C7               CMP     AL,BH          ; EXACTLY = TO MASK
0165  74 08               JE      WFS_END        ; RETURN WITH ZERO FLAG ON
0167  E2 F5               LOOP    WFS1           ; TRY AGAIN
0169  FE CB               DEC     BL
016B  75 EF               JNZ     WFS0

016D  0A FF               OR      BH,BH          ; SET ZERO FLAG OFF
016F                   WFS_END:
016F  C3                  RET
0170                   WAIT_FOR_STATUS ENDP
0170                   COMMO_IO        ENDP
```

Printer Interrupt Hex 17 (PRT_IO)

```
               ;--- INT 17H ------------------------------------------------------
               ; PRT_IO
               ;       THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
               ; INPUT
               ;       (AH)=0  PRINT THE CHARACTER IN (AL)
               ;               ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED
               ;               (TIME OUT). OTHER BITS SET AS ON NORMAL STATUS CALL
               ;       (AH)=1  INITIALIZE THE PRINTER PORT
               ;               RETURNS WITH (AH) SET WITH PRINTER STATUS
               ;       (AH)=2  READ THE PRINTER STATUS INTO (AH)
               ;               7       6       5       4       3       2-1  0
               ;               |       |       |       |       |       |    |_TIMEOUT
               ;               |       |       |       |       |       |_ UNUSED
               ;               |       |       |       |       |_ 1 = I/O ERROR
               ;               |       |       |       |_ 1 = SELECTED
               ;               |       |       |_ 1 = OUT OF PAPER
               ;               |       |_ 1 = ACKNOWLEDGE
               ;               |_ 1 = NOT BUSY
               ;
               ;       (AH)=OTHER
               ;               NO ACTION TAKEN.
               ;
               ;       (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL
               ;              VALUES IN PRINTER_BASE AREA
               ;
               ; DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER
               ; CARD(S) AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT,
               ; 408H ABSOLUTE, 3 WORDS).
               ;
               ; DATA AREA PRINT_TIM_OUT (BYTE) MAY BE CHANGED TO CAUSE DIFFERENT
               ; TIME-OUT WAITS. DEFAULT=20.
               ;
               ; IF NO PRINTER ATTACHED OR INVALID FUNCTION CODE RECEIVED, A RETURN
               ; IS MADE WITH AH= PRINTER TIMEOUT (01)
               ;
               ; REGISTERS USED
               ;       AH IS MODIFIED
               ;       ALL OTHERS UNCHANGED
               ;
               ; INTERRUPTS:
               ;       INTERRUPTS ARE ENABLE UPON ROUTINE ENTRY, RESTORED ON EXIT.
               ;-----------------------------------------------------------------
                           ASSUME  CS:ROMCODE,DS:DATA
0170                    PRT_IO  PROC    FAR
0170  FB                   STI                      ; SHIELDS DOWN
0171  1E                   PUSH    DS               ; SAVE SEGMENT
0172  52                   PUSH    DX               ; SAVE REGISTERS USED HERE
0173  56                   PUSH    SI
0174  51                   PUSH    CX
0175  53                   PUSH    BX
0176  E8 0000 E            CALL    DDS              ; BIOS DATA SEG TO DS
0179  8B F2                MOV     SI,DX            ; GET PRINTER PARM
017B  8A 9C 0078 R         MOV     BL,PRINT_TIM_OUT[SI] ; LOAD TIME-OUT PARM
017F  D1 E6                SHL     SI,1             ; WORD OFFSET INTO TABLE
0181  8B 94 0008 R         MOV     DX,PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER
0185  0B D2                OR      DX,DX            ; ANY PRINTERS PRESENT?
0187  74 0C                JZ      B0               ; NO,SIGNAL ERROR
0189  0A E4                OR      AH,AH            ; TEST FOR (AH)=0
018B  74 10                JZ      B2               ; YES, PRINT CHAR
018D  FE CC                DEC     AH               ; TEST FOR (AH)=1
018F  74 57                JZ      B8               ; YES, INIT PRINTER
0191  FE CC                DEC     AH               ; TEST FOR (AH)=2
0193  74 3F                JZ      B5               ; YES, GET STATUS
               ;
               ; INVALID COMMAND OR NO PRINTER ATTACHED; SIGNAL TIMEOUT
```

```
                     ;
0195  B4 01                 B0:     MOV     AH,01H          ; NO, RETURN TIMEOUT STATUS
                     ;
                     ; COMMON EXIT FOR ALL PRINTER FUNCTIONS
                     ;
0197  5B                    B1:     POP     BX
0198  59                        POP     CX
0199  5E                        POP     SI                  ; RECOVER REGISTERS
019A  5A                        POP     DX                  ; RECOVER REGISTERS
019B  1F                        POP     DS
019C  CF                        IRET

                     ;------ PRINT THE CHARACTER IN (AL)

019D                        B2:
019D  50                        PUSH    AX                  ; SAVE VALUE TO PRINT
019E  42                        INC     DX                  ; POINT TO STATUS PORT

                     ; CHECK FOR PRINTER BUSY

019F  EC                        IN      AL,DX               ; GET STATUS
01A0  A8 80                     TEST    AL,NOT_BUSY         ; IS PRINTER CURRENTLY BUSY?
01A2  75 23                     JNZ     B4                  ; NO THEN OUT_STROBE

                     ; NOTIFY OPERATING SYSTEM THAT A PRINTER BUSY WAIT IS IN AFFECT

01A4  F8                        CLC                         ; ***INT 15 DEVICE BUSY
01A5  B8 90FE                   MOV     AX,90FEH            ; FUNCTION 90 PRINTER ID
01A8  CD 15                     INT     15H
01AA  72 13                     JC      B3_2                ; JUMP IF TIMEOUT OCCURRED
01AC                        B3:
01AC  2B C9                     SUB     CX,CX               ; WAIT_BUSY
01AE                        B3_1:
01AE  EC                        IN      AL,DX               ; GET STATUS
01AF  8A E0                     MOV     AH,AL               ; STATUS TO AH ALSO
01B1  A8 80                     TEST    AL,NOT_BUSY         ; IS PRINTER CURRENTLY BUSY
01B3  75 12                     JNZ     B4                  ; NO, OUT_STROBE
01B5  E2 F7                     LOOP    B3_1                ; YES, TRY AGAIN
01B7  A8 08                     TEST    AL,NOT_ERROR        ; CHECK FOR I/O ERROR
01B9  74 07                     JZ      B3_3                ; YES, JUMP
01BB  FE CB                     DEC     BL                  ; FINISHED 500 MS DELAY?
01BD  75 ED                     JNZ     B3                  ; NO, DO AGAIN

                     ; BUSY TIMEOUT OCCURRED                ; YES, TIMEOUT

01BF                        B3_2:
01BF  80 CC 01                  OR      AH,1                ; SET ERROR FLAG
01C2                        B3_3:
01C2  80 E4 F9                  AND     AH,0F9H             ; TURN OFF THE OTHER BITS
01C5  EB 19                     JMP     SHORT B7            ; RETURN WITH ERROR FLAG SET

                     ; PRINTER IS NOT BUSY SO STROBE OUT CHARACTER

01C7  58                    B4:     POP     AX              ; GET VALUE TO PRINT
01C8  4A                        DEC     DX                  ; POINT TO DATA REG
01C9  EE                        OUT     DX,AL               ; OUTPUT VALUE TO DATA REG
01CA  42                        INC     DX                  ; POINT TO STATUS PORT
01CB  50                        PUSH    AX                  ; SAVE VALUE TO PRINT

                                                            ; OUT_STROBE
01CC  B0 0D                     MOV     AL,SELECT+NO_INIT+STROBE ; SET THE STROBE HIGH
01CE  42                        INC     DX                  ; STRB IS BIT 0 OF PORT C
01CF  EE                        OUT     DX,AL
01D0  24 FE                     AND     AL,NOT STROBE       ; SET THE STROBE LOW
01D2  EE                        OUT     DX,AL
01D3  58                        POP     AX                  ; RECOVER THE OUTPUT CHAR

                     ;------ PRINTER STATUS

01D4                        B5:
01D4  50                        PUSH    AX                  ; SAVE AL REG
01D5                        B6:
01D5  8B 94 0008 R              MOV     DX,PRINTER_BASE[SI]
01D9  42                        INC     DX
```

```
01DA  EC                    IN      AL,DX               ; GET PRINTER STATUS
01DB  8A E0                 MOV     AH,AL
01DD  80 E4 F8              AND     AH,0F8H             ; TURN OFF UNUSED BITS
01E0                     B7:                            ; STATUS_SET
01E0  5A                    POP     DX                  ; RECOVER AL REG
01E1  8A C2                 MOV     AL,DL               ; GET CHARACTER INTO AL
01E3  80 F4 48              XOR     AH,ACK+NOT_ERROR    ; FLIP A COUPLE OF BITS
01E6  EB AF                 JMP     B1                  ; RETURN FROM ROUTINE

               ;------ INITIALIZE THE PRINTER PORT

01E8                     B8:
01E8  50                    PUSH    AX                  ; SAVE AL
01E9  42                    INC     DX                  ; POINT TO OUTPUT PORT
01EA  42                    INC     DX
01EB  B0 08                 MOV     AL,SELECT           ; SET INIT LINE LOW, SLCT HIGH
01ED  EE                    OUT     DX,AL
01EE  B9 0352               MOV     CX,850              ; SET MAX TIME CNT TO 4.25 SEC
               ;
               ; WAIT FOR UP TO 4 SECONDS FOR COMPACT PRINTER TO DROP BUSY
               ; IF PARALLEL PRINTER THEN JUST WAIT 5 MSECS AND EXIT
               ;
01F1  51                 B10:    PUSH    CX
01F2  B9 050A               MOV     CX,5*MS_DELAY       ; DELAY FOR 5 MSECS
01F5  E2 FE                 LOOP    $
01F7  59                    POP     CX
01F8  83 FA 7A              CMP     DX,CPRT_MODE        ; COMPACT PRINTER?
01FB  75 1B                 JNE     B11                 ; EXIT LOOP IF NOT
               ;
               ;        WAIT FOR COMPACT PRINTER TO DROP BUSY
               ;
01FD  E4 79                 IN      AL,CPRT_STAT        ; CHK FOR BUSY DROP IN STATUS
01FF  A8 02                 TEST    AL,02               ; NOT BUSY?
0201  E1 EE                 LOOPZ   B10                 ; LOOP IF STILL BUSY
               ;
               ; DELAY 2 SECONDS AFTER BUSY DROP TO LET PRINTER MOVE HEAD TO HOME
               ;
0203  B3 08                 MOV     BL,SELECT
0205  B9 FBF4            B10_1:  MOV     CX,250*MS_DELAY
0208  E2 FE                 LOOP    $
020A  FE CB                 DEC     BL
020C  75 F7                 JNE     B10_1
               ;
               ; ACTIVATE PRINTER POWER
               ;
020E  B0 0C                 MOV     AL,SELECT+NO_INIT   ; NO INTERRUPTS, NON AUTO LF,
0210  EE                    OUT     DX,AL               ; SELECT AND INIT LINE HIGH

               ; DELAY 60 MSEC AFTER PRINTER POWER HAS BEEN ENABLED

0211  B9 3C78               MOV     CX,60*MS_DELAY      ; 60 MSEC DELAY
0214  E2 FE                 LOOP    $
0216  EB BD                 JMP     SHORT B6            ; EXIT ROUTINE

0218  B0 0C              B11:    MOV     AL,SELECT+NO_INIT ; NO IRPTS, NON AUTO LF,
021A  EE                    OUT     DX,AL               ; SELECT AND INIT LINE HIGH
021B  EB B8                 JMP     SHORT B6            ; PRT_STATUS_1
021D                     PRT_IO  ENDP

021D                     ROMCODE ENDS
                            END
```

Time of Day (B15TOD)

```
0000                    ROMCODE SEGMENT BYTE PUBLIC
                           ASSUME  CS:ROMCODE,DS:DATA
                           IDENT   B15TOD,15,00

                ;*********************************************************************
                ;
                ; MODULE-NAME :      B15TOD
                ;
                ;       DATE LAST MODIFIED:  9/12/85
                ;
                ; DESCRIPTIVE-NAME : TIME OF DAY SUPPORT
                ;
                ;
                ; COPYRIGHT : 7396-917 (C) COPYRIGHT IBM CORP. 1985
                ;             REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
                ;
                ; CHANGE LEVEL: 0.0
                ;
                ; FUNCTION:  TOD_PROC            ALLOW READ/SET OF REAL TIME CLOCK
                ;            SET_TOD             INITIALIZE TIMER DATA AREA
                ;            TMRO_INT8           SYSTEM TIMER INTERRUPT HANDLER
                ;            INITIALIZE_STATUS   INITIALIZE REAL TIME CLOCK
                ;            RTC2_TST            TEST REAL TIME CLOCK FOR TIME UPDATE
                ;
                ; MODULE SIZE:  936 BYTES
                ;
                ; ENTRY CONDITIONS:
                ;        REFER TO ROUTINE PROLOGUES
                ;
                ; EXIT CONDITIONS:
                ;        REFER TO ROUTINE PROLOGUES
                ;
                ; ROUTINES IN MODULE:
                ;        TOD_PROC                  ALLOW READ/SET OF REAL TIME CLOCK
                ;        SET_TOD                   INITIALIZE TIMER DATA AREA
                ;        TMRO_INT8                 TIMER 0 INTERRUPT HANDLER
                ;
                ; INTERNAL DATA AREAS / TABLES:   NONE
                ;
                ; EXTERNALLY REFERENCED ROUTINES: REFER TO EXTRN LIST
                ;
                ; EXTERNALLY REFERENCED DATA AREAS:  TIMER DATA AREA
                ;
                ; CHANGE ACTIVITY:  NONE
                ;
                ;*********************************************************************

                ;*****************************************************************
                ;*     E X T E R N A L   R E F E R E N C E S
                ;*****************************************************************

                           EXTRN   GET_RTC_NMI:NEAR
                           EXTRN   PUT_RTC_NMI:NEAR
                           EXTRN   DDS:NEAR
                           EXTRN   ENABLE_NMI:NEAR
                           EXTRN   DISABLE_NMI:NEAR
                           EXTRN   KB_NOISE:NEAR
```

```
;*****************************************************************
;*    P U B L I C S  D E C L A R A T I O N              *
;*****************************************************************

            PUBLIC  TOD_PROC
            PUBLIC  SET_TOD
            PUBLIC  TMRO_INT8
            PUBLIC  INITIALIZE_STATUS
            PUBLIC  RTC2_TST

;*****************************************************************
;*       S T A R T  O F  C O D E                          *
;*****************************************************************
```

Time of Day Interrupt Hex 1A (TOD_PROC)

```
            SUBTTL  TOD_PROC
;********************************************************************
;
; ROUTINE-NAME :  TOD_PROC   (INT 1AH)
;
; FUNCTION:   ALLOW REAL TIME CLOCK TO BE READ/SET
;       ALLOW TIMER DATA AREA TO BE READ/SET
;
; ENTRY AND EXIT CONDITIONS:
;
;       1.  PURPOSE OF ENTRY:    READ THE CURRENT CLOCK SETTING
;                                (TIMER DATA AREA)
;           INPUT CONDITIONS:      AH = 00H
;           NORMAL EXIT CONDITIONS:  AL = 0 IF TIMER HAS NOT PASSED
;                                          24 HRS SINCE LAST READ.
;                                       <>0 IF ON ANOTHER DAY.
;                                    CX = HIGH WORD OF COUNT
;                                    DX = LOW WORD OF COUNT
;           REGISTERS MODIFIED:      AL, CX, DX
;
;       2.  PURPOSE OF ENTRY:    SET CURRENT CLOCK (TIMER DATA AREA)
;           INPUT CONDITIONS:      AH = 01H
;                                  CX = HIGH WORD OF COUNT
;                                  DX = LOW WORD OF COUNT
;           REGISTERS MODIFIED:      AH
;
;       3.  PURPOSE OF ENTRY:    READ TIME OF REAL TIME CLOCK
;           INPUT CONDITIONS:      AH = 02H
;           NORMAL EXIT CONDITIONS:  CH = HOURS IN BCD
;                                    CL = MINUTES IN BCD
;                                    DH = SECONDS IN BCD
;                                    DL = 00 - NOT DAYLIGHT SAVINGS
;                                       = 01 - DAYLIGHT SAVINGS
;           ERROR EXIT CONDITIONS:   CF FLAG SET IF CLK NOT OPERATING
;           REGISTERS MODIFIED:      AX, CX, DX
;
;       4.  PURPOSE OF ENTRY:    SET TIME OF REAL TIME CLOCK
;           INPUT CONDITIONS:      AH = 03H
;                                  CH = HOURS IN BCD
;                                  CL = MINUTES IN BCD
;                                  DH = SECONDS IN BCD
;                                  DL = 1 IF DAYLIGHT SAVINGS OPTION.
;                                        ELSE 0.
;           RESTRICTIONS:        DL SHOULD = 1 OR = 0 ONLY.
;                                DL MADE TO = ITS OWN BITO.
;           REGISTERS MODIFIED:      AX
;
```

```
;        5. PURPOSE OF ENTRY:    READ DATE FROM REAL TIME CLOCK
;           INPUT CONDITIONS:      AH = 04H
;           NORMAL EXIT CONDITIONS:  CH = CENTURY IN BCD (19H OR 20H)
;                                    CL = YEAR IN BCD
;                                    DH = MONTH IN BCD
;                                    DL = DAY OF MONTH IN BCD
;           ERROR EXIT CONDITIONS:   CF FLAG SET IF CLK NOT OPERATING
;           REGISTERS MODIFIED:      AX, CX, DX
;
;        6. PURPOSE OF ENTRY:    SET DATE OF REAL TIME CLOCK
;           INPUT CONDITIONS:      AH = 05H
;                                  CH = CENTURY IN BCD (19H OR 20H)
;                                  CL = YEAR IN BCD
;                                  DH = MONTH IN BCD
;                                  DL = DAY OF MONTH IN BCD
;           REGISTERS MODIFIED:      AX
;
;        7. PURPOSE OF ENTRY:    SET ALARM TIME OF REAL TIME CLOCK
;           INPUT CONDITIONS:      AH = 06H
;                                  CH = HOURS IN BCD
;                                  CL = MINUTES IN BCD
;                                  DH = SECONDS IN BCD
;           NORMAL EXIT CONDITIONS:  CF FLAG = 0
;           ERROR EXIT CONDITIONS:   CF FLAG SET IF CLK NOT OPERATING
;                                CF FLAG SET IF ALARM ALREADY ENABLED
;           RESTRICTIONS:          THE USER MUST CODE A RTN AND PUT
;                                  CORRECT ADDRESS IN THE VECTOR TABLE
;                                  FOR INT 4AH.
;           REGISTERS MODIFIED:      AX
;
;        8. PURPOSE OF ENTRY:    RESET THE ALARM
;           INPUT CONDITIONS:      AH = 07H
;           NORMAL EXIT CONDITIONS:  NONE
;           ERROR EXIT CONDITIONS:   NONE
;           REGISTERS MODIFIED:      AX
;
;        9. PURPOSE OF ENTRY:    SET ALARM TIME OF RTC FOR POWER ON
;           INPUT CONDITIONS:      AH = 08H
;                                  CH = HOURS IN BCD
;                     CL = MINUTES IN BCD
;                     DH = SECONDS IN BCD
;           NORMAL EXIT CONDITIONS:  CF FLAG = 0
;           ERROR EXIT CONDITIONS:   CF FLAG SET IF CLK NOT OPERATING
;                                 CF FLAG SET IF ALARM ALREADY ENABLED
;           REGISTERS MODIFIED:      AX
;
;       10. PURPOSE OF ENTRY:    READ ALARM TIME OF RTC AND STATUS
;           INPUT CONDITIONS:      AH = 09H
;           NORMAL EXIT CONDITIONS:
;                       CH = HOURS IN BCD
;                       CL = MINUTES IN BCD
;                       DH = SECONDS IN BCD
;                       DL = ALARM STATUS
;                            00 - ALARM NOT ENABLED (AIE=0)
;                            01 - ALARM ENABLED BUT WILL NOT POWER
;                                 ON SYSTEM (AIE=1, EN_PON_ALRM=0)
;                            02 - ALARM ENABLED AND WILL POWER ON
;                                 SYSTEM (AIE=1, EN_PON_ALRM=1)
;           ERROR EXIT CONDITIONS:   CF FLAG SET IF CLK NOT OPERATING
;           REGISTERS MODIFIED:      AX, CX, DX
;
;       NOTE:  FUNCTION CALLS WITH AH NOT EQUAL TO THE
;              ABOVE RESULT IN NO OPERATION
;              WITH THE CARRY FLAGE BEING CLEARED
;
; INTERRUPTS:  FUNCTIONS 0, 1   FORCED OFF (CLI)
;              FUNCTIONS 2 - 9  FORCED ON (STI)
;              ON EXIT RESTORED AS WERE
;
```

```
              ; INTERNALLY REFERENCED ROUTINES:
              ;      INITIALIZE_STATUS      INITIALIZES RTC REGISTERS A THRU D
              ;      UPD_IN_PR              WAITS UNTIL CLOCK NOT BEING UPDATED
              ;      SET_ALRM               SETS RTC ALARM TIME BYTES AND ENABLES
              ;                             THE RTC ALARM
              ;
              ; EXTERNALLY REFERENCED ROUTINES:
              ;          DDS                    ESTABLISH SEGMENT
              ;          GET_RTC_NMI            ADDRESSES AND READS A RTC REGISTER
              ;                                 AH = REGISTER #, AL = DATA READ
              ;          PUT_RTC_NMI            ADDRESSES AND PUTS A RTC REGISTER
              ;                                 AH = REGISTER #, AL = DATA PUT
              ;          ENABLE_NMI             ENABLE NMI'S
              ;          DISABLE_NMI            DISABLE NMI'S
              ;
              ;*********************************************************************

= 0001                  SET_CF  EQU   01H

0000                    TOD_PROC    PROC   FAR
0000  FB                   STI                          ; INTERRUPTS ON FOR TIMER
0001  1E                   PUSH    DS                   ; SAVE SEGMENT
0002  E8 0000 E            CALL    DDS                  ; SET DATA SEGMENT
              ;
              ;-------CHECK FOR FUNCTION REQUESTS 0 AND 1-------------------------
              ;
0005                    RTC_CHK0:
0005  0A E4                OR      AH,AH                ; IF PARAMETER AH=0
0007  74 2B                JZ      RTC_0                ; THEN READ TIMER
0009  FE CC                DEC     AH                   ; IF PARAMETER AH=1
000B  74 3A                JZ      RTC_1                ; THEN SET TIME
000D  80 FC 09             CMP     AH,9                 ; CHECK IF VALID DATA PASSED IN AH
                                                        ; (0 <= AH <= 9)
0010  73 06                JAE     TOD_NRET             ; NORMAL RETURN IF NOT VALID
0012  E8 0000 E            CALL    DISABLE_NMI          ; DISABLE NMI'S
0015  EB 40 90             JMP     RTC_CHK1             ; GO CHECK FOR OTHER FUNCTIONS
              ;
              ;-------TOD_PROC NORMAL RETURN----------------------------------------
              ;
0018                    TOD_NRET:
0018  E8 0000 E            CALL    ENABLE_NMI           ; RE-ENABLE NMI'S
001B                    TOD_RET:
001B  FB                   STI                          ; ENABLE INTERRUPTS FOR TIMER
001C  1F                   POP     DS                   ; RECOVER SEGMENT
001D  55                   PUSH    BP
001E  8B EC                MOV     BP,SP
0020  80 66 06 FE          AND     BYTE PTR SS:[BP+6],NOT SET_CF
                                                        ; RST CARRY OF FLAGS IN STACK
0024  5D                   POP     BP
0025  CF                   IRET                         ; RETURN WITH GOOD CODE (CF=0)
              ;
              ;-------TOD_PROC ERROR RETURN-----------------------------------------
              ;
0026                    TOD_ERET:
0026  E8 0000 E            CALL    ENABLE_NMI
0029  FB                   STI                          ; ENABLE INTERRUPTS FOR TIMER
002A  1F                   POP     DS                   ; RECOVER SEGMENT
002B  55                   PUSH    BP
002C  8B EC                MOV     BP,SP
002E  80 4E 06 01          OR      BYTE PTR SS:[BP+6-,SET_CF ; SET CARRY OF FLAGS
0032  5D                   POP     BP
0033  CF                   IRET                         ; RETURN WITH ERROR CD (CF=1)
              ;
              ;-------G E T   T I M E R   D A T A----- FUNCTION 00H ---------------
              ;
0034                    RTC_0:
0034  FA                   CLI                          ; DISABLE INTERRUPTS FOR READ
0035  A0 0070 R            MOV     AL,TIMER_OFL         ; GET OVERFLOW, AND
0038  C6 06 0070 R 00      MOV     TIMER_OFL,0          ; RESET THE FLAG
003D  8B 0E 006E R         MOV     CX,TIMER_HIGH        ; CX RETURNS TIMER_HIGH DATA
0041  8B 16 006C R         MOV     DX,TIMER_LOW         ; DX RETURNS TIMER_LOW DATA
0045  EB D4                JMP     TOD_RET              ; NORMAL RETURN WITH TMR DATA
              ;
              ;-------S E T   T I M E R-------------- FUNCTION  01H -------------
```

```
                ;
0047                    RTC_1:
0047  FA                   CLI                          ; DISABLE INTERRUPTS FOR READ
0048  89 16 006C R         MOV     TIMER_LOW,DX         ; SET TIMER_LOW = TO PARAM DX
004C  89 0E 006E R         MOV     TIMER_HIGH,CX        ; SET TIMER_HIGH = TO PARAM CX
0050  C6 06 0070 R 00      MOV     TIMER_OFL,0          ; RESET OVERFLOW
0055  EB C4                JMP     TOD_RET              ; NORMAL RTN AFTER SETTING TMR
                ;
                ;-------CHECK FOR FUNCTION REQUESTS 2 AND 3-------------------------
                ;
0057                    RTC_CHK1:
0057  FE CC                DEC     AH                   ; IF PARAMETER AH=2
0059  74 07                JZ      RTC_2                ; THEN GET RTC TIME
005B  FE CC                DEC     AH                   ; IF PARAMETER AH=3
005D  74 2D                JZ      RTC_3                ; THEN SET RTC TIME
005F  E9 00EE R            JMP     RTC_CHK2             ; GO CHECK FOR OTHER FUNCTIONS
                ;
                ;-------G E T   R T C   T I M E--------- FUNCTION 02H ---------------
                ;
0062                    RTC_2:
0062  E8 01FE R            CALL    UPD_IN_PR            ; CLOCK OPERATING?
0065  73 02                JNC     RTC_2A               ; YES, GO AROUND
0067  EB BD                JMP     TOD_ERET             ; NO, ERROR RETURN
0069                    RTC_2A:
0069  B4 00                MOV     AH,RTC_TSEC          ; ADDRESS AND
006B  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_TSEC (Reg0)
006E  8A F0                MOV     DH,AL                ; DH RETURNS TSEC DATA

0070  B4 02                MOV     AH,RTC_TMIN          ; ADDRESS AND
0072  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_TMIN (Reg2)
0075  8A C8                MOV     CL,AL                ; CL RETURNS TMIN DATA

0077  B4 04                MOV     AH,RTC_THRS          ; ADDRESS AND
0079  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_THRS (Reg4)
007C  8A E8                MOV     CH,AL                ; CH RETURNS THRS DATA

007E  2A D2                SUB     DL,DL                ; SET NOT-DAYLIGHT-SAVINGS
0080  B4 0B                MOV     AH,RTC_MODE          ; GET RTC_MODE (RegB)
0082  E8 0000 E            CALL    GET_RTC_NMI
0085  A8 01                TEST    AL,SET_DAYLIGHT      ; DAYLIGHT SAVINGS SET?
0087  74 01                JZ      RTC_2B               ; NO, JUMP AROUND
0089  42                   INC     DX                   ; YES, SET DL=01
008A  EB 8C             RTC_2B: JMP     TOD_NRET        ; NORMAL RETURN WITH TIME
                ;
                ;-------S E T   R T C   T I M E--------- FUNCTION 03H ----------------
                ;
008C                    RTC_3:
008C  E8 01FE R            CALL    UPD_IN_PR            ; CLOCK OPERATING?
008F  73 03                JNC     RTC_3A               ; YES, GO AROUND
0091  E8 01E5 R            CALL    INITIALIZE_STATUS    ; NO, INIT CONTROL/STATUS
                                                        ; REG.S (A-D)
0094                    RTC_3A:
0094  B4 00                MOV     AH,RTC_TSEC          ; ADDRESS AND SET
0096  8A C6                MOV     AL,DH                ; RTC_TSEC (Reg0) = TO
0098  E8 0000 E            CALL    PUT_RTC_NMI          ; TSEC PARAMETER, CH

009B  B4 02                MOV     AH,RTC_TMIN          ; ADDRESS AND SET
009D  8A C1                MOV     AL,CL                ; RTC_TMIN (Reg2) = TO
009F  E8 0000 E            CALL    PUT_RTC_NMI          ; TMIN PARAMETER, CL

00A2  B4 04                MOV     AH,RTC_THRS          ; ADDRESS AND SET
00A4  8A C5                MOV     AL,CH                ; RTC_THRS (Reg4) = TO
00A6  E8 0000 E            CALL    PUT_RTC_NMI          ; THRS PARAMETER, CH

00A9  B4 0B                MOV     AH,RTC_MODE          ; ADDRESS AND
00AB  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_MODE (RegB)
00AE  24 71                AND     AL,AIE_ENABLE+UIE_ENABLE+PIE_ENABLE+SET_DAYLIGHT
                                                        ; RESET UNUSED FLAGS
                                                        ; CLEAR ALL OTHERS.
00B0  80 E2 01             AND     DL,SET_DAYLIGHT      ; SET DL = TO ITS BIT0
00B3  0A C2                OR      AL,DL                ; IF DSE=1, KEEP, ELSE DSE=DL.
00B5  0C 02                OR      AL,SET_24HR          ; TURN ON 24HR-MODE
00B7  E8 0000 E            CALL    PUT_RTC_NMI          ; ADDR AND SET RTC_MODE (REG B)
```

```
00BA  B4 0E                   MOV     AH,RTC_DIAG_STAT    ; ADDRESS AND
00BC  E8 0000 E               CALL    GET_RTC_NMI         ; GET RTC_DIAG_STAT (RegE)
00BF  24 7F                   AND     AL,NOT RTC_TIME_BAD ; RESET RTC_TIME_BAD FLAG (=0)
00C1  E8 0000 E               CALL    PUT_RTC_NMI         ; PUT BACK RTC_DIAG_STAT (RegE)
00C4  E9 0018 R               JMP     TOD_NRET            ; NORMAL RTN AFTER SETTING TIME
                ;
                ;-------G E T   R T C   D A T E--------- FUNCTION 04H ----------------
                ;
00C7                       RTC_4:
00C7  E8 01FE R               CALL    UPD_IN_PR           ; CLOCK OPERATING?
00CA  73 03                   JNC     RTC_4A              ; YES, GO AROUND
00CC  E9 0026 R               JMP     TOD_ERET            ; NO, ERROR RETURN
00CF                       RTC_4A:
00CF  B4 07                   MOV     AH,RTC_MDAY         ; ADDRESS AND
00D1  E8 0000 E               CALL    GET_RTC_NMI         ; GET RTC_MDAY (Reg7)
00D4  8A D0                   MOV     DL,AL               ; DL RETURNS DAY OF MON. DATA

00D6  B4 08                   MOV     AH,RTC_MON          ; ADDRESS AND
00D8  E8 0000 E               CALL    GET_RTC_NMI         ; GET RTC_MON (Reg8)
00DB  8A F0                   MOV     DH,AL               ; DH RETURNS MONTH DATA

00DD  B4 09                   MOV     AH,RTC_YEAR         ; ADDRESS AND
00DF  E8 0000 E               CALL    GET_RTC_NMI         ; GET RTC_YEAR (Reg9)
00E2  8A C8                   MOV     CL,AL               ; CL RETURNS YEAR DATA

00E4  B4 32                   MOV     AH,RTC_CENTURY      ; ADDRESS AND
00E6  E8 0000 E               CALL    GET_RTC_NMI         ; GET RTC_CENTURY (Reg50/32H)
00E9  8A E8                   MOV     CH,AL               ; CH RETURNS CENTURY DATA

00EB  E9 0018 R               JMP     TOD_NRET            ; NORMAL RETURN WITH DATE
                ;
                ;-------CHECK FOR FUNCTION REQUESTS 4, 5, AND 6---------------------
                ;
00EE                       RTC_CHK2:
00EE  FE CC                   DEC     AH                  ; IF PARAMETER AH=4
00F0  74 D5                   JZ      RTC_4               ; THEN GET RTC DATE
00F2  FE CC                   DEC     AH                  ; IF PARAMETER AH=5
00F4  74 07                   JZ      RTC_5               ; THEN SET RTC DATE
00F6  FE CC                   DEC     AH                  ; IF PARAMETER AH=6
00F8  74 3B                   JZ      RTC_6               ; THEN SET RTC ALARM
00FA  EB 59 90                JMP     RTC_CHK3            ; GO CHECK FOR REMAINING FUNCT
                ;
                ;-------S E T   R T C   D A T E--------- FUNCTION 05H ----------------
                ;
00FD                       RTC_5:
00FD  E8 01FE R               CALL    UPD_IN_PR           ; CLOCK OPERATING?
0100  73 03                   JNC     RTC_5A              ; YES, GO AROUND
0102  E8 01E5 R               CALL    INITIALIZE_STATUS   ; NO, INIT CONTROL/STATUS
                                                          ; REG.S (A-D)
0105                       RTC_5A:
0105  B4 06                   MOV     AH,RTC_WDAY         ; ADDRESS AND
0107  B0 00                   MOV     AL,0                ; CLEAR
0109  E8 0000 E               CALL    PUT_RTC_NMI         ; RTC_WDAY (Reg6)

010C  B4 07                   MOV     AH,RTC_MDAY         ; ADDRESS AND SET
010E  8A C2                   MOV     AL,DL               ; RTC_MDAY (Reg7) = TO
0110  E8 0000 E               CALL    PUT_RTC_NMI         ; DAY OF MON. PARAMETER, DL

0113  B4 08                   MOV     AH,RTC_MON          ; ADDRESS AND SET
0115  8A C6                   MOV     AL,DH               ; RTC_MON (Reg8) = TO
0117  E8 0000 E               CALL    PUT_RTC_NMI         ; MONTH PARAMETER, DH

011A  B4 09                   MOV     AH,RTC_YEAR         ; ADDRESS AND SET
011C  8A C1                   MOV     AL,CL               ; RTC_YEAR (Reg9) = TO
011E  E8 0000 E               CALL    PUT_RTC_NMI         ; YEAR PARAMETER, CL

0121  B4 0B                   MOV     AH,RTC_MODE         ; ADDRESS AND
0123  E8 0000 E               CALL    GET_RTC_NMI         ; GET RTC_MODE (RegB)
0126  24 7F                   AND     AL,NOT SET_CLOCK    ; TURN OFF SET-MODE ONLY
0128  E8 0000 E               CALL    PUT_RTC_NMI         ; ADDR AND SET RTC_MODE (RegB)

012B  B4 32                   MOV     AH,RTC_CENTURY      ; ADDRESS AND SET
012D  8A C5                   MOV     AL,CH               ; RTC_CENTURY (Reg50/32H) = TO
012F  E8 0000 E               CALL    PUT_RTC_NMI         ; CENTURY PARAMETER, CH
```

```
0132  E9 0018 R            JMP     TOD_NRET             ; NML RTN AFTER SETTING DATE
               ;
               ;-------S E T   R T C   A L A R M------- FUNCTION 06H ----------------
               ;
0135                     RTC_6:
0135  E8 01A9 R            CALL    SET_ALRM             ; SET ALARM REGISTERS
0138  3D 0000              CMP     AX,0                 ; GOOD RET_CODE RETURNED?
013B  75 03                JNE     RTC_6A               ; ...YES, CONTINUE.
013D  E9 0026 R            JMP     TOD_ERET             ; ...NO, ERROR RET.
0140                     RTC_6A:
0140  E9 0018 R            JMP     TOD_NRET             ; NML RTRN AFTER SETTING ALARM
               ;
               ;-------R E S E T   R T C   A L A R M--- FUNCTION 07H ----------------
               ;
0143                     RTC_7:
0143  B4 0B                MOV     AH,RTC_MODE          ; ADDRESS AND
0145  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_MODE (RegB)
0148  24 DF                AND     AL,NOT AIE_ENABLE    ; RESET ALARM INT ENABLE
014A  E8 0000 E            CALL    PUT_RTC_NMI          ; ADDR & PUT RTC_MODE (RegB)

014D  80 26 00A0 R FB      AND     RTC_WAIT_FLAG,NOT PON_ALRM_PEND ; RST PWR BY ALARM

0152  E9 0018 R            JMP     TOD_NRET             ; NML RTRN RESETTING ALARM
               ;
               ;-------CHECK FOR FUNCTION REQUESTS 7, 8, AND 9----------------------
               ;
0155                     RTC_CHK3:
0155  FE CC                DEC     AH                   ; IF PARAMETER AH=7
0157  74 EA                JZ      RTC_7                ; THEN RESET ALARM
0159  FE CC                DEC     AH                   ; IF PARAMETER AH=8
015B  74 03                JZ      RTC_8                ; THEN SET RTC ALARM POWER-ON
015D  EB 14 90             JMP     RTC_9                ; PARAM AH=9, GET ALARM TIME &
                                                        ; STATUS
               ;
               ;-------S E T   R T C   A L A R M   F O R   P O W E R   O N-----------
               ;                                FUNCTION 08H
0160                     RTC_8:
0160  E8 01A9 R            CALL    SET_ALRM             ; SET ALARM REGISTERS
0163  3D 0000              CMP     AX,0                 ; GOOD RET_CODE RETURNED?
0166  75 03                JNE     RTC_8A               ; ...YES, CONTINUE.
0168  E9 0026 R            JMP     TOD_ERET             ; ...NO, ERROR RET.
016B  80 0E 00A0 R 04    RTC_8A: OR      RTC_WAIT_FLAG,PON_ALRM_PEND
                                                        ; SET PWR ON BY ALARM PENDING
0170  E9 0018 R            JMP     TOD_NRET             ; NML RTRN AFTER SETTING ALARM
               ;
               ;-------G E T   R T C   A L A R M   A N D   S T A T U S---------------
               ;                                FUNCTION 09H
0173                     RTC_9:
0173  E8 01FE R            CALL    UPD_IN_PR            ; CLOCK OPERATING?
0176  73 03                JNC     RTC_9A               ; YES, GO AROUND
0178  E9 0026 R            JMP     TOD_ERET             ; NO, ERROR RETURN
017B                     RTC_9A:
017B  B4 01                MOV     AH,RTC_ASEC          ; ADDRESS AND
017D  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_ASEC (Reg1)
0180  8A F0                MOV     DH,AL                ; DH RETURNS ASEC DATA

0182  B4 03                MOV     AH,RTC_AMIN          ; ADDRESS AND
0184  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_AMIN (Reg3)
0187  8A C8                MOV     CL,AL                ; CL RETURNS AMIN DATA

0189  B4 05                MOV     AH,RTC_AHRS          ; ADDRESS AND
018B  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_AHRS (Reg5)
018E  8A E8                MOV     CH,AL                ; CH RETURNS AHRS DATA

0190  2A D2                SUB     DL,DL                ; SET DL TO ZERO
0192  B4 0B                MOV     AH,RTC_MODE          ; ADDRESS AND
0194  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_MODE (RegB)
0197  A8 20                TEST    AL,AIE_ENABLE        ; RTC ALARM DISABLED (AIE=0)?
0199  74 0B                JZ      EXIT9                ; YES, RETURN DL=0
019B  FE C2                INC     DL                   ; DL=1
019D  F6 06 00A0 R 04      TEST    RTC_WAIT_FLAG,PON_ALRM_PEND
```

```
                                                          ; IS PWR ON BY ALARM PENDING?
01A2  74 02               JZ      EXIT9                   ; NO, RETURN DL=1
01A4  FE C2               INC     DL                      ; YES, RETURN DL=2
01A6  E9 0018 R        EXIT9:  JMP     TOD_NRET           ; NML RTRN W/ALRM TIME & STAT

                ;-------SUBROUTINES-----------------------------------------------
                ;-----------------------------------------------------------------
                ; SET_ALARM
                ;
                ; FUNCTION:  IF RTC ALARM IS DISABLED AND THE CLOCK IS OPERATING, SET
                ;            THE ALARM TIME AND ENABLE THE RTC ALARM.
                ;
                ; INPUT:   DH = ALARM SECONDS IN BCD
                ;          CL = ALARM MINUTES IN BCD
                ;          CH = ALARM HOURS IN BCD
                ;
                ; OUTPUT:  AX = 0  ERROR RETURN CODE
                ;          AX <> 0  GOOD RETURN CODE
                ;
                ; REGISTERS MODIFIED:  AX
                ;-----------------------------------------------------------------

01A9                   SET_ALRM  PROC  NEAR
01A9  B4 0B               MOV     AH,RTC_MODE             ; ADDRESS AND
01AB  E8 0000 E           CALL    GET_RTC_NMI             ; GET RTC_MODE (RegB)
01AE  A8 20               TEST    AL,AIE_ENABLE           ; IF ALARM DISABLED (AIE=0)
01B0  74 03               JZ      CONT1                   ; THEN CONTINUE
01B2  33 C0               XOR     AX,AX                   ; ELSE CLEAR AX AND
01B4  C3                  RET                             ; RETURN
01B5                   CONT1:
01B5  E8 01FE R           CALL    UPD_IN_PR               ; CLOCK OPERATING?
01B8  73 03               JNC     CONT2                   ; YES, GO AROUND
01BA  E8 01E5 R           CALL    INITIALIZE_STATUS       ; NO, INIT CONTROL/STATUS
                                                          ; REG.S (A-D)
01BD                   CONT2:
01BD  B4 01               MOV     AH,RTC_ASEC             ; ADDRESS AND SET
01BF  8A C6               MOV     AL,DH                   ; RTC_ASEC (Reg1) = TO
01C1  E8 0000 E           CALL    PUT_RTC_NMI             ; ASEC PARAMETER, DH

01C4  B4 03               MOV     AH,RTC_AMIN             ; ADDRESS AND SET
01C6  8A C1               MOV     AL,CL                   ; RTC_AMIN (Reg3) = TO
01C8  E8 0000 E           CALL    PUT_RTC_NMI             ; AMIN PARAMETER, CL

01CB  B4 05               MOV     AH,RTC_AHRS             ; ADDRESS AND SET
01CD  8A C5               MOV     AL,CH                   ; RTC_AHRS (Reg5) = TO
01CF  E8 0000 E           CALL    PUT_RTC_NMI             ; AHRS PARAMETER, CH

01D2  B4 0B               MOV     AH,RTC_MODE             ; ADDRESS AND
01D4  E8 0000 E           CALL    GET_RTC_NMI             ; GET RTC_MODE (RegB)
01D7  24 7F               AND     AL,NOT SET_CLOCK        ; SET MODE OFF (SET=0)
01D9  0C 20               OR      AL,AIE_ENABLE           ; ENABLE ALARM INT. (AIE=1)
01DB  E8 0000 E           CALL    PUT_RTC_NMI             ; ADDR AND SET RTC_MODE (RegB)

01DE  E4 61               IN      AL,NMI_CNTL             ; ENABLE RTC NMI FOR ALARM
01E0  24 F7               AND     AL,NOT DIS_ALARM
01E2  E6 61               OUT     NMI_CNTL,AL

01E4  C3                  RET                             ; RETURN TO RTC_6 OR RTC_8
01E5                   SET_ALRM  ENDP
```

```
                ;-----------------------------------------------------------------
                ; INITIALIZE_STATUS
                ;
                ; FUNCTION:  INITIALIZE RTC CONTROL AND STATUS REGISTERS
                ;            REG A = 976.6 USEC PERIODIC INTERRUPT
                ;                    32.768 KHZ TIME BASE
                ;            REG B = ENABLE SET, 24HR, BCD MODES
                ;                    DISABLE ALL INTERRUPTS AND DAYLIGHT SAVINGS
                ;            READ REG C AND D TO CLEAR STATUS
                ;
                ; INPUT:   NONE
                ;
                ; OUTPUT:  NONE
                ;
                ; REGISTERS MODIFIED:  AX
                ;-----------------------------------------------------------------
01E5                    INITIALIZE_STATUS       PROC    NEAR
01E5  B4 0A                MOV     AH,RTC_UP_STAT       ; ADDRESS AND INITIALIZE
01E7  B0 26                MOV     AL,26H               ; RTC_UP_STAT (RegA) TO GIVE
01E9  E8 0000 E            CALL    PUT_RTC_NMI          ; **INT.=976.6 MICRO SEC.

01EC  B4 0B                MOV     AH,RTC_MODE          ; ADDRESS AND INITIALIZE
01EE  B0 82                MOV     AL,SET_CLOCK+SET_24HR ; RTC_MODE (RegB):  (=82H)
01F0  E8 0000 E            CALL    PUT_RTC_NMI          ; * MODES ON...SET, 24HR, BCD
                                                        ; * DSABLE ALL OTH (NO DYLGHT)

01F3  B4 0C                MOV     AH,RTC_INT_STAT      ; GET RTC_INT_STAT (RegC)
01F5  E8 0000 E            CALL    GET_RTC_NMI          ; TO INITIALIZE IT
01F8  B4 0D                MOV     AH,RTC_COND_STAT     ; GET RTC_COND_STAT (RegD)
01FA  E8 0000 E            CALL    GET_RTC_NMI          ; TO INITIALIZE IT
01FD  C3                   RET                          ; RETURN TO CALLER
01FE                    INITIALIZE_STATUS       ENDP

                ;-----------------------------------------------------------------
                ; UPD_IN_PR
                ;
                ; FUNCTION: WAIT FOR RTC CLOCK TO UPDATE TIME.  SET CARRY IF CLK NOT
                ;           OPERATING (DOESN'T LEAVE UPDATE MODE AFTER 600 MSEC).
                ;
                ; INPUT:   NONE
                ;
                ; OUTPUT:  AX = 0   CLOCK NOT UPDATING
                ;          CARRY FLAG = 1  CLOCK NOT OPERATING
                ;                     = 0  CLOCK HAS BEEN UPDATED
                ;
                ; REGISTERS MODIFIED:  AX
                ;-----------------------------------------------------------------
01FE                    UPD_IN_PR   PROC  NEAR
01FE  51                   PUSH    CX                   ; SAVE CX
01FF  B9 0258              MOV     CX,600               ; SET LOOP COUNT
0202                    UPDATE:
0202  B4 0A                MOV     AH,RTC_UP_STAT       ; ADDRESS AND
0204  E8 0000 E            CALL    GET_RTC_NMI          ; GET RTC_UP_STAT (RegA)
0207  A8 80                TEST    AL,RTC_UIP           ; IF NOT UPDATING (UIP=0)
0209  74 05                JZ      UPD_IN_PREND         ; THEN RETURN
020B  E2 F5                LOOP    UPDATE               ; ELSE IF CX <> 0 THEN LOOP
020D  33 C0                XOR     AX,AX                ; ELSE CLEAR AX AND
020F  F9                   STC                          ; SET CARRY FOR ERROR SIGNAL
0210                    UPD_IN_PREND:
0210  59                   POP     CX                   ; RESTORE CX
0211  C3                   RET                          ; RETURN TO CALLER
0212                    UPD_IN_PR       ENDP

0212                    TOD_PROC        ENDP

                           SUBTTL  SET TIME OF DAY
                ;*********************************************************************
                ;
                ; ROUTINE-NAME :  SET_TOD
                ;
                ; FUNCTION:   SET TIMER DATA AREA FROM RTC TIME OR INITIAL TO ZERO
                ;             (FOR COLD_MODE)
                ;
                ; ENTRY CONDITIONS:
                ;       PURPOSE OF ENTRY: SET TIMER DATA AREA
                ;       INPUT CONDITIONS: NONE
                ;       RESTRICTIONS:     NONE
                ;
                ; EXIT CONDITIONS:
                ;       NORMAL EXIT CONDITIONS: TIMER DATA AREA SET FROM RTC TIME
                ;       ERROR EXIT CONDITIONS:  TIMER DATA AREA SET TO ZERO
                ;       REGISTERS MODIFIED:     AX RETURNS STATUS CODE FOR DISPATCHER
                ;                     AX = 4000  NON-FATAL ERROR  AX = 0  GOOD
                ;
                ; INTERRUPTS:  LEFT AS ARE
                ;
                ; INTERNALLY REFERENCED ROUTINES: CVT_BINARY
                ;
                ; EXTERNALLY REFERENCED ROUTINES: DDS, GET_RTC_NMI, PUT_RTC_NMI,
                ;                                 INT_1A
                ;
                ;*********************************************************************

0212                        SET_TOD PROC    NEAR

0212  1E                       PUSH    DS                  ; SAVE SEGMENT

                ;-------RESET TIMER DATA AREA
0213  2B C0                    SUB     AX,AX               ; RESET
0215  A2 0070 R                MOV     TIMER_OFL,AL        ; TIMER
0218  A3 006C R                MOV     TIMER_LOW,AX        ; DATA
021B  A3 006E R                MOV     TIMER_HIGH,AX       ; AREA

021E  B4 0E                    MOV     AH,RTC_DIAG_STAT    ; ADDRESS AND
0220  E8 0000 E                CALL    GET_RTC_NMI         ; GET RTC_DIAG_STAT (RegE)
0223  24 84                    AND     AL,RTC_TIME_BAD+RTC_FAILED
                                                           ; TIME BAD OR CLOCK ERROR?
0225  75 56                    JNZ     TOD_NOTSET          ; YES.  TIMER SET TO ZERO

                ;-------GET RTC TIME
0227  F8                       CLC                         ; SET CF=0 "NO ERROR" STATE
0228  B4 02                    MOV     AH,GET_RTC_TIME     ; FOR INT_1A, AH=2=GET RTC
022A  CD 1A                    INT     1AH                 ; GET RTC TIME: CH=H CL=M DH=S
022C  72 3F                    JC      SET_RET             ; IF CF=1, NML RETURN.  CF=1
                                                           ; CLK NOT OPERATING (UIP=1)

                ;-------CHECK VALIDITY OF RTC TIME
022E  80 FE 59                 CMP     DH,SEC_MAX_LIMIT    ; IF RTC_TSEC NOT IN LIMITS
0231  77 3E                    JA      BAD_TIME            ; THEN RETURN
0233  80 F9 59                 CMP     CL,MIN_MAX_LIMIT    ; IF RTC_TMIN NOT IN LIMITS
0236  77 39                    JA      BAD_TIME            ; THEN RETURN
0238  80 FD 23                 CMP     CH,HRS_MAX_LIMIT    ; IF RTC_THRS NOT IN LIMITS
023B  77 34                    JA      BAD_TIME            ; THEN RETURN

                ;-------SET TIMER FROM RTC TIME
023D  8A C6                    MOV     AL,DH               ; BCD TSEC INTO AL
023F  E8 0286 R                CALL    CVT_BINARY          ; BINARY TSEC RETURNED IN AL
0242  B3 12                    MOV     BL,COUNTS_SEC       ; BL <-- # OF COUNTS PER SEC
0244  F6 E3                    MUL     BL                  ; AX <-- # OF CNTS IN RTC_TSEC
0246  8B D0                    MOV     DX,AX               ; DX IS TOTAL-CNTS ACCUMULATOR

0248  8A C1                    MOV     AL,CL               ; BCD TMIN INTO AL
024A  E8 0286 R                CALL    CVT_BINARY          ; BINARY TMIN RETURNED IN AL
024D  BB 0444                  MOV     BX,COUNTS_MIN       ; BX <-- # OF COUNTS PER MIN
0250  52                       PUSH    DX                  ; SAVE TOTAL
0251  F7 E3                    MUL     BX                  ; AX <-- # OF CNTS IN RTC_TMIN
0253  5A                       POP     DX                  ; RETRIEVE TOTAL
0254  03 D0                    ADD     DX,AX               ; ADD TMIN-CNTS TO TOTAL-CNTS
```

```
0256  8A C5                MOV    AL,CH               ; BCD THRS INTO AL
0258  E8 0286 R            CALL   CVT_BINARY          ; BINARY THRS RETURNED IN AL
025B  8B C8                MOV    CX,AX               ; CX = HIGH WD  (THRS-CNTS   X
                                                      ; # OF COUNTS PER HR.)
025D  B3 07                MOV    BL,COUNTS_HOUR      ; BL <-- # CNTS PER H (LOW WD)
                                                      ; **TRUE CNTS/H IS 17 BIT VAL
025F  F6 E3                MUL    BL                  ; AX <-- # OF COUNTS IN
                                                      ; RTC_THRS (LOW WORD)
0261  03 C2                ADD    AX,DX               ; AX = TOTAL-COUNTS (LOW WORD)
0263  83 D1 00             ADC    CX,0000H            ; CX = TOTAL-COUNTS (HIGH WD)

0266  89 0E 006E R         MOV    TIMER_HIGH,CX       ; MOVE TOTAL-COUNTS
026A  A3 006C R            MOV    TIMER_LOW,AX        ; TO TIMER DATA AREA

               ;-------RETURN ROUTINES
026D                    SET_RET:
026D  2B C0                SUB    AX,AX               ; SET GOOD RETURN CODE
026F  1F                   POP    DS                  ; RETSORE SEGMENT
0270  C3                   RET                        ; RETURN TO CALLER

0271                    BAD_TIME:
0271  B4 0E                MOV    AH,RTC_DIAG_STAT    ; ADDRESS AND
0273  E8 0000 E            CALL   GET_RTC_NMI         ; GET RTC_DIAG_STAT (RegE)
0276  0C 04                OR     AL,RTC_FAILED       ; SET RTC_FAILED BIT (=1)
0278  E8 0000 E            CALL   PUT_RTC_NMI         ; PUT RTC_DIAG_STAT (RegE)
027B  EB F0                JMP    SET_RET             ; RETURN WITH GOOD RETURN CODE

027D                    TOD_NOTSET:
027D  A8 04                TEST   AL,RTC_FAILED       ; IS RTC_FAILED SET (=1)?
027F  75 EC                JNZ    SET_RET             ; YES, RTN WITH GOOD RTN CODE
0281  B8 4000              MOV    AX,NON_FATAL_ERR*100H ; NO, RTN WITH ERR CD IN AH
0284  1F                   POP    DS                  ; RESTORE SEGMENT
0285  C3                   RET                        ; RETURN TO CALLER

               ;-------SUBROUTINES
               ;-------------------------------------------------------------------
               ; CVT_BINARY
               ;
               ; FUNCTION:  CONVERT A 1 BYTE BCD NUMBER TO BINARY
               ;
               ; INPUT:   AL = BCD NUMBER
               ; OUTPUT:  AH = 0,  AL = BINARY NUMBER
               ;
               ; RESTRICTIONS:  INVALID BCD NUMBER RESULTS IN INVALID BINARY NUMBER.
               ;                NO RANGE CHECKING DONE
               ;------------------------------------------------------------------

0286                    CVT_BINARY      PROC    NEAR  ; INPUT:    AL = BCD NUMBER
                                                      ; OUTPUT: AH = 0,  AL = BINARY
0286  8A E0                MOV    AH,AL               ; PLACE LEFT
0288  51                   PUSH   CX                  ; BCD DIGIT
0289  B1 04                MOV    CL,4                ; INTO RIGHT
028B  D2 EC                SHR    AH,CL               ; NIBBLE OF
028D  59                   POP    CX                  ; AH
028E  24 0F                AND    AL,0FH              ; GET RID OF LEFT BCD DIGIT AL
0290  D5 0A                AAD                        ; CONVERT UNPACKED BCD TO BIN
                                                      ; (BINARY # IN AL, AH=0)
0292  C3                   RET                        ; RETURN TO CALLER
0293                    CVT_BINARY      ENDP
0293                    SET_TOD ENDP
```

```
                ;*******************************************************************
                ;
                ; ROUTINE-NAME :  RTC2_TST
                ;
                ; FUNCTION:   CHECK REAL TIME CLOCK FOR TIME UPDATE FUNCTION.
                ;        IF RTC_TIME_BAD FLAG IS SET INDICATED INVALID TIME, A
                ;        CALL TO INITIALIZE_STATUS IS MADE TO SET THE RTC REGISTERS
                ;        AND THE TIME,DATA, AND ALARM REGISTERS ARE SET TO 0.
                ;
                ; ENTRY CONDITIONS:
                ;       PURPOSE OF ENTRY: CALLED BY POST TO CHECK RTC TIME FUNCTION
                ;       INPUT CONDITIONS: DS:DATA
                ;       RESTRICTIONS:     NONE
                ;
                ; EXIT CONDITIONS:
                ;       NORMAL EXIT CONDITIONS: AX = 0
                ;       ERROR EXIT CONDITIONS:  AH=40 (NON_FATAL ERR) AL=63 (RTC ERR)
                ;       REGISTERS MODIFIED:     AX RETURNS STATUS CODE FOR POST
                ;
                ; INTERRUPTS:  LEFT AS ARE
                ;
                ; INTERNALLY REFERENCED ROUTINES: INITIALIZE_STATUS
                ;
                ; EXTERNALLY REFERENCED ROUTINES:  GET_RTC_NMI, PUT_RTC_NMI
                ;
                ;*******************************************************************

0293                    RTC2_TST PROC NEAR

0293  B4 0E                MOV     AH,RTC_DIAG_STAT    ; CHECK FOR CLOCK NOT SET
0295  E8 0000 E            CALL    GET_RTC_NMI
0298  A8 80                TEST    AL,RTC_TIME_BAD
029A  74 11                JZ      RTC2_02             ; JUMP IF CLOCK IS SET
                ;
                ; MUST INITIALIZE AND SET TIME TO 0:0:0 DATE TO 0:0:0
                ;
029C  E8 01E5 R            CALL    INITIALIZE_STATUS
029F  B4 00                MOV     AH,RTC_TSEC
02A1  B0 00                MOV     AL,0
02A3  B9 000A              MOV     CX,10               ; SET TIME AND DATE TO 0

02A6                    RTC2_01:
02A6  E8 0000 E            CALL    PUT_RTC_NMI         ; SET REGISTER TO 0
02A9  FE C4                INC     AH                  ; INCREMENT REGISTER NUMBER
02AB  E2 F9                LOOP    RTC2_01
                ;
                ; INSURE SET_CLOCK  BIT IS RESET
                ;
02AD                    RTC2_02:
02AD  B4 0B                MOV     AH,RTC_MODE
02AF  E8 0000 E            CALL    GET_RTC_NMI
02B2  24 7F                AND     AL,NOT SET_CLOCK    ; TURN OFF SET_CLOCK
02B4  E8 0000 E            CALL    PUT_RTC_NMI

02B7  B3 02                MOV     BL,02               ; OUTER LOOP COUNTER
02B9  2B C9                SUB     CX,CX               ; SET MAX LOOP TO 1.3  SECONDS
                ;
                ; WAIT FOR UPDATE IN PROGRESS FLAG TO BE SET
                ;
02BB                    RTC2_03:
02BB  B0 0A                MOV     AL,RTC_UP_STAT      ; READ CLOCK UPDATE STATUS
02BD  E6 70                OUT     RTCR_PORT,AL
02BF  E4 71                IN      AL,RTCD_PORT
02C1  A8 80                TEST    AL,RTC_UIP          ; UPDATE IN PROGRESS?
02C3  75 09                JNZ     RTC2_04             ; JUMP IF YES
02C5  E2 F4                LOOP    RTC2_03             ; KEEP LOOKING
02C7  FE CB                DEC     BL
02C9  75 F0                JNE     RTC2_03
02CB  EB 10 90             JMP     RTC2_FAIL           ; JUMP IF NEVER CAME ON
                ;
                ; WAIT FOR UPDATE IN PROGRESS FLAG TO BE RESET
                ;
02CE                    RTC2_04:
02CE  B9 0258              MOV     CX,600              ; SET LOOP FOR 6 MSEC
```

```
02D1                   RTC2_05:
02D1  B0 0A               MOV     AL,RTC_UP_STAT        ; READ CLOCK UPDATE STATUS
02D3  E6 70               OUT     RTCR_PORT,AL
02D5  E4 71               IN      AL,RTCD_PORT
02D7  A8 80               TEST    AL,RTC_UIP            ; UPDATE IN PROGRESS?
02D9  74 14               JZ      RTC2_INTS             ; JUMP IF RESET
02DB  E2 F4               LOOP    RTC2_05               ; KEEP LOOKING
              ;
              ; RTC FAILED TO UPDATE
              ;
02DD                   RTC2_FAIL:
02DD  B4 0E               MOV     AH,RTC_DIAG_STAT
02DF  E8 0000 E           CALL    GET_RTC_NMI
02E2  0C 04               OR      AL,RTC_FAILED         ; SET CLOCK FAILED FLAG
02E4  E8 0000 E           CALL    PUT_RTC_NMI
02E7  B8 4063             MOV     AX,4063H              ; SET NON-FATAL ERROR FLAG
02EA  B3 01               MOV     BL,01                 ; SET MFG ERROR CODE
02EC  EB 46 90            JMP     RTC2_EXIT
              ;
              ; TEST PERIODIC INTERRUPT FOR POST AFTER 100 MSECS
              ;
02EF                   RTC2_INTS:
02EF  1E                  PUSH    DS                    ; SET ES:BX PTR RTC_WAIT_FLAG
02F0  07                  POP     ES
02F1  BB 00A0 R           MOV     BX,OFFSET RTC_WAIT_FLAG ; SET UP FOR INT 15 CALL
02F4  B9 0001             MOV     CX,01
02F7  BA 86A0             MOV     DX,86A0H              ; POST AFTER 100 MSECS
02FA  B8 8300             MOV     AX,8300H              ; POST ON EVENT FUNCTION CALL
02FD  CD 15               INT     15H
02FF  B2 52               MOV     DL,82                 ; SET MAX TIME CNT TO 110 MSEC
              ;
              ; WAIT FOR POST FLAG TO BE SET BY NMI HNDLR - EACH CNT ON DL = 1.35MS
              ;
0301                   RTC2_WAIT:
0301  26: F6 07 80        TEST    BYTE PTR ES:[BX-,PSTD ; WAIT FOR POSTED FLAG
0305  75 0B               JNZ     INT_DONE
0307  B9 0102             MOV     CX,MS_DELAY           ; DELAY FOR 1 MSEC
030A  E2 FE               LOOP    $
030C  FE CA               DEC     DL
030E  75 F1               JNE     RTC2_WAIT
0310  EB 0D               JMP     SHORT RTC2_BAD        ; BAD IF > 110 MSECS

0312                   INT_DONE:
0312  80 FA 0F            CMP     DL,15                 ; OKAY IF > = 90 MSECS
0315  77 08               JA      RTC2_BAD
0317  26: 80 27 7F        AND     BYTE PTR ES:[BX-,NOT PSTD ; CLR POSTED FLAG
031B  2B C0               SUB     AX,AX                 ; SET GOOD RETURN CODE
031D  EB 15               JMP     SHORT RTC2_EXIT       ; EXIT TEST
              ;
              ; PERIODIC INTERRUPT TEST FAILED
              ;
031F                   RTC2_BAD:
031F  B4 0B               MOV     AH,RTC_MODE
0321  B0 00               MOV     AL,0                  ; CLEAR ALL INTERRUPT ENABLES
0323  E8 0000 E           CALL    PUT_RTC_NMI
0326  FE C4               INC     AH
0328  E8 0000 E           CALL    GET_RTC_NMI           ; RD STAT TO CLR PENDING INTS
032B  26: C6 07 00        MOV     BYTE PTR ES:[BX],0    ; CLEAR WAIT FLAG
032F  B8 4063             MOV     AX,4063H              ; SET NON-FATAL ERROR FLAG
0332  B3 02               MOV     BL,02                 ; SET MFG ERR CODE   INTS BAD
              ;
              ; PERIODIC INTERRUPT TEST PASSED
              ;
0334                   RTC2_EXIT:
0334  50                  PUSH    AX
0335  E4 61               IN      AL,NMI_CNTL
0337  0C 08               OR      AL,DIS_ALARM          ; DISABLE ALARM NMI
0339  E6 61               OUT     NMI_CNTL,AL
033B  58                  POP     AX
033C  C3                  RET
033D                   RTC2_TST ENDP
```

Timer 0 Interrupt 8 (TMR0_INT8)

```
              ;*******************************************************************
              ;
              ; TMRO_INT8
              ;
              ; DESCRIPTION:
              ;
              ; 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.
              ; 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.
              ;
              ; INPUT:  DS = DATA
              ;
              ; OUTPUT:  AX=0 (GOOD RETURN)
              ;          AH=40 (NON_FATAL ERROR) AL=63 (RTC ERROR)
              ;
              ; RESTRICTIONS:     NONE
              ;
              ; REGISTERS MODIFIED:  NONE
              ;
              ; INTERRUPTS:  FORCED ON (STI)
              ;
              ; INTERNALLY REFERENCED ROUTINES:  NONE
              ;
              ; EXTERNALLY REFERENCED ROUTINES:  DDS, INT 1CH, INT 4AH
              ;*******************************************************************
033D                   TMRO_INT8       PROC    FAR
033D  FB                  STI                          ; INTERRUPTS BACK ON
033E  1E                  PUSH    DS
033F  50                  PUSH    AX
0340  52                  PUSH    DX                   ; SAVE MACHINE STATE
0341  E8 0000 E           CALL    DDS
0344  FE 0E 007B R        DEC     EVENT_TIM_OUT        ; DEC WAIT ON EVNT TIMEOUT CTR
0348  FF 06 006C R        INC     TIMER_LOW            ; INCREMENT TIME
034C  75 04               JNZ     T4                   ; TEST_DAY

034E  FF 06 006E R        INC     TIMER_HIGH           ; INCREMENT HIGH WORD OF TIME
              ;
              ; CHECK TIMER FOR 24 HOUR ROLL OVER
              ;
0352                   T4:                             ; TEST_DAY
0352  83 3E 006E R 18     CMP     TIMER_HIGH,COUNTS_DAY_HI ; TST FOR CNT = 24 HOURS
0357  75 15               JNZ     T5                   ; DISKETTE_CTL
0359  81 3E 006C R 00B0   CMP     TIMER_LOW,COUNTS_DAY_LO
035F  75 0D               JNZ     T5                   ; DISKETTE_CTL

              ;------ TIMER HAS GONE 24 HOURS

0361  2B C0               SUB     AX,AX
0363  A3 006E R           MOV     TIMER_HIGH,AX
0366  A3 006C R           MOV     TIMER_LOW,AX
0369  C6 06 0070 R 01     MOV     TIMER_OFL,1

              ;------ TEST FOR DISKETTE TIME OUT

036E                   T5:                             ; DISKETTE_CTL
036E  80 3E 0040 R 00     CMP     MOTOR_COUNT,0        ; SKIP MOTOR OFF IF ALREADY 0
```

```
0373  74 1D                   JE      T6
0375  FE 0E 0040 R            DEC     MOTOR_COUNT
0379  75 17                   JNZ     T6                    ; RETURN IF COUNT NOT OUT
                      ;
                      ; DEGATE DISKETTE DRIVES BEFORE TURNING OFF MOTOR TO INSURE THAT A
                      ; RE-SELECT OF A DRIVE WITHIN 800 USECS OF MTR OFF WILL NOT GLITCH DRV
                      ; LINES
                      ;
037B  E4 77                   IN      AL,DSKT_CNTL          ; DEGATE DISKETTE DRIVES
037D  A8 80                   TEST    AL,DSKT_NMI           ; DISKETTE POWER NMIS ENABLED?
037F  74 06                   JZ      T5A                   ; JUMP IF NOT
0381  24 BF                   AND     AL,NOT FDC_PWR
0383  0C 20                   OR      AL,DSKT_DEGATE        ; ONLY DEGATE IF NMI ENABLED
0385  E6 77                   OUT     DSKT_CNTL,AL
                      ;
                      ; TURN OFF DISKETTE MOTORS AND DESELECT DRIVE
                      ;
0387  80 26 003F R C0    T5A:    AND     MOTOR_STATUS,0C0H ; TURN OFF MTR STATUS BITS
038C  B0 04                   MOV     AL,FDC_RUN            ; TRN OFF MTRS SLCTS & DMA/INTS
038E  BA 03F2                 MOV     DX,DRIVE_CNTL
0391  EE                      OUT     DX,AL
0392                     T6:
0392  CD 1C                   INT     1CH                   ; XSFER CONTROL TO A USER RTNE
                      ;
                      ; END OF INTERRUPT
                      ;
0394  B0 20                   MOV     AL,EOI
0396  E6 20                   OUT     INTA00,AL             ; END OF INTERRUPT TO CNTLR
                      ;
                      ; CHECK FOR ANY OTHER INTERRUPTS IN SERVICE
                      ;
0398  B0 0B                   MOV     AL,READ_ISR           ; SET TO READ INSERVICE REG
039A  E6 20                   OUT     INTA00,AL
039C  E4 20                   IN      AL,INTA00             ; READ IN-SERVICE-REG
039E  0A C0                   OR      AL,AL                 ; ANYTHING IN SERVICE?
03A0  75 0E                   JNZ     T7                    ; YES, ALARM SERVICE MUST WAIT
                      ;
                      ; CHECK FOR USER RTC ALARM ROUTINE PENDING ACTIVATION
                      ;
03A2  F6 06 00A0 R 02         TEST    RTC_WAIT_FLAG,ALARM_PEND ; ALRM INTERRUPT PENDING?
03A7  74 07                   JZ      T7                    ; JUMP IF NOT
                      ;
                      ; CALL USER ALARM ROUTINE DUE TO RTC ALARM PENDING FLAG BEING SET
                      ;
03A9  80 26 00A0 R FD         AND     RTC_WAIT_FLAG,NOT ALARM_PEND ; TURN OFF FLAG
03AE  CD 4A                   INT     4AH                   ; XSFER CTOL TO USR ALARM RTNE
03B0                     T7:
03B0  5A                      POP     DX
03B1  58                      POP     AX
03B2  1F                      POP     DS                    ; RESET MACHINE STATE
03B3  CF                      IRET                          ; RETURN FROM INTERRUPT
03B4                     TMR0_INT8       ENDP


03B4                     ROMCODE ENDS
                              END
```

System Services (B16SYSV)

```
0000                    ROMCODE SEGMENT BYTE PUBLIC
                           ASSUME  CS:ROMCODE
                           IDENT   B16SYSV,16,00

                ;******************************************************************************
                ;
                ; INT 15H
                ;
                ; MODULE-NAME:          B16SYSV
                ;
                ; CHANGE LEVEL:         0.0
                ;
                ; DATE LAST MODIFIED:   9/12/85
                ;
                ; DESCRIPTIVE-NAME:   SYSTEM SERVICES (INT 15H) FUNCTION CALL SUPPORT
                ;                     ROUTINES.
                ;
                ; COPYRIGHT:       7396-917 (C) COPYRIGHT IBM CORP. 1985
                ;                  REFER TO COPYRIGHT INSTRUCTIONS FORM NO. G120-2083
                ;
                ; FUNCTION:             REFER TO ROUTINE PROLOGUE BELOW.
                ;
                ; MODULE SIZE:          1014 BYTES.
                ;
                ; ENTRY CONDITIONS:
                ;
                ;  PURPOSE OF ENTRY: EXECUTE SYSTEM SERVICE INDICATED BY INPUT PARAM.
                ;  INPUT CONDITIONS: REFER TO ROUTINE PROLOGUES BELOW.
                ;  RESTRICTIONS: FOR SPECIAL RESTRICTIONS REFER TO INDIVIDUAL FUNCTION
                ;                ROUTINE PROLOGUES.
                ;  INTERNALLY
                ;  REFERENCED ROUTINES: SYS_PROF, EXT_EVENT, SYS_POWER_OFF, SYS_STAUS,
                ;                  POST_INTV, WAIT_INTV, DEV_BUSY, TMRO_CHK,
                ;                  INT_COMPLETE
                ;
                ; EXIT CONDITIONS:
                ;
                ;  NORMAL EXIT
                ;  CONDITIONS:          REFER TO ROUTINE PROLOGUE BELOW.
                ;  ERROR EXIT
                ;  CONDITIONS: CARRY FLAG AND AH = 86H IS SET FOR INVALID FUNCTION
                ;              REQUEST OR INVALID PARAMETER ERROR.
                ;
                ; REGISTERS MODIFIED:   AH & RETURNED OUTPUT PARAM (REFER TO PLOGUES).
                ;
                ; INTERNAL DATA AREAS/
                ; TABLES             :  SYS_TABLE.
                ;
                ; EXTERNALLY
                ; REFERENCED ROUTINES:  REFER TO EXTERNAL REFERENCES LIST.
                ;
                ; EXTERNALLY REFERENCED
                ; DATA AREAS          : REFER TO EXTERNAL REFERENCES LIST
                ; CHANGE ACTIVITY: NONE
                ;*************************************************************************
```

```
                 ;*******************************************************************
                 ;
                 ;                E X T E R N A L   R E F E R E N C E S
                 ;
                 ;*******************************************************************

                             EXTRN   GET_RTC_REG      :NEAR
                             EXTRN   PUT_RTC_REG      :NEAR
                             EXTRN   DISABLE_NMI      :NEAR
                             EXTRN   ENABLE_NMI       :NEAR
                             EXTRN   DDS              :NEAR
                             EXTRN   VIDEO_IO_1       :NEAR
                             EXTRN   COM_POWER        :NEAR
                             EXTRN   BAT_SAV_SETUP    :NEAR
                             EXTRN   TMRO_INT8        :NEAR
                             EXTRN   SYS_DESCR_TABLE  :NEAR
                             EXTRN   COMMO_IO         :NEAR
                             EXTRN   MODEM_POWER_ON   :NEAR
                             EXTRN   SEND_COM         :NEAR
                             EXTRN   MODEM_CONFIG     :NEAR

                 ;*******************************************************************
                 ;
                 ;                          P U B L I C S
                 ;
                 ;*******************************************************************

                             PUBLIC  SYS_SERVICES
                             PUBLIC  EXT_EVENT

                             SUBTTL  SYSTEM SERVICE ROUTINES

0000                      SYS_TABLE LABEL   WORD

                 ;ENTRIES FOR 40H - 44H
0000  0083 R                 DW      OFFSET SYS_PROF      ; 40H - READ/MOD SYS PROFILE
0002  01BE R                 DW      OFFSET EXT_EVENT     ; 41H - WAIT ON EXTERNAL EVENT
0004  0298 R                 DW      OFFSET SYS_POWER_OFF ; 42H - SYSTEM POWER OFF
0006  02C1 R                 DW      OFFSET SYS_STATUS    ; 43H - READ SYSTEM STATUS
0008  030F R                 DW      OFFSET SYS_MODEM_PWR ; 44H - MODEM POWER CONTROL

                 ;ENTRIES FOR 80H - 86H
000A  0472 R                 DW      OFFSET RETURN        ; 80H - DEVICE OPEN
000C  0472 R                 DW      OFFSET RETURN        ; 81H - DEVICE CLOSE
000E  0472 R                 DW      OFFSET RETURN        ; 82H - PROGRAM TERMINATION
0010  032A R                 DW      OFFSET POST_INTV     ; 83H - POST ON ELPS TME INTV
0012  007F R                 DW      OFFSET JOYSTICK      ; 84H - JOYSTICK SUPPORT
0014  0472 R                 DW      OFFSET RETURN        ; 85H - SYSTEM REQUEST
0016  0380 R                 DW      OFFSET WAIT_INTV     ; 86H - WAIT ON ELPS TME INTV

                 ;ENTRIES FOR 90H - 91H
0018  03DB R                 DW      OFFSET DEV_BUSY      ; 90H - DEVICE BUSY
001A  0472 R                 DW      OFFSET INT_COMPLETE  ; 91H - DEV INTERRUPT COMPLETE
```

System Services Interrupt Hex 15 (SYS_SERVICES)

```
                 ;-------------------------------------------------------------------
                 ;
                 ; ROUTINE-NAME :  INT 15H, SYSTEM SERVICES
                 ;
                 ; FUNCTION:       PROVIDES ACCESS TO SYSTEM SERVICES.
                 ;
                 ;  ENTRY/EXIT CONDITIONS
                 ;
                 ;                 INPUT:                            OUTPUT:
```

```
;  CASSETTE I/O:
;                AH = 0-3                         AH = 86H, CARRY SET
;                                                 PORT NOT PRESENT
;
;  READ/MODIFY
;  SYSTEM PROFILE: AH = 40H
;                  AL = 0 READ                    BX,CX CONTAINS
;                                                 PROFILE INFORMATION
;                  AL = 1 MODIFY                  PROFILE IS MODIFIED
;                      BX,CX CONTAINS PROFILE     AND EXECUTED
;                  AL = 2 READ MODEM CONFIG
;                  AL = 3 MODIFY MODEM CONFIG
;                  (REFER TO SYS_PROF PROLOG FOR DETAILS)
;
;  EXTERNAL EVENT: AH = 41H
;                  (REFER TO EXT_EVENT PROLOG FOR DETAILS)
;
;  SYSTEM POWER OFF:
;                  AH = 42H
;                  (REFER TO SYS_POWER_OFF PROLOG FOR DETAILS)
;
;  SYSTEM STATUS:  AH = 43H
;                  (REFER TO SYS_STATUS PROLOG FOR DETAILS)
;
;  MODEM POWER:    AH = 44H
;                  (REFER TO SYS_MODEM_POWER PROLOG FOR DETAILS)
;
;  KEYBOARD INTERCEPT:
;                  AH = 4FH
;          ENTRY CONDITION: AL = SCAN CODE OF KEY FROM INT 9 (PORT 60)
;          EXIT CONDITION:  CARRY SET IF INT 9 SHOULD PROCESS KEY
;                           CARRY CLEAR IF INT 9 SHOULD NOT PROCESS
;                           KEY BUT JUST ISSUE EOI AND EXIT LEVEL.
;
;   (RESERVED FOR OPERATING SYSTEM USE - THIS BIOS WILL RETURN WITH Y
;                                        CARRY SET TO
;                                        ENABLE INT 9 TO HANDLE KEY)
;
;  DEVICE OPEN:
;                  AH = 80H                       IMMEDIATE RETURN
;                  BX = DEVICE ID
;                  CX = PROCESS ID
;
;  DEVICE CLOSE:
;                  AH = 81H                        IMMEDIATE RETURN
;                  BX = DEVICE ID
;                  CX = PROCESS ID
;
;  PROGRAM
;  TERMINATION:    AH = 82H                        IMMEDIATE RETURN
;                  BX = DEVICE ID
;
;  POST ON TIME
;  INTERVAL   :    AH = 83H
;                  AH = 0 SET INTERVAL            BIT 7 SET IN ES:-BX-
;                  ES.BX PTR TO BYTE IN CALLERS   WHEN INTERVAL HAS
;                  STORAGE FOR POST NOTIFICATON   ELAPSED
;                  CX,DX NUMBER OF MICROSECONDS TO
;                  ELAPSE BEFORE POSTING.
;                  AL = 1 CANCEL THE INTERVAL POST POST IS CANCELLED
;
;  JOYSTICK
;  SUPPORT:        AH = 84H                       AH = 86H, CARRY SET
;                                                 PORT NOT PRESENT
;
;  SYSTEM REQ.
;  KEY CHANGE:     AH = 85H                       IMMEDIATE RETURN
;                  AL = 00 - MAKE OF KEY
;                  AL = 01 - BREAK OF KEY
;
```

```
;  WAIT ON
;  TIME INTERVAL:  AH = 86H
;                  CX,DX NUMBER OF MSECONDS       RETURN AFTER TIME
;                  TO ELAPSE BEFORE RETURN        ELAPSED XXX
;                  TO CALLER
;
;  DEVICE BUSY:
;                  AH = 90H
;                  AL = DEVICE TYPE (SEE CODE)
;                       01 - DISKETTE             RETURN AFTER IRPT
;                                                 RECEIVED OR 2 SECOND
;                                                 TIMEOUT HAS OCCURRED
;                                                 CARRY SET IF TIMEOUT
;
;                  02 - KEYBOARD                  RETURN WHEN KEYBOARD
;                                                 BUFFER TAIL <>
;                                                 CURRENT  VALUE
;
;                  03 - FFH ALL OTHERS            IMMEDIATE RETURN
;
;  DEVICE
;  INTERRUPT       AH = 91H                       IMMEDIATE RETURN
;  COMPLETE:       AL = DEVICE TYPE
;                     00H -> 7FH
;                       SERIALLY REUSABLE DEVICES
;                       OPERATING SYSTEM MUST SERIALIZE
;                       ACCESS
;                     80H -> BFH
;                       REENTRANT DEVICES; ES:BX IS
;                       USED TO DISTINGUISH DIFFERENT
;                       CALLS (MULTIPLE I/O CALLS ARE
;                       ALLOWED SIMULTANEUSLY)
;                     C0H -> FFH
;                       WAIT ONLY CALLS; THERE IS NO
;                       COMPLEMENTARY 'POST' FOR THESE
;                       WAITS - - THESE ARE TIMEOUT
;                       ONLY.  TIMES ARE FUNCTION NUMBER
;                       DEPENDENT.
;
;                  DEVICE  TYPE  DESCRIPTION      TIMEOUT
;
;                      00H = DISK                   YES
;                      01H = DISKETTE               YES
;                      02H = KEYBOARD                NO
;                      FDH = DISKETTE MOTOR START   YES
;
;  READ SYSTEM
;  DESCRIPTION     AH = C0H                       AH=0, CARRY CLEAR
;  TABLE     :                                    ES:BX TABLE POINTER
;                  TABLE DEFINED AS FOLLOWS:
;                   BYTE 1&2: LENGTH OF TABLE IN BYTES
;                             STARTING AT BYTE 3
;                   BYTE   3: SYSTEM MODEL BYTE
;                   BYTE   4: SECONDARY MODEL BYTE
;                   BYTE   5: BIOS REVISION NUMBER
;                   BYTE   6: FEATURE INFORMATION BYTE 1:
;                     MSB:
;                      BIT 7 = 1 - BIOS USES DMA CHANNEL 3
;                              0 - DMA CHANNEL 3 NOT USED
;                      BIT 6 = 1 - 2ND INTERRUPT CONTROLLER INSTALLED
;                              0 - 2ND INTERRUPT CNTLR NOT INSTALLED
;                      BIT 5 = 1 - REAL TIME CLOCK PRESENT
;                              0 - REAL TIME CLOCK NOT PRESENT
;                      BIT 4 = 1 - INT 15H FUNCTION 4FH LINKAGE FROM
;                                  BIOS INT9 SUPPORTED
;                              0 - LINKAGE NOT SUPPORTED
;                      Bit 3 = 1 - INT 15H FUNCTION 41H WAIT ON
;                                  EXTERNAL EVENT SUPPORTED
;                              0 - INT 15H FUNCTION 41H NOT SUPPORTED
;                     Bit 2-0=  RESERVED
;                  BYTES
;                    7-10: RESERVED FOR FUTURE FEATURE EXPANSION
;
```

```
                ; REGISTERS MODIFIED : AH AND RETURNED PARAMETER REGISTERS
                ;
                ; INVALID FUNCTION REQUESTS (INVALID REGISTER AH VALUES) WILL RETURN  AH = 86H
                ; AND THE CARRY FLAG WILL BE SET.
                ;
                ;-----------------------------------------------------------------------------
                                ASSUME  DS:DATA,ES:NOTHING

001C                        SYS_SERVICES PROC    FAR
001C  53                        PUSH    BX
001D  8B DC                     MOV     BX,SP              ; GET POINTER TO SYSTEM STACK
001F  36: 8B 5F 06              MOV     BX,SS:[BX+6]       ; GET FLAGS FROM SYSTEM STACK
0023  53                        PUSH    BX                 ; & SAVE IN STK FOR FLG REG LD
0024  9D                        POPF                       ; RETORE FLAGS
0025  5B                        POP     BX                 ; RESTORE BX
0026  80 FC 40                  CMP     AH,40H             ; BELOW SYS SVCS 1 (CSST I/O)?
0029  72 4E                     JB      INV_FUN            ; INVLD PARM OR UNSUPTD CSST
002B  80 FC 80                  CMP     AH,80H             ; IS SELECT PARM ABOVE 80H?
002E  73 0A                     JAE     SYS_SV2            ; YES, CK SYS SVCS 2 REQUEST
                ;
                ; CHECK FOR SYSTEM SERVICES 1 (AH = 40-44H)
                ;
0030  80 FC 44                  CMP     AH,44H             ; IN VALID SYS SVCS UP RANGE?
0033  77 44                     JA      INV_FUN            ; NO, INVALID FUNCTION
0035  80 EC 40                  SUB     AH,40H             ; YES, SET TABLE OFFSET
0038  EB 17                     JMP     SHORT SYS_SVC      ; GO LINK TO PROPER ROUTINE...
                ;
                ; CHECK FOR SYSTEM SERVICES  2 (AH = 80-91H)
                ;
003A                        SYS_SV2:
003A  80 FC 91                  CMP     AH,91H             ; IN VLD SYS SVCS 2 UP RANGE?
003D  77 2C                     JA      CHK_DESCR          ; YES, CK READ DESCRIPTOR...
003F  80 EC 7B                  SUB     AH,7BH             ; MAKE REL TO 4TH ENTRY IN TBL
0042  80 FC 15                  CMP     AH,15H             ; FUNCTION > = 90H?
0045  72 05                     JB      SYS_SV3            ; NO, GO SYS SERV 3 REQUEST...
0047  80 EC 09                  SUB     AH,9               ; YES, CORRECT THE OFFSET
004A  EB 05                     JMP     SHORT SYS_SVC      ; GO LINK TO THE ROUTINE...
                ;
                ; CHECK FOR FUNCTIONS 87H-8FH THAT ARE INVALID
                ;
004C                        SYS_SV3:
004C  80 FC 0C                  CMP     AH,12              ; IS PARM =  87H - 8FH CODE?
004F  73 28                     JAE     INV_FUN            ; YES, INVALID PARAMETER...
                ;
                ; LINK TO PROPER ROUTINE
                ;
0051                        SYS_SVC:
0051  1E                        PUSH    DS                 ; SAVE DS,BP,AX USE IN SY SERV
0052  55                        PUSH    BP
0053  50                        PUSH    AX
0054  BD ---- R                 MOV     BP,DATA            ; SET DS <-- DATA SEGMENT - BP
0057  8E DD                     MOV     DS,BP
0059  2A C0                     SUB     AL,AL              ; SET AL TO 0
005B  86 E0                     XCHG    AH,AL              ; & FUNCT CODE IN LOW BYTE &
                                                           ; AH=0 RET CODE CLR
005D  D1 E0                     SHL     AX,1               ; DOUBLE LINK TABLE OFFSET
005F  8B E8                     MOV     BP,AX              ; GET OFFSET IN BP
0061  58                        POP     AX                 ; RESTORE AX
0062  2E: FF 96 0000 R          CALL    SYS_TABLE[BP]      ; JUMP TO PROPER ROUTINE
0067  5D                        POP     BP                 ; RESTORE BP,DS
0068  1F                        POP     DS
0069  EB 11                     JMP     SHORT IMM_RET      ; SERVICE EXECUTED LEAVE...
                ;
                ; CHECK FOR READ SYSTEM DESCRIPTOR TABLE ADDRESS
                ;
006B                        CHK_DESCR:
006B  80 FC C0                  CMP     AH,0C0H            ; IS REQ PARM SYS DESC READ?
006E  75 09                     JNE     INV_FUN            ; NO, INVALID FN PARM REQUEST
0070  0E                        PUSH    CS                 ; SAVE CURRENT CODE SEGMENT
0071  07                        POP     ES                 ; SET ES TO POINT TO TABLE
0072  BB 0000 E                 MOV     BX,OFFSET SYS_DESCR_TABLE ; POINT TO TABLE
0075  2A E4                     SUB     AH,AH              ; CLEAR RETURN CODE
0077  EB 03                     JMP     SHORT IMM_RET      ; EXIT
                ;
                ; INVALID FUNCTION CODE RETURN POINT
                ;
0079                        INV_FUN:
0079  B4 86                    MOV     AH,86H           ; SET BAD COMMAND RETURN CODE
007B  F9                       STC                      ; SET CARRY FLAG ON - ERROR
007C                        IMM_RET:
007C  CA 0002                  RET     2                ; RETURN WITH CURRENT FLAGS
007F                        SYS_SERVICES ENDP

007F                        JOYSTICK        PROC    NEAR
007F  B4 86                    MOV     AH,86H           ; SET JOYSTICK NOT SUPPORTED
0081  F9                       STC                      ; SET CARRY FLAG - ERROR
0082  C3                       RET
0083                        JOYSTICK ENDP
```

System Profile Services (SYS_PROF)

```
;---------------------------------------------------------------------
;
; INT 15H
;
; ROUTINE-NAME:  SYS_PROF  (AH = 40H)
;
; FUNCTION:      ROUTINE TO ALLOW READ/MODIFY OPS ON THE SYS PROFILE
;                INFORMATION.  SYS PROFILE IS STORED IN CLK CHIP RAM.
;
; INPUT:    AL = 0 READ SYSTEM PROFILE INTO BX AND CX
;           AL = 1 MODIFY SYSTEM PROFILE FROM BX AND CX
;           AL = 2 READ MODEM SWITCH SETTING INTO BX
;           AL = 3 WRITE MODEM SETTINGS FROM BX AND TURN ON MODEM
;
;
;DEFINITION OF BITS IN SYSTEM PROFILE REGISTERS (AL = 0,1)
;---------------------------------------------------------
;  BH:     BIT 7 - 0 = SET SYSTEM COLD START MODE
;                 *1 = SET SYSTEM WARM START MODE
;
;          BIT 6 - 0 = DISABLE LOW BATTERY WARNING MESSAGE
;                 *1 = ENABLE LOW BATTERY WARNING MESSAGE
;
;          BITS 5,4 - INITIAL VIDEO MODE
;                   00 = RESERVED
;                   01 = 40 X 25 MONOCHROME USING CGA/LCD
;                  *10 = 80 X 25 MONOCHROME USING CGA/LCD
;                   11 = 80 X 25 MONOCHROME USING MONO/LCD
;
;         BITS 3,2 - *00 = IGNORE LCD HIGH INTENSITY ATTRIBUTE
;                     01 = MAP LCD HIGH INTENSITY TO UNDERSCORE
;                     10 = MAP LCD HIGH INTENSITY TO REVERSE VIDEO
;                     11 = MAP LCD HIGH INTENSITY TO ALTERNATE FONT
;
;         BIT 1 - *0 = INTERNAL MODEM NOT AVAILABLE ON BATTERY PWR
;                  1 = INTERNAL MODEM AVAILABLE ON BATTERY POWER
;
;         BIT 0 - *0 = RS232/PARALLEL NOT AVAILABLE ON BATTERY
;                  1 = RS232/PARALLEL AVAILABLE ON BATTERY
;
;  BL:   BITS 7-0   -    RESERVED
;
;  CH:   BITS 7-0 - KEYBOARD INACTIVITY TIME BEFORE LCD BLANKING.
;                   (TIME IN MINUTES, *0 = DISABLE BLANKING).
;
```

```
                ;  CL:   BITS 7-0 - KEYBOARD INACTIVITY TIME BEFORE SYSTEM POWER
                ;                   OFF (TIME IN MINUTES, *0 = DISABLE POWER OFF).
                ;
                ;
                ;        DEFINITION OF BITS IN MODEM PROFILE REGISTER (AL = 2,3)
                ;        ------------------------------------------------------
                ;  BL:   BIT 6,7 - NOT USED.
                ;
                ;        BIT 5 -  *0 = MANUAL ANSWER. / 1 = AUTO ANSWER.
                ;
                ;        BITS 4-2 - B4  B3  B2      PARITY AND FRAMING
                ;                   ---------------------------------
                ;                    0   0   0      MARK, 7 BITS DATA
                ;                    0   0   1      SPACE, 7 BITS DATA
                ;                    0   1   0      ODD, 7 BITS DATA
                ;                   *0   1   1      EVEN, 7 BITS DATA
                ;                    1   0   0      NONE, 8 BITS DATA
                ;                    1   0   1      RESERVED
                ;                    1   1   0      RESERVED
                ;                    1   1   1      RESERVED
                ;
                ;       BITS 1-0 - B1  B0         MODEM DATA RATE
                ;                  ----------------------------------
                ;                   0   0          110 BITS PER SECOND
                ;                   0   1          300 BITS PER SECOND
                ;                  *1   0         1200 BITS PER SECOND
                ;                 **1   1         2400 BITS PER SECOND
                ;
                ;                 *DEFAULT SETTINGS AFTER STANDY POWER LOST
                ;                **NOT PRESENTLY SUPPORTED
                ;
                ;  BH:     RESERVED FOR FUTURE USE.
                ;
                ;
                ; OUTPUT: AL = 80H IF MODEM CANNOT BE CONFIGURED OR IS NOT PRESENT.
                ;         AL = 00H IF MODEM CONFIGURATION IS PERFORMED OK.
                ;         AL = UNDEFINED FOR OTHER FUNCTIONS.
                ;
                ; REGISTERS
                ; MODIFIED:      AX
                ;
                ;--------------------------------------------------------------------

0083                    SYS_PROF        PROC    NEAR
0083  3C 01              CMP     AL,1                   ; IS REQUEST FOR MODIFY?
0085  75 03              JNE     SYS1
0087  E9 0125 R          JMP     SPROF_MODIFY           ; YES, GO SYS PROF MODIFY..
008A  3C 00             SYS1:           CMP     AL,0    ; NO,  REQUEST FOR SYS READ?
008C  74 57              JE      SPROF_READ             ; YES, EXECUTE SYS PROF READ..
008E  3C 02              CMP     AL,2                   ; NO,  IS REQ FOR MODEM READ?
0090  74 08              JE      MPROF_READ             ; YES
0092  3C 03              CMP     AL,3                   ; NO,  IS REQ FOR MODEM WRITE?
0094  74 15              JE      MPROF_MODIFY           ; YES
0096  F9                 STC                            ; SET ERROR RETURN CODE
0097  E9 01BD R          JMP     SPROF_OUT              ; EXIT ROUTINE
                ;
                ; READ MODEM PROFILE
                ;
009A  B4 1D             MPROF_READ:     MOV     AH,RTC_MOD_PROF1 ; ADDR PROFILE BYTE
009C  E8 0000 E          CALL    GET_RTC_REG            ; READ IT
009F  8A D8              MOV     BL,AL
00A1  FE C4              INC     AH                     ; ADDRESS NEXT BYTE
00A3  E8 0000 E          CALL    GET_RTC_REG            ; READ IT
00A6  8A F8              MOV     BH,AL
00A8  E9 01BB R          JMP     SPROF_EXIT             ; RETURN TO CALLER
                ;
                ; WRITE MODEM PROFILE AND CONFIGURE MODEM (IF POWER ON)
                ;
00AB  B4 1D             MPROF_MODIFY:   MOV     AH,RTC_MOD_PROF1 ; ADDR PROFILE BYTE
00AD  8A C3              MOV     AL,BL
00AF  E8 0000 E          CALL    PUT_RTC_REG            ; WRITE IT
00B2  FE C4              INC     AH                     ; ADDRESS NEXT BYTE
00B4  8A C7              MOV     AL,BH
00B6  E8 0000 E          CALL    PUT_RTC_REG            ; PROFILE IS NOW STORED
```

```
                ;
                ; CHECK TO SEE IF MODEM IS PRESENT
                ;
00B9  A1 0010 R          MOV    AX,EQUIP_FLAG          ; GET EQUIPMENT FLAG
00BC  F6 C4 20           TEST   AH,20H                 ; TEST MODEM INSTALLED BIT
00BF  74 19              JZ     MODEM_ERROR            ; EXIT WITH ERROR
                ;
                ; IF PRESENT, CHECK BATT PWR AND MODEM POWER-ON-BATT ALLOWED BIT
                ;
00C1  E4 7F              IN     AL,PWR_STAT            ; EXTERNAL POWER?
00C3  A8 40              TEST   AL,EXT_PWR
00C5  75 09              JNZ    MPROF1                 ; YES, SKIP NEXT TEST
                ;
00C7  B4 17              MOV    AH,RTC_SYS_PROF1       ; FIND PWR_ON_BATT BIT
00C9  E8 0000 E          CALL   GET_RTC_REG            ; FROM PROFILE
00CC  A8 02              TEST   AL,MODEM_BATT          ; IS BATT PWRED MODEM OK?
00CE  74 0A              JZ     MODEM_ERROR            ; NO, TURN OFF MODEM
                ;
                ; SET UP CONFIG DATA IN AL, AND CALL SET ROUTINE
                ;
00D0  8A C3             MPROF1:        MOV     AL,BL   ; GET CONFIG INPUT DATA
00D2  E8 0000 E          CALL   MODEM_CONFIG           ; AND DO IT
00D5  72 03              JC     MODEM_ERROR            ; ERROR OUT IF PROBLEM
00D7  E9 01BB R          JMP    SPROF_EXIT             ; INT RETURN
                ;
                ; ERROR FOUND IN CONFIGURING MODEM, TURN OFF AND EXIT
                ;
00DA                    MODEM_ERROR:
00DA  BB 0002            MOV    BX,0002H               ; MODEM POWER OFF
00DD  E8 0000 E          CALL   COM_POWER              ; NOW
00E0  B0 80              MOV    AL,80H                 ; ERROR RETURN
00E2  E9 01BD R          JMP    SPROF_OUT
                ;
                ; READ PROFILE ONLY
                ;
00E5  B4 17             SPROF_READ:    MOV     AH,RTC_SYS_PROF1 ; GET HIGH PROF BYTE
00E7  E8 0000 E          CALL   GET_RTC_REG
00EA  8A F8              MOV    BH,AL
00EC  FE C4              INC    AH
00EE  E8 0000 E          CALL   GET_RTC_REG
00F1  8A D8              MOV    BL,AL
00F3  FE C4              INC    AH
                ;
                ; GET LCD TIMEOUT VALUE
                ;
00F5  52                 PUSH   DX                     ; SAVE DX
00F6  E8 0000 E          CALL   GET_RTC_REG            ; GET DISPLAY TIMEOUT (LO)
00F9  8A D0              MOV    DL,AL
00FB  FE C4              INC    AH
00FD  E8 0000 E          CALL   GET_RTC_REG            ; GET DISPLAY TIMEOUT (HI)
0100  8A F0              MOV    DH,AL
0102  B1 3C              MOV    CL,60                  ; DIVIDE TO GET MINUTES
0104  8B C2              MOV    AX,DX
0106  F6 F1              DIV    CL
0108  8A E8              MOV    CH,AL                  ; SAVE  TIMEOUT IN SECS
                ;
                ; GET SYSTEM OFF TIMEOUT VALUE
                ;
010A  B4 1B              MOV    AH,RTC_SYS_INACT       ; GET SYSTEM INACTIVITY VALUE
010C  E8 0000 E          CALL   GET_RTC_REG            ; GET DISPLAY TIMEOUT (LO)
010F  8A D0              MOV    DL,AL
0111  FE C4              INC    AH
0113  E8 0000 E          CALL   GET_RTC_REG            ; GET DISPLAY TIMEOUT (HI)
0116  8A F0              MOV    DH,AL
0118  51                 PUSH   CX                     ; SAVE CX REG
0119  B1 3C              MOV    CL,60                  ; DIVIDE TO GET MINUTES
011B  92                 XCHG   AX,DX
011C  F6 F1              DIV    CL
011E  59                 POP    CX                     ; RESTORE CX
011F  8A C8              MOV    CL,AL                  ; SAVE  TIMEOUT IN SECS
0121  5A                 POP    DX                     ; RESTORE DX REGISTER
0122  E9 01BB R          JMP    SPROF_EXIT
                ;
                ; MODIFY PROFILE
                ;
0125  9C              SPROF_MODIFY:   PUSHF
0126  E8 0000 E        CALL    DISABLE_NMI          ; DISABLE INTERRUPTS
0129  B4 17            MOV     AH,RTC_SYS_PROF1     ; SELECT HIGH PROFILE BYTE
012B  8A C7            MOV     AL,BH
012D  E8 0000 E        CALL    PUT_RTC_REG
0130  FE C4            INC     AH
0132  8A C3            MOV     AL,BL
0134  E8 0000 E        CALL    PUT_RTC_REG
0137  53               PUSH    BX                   ; SAVE REGISTERS
0138  52               PUSH    DX
                ;
                ; CONVERT UNITS FROM MINUTES TO SECONDS FOR LCD INACTIVITY
                ;
0139                  SPROF_M01:
0139  8A C5            MOV     AL,CH                ; GET TIMEOUT IN MINUTES
013B  2A E4            SUB     AH,AH
013D  B3 3C            MOV     BL,60                ; MULTIPLY TO GET SECONDS
013F  F6 E3            MUL     BL
0141  8B D8            MOV     BX,AX
0143  B4 19            MOV     AH,RTC_LCD_INACT     ; GET INACT VALUE ADDRESS
0145  E8 0000 E        CALL    PUT_RTC_REG
0148  FE C4            INC     AH                   ; INCREMENT ADDRESS
014A  8A C7            MOV     AL,BH                ; WRITE HIGH VALUE
014C  E8 0000 E        CALL    PUT_RTC_REG
                ;
                ; CONVERT UNITS FROM MINUTES TO SECONDS FOR SYSTEM OFF INACTIVITY
                ;
014F  8A C1            MOV     AL,CL                ; GET TIMEOUT IN MINUTES
0151  2A E4            SUB     AH,AH
0153  B3 3C            MOV     BL,60                ; MULTIPLY TO GET SECONDS
0155  F6 E3            MUL     BL
0157  8B D8            MOV     BX,AX
0159  B4 1B            MOV     AH,RTC_SYS_INACT     ; GET INACT VALUE ADDRESS
015B  E8 0000 E        CALL    PUT_RTC_REG
015E  FE C4            INC     AH                   ; INCREMENT ADDRESS
0160  8A C7            MOV     AL,BH                ; WRITE HIGH VALUE
0162  E8 0000 E        CALL    PUT_RTC_REG
0165  E8 0000 E        CALL    BAT_SAV_SETUP        ; SETUP FOR BATTERY SAVE
0168  E8 0000 E        CALL    ENABLE_NMI
016B  5A               POP     DX                   ; RESTORE REGISTERS
016C  5B               POP     BX
016D  9D               POPF                         ; RESTORE INTERRUPTS
                ;
                ; SET MODEM AND RS-232 POWER STATE IF ON BAT PWR ACCORDING TO PROFILE
                ; IF ON EXTERNAL POWER THEN FORCE MODEM AND RS-232 ON
                ;
016E  E4 7F            IN      AL,PWR_STAT          ; GET POWER STATUS
0170  A8 40            TEST    AL,EXT_PWR           ; ON EXTERNAL POWER?
0172  74 03            JZ      EXEC_PROF            ; NO, GO ACCORDING TO PROF
0174  80 CF 03         OR      BH,03                ; YES, SELECT FORCE TO POWER
0177  52              EXEC_PROF :     PUSH DX       ; SAVE REGISTER
0178  8B D3            MOV     DX,BX                ; SAVE & SET UP SYS PROF TEST
017A  2A FF            SUB     BH,BH                ; SET DEFAULT TO POWER OFF
017C  F6 C6 02         TEST    DH,MODEM_BATT        ; PROF MODEM BAT POWER ACTIVE?
017F  74 03            JZ      SET_MOD              ; NO, EXECUTE MODEM POWER OFF
0181  80 CF 01         OR      BH,01                ; YES, ACTIVATE MODEM BAT PWR
0184                  SET_MOD:
0184  B3 02            MOV     BL,ACT_MODEM         ; SELECT MODEM
0186  E8 0000 E        CALL    COM_POWER            ; SET CORRESPONDING PWR STATE
                                                    ; IN DEVICE SPECIFIED BY BL

0189  2A FF            SUB     BH,BH                ; SET DEFAULT TO POWER OFF
018B  F6 C6 01         TEST    DH,RS232_BATT        ; PROF RS-232 BATT PWR ACTIVE?
018E  74 03            JZ      SET_RSP              ; NO, EXECUTE RS-232 POWER OFF
0190  80 CF 01         OR      BH,01                ; YES, ACTIVATE RS-232 BAT PWR
0193                  SET_RSP:
0193  B3 04            MOV     BL,ACT_RS232         ; SET DEFAULT
0195  B4 1F            MOV     AH,RTC_FEAT_CON      ; SELECT FEATURE CONFIG
0197  E8 0000 E        CALL    GET_RTC_REG          ; GET FEATURE CONFIG
019A  A8 10            TEST    AL,PRI_INST          ; IS RS-232 PRIMARY COM PORT?
019C  74 03            JZ      SET_RS232            ; NO, ITS NOT PRIMARY
019E  80 CB 01         OR      BL,SET_RS232_PRIM    ; YES, SELECT RS-232 PRIMARY
01A1  E8 0000 E       SET_RS232:      CALL    COM_POWER ; SET POWER STATE
```

```
01A4  8B DA              MOV     BX,DX                  ; RESTORE SYS PROF
01A6  5A                 POP     DX                     ; SAVE REGISTER
               ;
               ; SET LCD HIGH INTENSITY SUBSTITUTE
               ;
01A7                    SET_LCD_OPT:
01A7  53                 PUSH    BX                     ; SAVE BX REG
01A8  8A DF              MOV     BL,BH
01AA  D0 EB              SHR     BL,1
01AC  D0 EB              SHR     BL,1
01AE  80 E3 03           AND     BL,03H                 ; SAVE ONLY LCD SUBSTITUTE TYP
01B1  B4 14              MOV     AH,20                  ; VIDEO FUNCTION CALL 20
01B3  B0 02              MOV     AL,2                   ; SET LCD HIGH INTENSITY SUB
01B5  9C                 PUSHF
01B6  0E                 PUSH    CS
01B7  E8 0000 E          CALL    VIDEO_IO_1             ; INT 10 CALL
01BA  5B                 POP     BX                     ; RESTORE BX REG
01BB                    SPROF_EXIT:
01BB  2A C0              SUB     AL,AL                  ; SET GOOD RETURN
01BD                    SPROF_OUT:
01BD  C3                 RET
01BE                    SYS_PROF        ENDP
```

External Event Services (EXT_EVENT)

```
               ;--------------------------------------------------------------------
               ;
               ; INT 15H
               ;
               ; ROUTINE-NAME:  EXT_EVENT (AH = 41H)
               ;
               ; FUNCTION:  TO WAIT FOR A SPECIFIC STATUS CHANGE AFTER AN EXT EVENT
               ;            (DMA OR INTERRUPT) WHILE KEEPING THE SYS CLKS STOPPED TO
               ;            CONSERVE BATTERY POWER.
               ;
```

```
                ; INPUT:  ES:DI CONTAINS POINTER TO BYTE IN USERS STORAGE FOR
                ;         EVENT DETERMINATION (FOR AL=01-04)
                ;         - O R -
                ;         DX CONTAINS ADDRESS OF I/O PORT ADDRESS TO BE READ FOR
                ;         EVENT DETERMINATION (FOR AL=11-14)
                ;
                ;         AL - EVENT TYPE CODE
                ;              00    - RETURN AFTER ANY EVENT HAS OCCURRED
                ;              01    - COMPARE: RETURN IF EQUAL
                ;              02    - COMPARE: RETURN IF NOT EQUAL
                ;              03    - TEST: RETURN IF NOT ZERO
                ;              04    - TEST: RETURN IF ZERO
                ;              11-14 - SAME FUNCTION AS ABOVE EXCEPT DX CONTAINS
                ;                      I/O PORT ADDRESS FOR EVENT DETERMINATION
                ;
                ;         BH - CONDITION COMPARE OR MASK VALUE
                ;         BL - TIMEOUT VALUE (IN 55 MSEC UNITS)
                ;         BL = 0 - NO TIME LIMIT
                ;
                ; OUTPUT:     CARRY FLAG : SET   - TIMEOUT REACHED
                ;             CLEAR - EVENT OCCURRED
                ;
                ; REGISTERS
                ; MODIFIED:      AX
                ;
                ; RESTRICTIONS:  THIS ROUTINE WILL ENABLE PROC IRPTS NO TIMEOUT WILL
                ;                OCCUR IF TMR 0 IS NOT ENABLED WAITING ON NMI IRPTS
                ;                IS NOT ALLOWED.
                ;
                ;------------------------------------------------------------------------------

 01BE                      EXT_EVENT PROC NEAR

 01BE  8A E0                  MOV     AH,AL                ; SAVE TYPE CODE
 01C0  80 FB 00               CMP     BL,0                 ; NO TIME LIMIT FOR TIMEOUT?
 01C3  74 07                  JE      EXT_WT1              ; NO, GO CHECK FOR WAIT TYPE
 01C5  80 CC 20               OR      AH,20H               ; YES, SET NO TIMEOUT FLAG
 01C8  88 1E 007B R           MOV     EVENT_TIM_OUT,BL     ; SAVE TIMEOUT VALUE

 01CC                      EXT_WT1:
 01CC  24 0F                  AND     AL,0FH               ; SAVE ONLY TYPE
 01CE  3C 00                  CMP     AL,0                 ; WAIT ON ANY EVENT?
 01D0  75 03                  JNE     EXT_WT2              ; NO, CHECK FOR SPECIFIC EVENT
 01D2  E9 0259 R              JMP     ANY_WAIT             ; YES, WAIT ON ANY EVNT TO RET

 01D5                      EXT_WT2:
 01D5  3C 01                  CMP     AL,01                ; COMPARE, RETURN EQUAL?
 01D7  74 64                  JE      CEQ_WAIT
 01D9  3C 02                  CMP     AL,02
 01DB  74 44                  JE      CNEQ_WAIT            ; COMPARE , RETURN NOT EQUAL
 01DD  3C 03                  CMP     AL,03
 01DF  74 24                  JE      TNZ_WAIT             ; TEST, RETURN NOT ZERO
 01E1  3C 04                  CMP     AL,04
 01E3  75 1C                  JNE     INV_EVENT            ; IF NOT 4 THEN INVALID EVENT
                ;
                ; TEST, RETURN IF ZERO
                ;
 01E5                      TZ_WAIT:
 01E5  E8 026B R              CALL    PREP_WAIT            ; PREPARE FOR WAIT
 01E8  F6 C4 10               TEST    AH,10H               ; TEST FOR I/O PORT READ
 01EB  75 08                  JNZ     TZ_WT1
 01ED  26: 84 3D              TEST    BYTE PTR ES:[DI],BH
 01F0  74 70                  JZ      WAIT_DONE            ; RETURN IF ZERO
 01F2  EB 06 90               JMP     TZ_WT2
```

```
01F5                    TZ_WT1:
01F5  EC                   IN      AL,DX              ; READ FROM I/O PORT
01F6  84 C7                TEST    AL,BH
01F8  74 68                JZ      WAIT_DONE

01FA                    TZ_WT2:
01FA  E8 0277 R            CALL    EVENT_WAIT
01FD  72 64                JC      WAIT_EXIT
01FF  EB E4                JMP     TZ_WAIT
               ;
               ; INVALID EVENT SO SET CARRY AND EXIT
               ;
0201                    INV_EVENT:
0201  F9                   STC                        ; SET CARRY FOR INVAL EVNT TYP
0202  EB 5F 90             JMP     WAIT_EXIT
               ;
               ; TEST , RETURN IF NOT ZERO
               ;
0205                    TNZ_WAIT:
0205  E8 026B R            CALL    PREP_WAIT          ; PREPARE FOR WAIT
0208  F6 C4 10             TEST    AH,10H             ; TEST FOR I/O PORT READ
020B  75 08                JNZ     TNZ_WT1
020D  26: 84 3D            TEST    BYTE PTR ES:[DI],BH
0210  75 50                JNZ     WAIT_DONE          ; RETURN IF ZERO
0212  EB 06 90             JMP     TNZ_WT2
0215                    TNZ_WT1:
0215  EC                   IN      AL,DX              ; READ FROM I/O PORT
0216  84 C7                TEST    AL,BH
0218  75 48                JNZ     WAIT_DONE

021A                    TNZ_WT2:
021A  E8 0277 R            CALL    EVENT_WAIT
021D  72 44                JC      WAIT_EXIT
021F  EB E4                JMP     TNZ_WAIT
               ;
               ; COMPARE, RETURN IF NOT EQUAL
               ;
0221                    CNEQ_WAIT:
0221  E8 026B R            CALL    PREP_WAIT          ; PREPARE FOR WAIT
0224  F6 C4 10             TEST    AH,10H             ; TEST FOR I/O PORT READ
0227  75 08                JNZ     CNEQ_WT1
0229  26: 38 3D            CMP     BYTE PTR ES:[DI],BH
022C  75 34                JNE     WAIT_DONE          ; RETURN IF ZERO
022E  EB 06 90             JMP     CNEQ_WT2

0231                    CNEQ_WT1:
0231  EC                   IN      AL,DX              ; READ FROM I/O PORT
0232  3A C7                CMP     AL,BH
0234  75 2C                JNE     WAIT_DONE

0236                    CNEQ_WT2:
0236  E8 0277 R            CALL    EVENT_WAIT
0239  72 28                JC      WAIT_EXIT
023B  EB E4                JMP     CNEQ_WAIT
               ;
               ; COMPARE, RETURN IF ZERO
               ;
023D                    CEQ_WAIT:
023D  E8 026B R            CALL    PREP_WAIT          ; PREPARE FOR WAIT
0240  F6 C4 10             TEST    AH,10H             ; TEST FOR I/O PORT READ
0243  75 08                JNZ     CEQ_WT1
0245  26: 38 3D            CMP     BYTE PTR ES:[DI],BH
0248  74 18                JE      WAIT_DONE          ; RETURN IF ZERO
024A  EB 06 90             JMP     CEQ_WT2

024D                    CEQ_WT1:
024D  EC                   IN      AL,DX              ; READ FROM I/O PORT
024E  3A C7                CMP     AL,BH
0250  74 10                JE      WAIT_DONE

0252                    CEQ_WT2:
0252  E8 0277 R            CALL    EVENT_WAIT
```

```
0255  72 0C                JC      WAIT_EXIT
0257  EB E4                JMP     CEQ_WAIT
                ;
                ; WAIT ON ANY EVENT
                ;
0259                    ANY_WAIT:
0259  E8 026B R            CALL    PREP_WAIT            ; SET UP FOR CLOCK STOP
025C  E8 0277 R            CALL    EVENT_WAIT           ; EXECUTE CLOCK STOP
025F  EB 02 90             JMP     WAIT_EXIT

0262                    WAIT_DONE:
0262  F8                   CLC                          ; CLEAR CARRY FLAG

0263                    WAIT_EXIT:
0263  9C                   PUSHF                        ; SAVE FLAGS
0264  B0 27                MOV     AL,DISABLE_SLEEP+CLOCK_RUN+GLOBAL_NMI ; ON NMIS
0266  E6 72                OUT     CLOCK_CTL,AL
0268  9D                   POPF                         ; RESTORE FLAGS
0269  FB                   STI                          ; RE-ENABLE INTERRUPTS
026A  C3                   RET                          ; EXIT
                ;
                ; THIS ROUTINE ENABLES THE SLEEP CLOCK BUT SETS THE STATE TO CLOCK RUN
                ; THIS MUST BE DONE IN ORDER TO COVER THE TIMING PROBLEM OF THE IRPT
                ; OCCURRING AFTER THE CONDITION TEST BUT BEFORE THE CLOCKS ARE STOPPED
                ;
026B                    PREP_WAIT  PROC NEAR
026B  FA                   CLI                          ; DISABLE INTERRUPTS
026C  B0 07                MOV     AL,CLOCK_RUN+DISABLE_SLEEP ; DSABL NMI ST FL SPEED
026E  E6 72                OUT     CLOCK_CTL,AL         ; NO ENABLE SLEEP CLOCK
0270  EB 00                JMP     $+2                  ; DELAY
0272  24 FB                AND     AL,NOT DISABLE_SLEEP ; ENABLE SLEEP CLOCK
0274  E6 72                OUT     CLOCK_CTL,AL
0276  C3                   RET
0277                    PREP_WAIT  ENDP
                ;
                ; THIS SUBROUTINE STOPS THE SYSTEM CLOCKS AND RTN AFTER THEY HAVE BEEN
                ; RESTARTED. IF AN INTERRUPT HAS OCCURRED BTWN PREP_WAIT & EVENT_WAIT
                ; THE SLEEP CLOCK WILL HAVE BEEN DISABLED BY HARDWARE & THE EVENT_WAIT
                ; ROUTINE WILL "FALL THROUGH" WITHOUT STOPPING THE CLOCKS.
                ; ONCE THE CLOCKS HAVE RESUMED, THE TIMEOUT VALUE IS CKED (IF USED) &
                ; THE CARRY FLAG SET IF TIMEOUT
                ;
                ;
                ;
0277                    EVENT_WAIT PROC NEAR
0277  E4 72                IN      AL,CLOCK_CTL
0279  24 FC                AND     AL,NOT CLOCK_RUN
027B  0C 20                OR      AL,GLOBAL_NMI        ; TURN ON NMIS
027D  FB                   STI                          ; ENABLE INTERRUPTS
027E  E6 72                OUT     CLOCK_CTL,AL         ; STOP CLOCKS
0280  EB 00                JMP     $+2                  ; DELAY
                ;
                ; CLOCKS HAVE RESTARTED
                ;
0282  F6 C4 20             TEST    AH,20H               ; TIMEOUT TO BE TAKEN
0285  74 0F                JZ      NO_TIMEOUT           ; JUMP IF NOT
0287  38 1E 007B R         CMP     EVENT_TIM_OUT,BL     ; CHECK ENOUGH TIME
028B  77 07                JA      TIMEOUT              ; IF ORIGINAL TIME THEN ERROR
028D  80 3E 007B R 00      CMP     EVENT_TIM_OUT,0      ; CHECK FOR TIME ELAPSED
0292  75 02                JNZ     NO_TIMEOUT
0294                    TIMEOUT:
0294  F9                   STC                          ; TIMEOUT - SET CARRY
0295  C3                   RET                          ; RETURN

0296                    NO_TIMEOUT:
0296  F8                   CLC                          ; NO TIME_OUT SO CLEAR CARRY
0297  C3                   RET                          ; RETURN
0298                    EVENT_WAIT ENDP

0298                    EXT_EVENT ENDP
```

System Power Off Services (SYS_POWER_OFF)

```
               ;-----------------------------------------------------------------------
               ;
               ; INT 15H
               ;
               ; ROUTINE-NAME:  SYS_POWER_OFF  (AH = 42H)
               ;
               ; FUNCTION:  THIS ROUTINE POWERS THE SYSTEM DOWN BY THE REQUEST OF THE
               ;            APPLICATION PROGRAM. IF WARM START WAS SELECTED THE PGM
               ;            WILL RESUME IN THIS ROUTINE AND RETURN TO THE CALLER.
               ;
               ; INPUT:  AL = 00 FOR IPL/RESUME DETERMINED BY PROFILE
               ;         AL = 01 FOR RESUME MODE FORCED
               ;
               ; OUTPUT:  THE SYSTEM IS POWERED OFF.  IF SYS SUCCESSFULLY SUSPENDED,
               ;          THIS ROUTINE WILL RETURN CONTROL TO THE USER WHEN THE POWER
               ;          IS RE-ACTIVATED.
               ;
               ; REGISTERS
               ; MODIFIED:      AX
               ;
               ;----------------------------------------------------------------

0298                     SYS_POWER_OFF   PROC    NEAR
0298  3C 01               CMP     AL,01                 ; REQUEST FOR RESUME MODE?
029A  75 05               JNE     SYS_POFF1
029C  80 0E 0016 R 40     OR      BIOS_STATUS,F_RESUME  ; SET FORCE RESUME FLAG
02A1                     SYS_POFF1:
02A1  80 26 003F R C0     AND     MOTOR_STATUS,0C0H     ; RESET DISKETTE MOTORS
02A6  BA 03F2             MOV     DX,03F2H
02A9  B0 04               MOV     AL,04H                ; TURN OFF MOTORS/SELECT
02AB  EE                  OUT     DX,AL

02AC  E4 7F               IN      AL,PWR_STAT           ; ENSURE SUSPEND NMI ENABLED
02AE  24 F7               AND     AL,NOT HDWR_RESET     ; TURN OFF RESET FLAG
02B0  0C 04               OR      AL,EN_SUS_NMI
02B2  E6 7F               OUT     PWR_STAT,AL
02B4  EB 00               JMP     $+2                   ; DELAY
02B6  0C 02               OR      AL,REQ_POFF           ; REQUEST SYSTEM POWER OFF
02B8  E6 7F               OUT     PWR_STAT,AL
02BA  B9 0102             MOV     CX,MS_DELAY           ; DELAY 1 MSEC
02BD  E2 FE               LOOP    $
               ;
               ; POWER HAS BEEN RESTORED
               ;
02BF  F8                  CLC                           ; SET GOOD RETURN
02C0  C3                  RET
02C1                     SYS_POWER_OFF   ENDP
```

System Status Services (SYS_STATUS)

```
                    ;------------------------------------------------------------------
                    ;
                    ; INT 15H
                    ;
                    ; ROUTINE-NAME:  SYS_STATUS   (AH = 43H)
                    ;
                    ; FUNCTION:  THIS ROUTINE RETURNS THE CURRENT SYSTEM STATUS IN THE
                    ;            AL REGISTER AS DEFINED BELOW.
                    ;
                    ; INPUT:         NONE.
                    ;
                    ; OUTPUT:        AL CONTAINS SYSTEM STATUS AS FOLLOWS:
                    ;
                    ;                ---------------------------------
                    ;                | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
                    ;                ---------------------------------
                    ;                  |   |   |   |   |   |   |   |
                    ;                  |   |   |   |   |   |   |   ----> LCD DETACHED
                    ;                  |   |   |   |   |   |   ----> RESERVED
                    ;                  |   |   |   |   |   ----> RS-232 POWERED ON
                    ;                  |   |   |   |   ----> MODEM POWERED ON
                    ;                  |   |   |   ----> POWER ACTIVATED BY RTC ALARM
                    ;                  |   |   ----> STANDBY POWER LOST (RTC TIME BAD)
                    ;                  |   ----> OPERATING ON EXTERNAL POWER
                    ;                  ----> LOW BATTERY
                    ;
                    ; REGISTERS
                    ; MODIFIED :     AX
                    ;
                    ;------------------------------------------------------------------

02C1                      SYS_STATUS      PROC    NEAR
02C1  53                   PUSH    BX                      ; SAVE BX
02C2  B4 21                MOV     AH,RTC_SYS_STAT         ; SELECT POWER ON STAT
02C4  E8 0000 E            CALL    GET_RTC_REG             ; GET POWER ON STAT
02C7  24 10                AND     AL,PON_ALRM             ; SAVE ONLY POWER ON
02C9  8A D8                MOV     BL,AL                   ; ALARM STATUS
02CB  E4 7F                IN      AL,PWR_STAT             ; GET CURRENT POWER STATUS
02CD  24 C0                AND     AL,LOW_BAT+EXT_PWR      ; SAVE ONLY POWER STATUS
02CF  0A D8                OR      BL,AL                   ; SET COMBINED POWER STATS
02D1  B4 0E                MOV     AH,RTC_DIAG_STAT        ; SELECT RTC TIME
02D3  E8 0000 E            CALL    GET_RTC_REG             ; GET RTC TIME
02D6  A8 80                TEST    AL,RTC_TIME_BAD         ; IS RTC TIME BAD?
02D8  74 03                JZ      SYS_STAT1               ; NO, THEN GO READ SETTINGS
02DA  80 CB 20             OR      BL,20H                  ; YES, SET RTC TIME BAD
02DD                      SYS_STAT1:                       ; STANDBY POWER LOST
02DD  E4 7C                IN      AL,KYBD_CNTL            ; GET COMMO POWER STATUS
```

```
02DF  8A F8              MOV     BH,AL                  ; SAVE THE STATUS
02E1  B4 1F              MOV     AH,RTC_FEAT_CON        ; SELECT FEATURE CONFIG
02E3  E8 0000 E          CALL    GET_RTC_REG            ; GET FEATURE CONFIG
02E6  A8 40              TEST    AL,INTMOD_INST         ; INTERNAL MODEM PRESENT?
02E8  74 08              JZ      CHK_FOR_RS             ; NO, CHECK FOR RS-232
                    ;
                    ; TEST MODEM FOR POWER ACTIVE
                    ;
02EA  F6 C7 02           TEST    BH,ACT_MODEM           ; YES, IS MODEM POWER ON?
02ED  74 03              JZ      CHK_FOR_RS             ; NO, CHECK FOR RS-232 CARD
02EF  80 CB 08           OR      BL,08                  ; YES, SET MODEM ACTIVE
                    ;
                    ; TEST RS-232 FOR POWER ACTIVE
                    ;
02F2                    CHK_FOR_RS:
02F2  A8 80              TEST    AL,SERPLL_INST         ; IS RS-232 CARD INSTALLED?
02F4  74 08              JZ      CHK_FOR_LCD            ; NO, JUMP TO LCD CHECK
02F6  F6 C7 04           TEST    BH,ACT_RS232           ; YES, IS RS-232 POWER ON?
02F9  74 03              JZ      CHK_FOR_LCD            ; NO, JUMP TO LCD CHECK
02FB  80 CB 04           OR      BL,04                  ; YES, SET RS POWER ACTIVE
                    ;
                    ; TEST IF LCD IS OPERATIVE
                    ;
02FE                    CHK_FOR_LCD:
02FE  B4 20              MOV     AH,RTC_DSP_CON         ; LOOK AT THE DISPLAY
0300  E8 0000 E          CALL    GET_RTC_REG            ; CONFIGURATION
0303  A8 03              TEST    AL,DSP_MLCD+DSP_CLCD   ; IS LCD CONFIGURED?
0305  75 03              JNZ     SYS_STAT_EXIT          ; YES, THEN LEAVE ROUTINE
0307  80 CB 01           OR      BL,01                  ; NO, LCD IS INOPERATIVE
030A                    SYS_STAT_EXIT:
030A  8A C3              MOV     AL,BL                  ; RESTORE STATUS BACK TO AL
030C  5B                 POP     BX                     ; RESTORE BX REG
030D  F8                 CLC                            ; SET NORMAL GOOD RETURN
030E  C3                 RET
030F                    SYS_STATUS      ENDP
```

Modem Power Services (SYS_MODEM_PWR)

```
                    ;------------------------------------------------------------------
                    ;
                    ; INT 15H   (AH = 44H)
                    ;
                    ;  SYS_MODEM_PWR:  THIS PROCEDURE TURNS ON OR OFF THE MODEM
                    ;
                    ;         INPUT:  AL = 0, POWER OFF; AL <> 0, POWER ON.
                    ;
                    ;         OUTPUT: AL = 0 IF SUCCESSFUL; AL = 80H IF NOT SUCCESSFUL
                    ;
                    ;         REGISTERS USED:  AX DESTROYED.
                    ;
                    ;------------------------------------------------------------------
030F                    SYS_MODEM_PWR   PROC    NEAR

030F  0A C0                OR      AL,AL                ; TEST AL FOR COMMAND
0311  74 0E                JZ      SYS_MOD_OFF          ; TURN MODEM OFF
                    ;
0313                    SYS_MOD_ON:
0313  B4 1D                MOV     AH,RTC_MOD_PROF1     ; GET THE PROFILE REG
0315  E8 0000 E            CALL    GET_RTC_REG          ; PLACE PROFILE IN AL
0318  E8 0000 E            CALL    MODEM_CONFIG         ; CONFIGURE AND POWER MODEM
031B  73 0A                JNC     SYS_MOD_END

031D  B0 80                MOV     AL,80H               ; ERROR EXIT
031F  EB 08                JMP     SHORT SYS_MOD_EXIT
```

```
0321                    SYS_MOD_OFF:
0321  BB 0002              MOV     BX,0002H            ; SELECT MODEM
0324  E8 0000 E            CALL    COM_POWER           ; AND TURN OFF

0327                    SYS_MOD_END:
0327  2A C0                SUB     AL,AL               ; RETURN W/O ERROR
0329                    SYS_MOD_EXIT:
0329  C3                   RET

032A                    SYS_MODEM_PWR   ENDP
```

POST Interval Services (POST_INTV)

```
                ;-----------------------------------------------------------------
                ;
                ; INT 15H
                ;
                ; ROUTINE-NAME:  POST_INTV  (AH = 83H)
                ;
                ; FUNCTION:  POST USER WHEN ELAPSED TIME INTERVAL HAS EXPIRED WITH NO
                ;            WAIT. (RETURNS IMMEDIATELY AND POSTS OCCURRENCE VIA RTC
                ;            INTERRUPTS)
                ;
                ; INPUT:  AL = 0  SET UP FOR POST ON ELAPSED TIME INTERVAL
                ;         ES:BX POINTER TO BYTE IN CALLERS STORAGE FOR POST
                ;         CX,DX NUMBER OF MICROSECONDS TO ELAPSE BEFORE POSTING
                ;         (CX MOST SIGNIFICANT)
                ;
                ;         AL  = 1 CANCEL THE INTERVAL POST
                ;
                ; OUTPUT: FOR POST SET - RTC PERIODIC INTERRUPT IS ACTIVATED TO KEEP
                ;         TRACK OF ELAPSED TIME. INTERRUPTS AT
                ;             APPROXIMATELY  A 1 MILLISECOND RATE.  WHEN TIME
                ;             HAS ELAPSED THE USER FLAG POINTED BY ES:BX WILL
                ;             HAVE BIT 7 SET TO A 1. THE PERIODIC INTERRUPT
                ;             WILL THEN BE TURNED OFF.
                ;         FOR CANCEL - RTC PERIODIC INTERRUPT IS TURNED OFF AND THE
                ;             POST ACTIVE FLAG IS RESET.
                ;
                ; REGISTERS
                ; MODIFIED:      AX
                ;
                ;-----------------------------------------------------------------

032A                       POST_INTV  PROC    NEAR
032A  B4 0B                   MOV     AH,RTC_MODE          ; SET REGISTER NUMBER FOR RTC
032C  3C 00                   CMP     AL,0                 ; CHECK FOR SET
032E  74 14                   JE      POSTI_1              ; MUST BE CLEAR
                ;
                ; CANCEL ANY OUTSTANDING INTERVAL
                ;
0330  9C                      PUSHF                        ; DISABLE INTERRUPTS
0331  E8 0000 E               CALL    DISABLE_NMI          ; AND 8259 INTERRUPTS
0334  E8 0000 E               CALL    GET_RTC_REG          ; GET MODE
0337  24 BF                   AND     AL,NOT PIE_ENABLE    ; RESET PERIODIC IRPT ENABLE
0339  E8 0000 E               CALL    PUT_RTC_REG          ; SET MODE
033C  80 26 00A0 R FE         AND     RTC_WAIT_FLAG,NOT INTERVAL_WAIT ; RST INTVL ACTIVE
0341  EB 37 90                JMP     POSTI_3              ; ENABLE INTERRUPTS AND EXIT
                ;
                ; SET INTERVAL UNLESS ALREADY ACTIVE
                ;
0344                       POSTI_1:
0344  F6 06 00A0 R 01         TEST    RTC_WAIT_FLAG,INTERVAL_WAIT ; CHECK WAIT IN PROG
0349  74 04                   JZ      POSTI_2
                ;
                ; WAIT IS ACTIVE SO SIGNAL ERROR
                ;
034B  F9                      STC                          ; SET ERROR
034C  EB 31 90                JMP     POSTI_EXIT           ; RETURN
                ;
                ; ACTIVATE RTC PERIODIC INTERRUPT
                ;
034F                       POSTI_2:
034F  9C                      PUSHF
0350  E8 0000 E               CALL    DISABLE_NMI          ; DISABLE INTERRUPTS
0353  8C 06 009A R            MOV     USER_FLAG_SEG,ES     ; SET UP TRANSFER TABLE
0357  89 1E 0098 R            MOV     USER_FLAG,BX         ; .
035B  89 0E 009E R            MOV     RTC_HIGH,CX          ; .
035F  89 16 009C R            MOV     RTC_LOW,DX           ; .
```

```
0363  80 0E 00A0 R 01       OR      RTC_WAIT_FLAG,INTERVAL_WAIT ; SET INTVL POST ACT
0368  26: 80 27 7F          AND     BYTE PTR ES:[BX-,NOT POSTED ; RST POST FLAG
036C  E8 0000 E             CALL    GET_RTC_REG         ; GET MODE
036F  0C 40                 OR      AL,PIE_ENABLE       ; ENABLE RTC PERIODIC IRPT
0371  E8 0000 E             CALL    PUT_RTC_REG         ; SET MODE
0374  E4 61                 IN      AL,NMI_CNTL         ; ENABLE ALARM INTERRUPT
0376  24 F7                 AND     AL,NOT DIS_ALARM
0378  E6 61                 OUT     NMI_CNTL,AL
037A                     POSTI_3:
037A  E8 0000 E             CALL    ENABLE_NMI          ; ENBL RTC ALARM AND SUSP NMIS
037D  9D                    POPF                        ; RESTORE INTERRUPT STATUS
037E  F8                    CLC                         ; SET GOOD RETURN

037F                     POSTI_EXIT:
037F  C3                    RET
0380                     POST_INTV ENDP
```

Wait Interval Services (WAIT_INTV)

```
                ;----------------------------------------------------------------
                ;
                ; INT 15H
                ;
                ; ROUTINE-NAME: WAIT_INTV   (AH = 86H)
                ;
                ; FUNCTION:  WAIT FOR ELAPSED TIME TO EXPIRE BEFORE PASSING CONTROL
                ;            THE USER. THIS FUNCTION USES THE RTC PERIODIC IRPT AND
                ;            WAIT ON EXTERNAL EVENT TO CONSERVE BATTERY POWER.
                ;
                ; INPUT:     CX,DX NUMBER OF MICROSECONDS TO ELAPSE BEFORE RETURNING.
                ;
                ; OUTPUT:    RETURN WITH CARRY CLEAR DONE AFTER ELAPSED TIME REACHED.
                ;
                ; REGISTERS
                ; MODIFIED:     AX
                ;
                ;----------------------------------------------------------------

0380                     WAIT_INTV   PROC    NEAR
0380  F6 06 00A0 R 01       TEST    RTC_WAIT_FLAG,INTERVAL_WAIT ; TEST INTERVAL ACTIVE
0385  74 04                 JZ      WAIT_1
0387  F9                    STC                         ; SET ERROR
0388  EB 50 90              JMP     WINTV_EXIT          ; RETURN
038B                     WAIT_1:
038B  9C                    PUSHF
038C  E8 0000 E             CALL    DISABLE_NMI
038F  8C 1E 009A R          MOV     USER_FLAG_SEG,DS    ; SET UP TRANSFER TABLE
0393  C7 06 0098 R          MOV     USER_FLAG,OFFSET RTC_WAIT_FLAG
00A0 R
0399  89 0E 009E R          MOV     RTC_HIGH,CX
039D  89 16 009C R          MOV     RTC_LOW,DX
03A1  80 0E 00A0 R 01       OR      RTC_WAIT_FLAG,INTERVAL_WAIT ; SET FUNCTION ACTIVE
03A6  80 26 00A0 R 7F       AND     RTC_WAIT_FLAG,NOT POSTED ; RESET POST FLAG
03AB  B4 0B                 MOV     AH,RTC_MODE         ; GET MODE REGISTER
03AD  E8 0000 E             CALL    GET_RTC_REG
03B0  0C 40                 OR      AL,PIE_ENABLE       ; ENABLE RTC PERIODIC IRPT
03B2  E8 0000 E             CALL    PUT_RTC_REG         ; SET MODE
03B5  E4 61                 IN      AL,NMI_CNTL         ; ENABLE ALARM INTERRUPT
03B7  24 F7                 AND     AL,NOT DIS_ALARM
03B9  E6 61                 OUT     NMI_CNTL,AL
03BB  E8 0000 E             CALL    ENABLE_NMI          ; RE-ENABLE NMI'S
03BE  9D                    POPF                        ; RESTORE FLAGS
                ;
                ; WAIT ON EXTERNAL EVENT TO CONSERVE BATTERY PWR FOR THE ALLOTTED TIME
```

```
                   ;
03BF  06                   PUSH    ES                  ; SAVE REGISTERS
03C0  57                   PUSH    DI
03C1  53                   PUSH    BX

03C2  1E                   PUSH    DS                  ; SET ES TO DATA SEGMENT
03C3  07                   POP     ES
03C4  8B 3E 0098 R         MOV     DI,USER_FLAG        ; GET POSTED BYTE ADDR IN ES:DI
03C8  B0 03                MOV     AL,03               ; SET TEST & RTRN IF NOT ZERO
03CA  B7 80                MOV     BH,POSTED           ; SET RTRN FOR INTERVAL POSTED
03CC  2A DB                SUB     BL,BL               ; SET NO TIME LIMIT
03CE  E8 01BE R            CALL    EXT_EVENT           ; WAIT ON EVENT
                   ;
                   ; TIME HAS ELAPSED, RESTORE REGISTERS AND TURN OF PERIODIC INTERRUPT
                   ;
03D1  5B                   POP     BX                  ; RESTORE REGISTERS
03D2  5F                   POP     DI
03D3  07                   POP     ES
03D4  80 26 00A0 R 7E      AND     RTC_WAIT_FLAG,NOT POSTED+INTERVAL_WAIT ; RST FUNCT
03D9  F8                   CLC

03DA                    WINTV_EXIT:
03DA  C3                   RET
03DB                    WAIT_INTV    ENDP
```

Device Busy Services (DEV_BUSY)

```
;---------------------------------------------------------------------
;
; INT 15H
;
; ROUTINE-NAME:  DEV_BUSY  (AH = 90H)
;
; FUNCTION:  THIS ROUTINE IS CALLED TO INDICATE THAT A DEV IS BUSY.
;            FOR THE DEVICE TYPE = DISKETTE, A RTRN IS NOT MADE UNTIL
;            A DISKETTE INTERRUPT HAS BEEN RECEIVED OR 2 SECONDS HAVE
;            ELAPSED (WHICHEVER IS FIRST). FOR THE DEV TYPE = KYBRD A
;            RETURN WILL NOT BE MADE UNTIL T*HE KYBD BFR IS IN A NOT
;            EMPTY CONDITION.
;
;
;  ENTRY/EXIT
;  CONDITIONS:
;    INPUT:
;       AL = DEVICE TYPE:
;            01 - DISKETTE         RETURN AFTER IRPT RECEIVED OR 2
;                                  SECOND TIMEOUT HAS OCCURRED CARRY
;                                  SET IF TIMEOUT
;
;            02 - KEYBOARD         RETURN WHEN KEYBOARD
;                                  BUFFER HEAD <> BUFFER TAIL
;
;            FD - WAIT FOR         RETURN WHEN TIME HAS EXPIRED IF
;                 MOTOR STARTUP    TIMER 0 IS RUNNING OR IMMEDIATE
;                                  RETURN WHEN NOT RUNNING ON ENTRY CH
;                                  CONTAINS TIME TO WAIT IN 1/8 SEC
;                                  UNITS
;
;     ALL OTHERS            IMMEDIATE RETURN WITH CARRY CLEAR
;
;  REGISTERS
;  MODIFIED:        AX
;---------------------------------------------------------------------
```

```
03DB                    DEV_BUSY        PROC    NEAR
03DB  57                   PUSH    DI                    ; SAVE REGISTERS
03DC  06                   PUSH    ES
03DD  53                   PUSH    BX
03DE  BF ---- R            MOV     DI,DATA
03E1  8E C7                MOV     ES,DI                 ; SET ES TO DATA
03E3  3C 01                CMP     AL,01                 ; IS DEVICE TYPE = DISKETTE?
03E5  74 4B                JE      DSKT_BUSY             ; YES, GO SERVICE FOR DISKETTE
03E7  3C 02                CMP     AL,02                 ; NO, DEVICE TYPE = KEYBOARD?
03E9  74 3A                JE      KYBD_WAIT             ; YES, GO SERVICE FOR KEYBOARD
03EB  3C FD                CMP     AL,0FDH               ; DEV TYPE = DSKT MTR STARTUP?
03ED  75 33                JNE     DEVB_CLEAR            ; NO, RETURN WITH CARRY CLEAR
                ;
                ; WAIT FOR DISKETTE MOTOR STARTUP TIME TO ELAPSE
                ; WAIT TIME IS IN 125 MILLISECOND UNITS IN CH
                ;
03EF  E8 0442 R            CALL    TMRO_CHK              ; CHECK FOR TIMER 0 RUNNING
03F2  73 4A                JNC     DEVB_EXIT             ; IF NOT RUNNING THEN EXIT, NO
03F4  51                   PUSH    CX                    ; WAIT
03F5  8A DD                MOV     BL,CH                 ; COPY TIME IN CH
03F7  D0 E5                SHL     CH,1                  ; MULTIPLY BY 3
03F9  02 DD                ADD     BL,CH                 ; WAIT TIME IS IN BL
                ;
                ; REQUEST CONTROLLER POWER ON TO OVERLAP PWR ON SEQ WITH MOTOR STARTUP
                ; DELAY
                ;
03FB  88 1E 007B R         MOV     EVENT_TIM_OUT,BL      ; SAVE TIME DLY IN TIMEOUT CTR
03FF  59                   POP     CX                    ; RESTORE CX
0400  52                   PUSH    DX                    ; SAVE DX
0401  BA 03F4              MOV     DX,3F4H               ; DO DUMMY READ TO PWR ON CNTL
0404  EC                   IN      AL,DX                 ; FOR MOTOR WAIT OVERLAP
                                                         ; TIMER 0 ROUTINE WILL DECRMNT
                                                         ; TIME VAL EVENT_TIM_OUT WHILE
                                                         ; CNTLR IS BEING POWERED ON
0405  90                   NOP                           ; BE BACK FROM NMI IF PWR WAS
                                                         ; OFF HERE
0406  5A                   POP     DX                    ; RESTORE DX
0407  38 1E 007B R         CMP     EVENT_TIM_OUT,BL
040B  77 31                JA      DEVB_EXIT             ; EXIT IF MORE THAN ENOUGH TME
040D  8A 1E 007B R         MOV     BL,EVENT_TIM_OUT      ; RETRIEVE ADJUSTED WAIT TIME
0411  0A DB                OR      BL,BL
0413  75 03                JNE     DEVB_01               ; DON'T WAIT IF TIME EXPIRED
0415  F9                   STC                           ; SET CARRY TO SHOW TIMEOUT
0416  EB 26                JMP     SHORT DEVB_EXIT       ; EXIT WITH TIME EXPIRED
0418                    DEVB_01:
0418  B7 00                MOV     BH,0                  ; SET COMPARE MASK TO 0
041A  0E                   PUSH    CS
041B  07                   POP     ES                    ; SET ES TO ROM SEGMENT
041C  2B FF                SUB     DI,DI                 ; SET ADDR 0 IN ROM FOR DUMMY
                                                         ; COMPARE
041E  B0 01                MOV     AL,01                 ; RETURN ON COMPARE EQUAL
0420  EB 19                JMP     SHORT BUSY_WAIT       ; WAIT FOR TIMEOUT TO OCCUR
                ;
                ; RETURN WITH CARRY CLEAR
                ;
0422                    DEVB_CLEAR:
0422  F8                   CLC
0423  EB 19                JMP     SHORT DEVB_EXIT       ; EXIT ROUTINE
                ;
                ; WAITING FOR KEY IN KEYBOARD BUFFER
                ; PERFORM WAIT_ON_EVENT FOR BUFFER_HEAD <> BUFFER TAIL  (LOW BYTES)
                ;
0425                    KYBD_WAIT:
0425  BF 001C R            MOV     DI,OFFSET BUFFER_TAIL ; POINT ES:DI TO BFR HEAD
0428  8A 3E 001A R         MOV     BH,BYTE PTR BUFFER_HEAD ; MASK TO CURRENT BFR TAIL
042C  B0 02                MOV     AL,02                 ; FN CODE=CMP, RTRN NOT EQUAL
042E  2A DB                SUB     BL,BL                 ; NO TIMEOUT
0430  EB 09                JMP     SHORT BUSY_WAIT
                ;
                ; WAIT FOR DISKETTE INTERRUPT COMPLETE
                ;  (INT_FLAG TO BE SET IN SEEK_STATUS)
                ;
0432                    DSKT_BUSY:
0432  BF 003E R            MOV     DI,OFFSET SEEK_STATUS ; GET ADD OF DSKT INTR FLAG
```

```
0435  B7 80              MOV     BH,INT_FLAG          ; INTERRUPT COMPLETE FLAG
0437  B0 03              MOV     AL,03                ; FN CODE = TEST, RETURN NOT 0
0439  B3 25              MOV     BL,25H               ; SET TWO SECOND TIMEOUT

043B                  BUSY_WAIT:
043B  E8 01BE R          CALL    EXT_EVENT

043E                  DEVB_EXIT:
043E  5B                 POP     BX
043F  07                 POP     ES
0440  5F                 POP     DI
0441  C3                 RET
0442                  DEV_BUSY ENDP
               ;-------------------------------------------------------------------
               ;
               ; INT 15H
               ;
               ; ROUTINE-NAME: TMRO_CHK
               ;
               ; FUNCTION: THIS ROUTINE CHECKS TO SEE IF TIMER 0 IS SET UP AS FOLLOWS
               ;           IRPT ENABLED TIMER COUNTING INT 8 VECTOR=TMRO_INT8 ADDR
               ;
               ; INPUT:    NONE.
               ;
               ; OUTPUT:   CARRY FLAG : SET   - TIMER IS  RUNNING
               ;                        CLEAR - TIMER NOT RUNNING
               ;
               ; REGISTERS
               ; MODIFIED:     NONE.
               ;
               ;-------------------------------------------------------------------

                         ASSUME  DS:ABS0
0442                  TMRO_CHK PROC  NEAR
0442  50                 PUSH    AX
0443  53                 PUSH    BX
0444  1E                 PUSH    DS
0445  B8 0000            MOV     AX,0
0448  8E D8              MOV     DS,AX
044A  A1 0020 R          MOV     AX,INT_ADDR          ; GET INT 8 OFFSET ADDRESS
044D  1F                 POP     DS
044E  3D 0000 E          CMP     AX,OFFSET TMRO_INT8 ; CHECK FOR OUR TIMER ROUTINE
0451  75 1B              JNE     TMRO_NORUN           ; IF NOT THEN EXIT
0453  E4 21              IN      AL,INTA01
0455  A8 01              TEST    AL,01                ; CK FOR TMR 0 IRPT ENABLED
0457  75 15              JNE     TMRO_NORUN           ; EXIT IF DISABLED
0459  E4 40              IN      AL,TIMERO            ; MAKE SURE TIMER IS RUNNING
045B  8A E0              MOV     AH,AL
045D  E4 40              IN      AL,TIMERO
045F  50                 PUSH    AX                   ; SAVE CURRENT TMR 0 TIME CNT
0460  5B                 POP     BX                   ; RETRIEVE ORIGINAL COUNT
0461  E4 40              IN      AL,TIMERO            ; RE-READ TIMER LOW
0463  8A E0              MOV     AH,AL
0465  E4 40              IN      AL,TIMERO            ; RE-READ TIMER HI
0467  3B C3              CMP     AX,BX                ; COMPARE WITH ORIGINAL
0469  74 03              JE      TMRO_NORUN           ; JUMP IF TIMER NOT RUNNING
046B                  TMRO_OK:
046B  F9                 STC                          ; SET CARRY TO INDICATE OKAY
046C  EB 01              JMP     SHORT TMRO_EXT

046E                  TMRO_NORUN:
046E  F8                 CLC                          ; CLEAR CARRY
046F                  TMRO_EXT:
046F  5B                 POP     BX                   ; RESTORE REGS
0470  58                 POP     AX
0471  C3                 RET
0472                  TMRO_CHK ENDP
```

Interrupt Complete Services (INT_COMPLETE)

```
             ;-----------------------------------------------------------------
             ;
             ; INT 15H
             ;
             ; ROUTINE-NAME: INT_COMPLETE  (AH = 91H)
             ;
             ; FUNCTION:     DUMMY INTERRUPT RETURN (FUNCTION NOT HANDLED BY BIOS)
             ;
             ; INPUT:        NONE.
             ;
             ; OUTPUT:       NONE.
             ;
             ; REGISTERS
             ; MODIFIED:     NONE.
             ;
             ;-----------------------------------------------------------------

0472                  INT_COMPLETE PROC NEAR
0472                  RETURN:
0472  F8                 CLC                          ; SET GOOD RETURN CODE
0473  C3                 RET
0474                  INT_COMPLETE ENDP

0474                  ROMCODE ENDS
                         END
```

General Subroutines and Tables (B17TABLE)

```
0000                    ROMCODE SEGMENT BYTE PUBLIC

                        ASSUME CS:ROMCODE, DS:DATA
                           IDENT   B17TABLE,0,0

               ;*********************************************************************
               ;
               ; MODULE-NAME :     B17TABLE
               ; DATE LAST MODIFIED: 09/13/1985
               ;
               ; DESCRIPTIVE-NAME : MODULE CONTAINING ALL TABLES USED BY POST/ BIOS,
               ;               GENERAL USE SUBROUTINES AND COMPATABLE LINK ADDRESSES
               ;
               ; COPYRIGHT : 7396-918 (C) COPYRIGHT IBM CORP. 1985
               ;             REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
               ;
               ; CHANGE LEVEL: EC000
               ;
               ; MODULE SIZE: 8K BYTES
               ;
               ; EXTERNALLY REFERENCED ROUTINES: REFER TO EXTRN LIST
               ;
               ; SUBROUTINES:
               ;    DDS             - SET DATA SEGMENT TO BIOS DATA AREA
               ;    ERR_BEEP        - BEEP THE SPEAKER FOR POST ERROR CONDITIONS
               ;    BEEP            - BBEP THE SPEAKER
               ;    DSP_BYTE        - TRANSLATE AND DISPLAY HEX BYTE TO SCREEN
               ;    XLAT_NIB        - TRANSLATE HEX NIBBLE TO ASCII CHARACTER
               ;    DSP_HEX         - WRITE HEX BYTE TO SCREEN
               ;    GET_RTC_REG     - READ REAL TIME CLOCK REGISTER
               ;    PUT_RTC_REG     - WRITE REAL TIME CLOCK REGISTER
               ;    D11             - DEFAULT INTERRUPT HANDLER
               ;    E_MSG           - POST ERROR MESSAGE HANDLER
               ;    ICON_PR         - ICON DISPLAY PROCESSING ROUTINE
               ;    STR_CON         - CONVERT HEX WORD TO ASCII STRING
               ;    ENABLE_NMI      - ENABLE KEYBOARD,RTC, AND SUSPEND NMIS
               ;    DISABLE_NMI     - DISABLE KEYBOARD, RTC, AND SUSPEND NMIS
               ;    BAT_SAV_SETUP   - ENABLES/DISABLES THE RTC 1 SECOND BAT SAVING
               ;                      INTERRUPT
               ;    GET_VECTOR@     - GET ABSOLUTE VECTOR ADDRESS FOR HARDWARE LEVEL
               ;    RES_ERR_CHK     - CHECK AND DISPLAY ANY RESUME ERRORS
               ;
               ; TABLES:
               ;    PRT_TAB         - PRINTER PORT SEARCH TABLE
               ;    VECTOR_TABLE    - INTERRUPT VECTOR INITIALIZATION TABLE
               ;    BAUD_TABLE      - COMMUNICATION BAUDE RATE SELECT TABLE
               ;    DSKT_BASE       - DISKETTE DRIVE TIMING PARAMETERS
               ;    VIDEO_PARMS     - VIDEO IO PARAMETER TABLES
               ;    M1-M7           - VIDEO LINKAGE TABLE
               ;    CHAR_GEN_LO     - LOWER 128 DISPLAY CHARACTER SET
               ;    CHAR_GEN_HI     - UPPER 128 DISPLAY CHARACTER SET
               ;    K6-K30          - KEYBOARD TRANSLATE TABLES
               ;    KBPAD_TBL       - PREPROCESSOR KEYPAD XLATE TABLE
               ;    KBFUN_TBL       - PREPROCESSOR FUNCTION KEY XLATE TABLE
               ;    KBNMI_TBL       - PREPROCESSOR NORMAL KEY XLATE TABLE
               ;    DEF_SYS_PROF    - DEFAULT SYSTEM PROFILE
               ;
```

```
; MESSAGES AND ICONS:
;    MEM_MSG          - 'KB OK' MEMORY MESSAGE
;    ROM_MSG          - 'XXXXX ROM' MESSAGE
;    SYS_UNIT         - SYSTEM UNIT ICON
;    FEAT_ICON        - FEATURE DEVICE ICON
;    CLOCK            - SYSTEM CLOCK NOT SET ICON
;    BAT_ICON         - BATTERY LOW ICON
;    PAR_CHK          - PARITY CHECK (I/O CHANNEL CHECK) ICON
;    F1_ICON          - PRESS F1 TO CONTINUE ICON
;
;
; CHANGE ACTIVITY:  NONE
;
;*********************************************************************

;******************************************************************
;*     E X T E R N A L   R E F E R E N C E S                       *
;******************************************************************
;
;  PRIVATE BIOS AND POST ROUTINE NAMES
;
            EXTRN   POSTMAIN:NEAR
            EXTRN   NMI_FLIH:NEAR
            EXTRN   SYS_BOOT:NEAR
            EXTRN   KYBD_IO:NEAR
            EXTRN   KYBD_INT9:NEAR
            EXTRN   DSKT_IO:NEAR
            EXTRN   DSKT_INTE:NEAR
            EXTRN   PRT_IO:NEAR
            EXTRN   VIDEO_IO_1:NEAR
            EXTRN   SYS_SERVICES:NEAR
            EXTRN   TOD_PROC:NEAR
            EXTRN   TMRO_INT8:NEAR
            EXTRN   COMMO_IO:NEAR
            EXTRN   PRT_SCRN:NEAR
            EXTRN   COM_POWER:NEAR
            EXTRN   MODEM_CONFIG:NEAR

; VIDEO I/O SUBROUTINES EXTERNALS

            EXTRN SET_MODE:NEAR
            EXTRN SET_CTYPE:NEAR
            EXTRN SET_CPOS:NEAR
            EXTRN READ_CURSOR:NEAR
            EXTRN READ_LPEN:NEAR
            EXTRN ACT_DISP_PAGE:NEAR
            EXTRN SCROLL_UP:NEAR
            EXTRN SCROLL_DOWN:NEAR
            EXTRN READ_AC_CURRENT:NEAR
            EXTRN WRITE_AC_CURRENT:NEAR
            EXTRN WRITE_C_CURRENT:NEAR
            EXTRN SET_COLOR:NEAR
            EXTRN WRITE_DOT:NEAR
            EXTRN READ_DOT:NEAR
            EXTRN WRITE_TTY:NEAR
            EXTRN VIDEO_STATE:NEAR

;******************************************************************
;*     P U B L I C S                                               *
;******************************************************************

; PRINTER, VECTOR, COMMUNICATION AND DISKETTE TABLE PUBLICS

            PUBLIC  PRT_TAB
            PUBLIC  PRT_TAB_END
            PUBLIC  VECTOR_TABLE
            PUBLIC  BAUD_TABLE
            PUBLIC  PARMS_TPI48
            PUBLIC  PARMS_TPI135
            PUBLIC  DSKT_BASE

;VIDEO PUBLICS

            PUBLIC  VIDEO_PARMS
            PUBLIC  M1
            PUBLIC  M1L
            PUBLIC  M4
            PUBLIC  M5
            PUBLIC  M6
            PUBLIC  M7
            PUBLIC  CHAR_GEN_LO
            PUBLIC  CHAR_GEN_HI
            PUBLIC  MONO_TBL
            PUBLIC  CGA_TBL
            PUBLIC  LCD_CGA_TBL
            PUBLIC  LCD_MONO_TBL

; KEYBOARD PUBLICS

            PUBLIC  K6
            PUBLIC  K6L
            PUBLIC  K7
            PUBLIC  K8
            PUBLIC  K9
            PUBLIC  K10
            PUBLIC  K11
            PUBLIC  K12
            PUBLIC  K13
            PUBLIC  K14
            PUBLIC  K15
            PUBLIC  K30
            PUBLIC  KBPAD_TBL
            PUBLIC  KBPADL
            PUBLIC  KBFUN_TBL
            PUBLIC  KBFUNL
            PUBLIC  KBNMI_TBL

; COMPATABILITY ENTRY POINTS

            PUBLIC  RESET
            PUBLIC  START
            PUBLIC  RS232_IO
            PUBLIC  DUMMY_RETURN
            PUBLIC  NMI_INT
            PUBLIC  BOOT_STRAP
            PUBLIC  KEYBOARD_IO
            PUBLIC  KB_INT
            PUBLIC  DISKETTE_IO
            PUBLIC  DSKT_INT
            PUBLIC  PRINTER_IO
            PUBLIC  VIDEO_IO
            PUBLIC  MEMORY_SIZE_DET
            PUBLIC  EQUIPMENT
            PUBLIC  CASSETTE_IO
            PUBLIC  PRINT_SCREEN
            PUBLIC  TIME_OF_DAY
            PUBLIC  TIMER_INT
            PUBLIC   P_O_R
;
;  GENERAL PURPOSE SUBROUTINES
;
            PUBLIC  DDS
            PUBLIC  ERR_BEEP
            PUBLIC  BEEP
            PUBLIC  BEEP_SUB
            PUBLIC  DSP_BYTE
            PUBLIC  XLAT_NIB
            PUBLIC  DSP_HEX
            PUBLIC  GET_RTC_REG
            PUBLIC  PUT_RTC_REG
            PUBLIC  D11
            PUBLIC  E_MSG
            PUBLIC  ICON_PR
            PUBLIC  STR_CON
            PUBLIC  KB_NOISE
            PUBLIC  DISABLE_NMI
            PUBLIC  ENABLE_NMI
            PUBLIC  SYS_SETUP
```

```
                        PUBLIC  SYS_CHK
                        PUBLIC  KYBD_RESET
                        PUBLIC  BAT_SAV_SETUP
                        PUBLIC  VECTOR_SETUP
                        PUBLIC  GET_VECTOR@
                        PUBLIC  RES_ERR_CHK
                ;
                ; POST ICONS AND MESSAGES
                ; AND TABLES

                        PUBLIC MEM_MSG
                        PUBLIC ROM_MSG
                        PUBLIC DEF_SYS_PROF
                        PUBLIC PAR_CHK
                        PUBLIC RTC_SIG_SAVE
                        PUBLIC F1_ICON
                        PUBLIC SYS_DSKT_ICON
                        PUBLIC DSKT_ICON
                        PUBLIC BAD_DSKT_ICON
                        PUBLIC SYS_DESCR_TABLE

                ;*******************************************************
                ;  I B M   C O P Y R I G H T   S T A T E M E N T
                ;*******************************************************
0000  37 33 39 36 39 31     DB      '7396918 COPR. IBM 1985' ; COPYRIGHT NOTICE
      38 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 35
```

Software Reset Routine

```
                ;ORG      0E05BH
005B                       ORG      0005BH
005B                      RESET       LABEL   FAR
005B                      START:
005B  B0 00                MOV      AL,0                 ; SET LCD INDEX
005D  E6 74                OUT      LCD_INDX,AL
005F  E4 75                IN       AL,LCD_DATA
0061  24 BF                AND      AL,NOT PANEL_ENABLE  ; TURN OFF PANEL POWER
0063  E6 75                OUT      LCD_DATA,AL
0065  B0 08                MOV      AL,HDWR_RESET        ; CAUSE HARDWARE RESET
0067  E6 7F                OUT      PWR_STAT,AL
0069  EB FE                JMP      $                    ; WAIT FOR RESET


                ;---------------------------------
                ; POST ERROR ICONS  AND MESSAGES
                ;---------------------------------
006B                      MEM_MSG         LABEL   BYTE
006B  20 4B 62 20 4F 4B          DB        ' Kb OK '     ; MEMORY OK MESSAG
      20
0072  30 20 52 4F 4D 04 ROM_MSG DB        '0 ROM',4     ; ROM CHECKSUM ERROR MSG

0078  00 00 14 28 3C     DIS_POS DB       0,0,20,40,60  ; VALUES FOR E_MSG ROUTINE

                ;***********************************************
                ; P O W E R  O N  S E L F   T E S T  I C O N S
                ;***********************************************

007D                      SYS_UNIT        LABEL   BYTE
007D  01 04 0B 08          DB      1,04,11,8
0081  C9 01 09 CD BB       DB      0C9H,01,09,0CDH,0BBH
0086  04 02 BA 01 09 DB    DB      4,2,0BAH,1,9,0DBH,0BAH
      BA
008D  C7 C4 DC C4 BF 20    DB      0C7H,0C4H,0DCH,0C4H,0BFH,020H,0DAH,0C4H,0DCH,0C4H,0B6H
      DA C4 DC C4 B6
0098  C7 C4 C4 C4 C1 C4    DB      0C7H,0C4H,0C4H,0C4H,0C1H,0C4H,0C1H,0C4H,0C4H,0C4H,0B6H
```

```
      C1 C4 C4 C4 B6
00A3  04 02 BA 01 09 FE   DB      4,2,0BAH,1,9,0FEH,0BAH
      BA
00AA  C8 01 09 CD BC      DB      0C8H,1,9,0CDH,0BCH

00AF                     FEAT_ICON        LABEL    BYTE
00AF  05 FB 0F 04         DB      05,0FBH,15,04
00B3  B9 BB 03            DB      0B9H,0BBH,03
00B6  20 BA 01 05 20 C9   DB      020H,0BAH,01,05,020H,0C9H,01,06,0CDH,0BBH
      01 06 CD BB
00C0  20 C8 01 05 CD B9   DB      020H,0C8H,01,05,0CDH,0B9H,020H,049H,
                                       02FH,04FH,020H,020H,0BAH
      20 49 2F 4F 20 20
      BA
00CD  01 07 20 C8 01 06   DB      01,07,020H,0C8H,01,06,0CDH,0BCH
      CD BC

00D5                     CLOCK      LABEL   BYTE
00D5  02 05 0A 07         DB      02,05,10,07
00D9  C9 01 08 CD BB      DB      0C9H,01,8,0CDH,0BBH
00DE  BA 5C 20 20 31 32   DB      0BAH,05CH,020H,020H,031H,032H,020H,020H,02FH,0BAH
      20 20 2F BA
00E8  BA 01 08 20 BA      DB      0BAH,01,8,020H,0BAH
00ED  BA 39 20 01 04 02   DB      0BAH,039H,020H,01,04,02,087H,03FH,020H,033H,0BAH
      87 3F 20 33 BA
00F8  BA 01 08 20 BA      DB      0BAH,01,8,020H,0BAH
00FD  BA 2F 20 20 20 36   DB      0BAH,02FH,020H,020H,020H,036H,020H,020H,05CH,0BAH
      20 20 5C BA
0107  C8 01 08 CD BC      DB      0C8H,01,8,0CDH,0BCH

010C                     BAT_ICON         LABEL    BYTE
010C  02 07 06 07         DB      02,07,06,07
0110  DA CA C4 C4 CA BF   DB      0DAH,0CAH,0C4H,0C4H,0CAH,0BFH
0116  B3 2B 20 20 2D B3   DB      0B3H,02BH,020H,020H,02DH,0B3H
011C  04 04 B3 01 04 20   DB      04,04,0B3H,01,04,020H,0B3H
      B3
0123  C0 01 04 C4 D9      DB      0C0H,01,04,0C4H,0D9H

0128                     F1_ICON          LABEL    BYTE
0128  02 0F 33 04         DB      02,15,51,4
012C  20 20 46 31 20 20   DB      020H,020H,46H,31H,20H,20H,20H,46H,
                                       32H,20H,20H,20H,46H,33H,20H
      20 46 32 20 20 20
      46 33 20
013B  20 20 46 34 20 20   DB      020H,020H,46H,34H,20H,20H,20H,46H,
                                       35H,20H,20H,20H,46H,36H,20H
      20 46 35 20 20 20
      46 36 20
014A  20 20 46 37 20 20   DB      020H,020H,46H,37H,20H,20H,20H,46H,
                                       38H,20H,20H,20H,46H,39H,20H
      20 46 38 20 20 20
      46 39 20
0159  20 46 31 30 20 20   DB      020H,46H,31H,30H,20H,20H

015F  20 02 87 19 20 20   DB      020H,2,87H,19H,20H,20H,2,87H,19H,
                                       0DAH,5,8,5,0C4H,0C4H,0C4H,0C4H,0C2
                         H
      02 87 19 DA 05 08
      05 C4 C4 C4 C4 C2
0171  01 04 C4 BF         DB      1,4,0C4H,0BFH

0175  DA 01 04 C4 B4 05   DB      0DAH,1,4,0C4H,0B4H,5,9,5,20H,20H,20H,20H,0B3H
      09 05 20 20 20 20
      B3

0182  C0 05 09 05 C4 C4   DB      0C0H,5,9,5,0C4H,0C4H,0C4H,0C4H,0C1H,1,4,0C4H,0D9H
      C4 C4 C1 01 04 C4
      D9
018F                     PAR_CHK          LABEL    BYTE
018F  02 07 0B 07         DB      02,07,11,07           ; I/O CHANNEL CHECK ICON
0193  01 09 20 FB FB      DB      01,09,020H,0FBH,0FBH
0198  01 08 20 FB FB 03   DB      01,08,020H,0FBH,0FBH,03
019E  01 07 20 FB FB 03   DB      01,07,020H,0FBH,0FBH,03
```

```
01A4  01 06 20 FB FB 03  DB      01,06,020H,0FBH,0FBH,03
01AA  FB FB 20 20 20 FB  DB      0FBH,0FBH,020H,020H,020H,0FBH,0FBH,03
      FB 03
01B2  20 FB FB 20 FB FB  DB      020H,0FBH,0FBH,020H,0FBH,0FBH,03
      03
01B9  20 20 20 FB FB 03  DB      020H,020H,020H,0FBH,0FBH,03
                  ;
                  ; SYSTEM UNIT WITH INSERT DISKETTE ICON (USED DURING BOOT)
                  ;
01BF                        SYS_DSKT_ICON   LABEL   BYTE
01BF  00 00 12 11        DB      0,0,18,17
01C3  C9 01 10 CD BB     DB      0C9H,01,16,0CDH,0BBH
01C8  04 04 BA 20 01 0E  DB      04,04,0BAH,20H,01,14,0DBH,20H,0BAH
      DB 20 BA
01D1  C8 D1 01 05 CD B8  DB      0C8H,0D1H,01,05,0CDH,0B8H,20H
      20
01D8  20 D5 01 05 CD D1  DB      20H,0D5H,01,05,0CDH,0D1H,0BCH
      BC
01DF  20 B3 DC DC F0 DC  DB      20H,0B3H,0DCH,0DCH,0F0H,0DCH
01E5  DC C3 C4 C4 B4 DC  DB      0DCH,0C3H,0C4H,0C4H,0B4H,0DCH
01EB  DC F0 DC DC B3 20  DB      0DCH,0F0H,0DCH,0DCH,0B3H,20H
01F1  20 C3 01 05 C4 C1  DB      20H,0C3H,01,05,0C4H,0C1H,0C4H
      C4
01F8  C4 C1 01 05 C4 B4  DB      0C4H,0C1H,01,05,0C4H,0B4H,20H
      20
01FF  04 03 20 B3 01 0E  DB      04,03,20H,0B3H,01,14,0FEH,0B3H,20H
      FE B3 20
0208  20 D4 01 0E CD BE  DB      20H,0D4H,01,14,0CDH,0BEH,20H
      20
020F  04 05 01 12 20     DB      04,05,01,18,020H
                  ;
                  ; DISKETTE ICON (USED DURING BOOT)
                  ;
0214                        DSKT_ICON       LABEL   BYTE
0214  00 00 05 04        DB      0,0,5,4
0218  DA C4 DC           DB      0DAH,0C4H,0DCH
021B  C4 BF              DB      0C4H,0BFH
021D  04 02 B3 20 20 20  DB      04H,02H,0B3H,20H,20H,020H,0B3H
      B3
0224  C0 01 03 C4 D9     DB      0C0H,01H,03H,0C4H,0D9H
                  ;
                  ; BAD DISKETTE ICON  (USED DURING BOOT)
                  ;
0229                        BAD_DSKT_ICON   LABEL   BYTE
0229  00 00 07 04        DB      0,0,7,4
022D  DA BF 20 DA C4 C4  DB      218,191,32,218,196,196,191
      BF
0234  B3 C0 BF C0 BF 20  DB      179,192,191,192,191,32,179
      B3
023B  B3 20 C0 BF C0 BF  DB      179,32,192,191,192,191,179
      B3
0242  C0 C4 C4 D9 20 C0  DB      192,196,196,217,32,192,217
      D9
                  ;
                  ; TABLE OF ICON ADDRESSES USED BY E_MSG ROUTINE
                  ;
0249                        ICON_ADR        LABEL   WORD
0249  0000 007D R 00AF R DW      00,OFFSET SYS_UNIT,OFFSET FEAT_ICON,OFFSET CLOCK
      00D5 R
0251  010C R 018F R      DW      OFFSET BAT_ICON,OFFSET PAR_CHK,OFFSET F_ICON
      0128 R
```

System Descriptor Table

```
              ;----------------------------------
              ; SYSTEM DESCRIPTOR TABLE
              ; ACCESSED VIA INT 15H FUNCTION 0C0H
              ;----------------------------------
0257                  SYS_DESCR_TABLE LABEL  BYTE
0257  0008             DW     8                     ; DESCRIPTOR TABLE LENGTH
0259  F9               DB     0F9H                  ; SYSTEM MODEL BYTE
025A  00               DB     0                     ; SECONDARY MODEL BYTE
025B  00               DB     0                     ; BIOS REVISION LEVEL
025C  38               DB     00111000B             ; FEATURE INFORMATION BYTE 1
025D  00               DB     0                     ; FEATURE INFORMATION BYTE 2
025E  00               DB     0                     ; FEATURE INFORMATION BYTE 3
025F  00               DB     0                     ; FEATURE INFORMATION BYTE 4
0260  00               DB     0                     ; FEATURE INFORMATION BYTE 5
```

Default System Profile

```
              ;--------------------------------------------------
              ; DEFAULT SYSTEM PROFILE
              ;  THIS IS THE SET OF PARAMETERS DEFINING THE
              ;  DEFAULT SYSTEM PROFILE WHICH IS LOADED AFTER
              ;  STANDBY POWER LOST CONDITIONS
              ;--------------------------------------------------

0261                  DEF_SYS_PROF   LABEL  BYTE
0261  E0 00            DB      0E0H,0               ; PROFILE BYTE 1 AND 2
                                                    ; WARMSTART, LOW BAT WNG ENAB
                                                    ; INIT VIDEO MODE = CGA 80X25
                                                    ; LCD HIGH INT = NO OPERATION
                                                    ; RS232 AND MDM NOT AV ON BATT
0263  0000 0000        DW      0,0                  ; KYBD INACT TIMEOUT VALUES
                                                    ; LCD BLANK = NO TIMEOUT
                                                    ; SYSTEM OFF = NO TIMEOUT
0267  0E 00            DB     0EH,0                 ; DFLT MDM SET 1200BPS, E-PRTY
                                                    ; NO AUTO-ANSWER
              ;
              ; REAL TIME CLOCK VALID SIGNATURE
              ;
0269                  RTC_SIG_SAVE    LABEL   BYTE
0269  52 54 43 47      DB      "RTCG"               ; REAL TIME CLOCK SIGNATURE
                                                    ; WRTN & CKD BY POST (P01MAIN)
```

Printer Configuration Table (PRT_TAB)

```
              ;----------------------------------------
              ; PRINTER PORT SEARCH TABLE USED DURING POST
              ;----------------------------------------
026D                  PRT_TAB         LABEL   WORD
026D  0078             DW      078H                 ; COMPACT PRINTER PORT
026F  0378             DW      378H                 ; PRIMARY PRINTER PORT
0271  03BC             DW      3BCH                 ; MONO ADAPTER PRINTER PORT
0273  0278             DW      278H                 ; ALTERNATE PRINTER PORT
0275                  PRT_TAB_END     LABEL   WORD
```

ASCII Conversion (STR_CON)

```
              ;****************************************************************
              ;
              ; ROUTINE-NAME :  STR_CON
              ;
              ; FUNCTION:   THIS ROUTINE CONVERTS HEX NUMBERS TO ASCII AND STORES
              ;             THEM AT ADDRESS ES:DI.
              ;
              ;
              ; ENTRY CONDITIONS:
              ;  PURPOSE OF ENTRY:  TO POST ERROR MESSAGES IN MEMORY
              ;  INPUT CONDITIONS:  DX = HEX NUMBER TO BE CONVERTED
              ;                     ES:DI = PLACE IN MEMORY TO PUT ASCII STRNG
              ;
              ;  RESTRICTIONS:  NONE
              ;
              ; EXIT CONDITIONS:
              ;  NORMAL EXIT CONDITIONS:
              ;
              ;
              ;  ERROR EXIT CONDITIONS:
              ;
              ;  REGISTERS MODIFIED:  CX,DX,ES,DI
              ;
              ; INTERNALLY REFERENCED ROUTINES:  NONE
              ;
              ; EXTERNALLY REFERENCED ROUTINES:  NONE
              ;
              ;****************************************************************

0275                    STR_CON         PROC    NEAR

0275  51                   PUSH    CX
0276  B1 0C                MOV     CL,12              ; CONVERT 2 BYTES
0278  26: C6 05 20         MOV     BYTE PTR ES:[DI-, 020H ; INSERT ASCII BLANK
027C  47                   INC     DI                 ; INC POINTER
027D  52                SC10:   PUSH    DX            ; SAVE COUNT
027E  D3 EA                SHR     DX,CL              ; MOVE HIGH NIBBLE TO LOW NIB
0280  80 E2 0F             AND     DL,0FH             ; MASK OUT NEW HIGH NIBBLE
0283  80 FA 09             CMP     DL,09              ; ALPHA OR NUMERIC
0286  7E 03                JLE     SC24               ; NO ADJUST FOR ALPHA
0288  80 C2 07             ADD     DL,07              ; ADJUST FOR ALPHA
028B  80 C2 30          SC24:   ADD     DL,30H        ; CONVERT TO ASCII
028E  26: 88 15            MOV     ES:[DI],DL         ; SEND ERROR TO SCREEN
0291  47                   INC     DI                 ; POINT TO NEXT MEMOERY LOCAT
0292  5A                   POP     DX                 ; RESTORE AX
0293  80 E9 04             SUB     CL,4               ; SUB 4 FROM SHIFT COUNT
0296  80 F9 00             CMP     CL,0               ; SEE IF MINUS
0299  7D E2                JGE     SC10               ; LOOP
029B  59                   POP     CX
029C  C3                SC_RET: RET                   ; RETURN
029D                    STR_CON         ENDP
```

NMI Handler Entry Point Address (NMI_INT)

```
              ;*************************************
              ;   NMI HANDLER ENTRY POINT ADDRESS
              ;*************************************

              ;ORG      0E2C3H
02C3                     ORG      002C3H
= 02C3                  NMI_INT          EQU      $
02C3  E9 0000 E          JMP      NMI_FLIH
```

Character Generator Graphics 128–255 (CHAR_GEN_HI)

```
                    ;*****************************************************************
                    ;  CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200
                    ;  GRAPHICS FOR CHARACTERS 80H THROUGH FFH AND FOR DEFAULT
                    ;  LCD CHARACTER GENERATOR
                    ;*****************************************************************

02C6                          CHAR_GEN_HI     LABEL   BYTE
02C6  1D 33 61 60 60 31          DB      01DH,033H,061H,060H,060H,031H,00EH,01CH ; D_80
      0E 1C
02CE  6C 00 66 66 66 66          DB      06CH,000H,066H,066H,066H,066H,03BH,000H ; D_81
      3B 00
02D6  0E 18 3C 66 7E 60          DB      00EH,018H,03CH,066H,07EH,060H,03EH,000H ; D_82
      3E 00
02DE  18 3C 3C 66 1E 66          DB      018H,03CH,03CH,066H,01EH,066H,07BH,000H ; D_83
      7B 00
02E6  36 00 3C 66 1E 66          DB      036H,000H,03CH,066H,01EH,066H,07BH,000H ; D_84
      7B 00
02EE  30 18 3C 66 1E 66          DB      030H,018H,03CH,066H,01EH,066H,07BH,000H ; D_85
      7B 00
02F6  18 18 3C 66 1E 66          DB      018H,018H,03CH,066H,01EH,066H,07BH,000H ; D_86
      7B 00
02FE  00 00 3E 66 60 36          DB      000H,000H,03EH,066H,060H,036H,01CH,038H ; D_87
      1C 38
0306  18 3C 3C 66 7E 60          DB      018H,03CH,03CH,066H,07EH,060H,03EH,000H ; D_88
      3E 00
030E  36 00 3C 66 7E 60          DB      036H,000H,03CH,066H,07EH,060H,03EH,000H ; D_89
      3E 00
0316  30 18 3C 66 7E 60          DB      030H,018H,03CH,066H,07EH,060H,03EH,000H ; D_8A
      3E 00
031E  6C 00 38 18 18 18          DB      06CH,000H,038H,018H,018H,018H,03CH,000H ; D_8B
      3C 00
0326  10 38 08 38 18 18          DB      010H,038H,008H,038H,018H,018H,03CH,000H ; D_8C
      3C 00
032E  30 18 08 38 18 18          DB      030H,018H,008H,038H,018H,018H,03CH,000H ; D_8D
      3C 00
0336  36 08 1C 16 36 3F          DB      036H,008H,01CH,016H,036H,03FH,063H,000H ; D_8E
      63 00
033E  1C 14 1C 1E 36 3F          DB      01CH,014H,01CH,01EH,036H,03FH,063H,000H ; D_8F
      63 00
0346  07 0C 7F 31 3C 31          DB      007H,00CH,07FH,031H,03CH,031H,07FH,000H ; D_90
      7F 00
034E  00 00 76 1B 3F 6C          DB      000H,000H,076H,01BH,03FH,06CH,077H,000H ; D_91
      77 00
0356  3F 3D 2C 3E 6C 6D          DB      03FH,03DH,02CH,03EH,06CH,06DH,06FH,000H ; D_92
      6F 00
035E  18 3C 3C 66 66 66          DB      018H,03CH,03CH,066H,066H,066H,03CH,000H ; D_93
      3C 00
0366  66 00 3C 66 66 66          DB      066H,000H,03CH,066H,066H,066H,03CH,000H ; D_94
      3C 00
036E  70 18 3C 66 66 66          DB      070H,018H,03CH,066H,066H,066H,03CH,000H ; D_95
      3C 00
0376  18 3C 42 66 66 66          DB      018H,03CH,042H,066H,066H,066H,03BH,000H ; D_96
      3B 00
037E  70 18 66 66 66 66          DB      070H,018H,066H,066H,066H,066H,03BH,000H ; D_97
      3B 00
0386  36 00 77 33 1A 0C          DB      036H,000H,077H,033H,01AH,00CH,06CH,038H ; D_98
      6C 38
038E  63 1C 36 63 63 36          DB      063H,01CH,036H,063H,063H,036H,01CH,000H ; D_99
      1C 00
0396  36 41 63 63 63 63          DB      036H,041H,063H,063H,063H,063H,03EH,000H ; D_9A
```

```
     3E 00
039E 06 04 3C 6E 68 6A     DB     006H,004H,03CH,06EH,068H,06AH,03CH,030H ; D_9B
     3C 30
03A6 1E 33 33 7C 39 5B     DB     01EH,033H,033H,07CH,039H,05BH,076H,000H ; D_9C
     76 00
03AE 66 66 3C 7E 18 7E     DB     066H,066H,03CH,07EH,018H,07EH,018H,000H ; D_9D
     18 00
03B6 78 6C 6C 7A 66 6F     DB     078H,06CH,06CH,07AH,066H,06FH,066H,003H ; D_9E
     66 03
03BE 0E 1B 18 3E 18 18     DB     00EH,01BH,018H,03EH,018H,018H,058H,070H ; D_9F
     58 70
03C6 0E 18 3C 66 1E 66     DB     00EH,018H,03CH,066H,01EH,066H,07BH,000H ; D_A0
     7B 00
03CE 1C 30 18 38 18 18     DB     01CH,030H,018H,038H,018H,018H,03CH,000H ; D_A1
     3C 00
03D6 0E 18 3C 66 66 66     DB     00EH,018H,03CH,066H,066H,066H,03CH,000H ; D_A2
     3C 00
03DE 0E 18 66 66 66 66     DB     00EH,018H,066H,066H,066H,066H,03BH,000H ; D_A3
     3B 00
03E6 1A 2C 76 3B 33 33     DB     01AH,02CH,076H,03BH,033H,033H,073H,000H ; D_A4
     73 00
03EE 1A 2C 73 7B 6F 67     DB     01AH,02CH,073H,07BH,06FH,067H,063H,000H ; D_A5
     63 00
03F6 1E 36 36 1F 00 3F     DB     01EH,036H,036H,01FH,000H,03FH,000H,000H ; D_A6
     00 00
03FE 1C 36 36 1C 00 3E     DB     01CH,036H,036H,01CH,000H,03EH,000H,000H ; D_A7
     00 00
0406 18 00 18 30 64 66     DB     018H,000H,018H,030H,064H,066H,03CH,000H ; D_A8
     3C 00
040E 00 00 00 7E 60 60     DB     000H,000H,000H,07EH,060H,060H,000H,000H ; D_A9
     00 00
0416 00 00 00 7E 06 06     DB     000H,000H,000H,07EH,006H,006H,000H,000H ; D_AA
     00 00
041E 60 66 6C 7E 3B 66     DB     060H,066H,06CH,07EH,03BH,066H,04CH,00FH ; D_AB
     4C 0F
0426 60 66 6C 7B 37 6B     DB     060H,066H,06CH,07BH,037H,06BH,04FH,003H ; D_AC
     4F 03
042E 30 00 30 30 30 30     DB     030H,000H,030H,030H,030H,030H,030H,000H ; D_AD
     30 00
0436 00 1B 36 6C 6C 36     DB     000H,01BH,036H,06CH,06CH,036H,01BH,000H ; D_AE
     1B 00
043E 00 6C 36 1B 1B 36     DB     000H,06CH,036H,01BH,01BH,036H,06CH,000H ; D_AF
     6C 00
0446 11 44 11 44 11 44     DB     011H,044H,011H,044H,011H,044H,011H,044H ; D_B0
     11 44
044E 55 AA 55 AA 55 AA     DB     055H,0AAH,055H,0AAH,055H,0AAH,055H,0AAH ; D_B1
     55 AA
0456 EE BB EE BB EE BB     DB     0EEH,0BBH,0EEH,0BBH,0EEH,0BBH,0EEH,0BBH ; D_B2
     EE BB
045E 18 18 18 18 18 18     DB     018H,018H,018H,018H,018H,018H,018H,018H ; D_B3
     18 18
0466 18 18 18 F8 F8 18     DB     018H,018H,018H,0F8H,0F8H,018H,018H,018H ; D_B4
     18 18
046E 18 18 F8 F8 18 F8     DB     018H,018H,0F8H,0F8H,018H,0F8H,018H,018H ; D_B5
     18 18
0476 34 34 34 F4 F4 34     DB     034H,034H,034H,0F4H,0F4H,034H,034H,034H ; D_B6
     34 34
047E 00 00 00 F8 FC 34     DB     000H,000H,000H,0F8H,0FCH,034H,034H,034H ; D_B7
     34 34
0486 00 00 F0 F8 18 F8     DB     000H,000H,0F0H,0F8H,018H,0F8H,018H,018H ; D_B8
     18 18
048E 34 34 F4 F4 04 F4     DB     034H,034H,0F4H,0F4H,004H,0F4H,034H,034H ; D_B9
     34 34
0496 34 34 34 34 34 34     DB     034H,034H,034H,034H,034H,034H,034H,034H ; D_BA
     34 34
049E 00 00 F8 FC 04 F4     DB     000H,000H,0F8H,0FCH,004H,0F4H,034H,034H ; D_BB
     34 34
04A6 34 34 F4 F4 04 FC     DB     034H,034H,0F4H,0F4H,004H,0FCH,000H,000H ; D_BC
     00 00
04AE 34 34 34 FC FC 00     DB     034H,034H,034H,0FCH,0FCH,000H,000H,000H ; D_BD
     00 00
04B6 18 18 F8 F8 18 F8     DB     018H,018H,0F8H,0F8H,018H,0F8H,000H,000H ; D_BE
     00 00
04BE 00 00 00 F8 F8 18     DB     000H,000H,000H,0F8H,0F8H,018H,018H,018H ; D_BF
```

```
        18 18
04C6    18 18 18 1F 1F 00      DB       018H,018H,018H,01FH,01FH,000H,000H,000H ; D_C0
        00 00
04CE    18 18 18 FF FF 00      DB       018H,018H,018H,0FFH,0FFH,000H,000H,000H ; D_C1
        00 00
04D6    00 00 00 FF FF 18      DB       000H,000H,000H,0FFH,0FFH,018H,018H,018H ; D_C2
        18 18
04DE    18 18 18 1F 1F 18      DB       018H,018H,018H,01FH,01FH,018H,018H,018H ; D_C3
        18 18
04E6    00 00 00 FF FF 00      DB       000H,000H,000H,0FFH,0FFH,000H,000H,000H ; D_C4
        00 00
04EE    18 18 18 FF FF 18      DB       018H,018H,018H,0FFH,0FFH,018H,018H,018H ; D_C5
        18 18
04F6    18 18 1F 1F 18 1F      DB       018H,018H,01FH,01FH,018H,01FH,018H,018H ; D_C6
        18 18
04FE    34 34 34 37 37 34      DB       034H,034H,034H,037H,037H,034H,034H,034H ; D_C7
        34 34
0506    34 34 37 37 30 1F      DB       034H,034H,037H,037H,030H,01FH,000H,000H ; D_C8
        00 00
050E    00 00 3F 3F 30 37      DB       000H,000H,03FH,03FH,030H,037H,034H,034H ; D_C9
        34 34
0516    34 34 F7 F7 00 FF      DB       034H,034H,0F7H,0F7H,000H,0FFH,000H,000H ; D_CA
        00 00
051E    00 00 FF FF 00 F7      DB       000H,000H,0FFH,0FFH,000H,0F7H,034H,034H ; D_CB
        34 34
0526    34 34 37 37 30 37      DB       034H,034H,037H,037H,030H,037H,034H,034H ; D_CC
        34 34
052E    00 00 FF FF 00 FF      DB       000H,000H,0FFH,0FFH,000H,0FFH,000H,000H ; D_CD
        00 00
0536    34 34 F7 F7 00 F7      DB       034H,034H,0F7H,0F7H,000H,0F7H,034H,034H ; D_CE
        34 34
053E    18 18 FF FF 00 FF      DB       018H,018H,0FFH,0FFH,000H,0FFH,000H,000H ; D_CF
        00 00
0546    34 34 34 FF FF 00      DB       034H,034H,034H,0FFH,0FFH,000H,000H,000H ; D_D0
        00 00
054E    00 00 FF FF 00 FF      DB       000H,000H,0FFH,0FFH,000H,0FFH,018H,018H ; D_D1
        18 18
0556    00 00 00 FF FF 34      DB       000H,000H,000H,0FFH,0FFH,034H,034H,034H ; D_D2
        34 34
055E    34 34 34 3F 1F 00      DB       034H,034H,034H,03FH,01FH,000H,000H,000H ; D_D3
        00 00
0566    18 18 1F 1F 18 0F      DB       018H,018H,01FH,01FH,018H,00FH,000H,000H ; D_D4
        00 00
056E    00 00 1F 1F 18 1F      DB       000H,000H,01FH,01FH,018H,01FH,018H,018H ; D_D5
        18 18
0576    00 00 00 3F 3F 34      DB       000H,000H,000H,03FH,03FH,034H,034H,034H ; D_D6
        34 34
057E    34 34 34 FF FF 34      DB       034H,034H,034H,0FFH,0FFH,034H,034H,034H ; D_D7
        34 34
0586    18 18 FF FF 18 FF      DB       018H,018H,0FFH,0FFH,018H,0FFH,018H,018H ; D_D8
        18 18
058E    18 18 18 F8 F8 00      DB       018H,018H,018H,0F8H,0F8H,000H,000H,000H ; D_D9
        00 00
0596    00 00 00 1F 1F 18      DB       000H,000H,000H,01FH,01FH,018H,018H,018H ; D_DA
        18 18
059E    FF FF FF FF FF FF      DB       0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH ; D_DB
        FF FF
05A6    00 00 00 00 FF FF      DB       000H,000H,000H,000H,0FFH,0FFH,0FFH,0FFH ; D_DC
        FF FF
05AE    F0 F0 F0 F0 F0 F0      DB       0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H ; D_DD
        F0 F0
05B6    0F 0F 0F 0F 0F 0F      DB       00FH,00FH,00FH,00FH,00FH,00FH,00FH,00FH ; D_DE
        0F 0F
05BE    FF FF FF FF 00 00      DB       0FFH,0FFH,0FFH,0FFH,000H,000H,000H,000H ; D_DF
        00 00
05C6    00 00 3D 6E 66 6E      DB       000H,000H,03DH,06EH,066H,06EH,03BH,000H ; D_E0
        3B 00
05CE    0E 1B 33 3E 33 73      DB       00EH,01BH,033H,03EH,033H,073H,06EH,060H ; D_E1
        6E 60
05D6    7F 33 31 30 30 30      DB       07FH,033H,031H,030H,030H,030H,078H,000H ; D_E2
        78 00
05DE    00 7E FC A8 28 6C      DB       000H,07EH,0FCH,0A8H,028H,06CH,0CCH,000H ; D_E3
        CC 00
05E6    FE 66 30 18 30 66      DB       0FEH,066H,030H,018H,030H,066H,0FEH,000H ; D_E4
```

```
        FE 00
05EE    00 00 3F 6C 6C 6C    DB      000H,000H,03FH,06CH,06CH,06CH,038H,000H ; D_E5
        38 00
05F6    00 00 33 33 33 37    DB      000H,000H,033H,033H,033H,037H,06DH,060H ; D_E6
        6D 60
05FE    00 3F 7E 50 18 1C    DB      000H,03FH,07EH,050H,018H,01CH,00CH,000H ; D_E7
        0C 00
0606    1C 08 3E 6B 3E 08    DB      01CH,008H,03EH,06BH,03EH,008H,01CH,000H ; D_E8
        1C 00
060E    1C 36 63 7F 63 36    DB      01CH,036H,063H,07FH,063H,036H,01CH,000H ; D_E9
        1C 00
0616    1C 36 63 63 36 55    DB      01CH,036H,063H,063H,036H,055H,077H,000H ; D_EA
        77 00
061E    1C 30 18 2C 66 66    DB      01CH,030H,018H,02CH,066H,066H,03CH,000H ; D_EB
        3C 00
0626    00 36 7F 4D 59 7F    DB      000H,036H,07FH,04DH,059H,07FH,036H,000H ; D_EC
        36 00
062E    01 03 3E 67 6B 73    DB      001H,003H,03EH,067H,06BH,073H,03EH,040H ; D_ED
        3E 40
0636    00 00 1E 30 3E 30    DB      000H,000H,01EH,030H,03EH,030H,01EH,000H ; D_EE
        1E 00
063E    3C 66 66 66 66 66    DB      03CH,066H,066H,066H,066H,066H,066H,000H ; D_EF
        66 00
0646    00 7E 00 7E 00 7E    DB      000H,07EH,000H,07EH,000H,07EH,000H,000H ; D_F0
        00 00
064E    18 18 7E 18 18 00    DB      018H,018H,07EH,018H,018H,000H,07EH,000H ; D_F1
        7E 00
0656    60 38 0E 38 60 00    DB      060H,038H,00EH,038H,060H,000H,07EH,000H ; D_F2
        7E 00
065E    06 1C 70 1C 06 00    DB      006H,01CH,070H,01CH,006H,000H,07EH,000H ; D_F3
        7E 00
0666    0E 1B 1A 18 18 18    DB      00EH,01BH,01AH,018H,018H,018H,018H,018H ; D_F4
        18 18
066E    18 18 18 18 58 D8    DB      018H,018H,018H,018H,058H,0D8H,070H,000H ; D_F5
        70 00
0676    18 18 00 7E 00 18    DB      018H,018H,000H,07EH,000H,018H,018H,000H ; D_F6
        18 00
067E    00 3B 6E 00 3B 6E    DB      000H,03BH,06EH,000H,03BH,06EH,000H,000H ; D_F7
        00 00
0686    1C 36 36 1C 00 00    DB      01CH,036H,036H,01CH,000H,000H,000H,000H ; D_F8
        00 00
068E    00 00 08 1C 1C 08    DB      000H,000H,008H,01CH,01CH,008H,000H,000H ; D_F9
        00 00
0696    00 00 00 18 18 00    DB      000H,000H,000H,018H,018H,000H,000H,000H ; D_FA
        00 00
069E    03 02 06 24 6C 38    DB      003H,002H,006H,024H,06CH,038H,018H,010H ; D_FB
        18 10
06A6    76 3B 33 33 33 00    DB      076H,03BH,033H,033H,033H,000H,000H,000H ; D_FC
        00 00
06AE    3C 66 0C 38 7E 00    DB      03CH,066H,00CH,038H,07EH,000H,000H,000H ; D_FD
        00 00
06B6    00 00 3C 3C 3C 3C    DB      000H,000H,03CH,03CH,03CH,03CH,000H,000H ; D_FE
        00 00
06BE    00 00 00 00 00 00    DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_FF
        00 00
```

Convert AX to ASCII (DSP_BYTE)

```
              ;**********************************************************************
              ;
              ; ROUTINE-NAME :  DSP_BYTE
              ;
              ; FUNCTION: THIS ROUTINE WILL CONVERT THE BYTE IN AL INTO TWO ASCII
              ;           CHARACTERS IN AX AND DISPLAY THEM AT THE CURRENT
              ;           CURSOR LOCATION. THE CURSOR WILL BE MOVED TWO CHARACTERS
              ;           TO THE RIGHT.
              ;
              ; ENTRY CONDITIONS:
              ;  INPUT CONDITIONS: DSP_BYTE: AL = BYTE TO BE CONVERTED
              ;                    XLAT_NIB: AL = LOW NIBBLE TO BE CONVERTED
              ;                    DSP_HEX:  AL = BYTE TO BE DISPLAYED
              ;
              ;  RESTRICTIONS:     NONE
              ;
              ; EXIT CONDITIONS:
              ;          DSP_BYTE: BYTE IS CONVERTED AND PRINTED, ASCII IN AX
              ;          XLAT_NIB: ASCII RETURNED IN AL
              ;          DSP_HEX:  CHARACTER DISPLAYED AND CURSOR ADVANCED
              ;
              ; REGISTERS MODIFIED:    AX
              ;
              ;**********************************************************************

06C6                  DSP_BYTE       PROC    NEAR
06C6  51               PUSH    CX
06C7  50               PUSH    AX                  ; SAVE FOR LOW NIBBLE DISPLAY
06C8  B1 04            MOV     CL,4                ; SHIFT COUNT
06CA  D2 E8            SHR     AL,CL               ; NYBBLE SWAP
06CC  E8 06E0 R        CALL    XLAT_NIB            ; CONVERT THE HIGH NIBBLE
06CF  E8 06E7 R        CALL    DSP_HEX             ; DISPLAY THE FIRST CHARACTER
06D2  8A E8            MOV     CH,AL               ; SAVE UPPER ASCII CHARACTER
06D4  58               POP     AX                  ; RECOVER THE NIBBLE
06D5  24 0F            AND     AL,0FH              ; ISOLATE TO LOW NIBBLE
                                                   ; DO LOW NIBBLE CONVERSION
06D7  E8 06E0 R        CALL    XLAT_NIB            ; 2ND ASCII CHARACTER IN AL
06DA  8A E5            MOV     AH,CH               ; RESTORE 1ST ASCII CHAR TO AH
06DC  59               POP     CX
06DD  EB 08 90         JMP     DSP_HEX             ; DISPLAY SECOND CHARACTER
06E0                  DSP_BYTE       ENDP
              ;
              ; CONVERT BYTE IN LOW NIBBLE IN AL TO ASCII IN AL
              ;
06E0                  XLAT_NIB       PROC    NEAR
06E0  04 90            ADD     AL,090H             ; ADD FIRST CONVERSION FACTOR
06E2  27               DAA                         ; ADJ FOR NUM AND ALPHA RANGE
06E3  14 40            ADC     AL,040H             ; ADD CONV AND ADJ LOW NIBBLE
06E5  27               DAA                         ; ADJ HIGH NIB TO ASCHI RANGE
06E6  C3               RET
06E7                  XLAT_NIB       ENDP
              ;
              ; PRINT CHARACTER IN AL USING TELETYPE INTERFACE TO VIDEO I/O
              ;
06E7                  DSP_HEX        PROC    NEAR
06E7  50               PUSH    AX                  ; SAVE REGS
06E8  53               PUSH    BX
06E9  B4 0E            MOV     AH,14               ; DISPLAY CHARACTER IN AL
06EB  B7 00            MOV     BH,0
06ED  CD 10            INT     10H                 ; CALL VIDEO_IO
06EF  5B               POP     BX                  ; RESTORE REGS
06F0  58               POP     AX
06F1  C3               RET
06F2                  DSP_HEX        ENDP
```

Boot Strap Loader Entry Address (BOOT_STRAP)

```
                ;***********************************
                ;   BOOT STRAP LOADER ENTRY ADDRESS
                ;***********************************
                ;ORG     0E6F2H
06F2                       ORG     006F2H
= 06F2                     BOOT_STRAP      EQU     $
06F2  E9 0000 E            JMP     SYS_BOOT
```

Keyboard Noise (KB_NOISE)

```
                ;---------------------------------------------------------------
                ;KB_NOISE
                ;       THIS ROUTINE IS CALLED WHEN GENERAL BEEPS ARE REQUIRED FROM
                ;       THE SYSTEM.
                ;INPUT
                ;       DS= BIOS DATA SEGMENT
                ;       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.
                ;
                ;---------------------------------------------------------------
06F5                       KB_NOISE        PROC   NEAR

06F5  F6 06 0016 R 01         TEST    BIOS_STATUS,KB_NOISE_ACT ; ROUTINE ALREADY ACTIVE?
06FA  75 29                   JNZ     KBN_EXIT             ; IF SO THEN EXIT
06FC  80 0E 0016 R 01         OR      BIOS_STATUS,KB_NOISE_ACT ; SET ROUTINE ACTIVE FLAG
0701  50                      PUSH    AX
0702  53                      PUSH    BX
0703  51                      PUSH    CX

0704  E4 61                   IN      AL,NMI_CNTL          ; GET CONTROL INFO
0706  50                      PUSH    AX                   ; SAVE
0707                       LOOP01:
0707  24 FC                   AND     AL,NOT SPKR_DATA+TMR2_GATE ; TURN OFF SPEAKER
                                                           ; DATA
0709  E6 61                   OUT     NMI_CNTL,AL          ; OUTPUT TO CONTROL
070B  51                      PUSH    CX                   ; HALF CYCLE TIME FOR TONE
070C  E2 FE                   LOOP    $                    ; SPEAKER OFF

070E  0C 02                   OR      AL,SPKR_DATA         ; TURN ON SPEAKER
0710  E6 61                   OUT     NMI_CNTL,AL          ; OUTPUT TO CONTROL
0712  59                      POP     CX
0713  51                      PUSH    CX                   ; RETRIEVE FREQUENCY
0714  E2 FE                   LOOP    $                    ; ANOTHER HALF CYCLE

0716  59                      POP     CX                   ; RETRIEVE  FREQ.
0717  4B                      DEC     BX                   ; TOTAL TIME COUNT
0718  75 ED                   JNZ     LOOP01               ; DO ANOTHER CYCLE

071A  58                      POP     AX                   ; RECOVER CONTROL

071B  E6 61                   OUT     NMI_CNTL,AL          ; RESTORE THE CONTROL REGISTER
071D  80 26 0016 R FE         AND     BIOS_STATUS,NOT KB_NOISE_ACT ; RESET ACTIVE FLAG
0722  59                      POP     CX
0723  5B                      POP     BX
0724  58                      POP     AX
0725                       KBN_EXIT:
```

```
0725  C3                      RET
0726                        KB_NOISE          ENDP
```

Communications Baud Rate Table (BAUD_TABLE)

```
               ;************************************
               ;  COMMUNICATIONS BAUD RATE TABLE
               ;************************************
               ;ORG     0E729H
0729                     ORG     00729H
0729                    BAUD_TABLE        LABEL   WORD
0729  0417                 DW      1047                 ; 110 BAUD  ; TBL OF INIT VAL
072B  0300                 DW      768                  ; 150
072D  0180                 DW      384                  ; 300
072F  00C0                 DW      192                  ; 600
0731  0060                 DW      96                   ; 1200
0733  0030                 DW      48                   ; 2400
0735  0018                 DW      24                   ; 4800
0737  000C                 DW      12                   ; 9600
```

RS-232 I/O Entry Point (RS232_IO)

```
               ;************************************
               ;  RS232 I/O ENTRY POINT
               ;************************************
               ;ORG     0E739H
0739                     ORG     00739H
= 0739                  RS232_IO        EQU     $
0739  E9 0000 E          JMP     COMMO_IO
```

Indicate POST Error (ERR_BEEP)

```
               ;*********************************************************************
               ;
               ; ROUTINE-NAME :  ERR_BEEP
               ;
               ; FUNCTION: THIS ROUTINE WILL ISSUE LONG (3 SEC) AND/OR SHORT (1 SEC)
               ;        TONES TO THE SPEAKER TO INDICATE POST STATUS TO THE OPERATOR
               ;
               ;
               ; ENTRY CONDITIONS:
               ;       PURPOSE OF ENTRY: SEND BEEP TONES TO THE SPEAKER
               ;       INPUT CONDITIONS: DH = NUMBER OF LONG TONES TO SOUND
               ;                         DL = NUMBER OF SHORT TONES TO SOUND
               ;       RESTRICTIONS:     NONE
               ;
               ; EXIT CONDITIONS:
               ;       NORMAL EXIT CONDITIONS: SPEAKER IS SOUNDED
               ;
               ;       ERROR EXIT CONDITIONS: NONE
               ;
```

```
                ;       REGISTERS MODIFIED:    AX,BX,CX,DX
                ;
                ;*********************************************************************
= 0080                  FRC_BEP          EQU       80H ; FORCED BEEP BIT

073C                    ERR_BEEP        PROC NEAR
073C  0A F6              OR      DH,DH                  ; ANY LONG TONES?
073E  74 0D              JZ      EB_03                  ; NO THEN DO SHORT
                ;
                ; ISSUE LONG BEEPS
                ;
0740                    EB_01:
0740  B3 86              MOV     BL,FRC_BEP+6           ; SET BEEP CNTR LNG & FRC BEEP
0742  E8 075F R          CALL    BEEP
0745  2B C9              SUB     CX,CX                  ; CLEAR CX REGISTER
0747  E2 FE             EB_02:    LOOP    EB_02         ; DELAY BETWEEN BEEPS
0749  FE CE              DEC     DH                     ; DECREMENT LONG BEEP COUNTER
074B  75 F3              JNZ     EB_01                  ; ISSUE NEXT LONG BEEP
                ;
                ; ISSUE SHORT BEEPS
                ;
074D                    EB_03:
074D  0A D2              OR      DL,DL                  ; ANY SHORT BEEPS?
074F  74 0D              JZ      EB_EXIT
0751                    EB_04:
0751  B3 81              MOV     BL,FRC_BEP+1           ; SET BEEP CNTR FOR SHORT BEEP
0753  E8 075F R          CALL    BEEP
0756  2B C9              SUB     CX,CX                  ; CLEAR CX REGISTER
0758  E2 FE             EB_05:    LOOP    EB_05         ; DELAY BETWEEN BEEPS
075A  FE CA              DEC     DL
075C  75 F3              JNZ     EB_04                  ; NEXT SHRT BEEP IF NOT COMPLT
075E                    EB_EXIT:
075E  C3                 RET                            ; RETURN TO CALLER
075F                    ERR_BEEP        ENDP
```

Beep to Speaker (BEEP)

```
                ;********************************************************************
                ;
                ; ROUTINE-NAME :  BEEP
                ;
                ; FUNCTION:   TO BEEP THE SPEAKER
                ;
                ; ENTRY CONDITIONS:
                ;     PURPOSE OF ENTRY: TO BEEP THE SPEAKER
                ;     INPUT CONDITIONS: BL CONTAINS THE COUNT FOR THE LENGTH OF TIME
                ;        MULTIPLIED BY 500 MSECS  FOR THE SPEAKER TO SOUND. IF THE MSB
                ;        IN BL IS SET THE SPEAKER IS SOUNDED WHETHER OR NOT IT WAS
                ;        DISABLED.
                ;     RESTRICTIONS: NONE
                ;
                ; EXIT CONDITIONS:
                ;     NORMAL EXIT CONDITIONS:
                ;     ERROR EXIT CONDITIONS:
                ;     REGISTERS MODIFIED:    NONE
                ;
                ; INTERNALLY REFERENCED ROUTINES: NONE
                ;
                ; EXTERNALLY REFERENCED ROUTINES:
                ;
                ;********************************************************************

075F                    BEEP       PROC      NEAR
075F  50                 PUSH      AX                   ; SAVE REGISTERS
0760  53                 PUSH      BX
0761  51                 PUSH      CX
0762                    BEEP_SUB:                       ; NO STACK ENTRY POINT
0762  B0 B6             BP1:       MOV       AL,0B6H    ; TIMER 2,MSB,LSB,BINARY
```

```
0764  E6 43            OUT     TIMER_CTL,AL        ; WRITE TO TIMER CONTROL PORT
0766  B8 0533          MOV     AX,533H             ; DIVISOR FOR 1000 HZ
0769  E6 42            OUT     TIMER2,AL           ; TIMER 2 COUNT LSB
076B  8A C4            MOV     AL,AH
076D  E6 42            OUT     TIMER2,AL           ; TIMER 2 COUNT MSB

076F  E4 61            IN      AL,NMI_CNTL         ; GET SPEAKER ENABLE BIT
0771  8A E0            MOV     AH,AL               ; SAVE SPEAKER ENABLE SETTING
0773  0C 03            OR      AL,SPKR_DATA+TMR2_GATE ; ENABLE SPEAKER
0775  F6 C3 80         TEST    BL,FRC_BEP          ; CHECK FOR FORCED BEEP BIT
0778  74 02            JZ      BP10                ; JUMP IF NOT FORCED BEEP

077A  0C 04           BP5:     OR        AL,EN_SPKR ; SET SPEAKER ENABLE BIT
077C  E6 61           BP10:    OUT       NMI_CNTL,AL ; TURN ON SPEAKER ENABLE
077E  80 E3 7F         AND     BL,0FFH-FRC_BEP     ; MASK OUT FORCED BEEP BIT

0781  B9 0000          MOV     CX,0                ; SET CNT TO WAIT 500 MSECS
0784  E2 FE           BP15:    LOOP      BP15      ; WAIT 500 MSECS
0786  FE CB            DEC     BL                  ; DECREMENT COUNTER
0788  75 FA            JNZ     BP15                ; LOOP IF COUNT NOT ZERO

078A  8A C4           BP20:    MOV       AL,AH     ; SET PREVIOUS PORT SETTING
078C  E6 61            OUT     NMI_CNTL,AL         ; SEND TO PPI

078E  B8 F000          MOV     AX,0F000H           ; CHECK FOR ROM STACK
0791  8C D3            MOV     BX,SS
0793  3B C3            CMP     AX,BX
0795  74 03            JE      BEEP_EXT            ; IF ROM STACK THEN RETURN

0797  59               POP     CX                  ; RESTORE REGISTERS
0798  5B               POP     BX
0799  58               POP     AX
079A                  BEEP_EXT:
079A  C3               RET                         ; RETURN
079B                  BEEP      ENDP
```

Disable All Interrupts (DISABLE_NMI)

```
               ;*************************
               ; DISABLE ALL INTERRUPTS
               ;*************************
079B                  DISABLE_NMI     PROC    NEAR
079B  FA               CLI                        ; DISABLE MASKABLE INTERRUPTS
079C  50               PUSH    AX
079D  B0 07            MOV     AL,DISABLE_SLEEP+CLOCK_RUN ; DISABLE NMIS
079F  E6 72            OUT     CLOCK_CTL,AL       ; WRITE TO PORT
07A1  58               POP     AX
07A2  C3               RET
07A3                  DISABLE_NMI     ENDP
```

Enable Global NMIs (ENABLE_NMI)

```
               ;***********************************************************
               ; ENABLE GLOBAL NMI'S
               ;***********************************************************
07A3                  ENABLE_NMI      PROC    NEAR
07A3  50               PUSH    AX
```

```
07A4  B0 27            MOV     AL,DISABLE_SLEEP+CLOCK_RUN+GLOBAL_NMI ; ENABLE NMIS
07A6  E6 72            OUT     CLOCK_CTL,AL          ; WRITE TO PORT
07A8  58               POP     AX
07A9  C3               RET
07AA                  ENABLE_NMI      ENDP
```

Get RTC Register (GET_RTC_REG)

```
                ;*********************************************************************
                ; ROUTINE-NAME :  GET_RTC_REG
                ;
                ; FUNCTION: THIS ROUTINE WILL GET THE DESIGNATED RTC REGISTER LOCATION
                ;      INTO THE AL REGISTER. ALL INTERRUPTS WILL BE DISABLED DURING
                ;      THIS PROCESS AND RESTORED WHEN COMPLETE.
                ; ENTRY CONDITIONS:
                ;      INPUT CONDITIONS: AH = REAL TIME CLOCK REGISTER NUMBER
                ;      RESTRICTIONS:     NONE
                ;
                ; EXIT CONDITIONS:        AL = CONTENTS OF SPECIFIED REGISTER
                ;
                ; REGISTERS MODIFIED:    AL
                ;*********************************************************************
07AA                  GET_RTC_REG     PROC    NEAR
07AA  9C               PUSHF
07AB  FA               CLI                           ; DISABLE INTERRUPTS
07AC  53               PUSH    BX
07AD  E4 72            IN      AL,CLOCK_CTL
07AF  8A D8            MOV     BL,AL                 ; BL <-- NMI CONTROL STATE
07B1  24 DF            AND     AL,NOT GLOBAL_NMI     ; DISABLE NMI
07B3  E6 72            OUT     CLOCK_CTL,AL
07B5  8A C4            MOV     AL,AH
07B7  E6 70            OUT     RTCR_PORT,AL          ; WRITE RAM ADDRESS
07B9  EB 00            JMP     $+2
07BB  E4 71            IN      AL,RTCD_PORT          ; READ DATA
07BD  8A F8            MOV     BH,AL                 ; BH <-- DATA
07BF  8A C3            MOV     AL,BL
07C1  E6 72            OUT     CLOCK_CTL,AL          ; RESTORE NMI STATE
07C3  8A C7            MOV     AL,BH                 ; RESTORE DATA
07C5  5B               POP     BX
07C6  9D               POPF                          ; RESTORE INTERRUPT STATE
07C7  C3               RET
07C8                  GET_RTC_REG     ENDP
```

Put RTC Register (PUT_RTC_REG)

```
                ;*********************************************************************
                ; ROUTINE-NAME :  PUT_RTC_REG
                ;
                ; FUNCTION: THIS RTNE WILL WRITE THE DESIGNATED RTC REGISTER LOCATION
                ;      WITH THE CONTENTS OF THE AL REGISTER. ALL INTERRUPTS WILL BE
                ;      DISABLE DURING THIS PROCESS AND RESTORED WHEN COMPLETE.
                ;
                ; ENTRY CONDITIONS:
                ;     INPUT CONDITIONS: AH = REAL TIME CLOCK REGISTER NUMBER
                ;                       AL = DATA TO BE STORED INTO REGISTER
                ;     RESTRICTIONS:     NONE
                ;
                ; EXIT CONDITIONS:       DATA IN REGISTER MODIFIED
                ;
                ; REGISTERS MODIFIED:    NONE
                ;*********************************************************************
07C8                    PUT_RTC_REG     PROC    NEAR
07C8  9C                 PUSHF
07C9  FA                 CLI                            ; DISABLE INTERRUPTS
07CA  53                 PUSH    BX
07CB  8A D8              MOV     BL,AL                  ; BL <-- DATA
07CD  E4 72              IN      AL,CLOCK_CTL
07CF  8A F8              MOV     BH,AL                  ; BH <-- NMI CONTROL SAVE
07D1  24 DF              AND     AL,NOT GLOBAL_NMI      ; DISABLE NMI
07D3  E6 72              OUT     CLOCK_CTL,AL
07D5  8A C4              MOV     AL,AH                  ; GET ADDRESS
07D7  E6 70              OUT     RTCR_PORT,AL           ; WRITE RAM ADDRESS
07D9  8A C3              MOV     AL,BL                  ; GET DATA
07DB  E6 71              OUT     RTCD_PORT,AL           ; WRITE DATA
07DD  8A C7              MOV     AL,BH
07DF  E6 72              OUT     CLOCK_CTL,AL           ; RESTORE NMI STATE
07E1  8A C3              MOV     AL,BL                  ; RESTORE DATA
07E3  5B                 POP     BX
07E4  9D                 POPF                           ; RESTORE INTERRUPT STATE
07E5  C3                 RET
07E6                    PUT_RTC_REG     ENDP
```

Setup for Battery Savings (BAT_SAV_SETUP)

```
                ;******************************************************************
                ; BAT_SAV_SETUP
                ;     THIS ROUTINE ENABLES THE RTC 1 SEC UPDATE ENDED INTERRUPT AS A
                ;     TIME BASE FOR THE LCD BLANK AND THE SYSTEM POWER OFF OPTIONS
                ;     IF THE SYSTEM PROFILE INDICATES THAT ONE OR BOTH OF THESE
                ;     OPTIONS ARE ENABLED. IF NOT OR WE ARE ON EXTERNAL PWR THEN THE
                ;     INTERRUPT IS DISABLED.
                ;
                ; RESTRICTIONS: RTC, AND SYSTEM SUSPEND NMIS MUST BE DISABLED
                ;     BEFORE CALL TO THIS ROUTINE
                ; REGISTERS MODIFIED:
                ;     AX,BX,CX
                ;
                ;******************************************************************
07E6                    BAT_SAV_SETUP   PROC    NEAR
07E6  B4 0B              MOV     AH,RTC_MODE            ; GET CURRENT MODE
07E8  E8 07AA R          CALL    GET_RTC_REG
07EB  8B D8              MOV     BX,AX                  ; SAVE ADDRESS AND MODE

07ED  80 CB 10           OR      BL,UIE_ENABLE          ; DEFAULT TO SET ENABLE ON
07F0  B9 0004            MOV     CX,4                   ; CHECK THE FOUR TIME VALUES
07F3  B4 19              MOV     AH,RTC_LCD_INACT       ; SPECIFY BEG @ OF PROFILE

07F5  E8 07AA R         BAT_S01:        CALL    GET_RTC_REG
07F8  3C 00              CMP     AL,0                   ; CHECK TIME FOR 0
07FA  75 10              JNE     BAT_S02                ; IF NOT 0, OPTION ACTIVATED
07FC  FE C4              INC     AH                     ; LOOP TO CHECK NEXT BYTE
07FE  E2 F5              LOOP    BAT_S01

0800  B4 17              MOV     AH,RTC_SYS_PROF1       ; GET LOW BAT WARNING STATE
0802  E8 07AA R          CALL    GET_RTC_REG
0805  A8 40              TEST    AL,LOWBAT_ENABLE
0807  75 03              JNZ     BAT_S02                ; JUMP IF WARNING ENABLED
                ;
                ;  DISABLE UPDATE ENDED INTERRUPT (BATTERY SAVINGS MODE IS DISABLED)
                ;
0809  80 E3 EF           AND     BL,NOT UIE_ENABLE      ; TURN OFF INTERRUPT
                ;
```

```
               ; UPDATE MODE REGISTER IN RTC
               ;
080C  8B C3          BAT_S02:       MOV     AX,BX  ; RESTORE MODE ADDR AND DATA
080E  E8 07C8 R       CALL    PUT_RTC_REG
0811  E4 61           IN      AL,NMI_CNTL
0813  24 F7           AND     AL,NOT DIS_ALARM      ; ENABLE RTC INTERRUPT
0815  E6 61           OUT     NMI_CNTL,AL
0817  1E              PUSH    DS
0818  E8 085C R       CALL    DDS                   ; SET DS TO DATA SEGMENT
                      ASSUME  DS:DATA
081B  80 0E 0016 R 20 OR      BIOS_STATUS,KYBD_ACTIVE ; FORCE RELOAD OF COUNTERS
0820  1F              POP     DS
0821  C3              RET
0822                 BAT_SAV_SETUP   ENDP
```

Keyboard I/O Entry Point (KEYBOARD_IO)

```
               ;*************************************
               ;  KEYBOARD I/O ENTRY POINT
               ;*************************************
               ;ORG     0E82EH
082E                  ORG     0082EH
= 082E               KEYBOARD_IO     EQU     $
082E  E9 0000 E       JMP     KYBD_IO
```

Keyboard Reset (KYBD_RESET)

```
                        SUBTTL  KYBD_RESET
               ;*********************************************************************
               ;
               ; ROUTINE-NAME :  KYBD_RESET
               ;
               ; FUNCTION: THIS ROUTINE INITIALIZES THE KEYBOARD CONTROL AREA TO A
               ;      NO KEY CONDITION.
               ;
               ; ENTRY CONDITIONS:
               ;     PURPOSE OF ENTRY: PERFORM KEYBOARD DATA AREA INITIALIZATION
               ;     INPUT CONDITIONS: DS:DATA, NMI MUST BE DISABLED
               ;     RESTRICTIONS:     KEYBOARD NMI'S MUST BE DISABLED
               ;
               ; EXIT CONDITIONS:
               ;     NORMAL EXIT CONDITIONS: KEYBOARD DATA AREA INITIALIZED
               ;
               ;     ERROR EXIT CONDITIONS:  NONE
               ;
               ;     REGISTERS MODIFIED:    ES,SI,DI,AX,CX
               ;
               ;*********************************************************************
0831                 KYBD_RESET      PROC    NEAR

                        ASSUME  ES:DATA,DS:DATA

0831  1E                PUSH    DS
0832  07                POP     ES
               ;
               ; CLEAR INTERRUPT 9 AND INTERRUPT 16 BUFFER AND CONTROL FLAGS
               ;
0833  2B C0             SUB     AX,AX
0835  BF 0017 R         MOV     DI,OFFSET KB_AREA1  ; CLEAR KEYBOARD AREA 1
0838  B9 0027           MOV     CX,KB_AREA1_LNG
```

```
083B  F3/ AA              REP     STOSB
083D  A2 0096 R           MOV     KB_FLAG_3,AL        ; CLEAR ADDITIONAL FLAGS
0840  89 3E 0082 R        MOV     BUFFER_END,DI       ; SET UP BUFFER_END ADDR
                ;
                ; SETUP UP DEFAULT KEYBOARD BUFFER POINTERS
                ;
0844  BE 001E R           MOV     SI,OFFSET KB_BUFFER ; SETUP KEYBOARD PARAMETERS
0847  89 36 001A R        MOV     BUFFER_HEAD,SI
084B  89 36 001C R        MOV     BUFFER_TAIL,SI
084F  89 36 0080 R        MOV     BUFFER_START,SI
                ;
                ; INITIALIZE KEYBOARD NMI CONTROL AREA AND BUFFER (KB_AREA2)
                ;
0853  BF 00B4 R           MOV     DI,OFFSET KB_AREA2
0856  B9 0019             MOV     CX,KB_AREA2_LNG
0859  F3/ AA              REP     STOSB
085B  C3                  RET
085C                    KYBD_RESET ENDP
```

Set Data Segment (DDS)

```
                ;*******************************************************************
                ; ROUTINE-NAME :  DDS
                ;
                ; FUNCTION: SET DATA SEGMENT (DS) TO BIOS DATA AREA
                ;
                ; ENTRY CONDITIONS:
                ;     INPUT CONDITIONS: NONE
                ;     RESTRICTIONS:     NONE
                ;
                ; EXIT CONDITIONS:    DS = SET TO BIOS DATA SEGMENT
                ; REGISTERS MODIFIED: DS
                ;*******************************************************************
085C                    DDS        PROC    NEAR
085C  50                 PUSH    AX                     ; SAVE AX
085D  B8 ---- R          MOV     AX,DATA
0860  8E D8              MOV     DS,AX                  ; SET SEGMENT
0862  58                 POP     AX                     ; RESTORE AX
0863  C3                 RET
0864                    DDS        ENDP
```

Calculate Absolute Vector Offset (GET_VECTOR@)

```
                ;**************************************************
                ; ROUTINE:  GET_VECTOR@
                ; FUNCTION:  CALCULATE ABSOLUTE VECTOR OFFSET OF
                ;            HARDWARE INTERRUPT LEVELS.
                ;
                ;    INPUT:    CL = HARDWARE INTERRUPT LEVEL # 0-7
                ;    OUTPUT:   SI = ABSOLUTE OFFSET ADDRESS OF VECTOR
                ;
                ;    REGISTERS MODIFIED:  SI
                ;***************************************************
0864                    GET_VECTOR@      PROC  NEAR
0864  50                  PUSH  AX
0865  2A E4               SUB   AH,AH
0867  E4 72               IN    AL,CLOCK_CTL
0869  24 BF               AND   AL,0BFH                 ; SET READ INTERRUPT REG 0
086B  E6 72               OUT   CLOCK_CTL,AL
086D  E4 63               IN    AL,63H                  ; READ CURRENT IRPT TYPE BITS
```

```
086F  24 F8                 AND   AL,0F8H               ; CLEAR UNUSED BITS
0871  02 C1                 ADD   AL,CL                 ; ADD OFFSET TO LEVEL #
0873  D1 E0                 SHL   AX,1                  ; MULTIPLY BY 4
0875  D1 E0                 SHL   AX,1
0877  8B F0                 MOV   SI,AX                 ; MOVE INTO OUTPUT REGISTER
0879  58                    POP   AX
087A  C3                    RET
087B                      GET_VECTOR@     ENDP
```

Keyboard Support Tables

```
               ;****************************************************
               ; KEYBOARD SUPPORT TABLES
               ;
               ; NOTE:  A -1 ENTRY INDICATES THE CORRESPONDING KEY IS
               ;   EXCLUDED FROM THE TABLE
               ;****************************************************
               ;ORG     0E87EH
087E                        ORG     0087EH
               ;-----------------------------------------------------------------
               ;----- SHIFT & STATE KEY PC1 SCAN CODES                          -
               ;-----------------------------------------------------------------
087E                      K6 LABEL   BYTE
087E  52 3A 45 46            DB      INS_KEY,CAPS_KEY,NUM_KEY,SCROLL_KEY
0882  38 1D 2A 36            DB      ALT_KEY,CTL_KEY,LEFT_KEY,RIGHT_KEY
= 0008                    K6LEQU     $-K6
               ;-----------------------------------------------------------------
               ;----- SHIFT & STATE KEY "MASKS"                                 -
               ;-----------------------------------------------------------------
0886                      K7 LABEL   BYTE
0886  80 40 20 10            DB      INS_SHIFT,CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT ; KB
088A  08 04 02 01            DB      ALT_SHIFT,CTL_SHIFT,LEFT_SHIFT,RIGHT_SHIFT ; KB
               ;-----------------------------------------------------------------
               ;----- CTL + TYPEWRITER KEYS  -  ASCII CODES                     -
               ;-----------------------------------------------------------------
088E  1B FF 00 FF FF FF K8         DB      ESC,-1,NUL,-1,-1,-1,RS,-1 ; ESC 1 2 3 4 5 6 7
      1E FF
0896  FF FF FF 1F FF 7F  DB      -1,-1,-1,US,-1,DEL,-1,DC1 ; 8 9 0 - = BKSPC TAB Q
      FF 11
089E  17 05 12 14 19 15  DB      ETB,ENQ,DC2,DC4,EM,NAK,HT,SI015 ; W E R T Y U I O
      09 0F
08A6  10 1B 1D 0A FF 01  DB      DLE,ESC,GS,LF,-1,SOH,DC3 ; P [ ] ENTER CTL A S
      13
08AD  04 06 07 08 0A 0B  DB      EOT,ACK006,BEL,BKSPC,LF,VT,FF,-1,-1 ; D F G H J K L ; '
      0C FF FF
08B6  FF FF 1C 1A 18 03  DB      -1,-1,FS,SUB,CAN,ETX,SYN,STX ; Ω LSHFT \ Z X C V B
      16 02
08BE  0E 0D FF FF FF FF  DB      SO,ENTER,-1,-1,-1,-1,-1,-1 ; B N M , . / RSHFT ALT
      FF FF
08C6  20 FF              DB      SPACE,-1               ; SPACE CAPSLOCK
               ;-----------------------------------------------------------------
               ;----- CTL + (F KEYS) & (CURSOR KEYS) - EXTENDED ASCII CODES     -
               ;-----------------------------------------------------------------
08C8                      K9 LABEL   BYTE
                                                        ; CTL + (F1 - F10)
08C8  5E 5F 60 61 62     DB      94,95,96,97,98
08CD  63 64 65 66 67     DB      99,100,101,102,103
                                                        ; CTL + CURSOR KEYS
08D2  FF FF 77 FF 84     DB      -1,-1,CTL_HOME,-1,CTL_PGUP
08D7  FF 73 FF 74        DB      -1,CTL_CUR_LFT,-1,CTL_CUR_RHT
08DB  FF 75 FF 76 FF FF  DB      -1,CTL_END,-1,CTL_PGDN,-1,-1
               ;-----------------------------------------------------------------
               ;----- LOWER CASE TYPEWRITER KEYS - ASCII CODES                  -
               ;-----------------------------------------------------------------
08E1                      K10LABEL   BYTE
08E1  1B 31 32 33 34 35  DB      ESC,'1234567890-=',BKSPC,TAB
      36 37 38 39 30 2D
      3D 08 09
08F0  71 77 65 72 74 79  DB      'qwertyuiop[]',ENTER,-1,'asdfghjkl; ',APOSTR
      75 69 6F 70 5B 5D
      0D FF 61 73 64 66
      67 68 6A 6B 6C 3B
      27
0909  60 FF 5C 7A 78 63  DB      'Ω',-1,'\zxcvbnm,./',-1,'*',-1,SPACE,-1
      76 62 6E 6D 2C 2E
      2F FF 2A FF 20 FF
                 ;-----------------------------------------------------------
                 ;----- UPPER CASE TYPEWRITER KEYS - ASCII CODES            -
                 ;-----------------------------------------------------------
091B                   K11LABEL   BYTE
091B  1B 21 40 23 24 25  DB      ESC,'!@#$%¬&*()_+',BKSPC,PSEUDO ; PSEUDO SHFT_TAB
      5E 26 2A 28 29 5F
      2B 08 00
092A  51 57 45 52 54 59  DB      'QWERTYUIOP{}',ENTER,-1,'ASDFGHJKL:"~'
      55 49 4F 50 7B 7D
      0D FF 41 53 44 46
      47 48 4A 4B 4C 3A
      22 7E
0944  FF 7C 5A 58 43 56  DB      -1,'|ZXCVBNM<>?',-1,PSEUDO ; PSEUDO FOR PRTSC
      42 4E 4D 3C 3E 3F
      FF 00
0952  FF 20 FF           DB      -1,SPACE,-1
                 ;-----------------------------------------------------------
                 ;----- SHIFT + (F1 - F10)  -  EXTENDED ASCII CODES         -
                 ;-----------------------------------------------------------
0955                   K12LABEL   BYTE
0955  54 55 56 57 58     DB      84,85,86,87,88
095A  59 5A 5B 5C 5D     DB      89,90,91,92,93
                 ;-----------------------------------------------------------
                 ;----- ALT + (F1 - F10)  -  EXTENDED ASCII CODES           -
                 ;-----------------------------------------------------------
095F                   K13LABEL   BYTE
095F  68 69 6A 6B 6C     DB      104,105,106,107,108
0964  6D 6E 6F 70 71     DB      109,110,111,112,113
                 ;-----------------------------------------------------------
                 ;----- KEYPAD KEYS - ASCII CODES                           -
                 ;-----------------------------------------------------------
0969                   K14LABEL   BYTE                ; IN KEYPAD_STATE
0969  37 38 39 2D 34 35  DB      '789-456+1230.'
      36 2B 31 32 33 30
      2E
                 ;-----------------------------------------------------------
                 ;----- BASE CASE KEYPAD KEYS - EXTENDED ASCII CODES        -
                 ;-----------------------------------------------------------
0976                   K15LABEL   BYTE
0976  47 48 49 FF        DB      HOME_KEY,CUR_UP,PGUP,-1
097A  4B FF 4D FF        DB      CUR_LFT,-1,CUR_RHT,-1
097E  4F 50 51           DB      END_KEY,CUR_DN,PGDN
0981  52 53              DB      INS_KEY,DEL_KEY
                 ;
                 ;
                 ;------ KEYBOARD INTERRUPT 9 ENTRY POINT
                 ;
                 ;ORG     0E987H
0987                     ORG     00987H
= 0987                 KB_INT     EQU     $
0987  E9 0000 E          JMP     KYBD_INT9
```

System Setup (SYS_SETUP)

```
                        SUBTTL  SYS_SETUP
                ;**********************************************************************
                ;
                ; ROUTINE-NAME :  SYS_SETUP
                ;
                ; FUNCTION: THIS ROUTINE INITIALIZES INTRPT VECTORS AND SETS UP DATA
                ;           AREAS FOR DEVICES PRIOR TO BOOTING FROM DISKETTE.
                ;
                ; ENTRY CONDITIONS:
                ;     PURPOSE OF ENTRY: PERFORM SYSTEM SETUP PRIOR TO BOOT
                ;     INPUT CONDITIONS: NONE
                ;     RESTRICTIONS:     NONE
                ;
                ; EXIT CONDITIONS:
                ;     NORMAL EXIT CONDITIONS: RETURN
                ;
                ;  ERROR EXIT CONDITIONS:  NONE
                ;
                ;  REGISTERS MODIFIED:    AX,BX,CX,DX
                ;
                ;**********************************************************************

                        ASSUME  DS:DATA,ES:ABSO

098A                 SYS_SETUP       PROC    NEAR
                ;
                ; ---- CLEAR THE RESET FLAG IF NOT LOOP MODE
                ;
098A  81 3E 0072 R ABCD    CMP     RESET_FLAG,LOOP_MODE
0990  74 06                JE      SYS_SET0
0992  C7 06 0072 R 0000    MOV     RESET_FLAG,0
                ;
                ;----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
                ;
0998                 SYS_SET0:
0998  E8 0A0C R            CALL    VECTOR_SETUP        ; SETUP INTERRUPT VECTORS
                ;
                ; COMPLETE THE VECTOR SETUP
                ;
099B  2B C0                SUB     AX,AX
099D  8E C0                MOV     ES,AX
099F  26: C7 06 01B0 R 1F53 R     MOV     WORD PTR[RESUME_PTR],OFFSET DUMMY_RETURN
09A6  26: 8C 0E 01B2 R     MOV     WORD PTR[RESUME_PTR+2],CS ; SET SEGMENT
09AB  26: C7 06 0128 R 1F53 R     MOV     WORD PTR[RTCA_PTR],OFFSET DUMMY_RETURN
09B2  26: 8C 0E 012A R     MOV     WORD PTR[RTCA_PTR+2],CS ; SET SEGMENT
                ;
                ; SET UP KEYBOARD DATA AREA
                ;
09B7  E8 0831 R            CALL    KYBD_RESET          ; RESET KEYBOARD
09BA  80 0E 00B4 R 08      OR      KB_NMI_CNTL,CLICK_ON ; ACTIVATE KEYBOARD CLICKER


                ;
                ; DISABLE RTC PERIODIC,ALARM AND UPDATE INTERRUPTS AND RESET SET
                ;
09BF  B4 0B                MOV     AH,RTC_MODE         ; GET RTC MODE CONTROL
09C1  E8 07AA R            CALL    GET_RTC_REG
09C4  24 0F                AND     AL,NOT PIE_ENABLE+SET_CLOCK+AIE_ENABLE+UIE_ENABLE
09C6  E8 07C8 R            CALL    PUT_RTC_REG         ; UPDATE RTC MODE CONTROL
09C9  E8 07E6 R            CALL    BAT_SAV_SETUP       ; SETUP FOR BATTERY SAVINGS
                ;
                ; TURN OFF MODEM IF ON BATTERY POWER AND PROFILE SO INDICATES
                ;
09CC  E4 7F                IN      AL,PWR_STAT         ; CHECK FOR BATTERY OPERATION
09CE  A8 40                TEST    AL,EXT_PWR
09D0  75 13                JNZ     SYS_SET2

09D2  B4 17                MOV     AH,RTC_SYS_PROF1    ; GET SYSTEM PROFILE
09D4  E8 07AA R            CALL    GET_RTC_REG
```

```
09D7  A8 02                    TEST   AL,MODEM_BATT        ; OPERATE MODEM ON BATTERY?
09D9  75 0A                    JNZ    SYS_SET2             ; YES THEN JUMP
                         ;
                         ; ON BATTERY POWER AND MODEM PROFILE INDICATES NO BATTERY OPERATION
                         ;
09DB  B3 02                    MOV    BL,ACT_MODEM         ; SPECIFY MODEM OFF
09DD  2A FF                    SUB    BH,BH                ; INDICATE POWER OFF REQUEST
09DF  E8 0000 E                CALL   COM_POWER            ; TURN OFF PRIMARY COM POWER
09E2  EB 09 90                 JMP    SYS_SET3
                         ;
                         ; MODEM POWER LEFT ON SO SET MODEM ACCORDING TO CONFIGURATION
                         ;
09E5                        SYS_SET2:
09E5  B4 1D                    MOV    AH,RTC_MOD_PROF1     ; RETRIEVE MODEM PROFILE
09E7  E8 07AA R                CALL   GET_RTC_REG          ; GET PROFILE IN AL REGISTER
09EA  E8 0000 E                CALL   MODEM_CONFIG         ; GO SETUP MODEM
                         ;
                         ; ENABLE DISKETTE NMIS
                         ;
09ED                        SYS_SET3:
09ED  E4 77                    IN     AL,DSKT_CNTL         ; READ FROM DISKETTE PORT
09EF  0C 80                    OR     AL,DSKT_NMI          ; SET ON DISKETTE NMI ENABLE
09F1  E6 77                    OUT    DSKT_CNTL,AL         ; OUT IT
                         ;
                         ; ENABLE KEYBOARD NMIS
                         ;
09F3  E4 7C                    IN     AL,KYBD_CNTL
09F5  0C 80                    OR     AL,EN_KYBD_NMI       ; ENABLE KEYBOARD NMI'S
09F7  E6 7C                    OUT    KYBD_CNTL,AL
                         ;
                         ; ENABLE SPEAKER, RTC ALARM NMI, AND I/O CHECK NMI
                         ;
09F9  E4 61                    IN     AL,NMI_CNTL          ; ENABLE RTC INTERRUPT
09FB  24 D7                    AND    AL,NOT DIS_ALARM+DIS_IOCHK
09FD  0C 04                    OR     AL,EN_SPKR           ; ENABLE SPEAKER
09FF  E6 61                    OUT    NMI_CNTL,AL
                         ;
                         ; ENABLE I/O CHANNEL CHECK
                         ;
0A01  B0 80                    MOV    AL,EN_IOCHK
0A03  E6 A0                    OUT    IONMI_CNTL,AL
                         ;
                         ; ENABLE KEYBOARD, TIMER 0 AND DISKETTE INTERRUPTS
                         ;
0A05  E4 21                    IN     AL,INTA01
0A07  24 BC                    AND    AL,0BCH              ; ENABLE KEYBOARD AND TIMER
0A09  E6 21                    OUT    INTA01,AL            ; AND DISKETTE INTERRUPTS
0A0B  C3                       RET
0A0C                        SYS_SETUP ENDP
```

Vector Setup (VECTOR_SETUP)

```
                               SUBTTL  VECTOR_SETUP
                         ;*********************************************************************
                         ;
                         ; ROUTINE-NAME :  VECTOR_SETUP
                         ;
                         ; FUNCTION: THIS ROUTINE INITIALIZES INTERRUPT VECTORS 0-32 AND THE
                         ;      PRINTER AND RS232 TIMEOUT VALUES. VIDEO VECTORS 10H,1DH,1FH,44H
                         ;      ARE NOT INITIALIZED BY THIS ROUTINE. THEY ARE INITIALIZED IN
                         ;      THE LCD_CONFIG ROUTINE DURING POST.
                         ;
                         ; ENTRY CONDITIONS:
                         ;      INPUT CONDITIONS: DS:DATA
                         ;      RESTRICTIONS:     NMI'S MUST BE DISABLED
                         ;
                         ; EXIT CONDITIONS:
                         ;      NORMAL EXIT CONDITIONS: INTERRUPT VECTORS AND TIMEOUTS SET
```

```
                 ;
                 ;      ERROR EXIT CONDITIONS:  NONE
                 ;
                 ;      REGISTERS MODIFIED:    ES,SI,DI,AX,CX
                 ;
                 ;*********************************************************************
0A0C                    VECTOR_SETUP PROC NEAR

                           ASSUME  DS:DATA,ES:ABS0

0A0C  1E                   PUSH    DS                   ; SAVE DS
0A0D  B9 001F              MOV     CX,31                ; FILL ALL 31 INTERRUPTS
0A10  2B FF                SUB     DI,DI                ; FIRST INTERRUPT LOCATION
0A12  8E C7                MOV     ES,DI                ; SET ES=ABS0
                 ;
                 ; INITIALIZE FIRST 31 VECTORS TO D11 (TEMPORARY INTERRUPT HANDLER)
                 ; VECTORS 0-1EH SET TO D11

0A14                    SETUP_1:
0A14  B8 1F23 R            MOV     AX,OFFSET D11        ; MOVE ADDR OF INTR PROC TO TBL

0A17  83 FF 40             CMP     DI,10H*4             ; VIDEO INT 10H?
0A1A  74 05                JE      SETUP_1A             ; YES THEN SKIP
0A1C  83 FF 74             CMP     DI,1DH*4             ; VIDEO INT 1DH?
0A1F  75 05                JNE     SETUP_1B             ; YES THEN SKIP

0A21                    SETUP_1A:
0A21  83 C7 04             ADD     DI,4                 ; SKIP VECTORS
0A24  E2 EE                LOOP    SETUP_1

0A26                    SETUP_1B:
0A26  AB                   STOSW                        ; SET VECTORS
0A27  8C C8                MOV     AX,CS                ; GET ADDR OF INTR PROC SEG
0A29  AB                   STOSW
0A2A  E2 E8                LOOP    SETUP_1
                 ;
                 ;-INITIALIZE BIOS VECTORS 8-1EH FROM VECTOR_TABLE
                 ; SKIPPING VIDEO VECTORS 10H AND 1DH
                 ;

0A2C  B9 0017              MOV     CX,23                ; GET VECTOR COUNT
0A2F  0E                   PUSH    CS                   ; SETUP DS SEG REG
0A30  1F                   POP     DS
0A31  BE 1EF3 R            MOV     SI,OFFSET VECTOR_TABLE
0A34  BF 0020 R            MOV     DI,OFFSET INT_PTR
0A37                    SETUP_2:
0A37  83 FF 40             CMP     DI,10H*4             ; VIDEO INT 10H?
0A3A  74 05                JE      SETUP_2A             ; YES THEN SKIP
0A3C  83 FF 74             CMP     DI,1DH*4             ; VIDEO INT 1DH?
0A3F  75 08                JNE     SETUP_2B             ; YES THEN SKIP

0A41                    SETUP_2A:
0A41  83 C7 04             ADD     DI,4                 ; SKIP VIDEO VECTORS
0A44  83 C6 02             ADD     SI,2
0A47  E2 EE                LOOP    SETUP_2

0A49                    SETUP_2B:
0A49  A5                   MOVSW                        ; SET VECTORS
0A4A  83 C7 02             ADD     DI,2                 ; SKIP OVER SEGMENT
0A4D  E2 E8                LOOP    SETUP_2
0A4F  1F                   POP     DS                   ; RESTORE DS
                 ;
                 ; SET UP NMI,PRINT SCREEN,BASIC SEGMENT,  RESUME VECTOR, AND
                 ; USER REAL TIME CLOCK ALARM VECTORS
                 ;
0A50  26: C7 06 0008 R     MOV     NMI_PTR,OFFSET NMI_FLIH ; NMI INTERRUPT HANDLER
          0000 E
0A57  26: C7 06 0014 R     MOV     INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
          1F54 R
0A5E  26: C7 06 0062 R     MOV     BASIC_PTR+2,0F600H  ; SEGMENT FOR RESIDENT BASIC
          F600

                           ASSUME  ES:DATA,DS:DATA
```

```
0A65  1E                   PUSH    DS
0A66  07                   POP     ES
               ;
               ; SET INITIAL RS232 AND PRINTER TIMEOUT VALUES
               ;
0A67  A2 006B R            MOV     INTR_FLAG,AL        ; CLEAR STRAY INTERRUPT FLAG
0A6A  BF 0078 R            MOV     DI,OFFSET PRINT_TIM_OUT ; SET DEFAULT PTR TIMEOUT
0A6D  B8 1414              MOV     AX,1414H            ; DEFAULT = 20
0A70  83 3E 0008 R 78      CMP     PRINTER_BASE,078H   ; PORTABLE PRINTER INSTALLED?
0A75  75 02                JNE     SETUP_3             ; JUMP IF NOT
0A77  B0 23                MOV     AL,23H              ; SET PORTABLE TO 25 SECS

0A79                    SETUP_3:
0A79  AB                   STOSW                       ; THREE TIMEOUT VALUES
0A7A  8A C4                MOV     AL,AH               ; USE LPT2 TIMEOUT VALUE
0A7C  AA                   STOSB
0A7D  B8 0101              MOV     AX,0101H            ; RS232 TIMEOUTS = 01
0A80  AB                   STOSW                       ; 4 TIMEOUTS
0A81  AB                   STOSW
0A82  C3                   RET                         ; RETURN TO CALLER
0A83                    VECTOR_SETUP ENDP


               ;********************************************
               ; KEYBOARD TABLES CONTINUED
               ;********************************************
               ;ORG     0EA87H
0A87                     ORG     00A87H
               ;---------------------------------------------------------------
               ;----- ALT + (KEYPAD 0 - 9) - EXTENDED ASCII CODES             -
               ;---------------------------------------------------------------
0A87                    K30     LABEL   BYTE
0A87  52 4F 50 51 4B       DB      82,79,80,81,75      ; 10 NUMBERS ON KEYPAD
0A8C  4C 4D 47 48 49       DB      76,77,71,72,73
               ;---------------------------------------------------------------
               ;----- ALT + (A - Z)  -  EXTENDED ASCII CODES                  -
               ;-  -- SUPER SHIFT --                                           -
               ;---------------------------------------------------------------
0A91  10 11 12 13 14 15    DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
      16 17
0A99  18 19 1E 1F 20 21    DB      24,25,30,31,32,33,34,35
      22 23
0AA1  24 25 26 2C 2D 2E    DB      36,37,38,44,45,46,47,48
      2F 30
0AA9  31 32                DB      49,50

               ;*****************************************************************
               ;  KEY TRANSLATION TABLE - INTERNAL SCAN CODES TO PC1 SCAN CODES WHILE
               ;  THE KEYPAD STATE IS ACTIVE (KEY PAD SCAN CODES).
               ;*****************************************************************
               ;
               ; TABLE CONTAINS INTERNAL AND PC1 SCAN CODE PAIRS. HIGH BYTE IS INTNL
               ; SCAN CODE AND THE LOW BYTE IS THE PC1 EQUIVALENT.
               ;
0AAB                    KBPAD_TBL  LABEL WORD
0AAB  1847 1948 1A49 1C4A    DW   1847H,1948H,1A49H,1C4AH,1D4EH,284BH,294CH,2A4DH
      1D4E 284B 294C 2A4D
0ABB  384F 3950 3A51 4852    DW   384FH,3950H,3A51H,4852H,4A53H,4B35H,4E37H
      4A53 4B35 4E37
= 001E                  KBPADL  EQU  $-KBPAD_TBL

               ;*****************************************************************
               ;  KEY TRANSLATION TABLE - INTERNAL SCAN CODES TO PC1 SCAN CODES WHILE
               ;  THE FUNCTION KEY IS HELD.
               ;*****************************************************************
               ;
               ; FIRST 6 ENTRIES OF THIS TABLE CONTAIN INTNL AND PC1 SCAN CODE PAIRS.
               ; HIGH BYTE IS THE INTERNAL SCAN CODE AND THE LOW BYTE IS THE PC1
               ; EQUIVALENT. ENTRIES 5-8 CONTAIN INTERNAL SCAN CODES AND THEIR STATE
               ; BIT IN KB_NMI_CNTL. HIGH BYTE IS THE INTERNAL SCAN CODE AND LOW BYTE
               ; CONTAINS STATE BIT FOR KEYPAD STATE, CLICKER STATE, AND AUDIO STATE
               ; change made for speaker from Fn+A to Fn+Scroll_Lock
```

```
0AC9                    KBFUN_TBL  LABEL WORD
0AC9  5B47 5C49 5F4F 5E51    DW   5B47H,5C49H,5F4FH,5E51H,0257H,0358H
      0257 0358
0AD5  0108 0C04 3102 0D01    DW   0108H,0C04H,3102H,0D01H
= 0014                  KBFUNL  EQU  $-KBFUN_TBL

              ;*****************************************************************
              ; KEY TRANSLATION TABLE
              ;  TABLE DISPLACEMENT 52H IS THE FUNCTION KEY AND IS SET TO 52H.
              ;  THIS WILL ALLOW TESTING OF THE FUNCTION KEY EVEN THOUGH IT DOES
              ;  NOT HAVE A PC1 SCAN CODE EQUIVALENT.
              ; change made for bksp and \ change
              ;
              ; EXPLANATION
              ;
              ; Each entry is the P60 make scan code for the NMI scan code
              ; corresponding to this position.
              ;
              ; P60 sc = 0FFH   -   means no such NMI scan code
              ; P60 sc > 080H   -   means the key may be a FN+key or Keypad key.
              ; P60 sc < 080H   -   means the key is a base key
              ;
              ;*****************************************************************

              ;------ TABLE OF INTERNAL SCAN CODES TO PC1 SCAN CODES

0ADD                    KBNMI_TBL  LABEL BYTE
0ADD  FF 81 BB BC 3D 3E    DB   0FFH,081H,0BBH,0BCH,03DH,03EH,03FH,040H
      3F 40
0AE5  41 42 43 44 C5 C6    DB   041H,042H,043H,044H,0C5H,0C6H,052H,053H
      52 53
0AED  FF 29 02 03 04 05    DB   0FFH,029H,002H,003H,004H,005H,006H,007H
      06 07
0AF5  88 89 8A 0B 8C 8D    DB   088H,089H,08AH,00BH,08CH,08DH,02BH,00EH
      2B 0E
0AFD  FF 0F 10 11 12 13    DB   0FFH,00FH,010H,011H,012H,013H,014H,015H
      14 15
0B05  96 97 98 19 1A 1B    DB   096H,097H,098H,019H,01AH,01BH,0FFH,0FFH
      FF FF
0B0D  FF BA 1E 1F 20 21    DB   0FFH,0BAH,01EH,01FH,020H,021H,022H,023H
      22 23
0B15  A4 A5 A6 27 28 2B    DB   0A4H,0A5H,0A6H,027H,028H,02BH,01CH,0FFH
      1C FF
0B1D  FF 2A 2C 2D 2E 2F    DB   0FFH,02AH,02CH,02DH,02EH,02FH,030H,031H
      30 31
0B25  B2 33 B4 B5 36 FF    DB   0B2H,033H,0B4H,0B5H,036H,0FFH,0B7H,0FFH
      B7 FF
0B2D  FF 1D D2 38 56 FF    DB   0FFH,01DH,0D2H,038H,056H,0FFH,039H,0FFH
      39 FF
0B35  FF FF 38 CB C8 FF    DB   0FFH,0FFH,038H,0CBH,0C8H,0FFH,0D0H,0CDH
      D0 CD
```

Error Message Routine (SYS_CHK)

```
              ;*****************************************************************
              ;
              ; ROUTINE-NAME :  SYS_CHK
              ;
              ; FUNCTION: THIS ROUTINE DSPLYS ERROR MESSAGES ACCORDING TO FLAGS SET
              ;      IN THE RTC_DIAG_STAT SAVE AREA. ONCE THE FLAG HAS BEEN FOUND
              ;      IT IS CLEARED IF AN ASSOCIATED ERROR CODE IS PRESENT.
              ; ENTRY CONDITIONS:
              ;      PURPOSE OF ENTRY: DISPLAY ERROR MESSAGES
              ;      INPUT CONDITIONS: DS:DATA
              ;      RESTRICTIONS:     NONE
              ;
              ; EXIT CONDITIONS:
              ;      NORMAL EXIT CONDITIONS: RETURN
              ;
              ;  ERROR EXIT CONDITIONS:  NON FATAL ERROR RETURN CODE WITH ERROR CODE
              ;                IF ASSOCIATED BIT SET. RE_DISPATCH FLAG SET
              ;                IF ALL BITS IN RTC_DIAG_STAT NOT YET CHECKED.
              ;
              ;  REGISTERS MODIFIED:    AX,BX,CX,DX
              ;
              ;*****************************************************************
              ;
0B3D                   ERROR_TABLE     LABEL WORD
0B3D  40 73               DB      BAD_RTC_SIG,73H     ; RTC SIGNATURE BAD
0B3F  20 71               DB      BAD_STOR_CKSUM,71H  ; STG CHECKSUM BAD ON RESUME
0B41  10 75               DB      LCD_ALT_FAILED,75H  ; ALTERNATE LCD MODE FAILED

0B43                   SYS_CHK PROC    NEAR
0B43  B4 0E               MOV     AH,RTC_DIAG_STAT
0B45  E8 07AA R           CALL    GET_RTC_REG
0B48  B9 0003             MOV     CX,3                ; THREE ENTRIES IN TABLE
0B4B  BE 0B3D R           MOV     SI,OFFSET ERROR_TABLE
0B4E                   SYS_LP:
0B4E  2E: 84 04           TEST    AL,BYTE PTR CS:[SI] ; CHECK FOR BIT ON
0B51  75 0A               JNZ     SYS_ERR
0B53  83 C6 02            ADD     SI,2
0B56  E2 F6               LOOP    SYS_LP
0B58  2B C0               SUB     AX,AX               ; CLEAR ERROR CODE
0B5A  EB 0D 90            JMP     SYS_EXIT
              ;
              ; ERROR MATCH FOUND SO CHECK FOR ERROR CODE
              ;
0B5D                   SYS_ERR:
0B5D  2E: 8B 1C           MOV     BX,WORD PTR CS:[SI] ; GET CODE FROM TABLE
0B60  32 C3               XOR     AL,BL               ; RESET ERROR BIT
0B62  E8 07C8 R           CALL    PUT_RTC_REG         ; SAVE IN RTC_DIAG STAT
0B65  B4 50               MOV     AH,NON_FATAL_ERR+RE_DISPATCH ; SET ERR AND REDISP
0B67  8A C7               MOV     AL,BH
; SET ERROR CODE FROM TABLE

0B69                   SYS_EXIT:
0B69  C3                  RET
0B6A                   SYS_CHK ENDP
```

Resume Error Check (RES_ERR_CHK)

```
                ;*******************************************************************
                ;
                ; ROUTINE-NAME :  RES_ERR_CHK
                ;
                ; FUNCTION: THIS ROUTINE CHKS AND DSPLYS ANY ERROR MESGS ASSOCIATED
                ;      WITH OPERATOR CAUSED RESUME ERRORS. THIS IS DONE AT BOOT TIME
                ;      AFTER AN ABORTED RESUME.
                ; ENTRY CONDITIONS:
                ;      PURPOSE OF ENTRY: DISPLAY ERROR MESSAGES
                ;      INPUT CONDITIONS: DS:DATA
                ;      RESTRICTIONS:     NONE
                ;
                ; EXIT CONDITIONS:
                ;      NORMAL EXIT CONDITIONS: RETURN  WITH ZERO FLAG SET
                ;
                ;  ERROR EXIT CONDITIONS:  ERROR NO. IS DISPLAYED ON ROW XX COLUMN XX
                ;              OF THE SCREEN AND THAT ERROR FLAG IS RESET
                ;              RETURN IS MADE WITH ZERO FLAG RESET
                ;
                ;  REGISTERS MODIFIED:    AX,BX,CX,DX,ES,SI,DI,BP
                ;*******************************************************************
0B6A                    ERR2_TABLE LABEL WORD
0B6A  02 70                DB      LCD_NOT_ACTIVE,70H
0B6C  01 72                DB      DSKT_ACTIVE,72H
0B6E  08 74                DB      LCD_CHANGE,74H

0B70                    RES_ERR_CHK PROC NEAR
0B70  BA 0632              MOV     DX,0632H             ; SET ROW =  6 COLUMN=50 FOR MSG
0B73                    RES_LP0:
0B73  B4 0E                MOV     AH,RTC_DIAG_STAT     ; GET THE ERROR FLAGS FROM RTC
0B75  E8 07AA R            CALL    GET_RTC_REG
0B78  B9 0003              MOV     CX,3                 ; THREE ENTRIES IN TABLE
0B7B  BE 0B6A R            MOV     SI,OFFSET ERR2_TABLE
0B7E                    RES_LP:
0B7E  2E: 84 04            TEST    AL,BYTE PTR CS:[SI]  ; CHECK FOR BIT ON
0B81  75 0B                JNZ     RES_ERR
0B83  83 C6 02             ADD     SI,2
0B86  E2 F6                LOOP    RES_LP

0B88  80 FE 06             CMP     DH,06                ; SAME ROW AS STARTED ON?
0B8B  EB 2C 90             JMP     RES_EXIT             ; IF SO NO ERRORS A Z FLAG SET
                                                        ; ELSE ERROR & Z FLAG CLEARED
                ;
                ; ERROR MATCH FOUND SO CHECK FOR ERROR CODE
                ;
0B8E                    RES_ERR:
0B8E  2E: 8B 1C            MOV     BX,WORD PTR CS:[SI]  ; GET CODE FROM TABLE
0B91  32 C3                XOR     AL,BL                ; RESET ERROR BIT
0B93  E8 07C8 R            CALL    PUT_RTC_REG          ; SAVE IN RTC_DIAG STAT

                           ASSUME  ES:XXDATA

0B96  B8 ---- R            MOV     AX,XXDATA
0B99  8E C0                MOV     ES,AX
0B9B  52                   PUSH    DX                   ; SAVE COORDINATES
0B9C  B6 01                MOV     DH,01                ; SET HIGH BYTE OF ERROR CODE
0B9E  8A D7                MOV     DL,BH                ; SET ERROR CODE FROM TABLE
0BA0  BF 0009 R            MOV     DI,OFFSET TEMP       ; POINT TO MESSAGE SAVE AREA
0BA3  E8 0275 R            CALL    STR_CON              ; CONVERT CODE TO ASCII
0BA6  5A                   POP     DX                   ; RESTORE COORDINATES
0BA7  BB 0007              MOV     BX,0007H             ; SET ATTRIBUTE AND PAGE
0BAA  B9 0004              MOV     CX,4                 ; SET STRING LENGTH
0BAD  BD 000A R            MOV     BP,OFFSET TEMP+1     ; GET ADDRESS OF STRING
0BB0  B8 1300              MOV     AX,1300H             ; SET WRITE STRING REQUEST
0BB3  CD 10                INT     10H                  ; DISPLAY ERROR CODE
0BB5  FE C6                INC     DH                   ; POINT TO NEXT ROW
```

```
0BB7  EB BA              JMP     SHORT RES_LPO        ; GO LOOK FOR MORE ERRORS
0BB9                  RES_EXIT:
0BB9  C3                 RET
0BBA                  RES_ERR_CHK ENDP
```

Diskette I/O Entry (DISKETTE_IO)

```
               ;*****************************************************
               ; DISKETTE I/O ENTRY POINT
               ;*****************************************************
               ;ORG     0EC59H
0C59                   ORG     00C59H
= 0C59                DISKETTE_IO     EQU     $
0C59  E9 0000 E        JMP     DSKT_IO
```

Icon and Error Message Routine (E_MSG)

```
               ;*******************************************************************
               ;
               ; ROUTINE-NAME :  E_MSG
               ;
               ; FUNCTION: THIS ROUTINE DISPLAYS ICONS AND ERROR MESSAGES FOR POST
               ;
               ;
               ; ENTRY CONDITIONS:
               ;      PURPOSE OF ENTRY: TO DISPLAY A POST ICON AND/OR ERROR MESSAGE
               ;      INPUT CONDITIONS: AX = ERROR CODE    ; IF AH = 0 THEN NO ERROR
               ;                                             CODE , ONLY AN ICON
               ;                   ES:DI = POINTS TO AN ADDITIONAL ASCII ERROR
               ;                                 MESSAGE. IF DI = 0 THEN NO MESSAGE
               ;                  CL = ICON NUMBER TO DISPLAY
               ;
               ;  RESTRICTIONS: NONE
               ;
               ; EXIT CONDITIONS:
               ;      NORMAL EXIT CONDITIONS:
               ;
               ;      ERROR EXIT CONDITIONS:
               ;
               ;
               ;
               ;      REGISTERS MODIFIED: AX,BX,CX,DX,BP,DI,SI
               ;
               ; INTERNALLY REFERENCED ROUTINES:  NONE
               ;
               ; EXTERNALLY REFERENCED ROUTINES:  STR_CON
               ;
               ;*******************************************************************

                          ASSUME CS:ROMCODE
                          ASSUME DS:XXDATA
                          ASSUME ES:NOTHING

0C5C                  E_MSG   PROC    NEAR

0C5C  1E                  PUSH    DS                    ; SAVE SEGMENT REGS
0C5D  06                  PUSH    ES
```

```
0C5E  57                    PUSH    DI                  ; SAVE PTR TO ASCII ERROR MSG
0C5F  06                    PUSH    ES
0C60  50                    PUSH    AX                  ; SAVE ERROR CODE

0C61  0E                    PUSH    CS
0C62  07                    POP     ES                  ; POINT ES TO CODE SEGMENT
0C63  B8 ---- R             MOV     AX,XXDATA           ; SET DS TO XXDATA SEGMENT
0C66  8E D8                 MOV     DS,AX
0C68  B5 00                 MOV     CH,0                ; CLEAR CH
                ;
                ; CHECK FOR NEED TO DISPLAY ICON AND DO SO IF NECESSARY
                ;
0C6A  8B F9             EM10:   MOV     DI,CX           ; SAVE ICON NUMBER
0C6C  D1 E7                 SHL     DI,1                ; MULTIPLY BY 2
0C6E  2E: 8B AD 0249 R      MOV     BP,ICON_ADR[DI]     ; BP POINTS TO ICON DATA
0C73  8B F9                 MOV     DI,CX
0C75  83 FF 06              CMP     DI,6                ; F1 ICON
0C78  75 52                 JNE     EM15
                ;
                ; DISPLAY F1 ICON AND WAIT FOR F1 TO BE PRESSED IF SYS UNIT DISPLAYED
                ; OTHERWISE WAIT 10 SECONDS AND THEN RETURN
                ;
0C7A  80 3E 002B R 00       CMP     ICON_DIS+1,0        ; SYSTEM UNIT DISPLAYED?
0C7F  74 29                 JE      EM13                ; JUMP IF NO SYSTEM UNIT

0C81  BA 1200               MOV     DX,01200H           ; DX POINTS TO DSPLY POSITION
0C84  E8 0D5F R             CALL    ICON_PR             ; DISPLAY F1 ICON
                ;
                ; KEYSTROKE FOUND SO CHECK FOR CORRECT KEYS
                ;
0C87                    EM11:
0C87  B4 00                 MOV     AH,0                ; SET GET KEYSTROKE FN CODE
0C89  CD 16                 INT     16H                 ; GET A KEYSTROKE
0C8B  80 FC 3B              CMP     AH,03BH             ; F1 KEY?
0C8E  74 11                 JE      EM12                ; YES THEN WAIT FOR REL OF KEY
0C90  3C 0C                 CMP     AL,00CH             ; CTL+L FOR LOOP MODE?
0C92  75 F3                 JNE     EM11                ; NO LOOK FOR NEXT KEY SEQ

                            ASSUME  DS:DATA

0C94  B8 ---- R             MOV     AX,DATA             ; SETUP DATA SEGMENT
0C97  8E D8                 MOV     DS,AX
0C99  C7 06 0072 R ABCD     MOV     RESET_FLAG,LOOP_MODE ; SET POST LOOP MODE
0C9F  EB 25                 JMP     SHORT EM14          ; EXIT
                ;
                ; WAIT FOR BREAK OF F1 KEY
                ;
0CA1  FB                EM12:   STI                     ; ENABLE INTERRUPTS
0CA2  E4 60                 IN      AL,KB_DATA          ; READ KEYBOARD PORT
0CA4  3C 3B                 CMP     AL,03BH             ; CHECK FOR STILL F1 KEY
0CA6  74 F9                 JE      EM12                ; WAIT UNTIL NOT F1 KEY MAKE
0CA8  EB 1C                 JMP     SHORT EM14          ; EXIT IF F1 KEY RELEASED
                ;
                ; NO SYSTEM UNIT SO ONLY WAIT 10 SECONDS OR UNTIL KEY HIT
                ;
0CAA                    EM13:
0CAA  BA 0060               MOV     DX,KB_DATA          ; KEYBOARD PORT
0CAD  EC                    IN      AL,DX
0CAE  8A F8                 MOV     BH,AL               ; GET PRESENT PORT 60 VALUE
0CB0  B3 B5                 MOV     BL,0B5H             ; SET TIMEOUT TO 10 SECONDS
0CB2  B0 12                 MOV     AL,12H              ; COMPARE RETURN NOT EQUAL
                                                        ; USING BH AND PORT READ (DX)
0CB4  B4 41                 MOV     AH,41H              ; WAIT ON EXTERNAL EVENT
0CB6  CD 15                 INT     15H                 ; WILL RETURN AFTER 10 SECS
0CB8  72 0C                 JC      EM14                ; IF TIMEOUT THEN JUMP
0CBA                    EM13_1:
0CBA  B4 01                 MOV     AH,1                ; CHECK FOR KEY IN BUFFER
0CBC  CD 16                 INT     16H
0CBE  74 06                 JZ      EM14                ; JUMP IF NO KEY IN BUFFER
0CC0  B4 00                 MOV     AH,0
0CC2  CD 16                 INT     16H                 ; OTHERWISE PURGE THE KEY
0CC4  EB F4                 JMP     SHORT EM13_1        ; LOOP UNTIL NO MORE KEYS
```

```
                  ;
                  ; F1 WAIT COMPLETED SO EXIT
                  ;
0CC6                         EM14:
0CC6  83 C4 06                  ADD     SP,6                 ; BALANCE STACK
0CC9  E9 0D5C R                 JMP     EMSG_RET             ; EXIT WHEN PRESSED
                  ;
                  ; NOT REQUEST TO DISPLAY F1 PROMPT
                  ;
                                ASSUME  DS:XXDATA

0CCC  8A 8D 002A R           EM15:   MOV     CL,ICON_DIS[DI- ; MOVE DSPLY SQUARE NO. TO SI
0CD0  80 F9 00                  CMP     CL,0                 ; HAS ICON BEEN DISPLAYED?
0CD3  8B F1                     MOV     SI,CX                ; SAVE DISPLAY SQUARE NO.
0CD5  75 1A                     JNE     EM20
0CD7  8A 0E 002A R              MOV     CL,ICON_DIS[0]       ; GET NO. ICONS ALREADY DSPLYD
0CDB  FE C1                     INC     CL
0CDD  88 8D 002A R              MOV     ICON_DIS[DI],CL      ; STORE DSPLY SQUARE WITH ICON
0CE1  88 0E 002A R              MOV     ICON_DIS[0],CL       ; SAVE NO. OF ICONS DISPLAYED
0CE5  8B F1                     MOV     SI,CX                ; SAVE DISPLAY SQUARE
0CE7  2E: 8A 94 0078 R          MOV     DL,DIS_POS[SI]       ; GET COL NUMBER FOR DISPLAY
0CEC  B6 00                     MOV     DH,0                 ; PUT ROW NUMBER IN DH
0CEE  E8 0D5F R                 CALL    ICON_PR              ; DISPLAY ICON
                  ;
                  ; CONVERT AND DISPLAY ERROR MESSAGE IN AX IF REQUIRED
                  ;
0CF1  58                     EM20:   POP     AX              ; RESTORE ERROR CODE
0CF2  0A E4                     OR      AH,AH                ; ERROR MESSAGE?
0CF4  74 3C                     JZ      EM30

0CF6  57                        PUSH    DI                   ; SAVE ICON NUMBER
0CF7  8B D0                     MOV     DX,AX                ; MOVE ERROR CODE
0CF9  B8 ---- R                 MOV     AX,XXDATA
0CFC  8E C0                     MOV     ES,AX                ; ES POINTS TO XXDATA
0CFE  BF 0009 R                 MOV     DI,OFFSET TEMP       ; POINTER FOR CONVERSION RTNE
0D01  E8 0275 R                 CALL    STR_CON              ; CONVERT ERROR CODE TO ASCII
0D04  5F                        POP     DI                   ; RESTORE ICON NUMBER
0D05  8A 9D 0031 R              MOV     BL,ICON_MSG[DI]      ; GET NO. OF MSGD UNDER ICON
0D09  FE 85 0031 R              INC     ICON_MSG[DI]         ; INCREMENT NO. OF MESSAGES
0D0D  2E: 8A 94 0078 R          MOV     DL,DIS_POS[SI]       ; GET SCREEN POSITION
0D12  80 C2 08                  ADD     DL,8                 ; CENTER MESSAGE
0D15  83 FF 02                  CMP     DI,2                 ; FEATURE ICON?
0D18  75 03                     JNE     EM25
0D1A  80 EA 04                  SUB     DL,4                 ; ADJ DSPLY POS FOR FEAT ICON

0D1D  B6 09                  EM25:   MOV     DH,9            ; FIRST MESSAGE GOES AT ROW 9
0D1F  02 F3                     ADD     DH,BL                ; ERROR MSG. IS 1 ROW BELOW
                                                             ; PREVIOUS ONE
0D21  B9 0004                   MOV     CX,4                 ; LENGTH OF STRING IS 4 CHARS
0D24  BD 000A R                 MOV     BP,OFFSET TEMP+1     ; POINT TO ASCII ERROR CODE
0D27  B4 0F                     MOV     AH,15
0D29  CD 10                     INT     10H                  ; GET CURRENT PAGE NUMBER
0D2B  B3 07                     MOV     BL,07H               ; NORMAL ATTRIBUTES
0D2D  B8 1300                   MOV     AX,01300H            ; WRITE STRING FOR INT 10
0D30  CD 10                     INT     10H                  ; DISPLAY ERROR MESSAGE
                  ;
                  ; DISPLAY ASCII MESSAGE IF NECESSARY
                  ;
0D32  07                     EM30:   POP     ES
0D33  5B                        POP     BX                   ; GET ASCII ERROR MSG POINTER
0D34  0B DB                     OR      BX,BX                ; ASCII ERROR MESSAGE
0D36  74 24                     JE      EMSG_RET
                  ;
                  ; SCAN TO GET LENGTH
                  ;
0D38  8B EB                     MOV     BP,BX                ; POINT TO MESSAGE FOR INT 10
0D3A  43                     EM35:   INC     BX              ; INC POINTER
0D3B  26: 80 3F 04              CMP     BYTE PTR ES:[BX],04H ; END OF MESSAGE ?
0D3F  75 F9                     JNE     EM35                 ; IF NOT END LOOK AT NEXT BYTE

0D41  2B DD                  EM40:   SUB     BX,BP
0D43  8B CB                     MOV     CX,BX                ; LENGTH OF STRING FOR INT 10
0D45  2E: 8A 94 0078 R          MOV     DL,DIS_POS[SI]       ; GET COLUMN FOR DISPLAY
0D4A  80 C2 0C                  ADD     DL,12                ; MOVE OVER 9 COLUMNS
```

```
0D4D  83 FF 02              CMP    DI,2               ; FEATURE ICON?
0D50  75 03                 JNE    EM45               ; NO THEN DONT COLUMN CORRECT
0D52  80 EA 04              SUB    DL,4               ; MOVE BACK 4 COLUMNS
0D55                     EM45:
0D55  B3 07                 MOV    BL,07H             ; SET NORMAL ATTRIBUTE
0D57  B8 1300               MOV    AX,01300H          ; WRITE STRING FOR INT 10
0D5A  CD 10                 INT    10H                ; DSIPLAY ERROR MESSAGE

0D5C                     EMSG_RET:
0D5C  07                    POP    ES                 ; RESTORE SEGMENT REGS
0D5D  1F                    POP    DS
0D5E  C3                    RET                       ; RETURN TO CALLER
0D5F                     E_MSG   ENDP
```

Icon Display Routine (ICON_PR)

```
                ;*****************************************************************
                ;
                ; ROUTINE-NAME :  ICON_PR
                ;
                ; FUNCTION: THIS ROUTINE DISPLAYS ICONS ON THE SCREEN
                ;
                ;
                ; ENTRY CONDITIONS:
                ;      PURPOSE OF ENTRY: TO DISPLAY AN ICON
                ;      INPUT CONDITIONS: DH = ROW NUMBER TO DISPALY ICON
                ;                        DL = COLUMN NUMBER TO DISPLAY ICON
                ;                        ES:BP  POINTS TO THE DATA TO DISPLAY
                ;
                ;      ICON FORMAT: DB    ROW OFFSET,COLUMN OFFSET,WIDTH,DEPTH
                ;                   DB    ASCII DATA BYTES FOR ICON AND CONTROL CODES
                ;
                ;      CONTROL CODES:
                ;             01 XX YY     : REPEAT CHARACTER
                ;                            XX= NUMBER OF TIMES TO REPEAT CHARACTER
                ;                            YY= CHARACTER TO REPEAT
                ;
                ;             02 XX YY     : DISPLAY CHARACTER WITH ATTRIBUTE
                ;                            XX= ATTRIBUTE AS DEFINED BY PC
                ;                            YY= CHARACTER
                ;             03           : NEXT ROW
                ;                            NEXT BYTE WILL BE DISPLAYED ONE ROW DOWN
                ;             04 XX YY ZZ ... : REPEAT THE FOLLOWING ROW
                ;                               XX= NUMBER OF TIMES TO REPEAT THE ROW
                ;                               YY ZZ... = DATA FOR THE ROW TO REPEAT
                ;
                ;             05 XX YY STR .. : REPEAT STRING FUNCTION
                ;                               WHERE
                ;                               XX = THE NO. OF TIMES TO REPEAT STRING
                ;                               YY = THE LENGTH OF THE STRING
                ;                               STR.. = THE STRING TO REPEAT
                ;             THE STRING MUST NOT CONTAIN ANY REPEAT CHARACTER
                ;             CONTROL CODES
                ;
                ;             06 XX YY ZZ STR.. : REPEAT STRING WITH COMMON ATTRIBUTE
                ;                     WHERE
                ;                     XX = THE NUMBER OF TIMES TO REPEAT THE STRING
                ;                     YY = THE LENGTH OF THE STRING
                ;                     ZZ = THE ATTRIBUTE FOR THE ENTIRE STRING
                ;                     STR.. = THE STRING TO REPEAT
                ;
                ;
                ;  RESTRICTIONS: CONTROL CODE TO REPEAT A CHARACTER MUST NOT REPEAT
                ;                THE CHARACTER PAST THE RIGHT MARGIN OF THE ICON.
                ;                THE CONTROL CODE TO REPEAT A ROW MUST NOT REPEAT
                ;                THE ROW PAST THE BOTTOM OF THE ICON.
                ;                REPEAT CODES CANNOT BE IMBEDDED IN STRINGS TO BE
                ;                REPEATED.
                ;
                ; EXIT CONDITIONS:
                ;      NORMAL EXIT CONDITIONS:
                ;
                ;      ERROR EXIT CONDITIONS:
                ;
                ;      REGISTERS MODIFIED: NONE
                ;
                ; INTERNALLY REFERENCED ROUTINES:  NONE
                ;
                ; EXTERNALLY REFERENCED ROUTINES:
                ;
                ;*****************************************************************
                           ICON_PL STRUC              ; defines resrvd area on stack
0000  00                   STR_ROW DB      0
0001  00                   STR_COL DB      0
0002  00                   MAX_ROW DB      0
```

```
0003  00                    MAX_COL DB      0
0004  00                    REPR_NO DB      0
0005  00                    RSTR_NO DB      0
0006                        ICON_PL ENDS

                            ASSUME CS:ROMCODE
                            ASSUME DS:NOTHING
                            ASSUME ES:NOTHING

0D5F                     ICON_PR PROC    NEAR
                            SAVE
0D5F  50                    PUSH    AX
0D60  53                    PUSH    BX
0D61  51                    PUSH    CX
0D62  52                    PUSH    DX
0D63  55                    PUSH    BP
0D64  56                    PUSH    SI
0D65  57                    PUSH    DI
0D66  06                    PUSH    ES
0D67  1E                    PUSH    DS
0D68  83 EC 06              SUB     SP,6                ; MAKE ROOM FOR TEMP PARMS
0D6B  8B F4                 MOV     SI,SP               ; POINT SI TO TEMPORARY AREA
0D6D  16                    PUSH    SS                  ; SET DS TO STACK SEGMENT
0D6E  1F                    POP     DS
0D6F  B7 00                 MOV     BH,0                ; SET CURRENT DISPLAY PAGE
0D71  B4 01                 MOV     AH,01               ; TURN OFF CURSOR
0D73  B5 20                 MOV     CH,20H
0D75  CD 10                 INT     10H
0D77  B3 07                 MOV     BL,07H              ; SET NORMAL ATTRIBUTES
0D79  B5 00                 MOV     CH,0                ; INIT CH TO 0
0D7B  8B F9                 MOV     DI,CX               ; MOVE ICON NO. TO DI
0D7D  D1 E7                 SHL     DI,1                ; MULTIPLY BY 2
0D7F  26: 8A 66 00          MOV     AH,ES:[BP]          ; MOVE ROW OFFSET OF ICON TO AH
0D83  02 F4                 ADD     DH,AH               ; ADD TO OFFSET SENT BY CALLER
0D85  88 34                 MOV     [SI].STR_ROW,DH     ; SAVE STARTING ROW
0D87  45                    INC     BP                  ; POINT TO NEXT LOCATION
0D88  26: 8A 46 00          MOV     AL,ES:[BP]          ; MOVE COLUMN OFFSET OF ICON TO AL
0D8C  02 D0                 ADD     DL,AL               ; ADD TO OFFSET SENT BY CALLER
0D8E  88 54 01              MOV     [SI].STR_COL,DL     ; SAVE STARTING COLUMN
0D91  45                    INC     BP                  ; POINT TO NEXT LOCATION
0D92  26: 8A 46 00          MOV     AL,ES:[BP]          ; MOVE WIDTH OF ICON TO AL
0D96  02 C2                 ADD     AL,DL               ; GET MAX. TOTAL COLUMN NUMBER
0D98  88 44 03              MOV     [SI].MAX_COL,AL     ; SAVE MAXIMUM COLUMN NO.
0D9B  45                    INC     BP                  ; POINT TO NEXT LOCATION
0D9C  26: 8A 66 00          MOV     AH,ES:[BP]          ; MOVE DEPTH OF ICON TO AL
0DA0  02 E6                 ADD     AH,DH               ; ADD TO OFFSET SENT BY CALLER
0DA2  88 64 02              MOV     [SI].MAX_ROW,AH     ; SAVE MAXIMUM ROW NUMBER
0DA5  C6 44 04 00           MOV     [SI].REPR_NO,0      ; CLEAR LINE REPEAT COUNTER
0DA9  C6 44 05 00           MOV     [SI].RSTR_NO,0
                  ;
                  ; WRITE STRING LOOP
                  ;
0DAD  45                 IPR20:  INC     BP             ; POINT TO NEXT LOCATION
0DAE  B9 0001               MOV     CX,1                ; LENGTH OF STRING IS 1
0DB1  3A 54 03              CMP     DL,[SI].MAX_COL     ; IS POINTER PAST MAX COLUMN
0DB4  7C 1A                 JL      IPR23               ; JUMP IF NOT
0DB6  8A 54 01              MOV     DL,[SI].STR_COL     ; POINT TO 1ST POS OF NEXT ROW
0DB9  FE C6                 INC     DH                  ; POINT TO NEXT ROW
0DBB  3A 74 02              CMP     DH,[SI].MAX_ROW     ; ARE WE PAST THE LAST ROW?
0DBE  7C 03                 JL      IPR22
0DC0  E9 0EB9 R             JMP     IPR_RET             ; RETURN TO CALLER

0DC3  FE 4C 04           IPR22:  DEC     [SI].REPR_NO   ; DEC NO. OF ROWS TO REPEAT
0DC6  80 7C 04 00           CMP     [SI].REPR_NO,0
0DCA  7E 04                 JLE     IPR23               ; JUMP IF NO REPEAT ROWS
0DCC  8B EF                 MOV     BP,DI               ; POINT BEG OF ROW TO REPEAT
0DCE  EB DD                 JMP     IPR20               ; CONTINUE

0DD0  26: 8A 46 00       IPR23:  MOV     AL,ES:[BP]     ; GET NEXT BYTE OF DATA
0DD4  3C 07                 CMP     AL,7
0DD6  72 12                 JB      IPR_CNT             ; JUMP IF CONTROL CHARACTER
0DD8                     IPR30:
0DD8  B4 02                 MOV     AH,02               ; POSITON CURSOR
0DDA  CD 10                 INT     10H
```

```
ODDC  B3 07              MOV     BL,07H              ; NORMAL ATTRIBUTES
ODDE  26: 8A 46 00       MOV     AL,ES:[BP]

ODE2  B4 09           IPR40:  MOV     AH,09          ; WRITE AT CURSOR
ODE4  CD 10              INT     10H                 ; DISPLAY CHARACTER
ODE6  FE C2              INC     DL                  ; POINT TO THE NEXT COLUMN
ODE8  EB C3              JMP     IPR20               ; LOOP UNTIL LAST CHARACTER

ODEA                  IPR_CNT:
ODEA  3C 01              CMP     AL,1                ; REPEAT CHARACTER FUNCTION
ODEC  74 13              JE      IPR_REP
ODEE  3C 02              CMP     AL,2                ; ATTRIBUTE OTHER THAN NORMAL
ODF0  74 2F              JE      IPR_ATT
ODF2  3C 03              CMP     AL,3                ; NEXT ROW
ODF4  74 33              JE      IPR_NXRW
ODF6  3C 04              CMP     AL,4                ; REPEAT ROW FUNCTION
ODF8  74 47              JE      IPR_REPR
ODFA  3C 05              CMP     AL,5                ; REPEAT STRING
ODFC  74 50              JE      IPR_REPS
ODFE  E9 0E90 R          JMP     IPR_RSTA            ; REPEAT STRING WITH ATTRIBUTE
0E01                  IPR_REP:
0E01  45                 INC     BP                  ; INC POINTER
0E02  26: 8A 4E 00       MOV     CL,ES:[BP]          ; GET MODIFIER FOR REPITITIONS
0E06  45                 INC     BP                  ; INC POINTER
0E07  26: 8A 46 00       MOV     AL,ES:[BP]          ; GET CHARACTER TO DISPLAY
0E0B  3C 05              CMP     AL,5                ; CHECK FOR ATTRIBUTE FUNCTION
0E0D  72 12              JB      IPR_ATT

0E0F  B4 02           REP10:  MOV     AH,02          ; PSN CURSOR MODE FOR INT 10
0E11  CD 10              INT     10H                 ; POSITION CURSOR
0E13  26: 8A 46 00       MOV     AL,ES:[BP]
0E17  B4 09              MOV     AH,09H              ; WRITE CHAR/ATT AT CURSOR
0E19  CD 10              INT     10H                 ; DISPLAY
0E1B  02 D1              ADD     DL,CL               ; ADD REPITIONS TO POSTION PTR
0E1D  B3 07              MOV     BL,07H              ; RETURN ATTRIBUTE TO NORMAL
0E1F  EB 8C              JMP     IPR20               ; CONTINUE

0E21                  IPR_ATT:
0E21  45                 INC     BP                  ; POINT TO NEXT BYTE
0E22  26: 8A 5E 00       MOV     BL,ES:[BP]          ; STORE ATTRIBUTE IN BL
0E26  45                 INC     BP
0E27  EB E6              JMP     REP10               ; CONTINUE

0E29                  IPR_NXRW:
0E29  FE C6              INC     DH                  ; INCREMEMT ROW POINTER
0E2B  8A 54 01           MOV     DL,[SI].STR_COL     ; START AT RIGHT MARGIN
0E2E  FE 4C 04           DEC     [SI].REPR_NO        ; DEC NO. OF ROWS TO REPEAT
0E31  80 7C 04 00        CMP     [SI].REPR_NO,0
0E35  7C 02              JL      NX10                ; JUMP IF NO REPEAT ROWS
0E37  8B EF              MOV     BP,DI               ; POINT BEG OF ROW TO REPEAT

0E39  3A 74 02        NX10:   CMP     DH,[SI].MAX_ROW ; ARE WE PAST THE LAST ROW?
0E3C  7D 7B              JGE     IPR_RET             ; RETURN TO CALLER
0E3E  E9 0DAD R          JMP     IPR20               ; CONTINUE

0E41                  IPR_REPR:
0E41  45                 INC     BP                  ; INC POINTER
0E42  26: 8A 46 00       MOV     AL,ES:[BP]          ; SAVE NO. TO REPEAT ROW
0E46  88 44 04           MOV     [SI].REPR_NO,AL
0E49  8B FD              MOV     DI,BP               ; SAVE START OF ROW
0E4B  E9 0DAD R          JMP     IPR20               ; CONTINUE

0E4E                  IPR_REPS:
0E4E  45                 INC     BP                  ; INC POINTER
0E4F  26: 8A 46 00       MOV     AL,ES:[BP]          ; GET NO. TO REPEAT ROW
0E53  88 44 05           MOV     [SI].RSTR_NO,AL     ; SAVE NO.  TO REPEAT ROW
0E56  45                 INC     BP                  ; INC POINTER
0E57  26: 8A 4E 00       MOV     CL,ES:[BP]          ; GET LENGTH OF STRING
0E5B  45                 INC     BP                  ; INC POINTER
0E5C  8B FD              MOV     DI,BP               ; SAVE START OF STRING
0E5E                  REPS09:
0E5E  51                 PUSH    CX                  ; SAVE LENGTH OF STRING
0E5F  26: 8A 46 00    REPS10: MOV     AL,ES:[BP]     ; GET CHARCTER
```

```
0E63  3C 02             CMP     AL,02              ; CONTROL CODE
0E65  77 06             JA      REPS30             ; JUMP IF NOT

0E67  45              REPS20: INC     BP           ; INC POINTER
0E68  26: 8A 5E 00      MOV     BL,ES:[BP]         ; MOVE ATTRIBUTE TO BL
0E6C  45                INC     BP                 ; INCREMENT BP
0E6D  B4 02           REPS30: MOV     AH,2
0E6F  CD 10             INT     10H                ; SET CURSOR POSITION
0E71  51                PUSH    CX                 ; SAVE LOOP COUNT
0E72  B9 0001           MOV     CX,1               ; LENGTH OF STRING IS 1 BYTE
0E75  B8 1300           MOV     AX,1300H           ; WRITE STRING FOR INT10
0E78  CD 10             INT     10H                ; WRITE CHARACTER/ATTRIBUTE
0E7A  59                POP     CX                 ; RESTORE LOOP COUNT
0E7B  FE C2             INC     DL                 ; INC COLUMN POINTER
0E7D  B3 07             MOV     BL,07              ; SET ATTRIBUTE TO NORMAL
0E7F  45                INC     BP                 ; INC DATA POINTER
0E80  E2 DD             LOOP    REPS10             ; WRITE CHAR TIL END OF STRING
0E82  59                POP     CX                 ; RESTORE LENGTH OF STRING
0E83  FE 4C 05          DEC     [SI].RSTR_NO       ; DEC NO. TO REPEAT STRING
0E86  74 04             JZ      REPS50             ; CONT IF NO MORE REPITITIONS
0E88  8B EF             MOV     BP,DI              ; POINT TO BEGINNING OF STRING
0E8A  EB D2             JMP     REPS09             ; REPEAT STRING

0E8C  4D              REPS50: DEC     BP           ; DECREMENT POINTER
0E8D  E9 0DAD R         JMP     IPR20              ; CONTINUE
0E90                  IPR_RSTA:
0E90  45                INC     BP                 ; INC POINTER
0E91  26: 8A 46 00      MOV     AL,ES:[BP]         ; GET NO. TO REPEAT ROW
0E95  88 44 05          MOV     [SI].RSTR_NO,AL    ; SAVE
0E98  45                INC     BP                 ; INC POINTER
0E99  26: 8A 4E 00      MOV     CL,ES:[BP]         ; GET LENGTH OF STRING
0E9D  45                INC     BP                 ; INC POINTER
0E9E  26: 8A 5E 00      MOV     BL,ES:[BP]         ; GET ATTRIBUTE
0EA2  45                INC     BP                 ; INC POINTER
0EA3  8B FD             MOV     DI,BP              ; SAVE START OF ROW
0EA5  8B EF           RSTA10: MOV     BP,DI        ; RESET POINTER
0EA7  B8 1300           MOV     AX,01300H          ; WRITE STRING FOR INT 10
0EAA  CD 10             INT     10H                ; DISPLAY STRING
0EAC  02 D1             ADD     DL,CL              ; ADD TO COLUMN POINTER
0EAE  FE 4C 05          DEC     [SI].RSTR_NO       ; DEC NO. TO REPEAT STRING
0EB1  75 F2             JNZ     RSTA10             ; PRINT STRING AGAIN
0EB3                  RSTA_RET:
0EB3  03 E9             ADD     BP,CX              ; POINT TO POSTION IN DATA
0EB5  4D                DEC     BP
0EB6  E9 0DAD R         JMP     IPR20              ; CONTINUE

0EB9                  IPR_RET:
0EB9  83 C4 06          ADD     SP,6               ; DE-ALLOCATE STACK SPACE
                        RESTORE
0EBC  1F                POP     DS
0EBD  07                POP     ES
0EBE  5F                POP     DI
0EBF  5E                POP     SI
0EC0  5D                POP     BP
0EC1  5A                POP     DX
0EC2  59                POP     CX
0EC3  5B                POP     BX
0EC4  58                POP     AX
0EC5  C3                RET                        ; RETURN TO CALLER

0EC6                  ICON_PR ENDP
```

Diskette Interrupt Entry (DSKT_INT)

```
                ;************************************
                ;    DISKETTE INTERRUPT ENTRY ADDRESS
                ;************************************
                ;ORG     0EF57H
0F57                     ORG     00F57H
= 0F57                   DSKT_INT        EQU     $
0F57  E9 0000 E          JMP     DSKT_INTE
```

Diskette Drive Parameters

```
                ;****************************************************************
                ; DISKETTE DRIVE TYPE PARAMETERS (FOR READ DRIVE PARAMETERS CALL)
                ;****************************************************************
                ;
                ; TWO SIDED 5 1/4" DRIVE PARMS
                ;
0F5A                     PARMS_TPI48     LABEL   BYTE
0F5A  F000               DW      0F000H                 ; POINTER TO DRIVE PARMS (SEG)
0F5C  0FC7 R             DW      OFFSET DSKT_BASE       ; POINTER TO DRIVE PARMS (OFF)
0F5E  2709               DW      2709H                  ; MAX TRK #/S IDE,SECTOR/TRACK
0F60  0100               DW      0100H                  ; MAX HEAD #, F ILL
                ;
                ; TWO SIDED 3 1/2" DRIVE PARMS
                ;
0F62                     PARMS_TPI135    LABEL   BYTE
0F62  F000                DW      0F000H                ; POINTER TO DRIVE PARMS (SEG)
0F64  0FC7 R              DW      OFFSET DSKT_BASE      ; POINTER TO DRIVE PARMS (OFF)
0F66  4F09                DW      4F09H                 ; MAX TRK #/SIDE,SECTOR/TRACK
0F68  0100                DW      0100H                 ; MAX HEAD #, FILL

                ;********************************
                ; DISKETTE PARAMETER TABLE
                ;********************************
                ;ORG     0EFC7H
0FC7                      ORG     00FC7H
```

Diskette Timing Parameters (DSKT_BASE)

```
                ;************************************************
                ; DSKT_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
                ;************************************************

0FC7                     DSKT_BASE       LABEL   BYTE
0FC7  D0                    DB      11010000B           ; SRT=D, HD UNLD=00 - 1ST
0FC8  02                    DB      2                   ; HD LOAD=1, MODE=DMA - 2ND
0FC9  25                    DB      MOTOR_WAIT          ; WAIT AFTER OPN TIL MOTOR OFF
0FCA  02                    DB      2                   ; 512 BYTES/SECTOR
0FCB  09                    DB      9                   ; EOT ( LAST SECTOR ON TRACK)
0FCC  2A                    DB      02AH                ; GAP LENGTH
```

```
OFCD  FF                     DB      OFFH              ; DTL
OFCE  50                     DB      050H              ; GAP LENGTH FOR FORMAT
OFCF  F6                     DB      0F6H              ; FILL BYTE FOR FORMAT
OFD0  OF                     DB      15                ; HEAD SETTLE TIME (MSEC)
OFD1  04                     DB      4                 ; MOTOR START TIME (1/8 SEC)
```

Printer I/O Entry (PRINTER_IO)

```
                   ;*********************************
                   ;    PRINTER_IO ENTRY POINT
                   ;*********************************
                   ;ORG     0EFD2H
OFD2                        ORG     00FD2H
= OFD2                     PRINTER_IO      EQU     $
OFD2  E9 0000 E             JMP     PRT_IO
```

Video Parameters

```
                   ;***************************************
                   ;    VIDEO DISPLAY TYPE PARAMETER TABLES
                   ;***************************************
                   ; WORD 1 = IBM DISPLAY NUMBER
                   ; WORD 2 = # VERTICAL PELS / MICROMETER
                   ; WORD 3 = # HORIZONTAL PELS / MICROMETER
                   ; WORD 4 = TOTAL # OF VERTICAL PELS
                   ; WORD 5 = TOTAL # OF HORIZONTAL PELS
                   ; WORD 6 = HEIGHT OF PEL IN MICROMETER (VERTICAL DIRECTION)
                   ; WORD 7 = WIDTH OF PEL IN MICROMETER (HORIZONTAL DIRECTION)

OFD5                       MONO_TBL        LABEL   WORD
OFD5  5151 0000 0000       DW      5151H,0,0,0,0,0,0      ; TABLE FOR MONO DISPLAY
      0000 0000 0000 0000

OFE3                       CGA_TBL         LABEL   WORD
OFE3  5153 0498 0A15       DW      5153H,498H,0A15H,0C8H,280H,352H,184H ; RGB DISPLAY
      00C8 0280 0352 0184

OFF1                       LCD_CGA_TBL     LABEL   WORD
OFF1  5140 08E1 0987       DW      5140H,8E1H,987H,0C8H,280H,172H,172H ; TABLE FOR LCD
      00C8 0280 0172 0172
                                                          ; AS A CGA DISPLAY
OFFF                       LCD_MONO_TBL    LABEL   WORD
OFFF  5140 0000 0000       DW      5140H,0,0,0,0,0,0      ; TABLE FOR LCD AS A MONO
      0000 0000 0000 0000
                                                          ; DISPLAY
                   ;***************************************
                   ;    VIDEO_IO  SUBROUTINE ADDRESS TABLE
                   ;***************************************
                   ;ORG     0F045H
1045                       ORG     01045H
                             ASSUME  CS:ROMCODE,DS:DATA,ES:VIDEO_RAM
1045                       M1 LABEL   WORD                ; TBL OF RTNS WITHIN VIDEO I/O
1045  0000 E                  DW      OFFSET  SET_MODE
1047  0000 E                  DW      OFFSET  SET_CTYPE
1049  0000 E                  DW      OFFSET  SET_CPOS
104B  0000 E                  DW      OFFSET  READ_CURSOR
104D  0000 E                  DW      OFFSET  READ_LPEN
104F  0000 E                  DW      OFFSET  ACT_DISP_PAGE
1051  0000 E                  DW      OFFSET  SCROLL_UP
1053  0000 E                  DW      OFFSET  SCROLL_DOWN
1055  0000 E                  DW      OFFSET  READ_AC_CURRENT
1057  0000 E                  DW      OFFSET  WRITE_AC_CURRENT
```

```
1059  0000 E               DW      OFFSET  WRITE_C_CURRENT
105B  0000 E               DW      OFFSET  SET_COLOR
105D  0000 E               DW      OFFSET  WRITE_DOT
105F  0000 E               DW      OFFSET  READ_DOT
1061  0000 E               DW      OFFSET  WRITE_TTY
1063  0000 E               DW      OFFSET  VIDEO_STATE
= 0020                  M1L     EQU     $-M1
```

Video I/O Entry (VIDEO_IO)

```
              ;**************************************
              ;  VIDEO_IO ROUTINE ENTRY POINT
              ;**************************************
              ;ORG     0F065H
1065                     ORG     01065H
= 1065                  VIDEO_IO        EQU     $
1065  E9 0000 E          JMP     VIDEO_IO_1
```

Video Parameters

```
              ;**************************************
              ; VIDEO PARMAMETER TABLE
              ;**************************************
              ;ORG     0F0A4H
10A4                     ORG     010A4H
10A4                    VIDEO_PARMS     LABEL   BYTE

              ;----- INIT_TABLE

10A4  38 28 2D 0A 1F 06    DB      38H,28H,2DH,0AH,1FH,6,19H ; SET UP FOR 40X25
      19
10AB  1C 02 07 06 07       DB      1CH,2,7,6,7
10B0  00 00 00 00          DB      0,0,0,0
= 0010                  M4 EQU     $-VIDEO_PARMS

10B4  71 50 5A 0A 1F 06    DB      71H,50H,5AH,0AH,1FH,6,19H ; SET UP FOR 80X25
      19
10BB  1C 02 07 06 07       DB      1CH,2,7,6,7
10C0  00 00 00 00          DB      0,0,0,0

10C4  38 28 2D 0A 7F 06    DB      38H,28H,2DH,0AH,7FH,6,64H ; SET UP FOR GRAPHICS
      64
10CB  70 02 01 06 07       DB      70H,2,1,6,7
10D0  00 00 00 00          DB      0,0,0,0

10D4  61 50 52 0F 19 06    DB      61H,50H,52H,0FH,19H,6,19H ; SET UP FOR 80X25 B&W
      19
10DB  19 02 0D 0B 0C       DB      19H,2,0DH,0BH,0CH
10E0  00 00 00 00          DB      0,0,0,0

10E4                    M5 LABEL   WORD                ; TABLE OF REGEN LENGTHS
10E4  0800                 DW      2048                ; 40X25
10E6  1000                 DW      4096                ; 80X25
10E8  4000                 DW      16384               ; GRAPHICS
10EA  4000                 DW      16384

              ;------ COLUMNS

10EC                    M6 LABEL   BYTE
10EC  28 28 50 50 28 28    DB      40,40,80,80,40,40,80,80
      50 50
```

```
                ;------ C_REG_TAB

10F4                        M7 LABEL    BYTE              ; TABLE OF MODE SETS
10F4  2C 28 2D 29 2A 2E        DB       2CH,28H,2DH,29H,2AH,2EH,1EH,29H
      1E 29
```

Memory Interrupt Hex 12 (MEMORY_SIZE_DET)

```
                ;--- INT 12 ----------------------------------------------------
                ; MEMORY_SIZE_DET
                ;    THIS ROUTINE DETERMINES THE AMOUNT OF MEMORY IN THE SYSTEM
                ;    AS REPRESENTED BY MEMORY_SIZE WORD. THE MEMORY_SIZE WORD IS SET
                ;    BY POST WHEN IT HAS DETERMINED HOW MUCH MEMORY IS USABLE. IT IS
                ;    SET IN 1K INCREMENTS ON 8K BYTE BOUNDRIES
                ; INPUT
                ;       NO REGISTERS
                ; OUTPUT
                ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
                ;              AS DETERMINED BY THE POWER ON SELF TEST ROUTINE
                ;---------------------------------------------------------------
                               ASSUME  DS:DATA
                ;  ORG      0F841H
1841                           ORG     01841H
1841                        MEMORY_SIZE_DET PROC    FAR
1841  1E                       PUSH    DS                 ; SAVE SEGMENT
1842  E8 085C R                CALL    DDS
1845  A1 0013 R                MOV     AX,MEMORY_SIZE     ; GET VALUE
1848  1F                       POP     DS                 ; RECOVER SEGMENT
1849  CF                       IRET                       ; RETURN TO CALLER
184A                        MEMORY_SIZE_DET  ENDP
```

Equipment Interrupt Hex 11 (EQUIPMENT)

```
                ;--- INT 11 -------------------------------------------------
                ; EQUIPMENT DETERMINATION                                   :
                ;       THIS ROUTINE ATEMPTS  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 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 = INTERNAL MODEM INSTALLED                   :
                ;       BIT 12 = GAME I/O ATTACHED                          :
                ;       BIT 11,10,9 = NUMBER OF SERIAL COMM DEVICES ATTACHED  :
                ;       BIT 8 UNUSED                                        :
                ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES                 :
                ;               00=1, 01=2                                  :
                ;       BIT 5,4 = INITIAL VIDEO MODE                        :
                ;                       00 - UNUSED                         :
                ;                       01 - 40X25 BW USING COLOR CARD      :
```

```
              ;                       10 - 80X25 BW USING COLOR CARD         :
              ;                       11 - 80X25 BW USING BW CARD            :
              ;        BIT 3,2,1 = RESERVED                                  :
              ;        BIT 0 = IPL FROM DISKETTE -- ALWAYS A 1 (SYSTEM DISKETTE:
              ;             INSTALLED)                                       :
              ;                                                              :
              ;        NO OTHER REGISTERS AFFECTED                           :
              ;---------------------------------------------------------------
              ;  ORG     0F84DH
184D                       ORG     0184DH
184D                     EQUIPMENT       PROC    FAR
184D  1E                   PUSH    DS                    ; SAVE SEGMENT REGISTER
184E  E8 085C R            CALL    DDS
1851  A1 0010 R            MOV     AX,EQUIP_FLAG         ; GET THE CURRENT SETTINGS
1854  1F                   POP     DS                    ; RECOVER SEGMENT
1855  CF                   IRET                          ; RETURN TO CALLER
1856                     EQUIPMENT       ENDP
```

Cassette I/O Entry (No BIOS Support)

```
              ;**********************************************
              ;CASSETTE  I/O ENTRY POINT (NO BIOS SUPPORT)
              ;**********************************************
              ;ORG     0F859H
1859                       ORG     01859H
= 1859                   CASSETTE_IO     EQU     $
1859  E9 0000 E            JMP     SYS_SERVICES
```

Character Generator Graphics 0–127 (CHAR_GEN_LO)

```
                           SUBTTL  CHARACTER GENERATOR LOW
              ;----------------------------------------------------------------
              ;  CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200 GRAPHICS
              ;  AND DEFAULT LCD CHARACTER GENERATOR
              ;  FOR CHARACTERS 00H - 7FH
              ;----------------------------------------------------------------
              ;ORG     0FA6EH
1A6E                       ORG     01A6EH

1A6E                     CHAR_GEN_LO     LABEL   BYTE
1A6E  00 00 00 00 00 00    DB      000H,000H,000H,000H,000H,000H,000H,000H ; D_00
      00 00
1A76  3C 42 A5 A5 81 BD    DB      03CH,042H,0A5H,0A5H,081H,0BDH,05AH,03CH ; D_01
      5A 3C
1A7E  3C 7E DB DB FF C3    DB      03CH,07EH,0DBH,0DBH,0FFH,0C3H,066H,03CH ; D_02
      66 3C
1A86  36 7F 7F 7F 3E 1C    DB      036H,07FH,07FH,07FH,03EH,01CH,008H,000H ; D_03
      08 00
1A8E  08 1C 3E 7F 3E 1C    DB      008H,01CH,03EH,07FH,03EH,01CH,008H,000H ; D_04
      08 00
1A96  1C 3E 1C 7F 7F 36    DB      01CH,03EH,01CH,07FH,07FH,036H,008H,01CH ; D_05
      08 1C
1A9E  08 1C 3E 7F 7F 36    DB      008H,01CH,03EH,07FH,07FH,036H,008H,01CH ; D_06
      08 1C
1AA6  00 00 18 3C 3C 18    DB      000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
      00 00
1AAE  FF FF E7 C3 C3 E7    DB      0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
```

```
       FF FF
1AB6   00 00 3C 66 66 3C     DB      000H,000H,03CH,066H,066H,03CH,000H,000H ; D_09
       00 00
1ABE   FF FF C3 99 99 C3     DB      0FFH,0FFH,0C3H,099H,099H,0C3H,0FFH,0FFH ; D_0A
       FF FF
1AC6   07 03 3E 66 66 66     DB      007H,003H,03EH,066H,066H,066H,03CH,000H ; D_0B
       3C 00
1ACE   3C 66 66 66 3C 18     DB      03CH,066H,066H,066H,03CH,018H,03CH,018H ; D_0C
       3C 18
1AD6   08 0C 0E 0A 0A 08     DB      008H,00CH,00EH,00AH,00AH,008H,038H,030H ; D_0D
       38 30
1ADE   18 16 19 17 71 61     DB      018H,016H,019H,017H,071H,061H,007H,006H ; D_0E
       07 06
1AE6   48 6B 3E E4 27 7C     DB      048H,06BH,03EH,0E4H,027H,07CH,0D6H,012H ; D_0F
       D6 12
1AEE   40 70 7C 7F 7C 70     DB      040H,070H,07CH,07FH,07CH,070H,040H,000H ; D_10
       40 00
1AF6   01 07 1F 7F 1F 07     DB      001H,007H,01FH,07FH,01FH,007H,001H,000H ; D_11
       01 00
1AFE   18 3C 7E 18 18 7E     DB      018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
       3C 18
1B06   6C 6C 6C 6C 6C 00     DB      06CH,06CH,06CH,06CH,06CH,000H,06CH,000H ; D_13
       6C 00
1B0E   3F 6A 6A 3A 0A 0A     DB      03FH,06AH,06AH,03AH,00AH,00AH,01AH,000H ; D_14
       1A 00
1B16   78 EC 70 D8 6C 38     DB      078H,0ECH,070H,0D8H,06CH,038H,0DCH,078H ; D_15
       DC 78
1B1E   00 00 00 00 7E 7E     DB      000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
       7E 00
1B26   18 3C 7E 18 7E 3C     DB      018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
       18 FF
1B2E   18 3C 7E 5A 18 18     DB      018H,03CH,07EH,05AH,018H,018H,018H,000H ; D_18
       18 00
1B36   18 18 18 5A 7E 3C     DB      018H,018H,018H,05AH,07EH,03CH,018H,000H ; D_19
       18 00
1B3E   00 0C 06 7F 7F 06     DB      000H,00CH,006H,07FH,07FH,006H,00CH,000H ; D_1A
       0C 00
1B46   00 18 30 7F 7F 30     DB      000H,018H,030H,07FH,07FH,030H,018H,000H ; D_1B
       18 00
1B4E   00 00 60 60 7F 7F     DB      000H,000H,060H,060H,07FH,07FH,000H,000H ; D_1C
       00 00
1B56   00 14 36 7F 7F 36     DB      000H,014H,036H,07FH,07FH,036H,014H,000H ; D_1D
       14 00
1B5E   08 08 1C 1C 3E 3E     DB      008H,008H,01CH,01CH,03EH,03EH,07FH,000H ; D_1E
       7F 00
1B66   7F 3E 3E 1C 1C 08     DB      07FH,03EH,03EH,01CH,01CH,008H,008H,000H ; D_1F
       08 00
1B6E   00 00 00 00 00 00     DB      000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
       00 00
1B76   30 30 30 30 30 00     DB      030H,030H,030H,030H,030H,000H,030H,000H ; ! D_21
       30 00
1B7E   36 36 14 00 00 00     DB      036H,036H,014H,000H,000H,000H,000H,000H ; " D_22
       00 00
1B86   0A 0A 3F 14 7E 28     DB      00AH,00AH,03FH,014H,07EH,028H,028H,000H ; # D_23
       28 00
1B8E   08 3E 68 3E 0B 7E     DB      008H,03EH,068H,03EH,00BH,07EH,008H,000H ; $ D_24
       08 00
1B96   01 3F 52 6C 1B 35     DB      001H,03FH,052H,06CH,01BH,035H,076H,000H ; % D_25
       76 00
1B9E   1C 36 1C 3B 6E 66     DB      01CH,036H,01CH,03BH,06EH,066H,03BH,000H ; & D_26
       3B 00
1BA6   18 18 30 00 00 00     DB      018H,018H,030H,000H,000H,000H,000H,000H ; ' D_27
       00 00
1BAE   06 0C 18 18 18 0C     DB      006H,00CH,018H,018H,018H,00CH,006H,000H ; ( D_28
       06 00
1BB6   30 18 0C 0C 0C 18     DB      030H,018H,00CH,00CH,00CH,018H,030H,000H ; ) D_29
       30 00
1BBE   00 36 1C 7F 1C 36     DB      000H,036H,01CH,07FH,01CH,036H,000H,000H ; * D_2A
       00 00
1BC6   00 18 18 7E 18 18     DB      000H,018H,018H,07EH,018H,018H,000H,000H ; + D_2B
       00 00
1BCE   00 00 00 00 00 18     DB      000H,000H,000H,000H,000H,018H,018H,030H ; , D_2C
       18 30
1BD6   00 00 00 7E 00 00     DB      000H,000H,000H,07EH,000H,000H,000H,000H ; - D_2D
```

```
      00 00
1BDE  00 00 00 00 00 18    DB      000H,000H,000H,000H,000H,018H,018H,000H ; . D_2E
      18 00
1BE6  03 06 0C 18 30 60    DB      003H,006H,00CH,018H,030H,060H,040H,000H ; / D_2F
      40 00
1BEE  18 2C 66 66 66 34    DB      018H,02CH,066H,066H,066H,034H,018H,000H ; 0 D_30
      18 00
1BF6  18 18 38 18 18 18    DB      018H,018H,038H,018H,018H,018H,03CH,000H ; 1 D_31
      3C 00
1BFE  3C 66 66 0C 18 32    DB      03CH,066H,066H,00CH,018H,032H,07EH,000H ; 2 D_32
      7E 00
1C06  3C 66 0C 1C 06 66    DB      03CH,066H,00CH,01CH,006H,066H,03CH,000H ; 3 D_33
      3C 00
1C0E  0C 1C 2C 6C 7E 0C    DB      00CH,01CH,02CH,06CH,07EH,00CH,01EH,000H ; 4 D_34
      1E 00
1C16  7E 60 7C 66 06 66    DB      07EH,060H,07CH,066H,006H,066H,03CH,000H ; 5 D_35
      3C 00
1C1E  1C 30 60 7C 66 66    DB      01CH,030H,060H,07CH,066H,066H,03CH,000H ; 6 D_36
      3C 00
1C26  7E 66 4C 0C 18 18    DB      07EH,066H,04CH,00CH,018H,018H,038H,000H ; 7 D_37
      38 00
1C2E  3C 66 76 3C 6E 66    DB      03CH,066H,076H,03CH,06EH,066H,03CH,000H ; 8 D_38
      3C 00
1C36  3C 66 66 3E 06 0C    DB      03CH,066H,066H,03EH,006H,00CH,038H,000H ; 9 D_39
      38 00
1C3E  00 18 18 00 00 18    DB      000H,018H,018H,000H,000H,018H,018H,000H ; : D_3A
      18 00
1C46  00 18 18 00 00 18    DB      000H,018H,018H,000H,000H,018H,018H,030H ; ; D_3B
      18 30
1C4E  06 0C 18 30 18 0C    DB      006H,00CH,018H,030H,018H,00CH,006H,000H ; < D_3C
      06 00
1C56  00 00 7E 00 7E 00    DB      000H,000H,07EH,000H,07EH,000H,000H,000H ; = D_3D
      00 00
1C5E  30 18 0C 06 0C 18    DB      030H,018H,00CH,006H,00CH,018H,030H,000H ; > D_3E
      30 00
1C66  3C 66 26 0C 18 00    DB      03CH,066H,026H,00CH,018H,000H,018H,000H ; ? D_3F
      18 00
1C6E  3E 41 5D 55 5F 40    DB      03EH,041H,05DH,055H,05FH,040H,03EH,000H ; @ D_40
      3E 00
1C76  1C 0C 1C 16 3E 23    DB      01CH,00CH,01CH,016H,03EH,023H,063H,000H ; A D_41
      63 00
1C7E  7C 36 36 3E 33 33    DB      07CH,036H,036H,03EH,033H,033H,07EH,000H ; B D_42
      7E 00
1C86  1D 33 61 60 60 31    DB      01DH,033H,061H,060H,060H,031H,01EH,000H ; C D_43
      1E 00
1C8E  7C 36 33 33 33 36    DB      07CH,036H,033H,033H,033H,036H,07CH,000H ; D D_44
      7C 00
1C96  7F 31 34 3C 34 31    DB      07FH,031H,034H,03CH,034H,031H,07FH,000H ; E D_45
      7F 00
1C9E  7F 31 34 3C 34 30    DB      07FH,031H,034H,03CH,034H,030H,078H,000H ; F D_46
      78 00
1CA6  1D 33 61 60 67 33    DB      01DH,033H,061H,060H,067H,033H,01FH,000H ; G D_47
      1F 00
1CAE  66 66 66 7E 66 66    DB      066H,066H,066H,07EH,066H,066H,066H,000H ; H D_48
      66 00
1CB6  3C 18 18 18 18 18    DB      03CH,018H,018H,018H,018H,018H,03CH,000H ; I D_49
      3C 00
1CBE  1F 06 06 06 66 66    DB      01FH,006H,006H,006H,066H,066H,03CH,000H ; J D_4A
      3C 00
1CC6  67 66 6C 78 6C 66    DB      067H,066H,06CH,078H,06CH,066H,067H,000H ; K D_4B
      67 00
1CCE  78 30 30 30 31 33    DB      078H,030H,030H,030H,031H,033H,07FH,000H ; L D_4C
      7F 00
1CD6  41 63 77 7F 6B 63    DB      041H,063H,077H,07FH,06BH,063H,063H,000H ; M D_4D
      63 00
1CDE  43 63 73 7B 6F 67    DB      043H,063H,073H,07BH,06FH,067H,063H,000H ; N D_4E
      63 00
1CE6  1C 36 63 63 63 36    DB      01CH,036H,063H,063H,063H,036H,01CH,000H ; O D_4F
      1C 00
1CEE  7E 33 33 3E 30 30    DB      07EH,033H,033H,03EH,030H,030H,078H,000H ; P D_50
      78 00
1CF6  1C 36 63 63 6B 36    DB      01CH,036H,063H,063H,06BH,036H,01CH,007H ; Q D_51
      1C 07
1CFE  7C 66 66 7C 6C 66    DB      07CH,066H,066H,07CH,06CH,066H,067H,000H ; R D_52
```

```
         67 00
1D06     3E 66 70 3C 0E 66     DB      03EH,066H,070H,03CH,00EH,066H,07CH,000H ; S D_53
         7C 00
1D0E     7E 5A 18 18 18 18     DB      07EH,05AH,018H,018H,018H,018H,03CH,000H ; T D_54
         3C 00
1D16     63 63 63 63 63 63     DB      063H,063H,063H,063H,063H,063H,03EH,000H ; U D_55
         3E 00
1D1E     77 62 36 34 1C 18     DB      077H,062H,036H,034H,01CH,018H,008H,000H ; V D_56
         08 00
1D26     63 63 6B 6B 3E 36     DB      063H,063H,06BH,06BH,03EH,036H,022H,000H ; W D_57
         22 00
1D2E     66 66 3C 18 3C 66     DB      066H,066H,03CH,018H,03CH,066H,066H,000H ; X D_58
         66 00
1D36     E7 66 34 18 18 18     DB      0E7H,066H,034H,018H,018H,018H,03CH,000H ; Y D_59
         3C 00
1D3E     7E 66 4C 18 32 66     DB      07EH,066H,04CH,018H,032H,066H,07EH,000H ; Z D_5A
         7E 00
1D46     1E 18 18 18 18 18     DB      01EH,018H,018H,018H,018H,018H,01EH,000H ; [ D_5B
         1E 00
1D4E     60 30 18 0C 06 03     DB      060H,030H,018H,00CH,006H,003H,001H,000H ; BSL D_5C
         01 00
1D56     3C 0C 0C 0C 0C 0C     DB      03CH,00CH,00CH,00CH,00CH,00CH,03CH,000H ; ] D_5D
         3C 00
1D5E     08 1C 36 63 00 00     DB      008H,01CH,036H,063H,000H,000H,000H,000H ; CIR D_5E
         00 00
1D66     00 00 00 00 00 00     DB      000H,000H,000H,000H,000H,000H,000H,0FEH ; _ D_5F
         00 FE
1D6E     18 18 0C 00 00 00     DB      018H,018H,00CH,000H,000H,000H,000H,000H ; Ω D_60
         00 00
1D76     00 00 3C 66 1E 66     DB      000H,000H,03CH,066H,01EH,066H,07BH,000H ; LC A D_61
         7B 00
1D7E     70 30 3E 3B 33 3B     DB      070H,030H,03EH,03BH,033H,03BH,06EH,000H ; LC B D_62
         6E 00
1D86     00 00 3E 66 60 66     DB      000H,000H,03EH,066H,060H,066H,03CH,000H ; LC C D_63
         3C 00
1D8E     0E 06 36 6E 66 66     DB      00EH,006H,036H,06EH,066H,066H,03BH,000H ; LC D D_64
         3B 00
1D96     00 00 3C 66 7E 60     DB      000H,000H,03CH,066H,07EH,060H,03EH,000H ; LC E D_65
         3E 00
1D9E     0E 1B 18 3E 18 18     DB      00EH,01BH,018H,03EH,018H,018H,03CH,000H ; LC F D_66
         3C 00
1DA6     00 00 3D 66 38 3E     DB      000H,000H,03DH,066H,038H,03EH,063H,03EH ; LC G D_67
         63 3E
1DAE     70 30 36 3B 33 33     DB      070H,030H,036H,03BH,033H,033H,073H,000H ; LC H D_68
         73 00
1DB6     18 00 38 18 18 18     DB      018H,000H,038H,018H,018H,018H,03CH,000H ; LC I D_69
         3C 00
1DBE     0C 00 1C 0C 0C 6C     DB      00CH,000H,01CH,00CH,00CH,06CH,06CH,038H ; LC J D_6A
         6C 38
1DC6     70 30 33 36 3C 36     DB      070H,030H,033H,036H,03CH,036H,077H,000H ; LC K D_6B
         77 00
1DCE     38 18 18 18 18 18     DB      038H,018H,018H,018H,018H,03CH,000H ; LC L D_6C
         3C 00
1DD6     00 00 76 7F 6B 6B     DB      000H,000H,076H,07FH,06BH,06BH,06BH,000H ; LC M D_6D
         6B 00
1DDE     00 00 76 3B 33 33     DB      000H,000H,076H,03BH,033H,033H,073H,000H ; LC N D_6E
         73 00
1DE6     00 00 3C 66 66 66     DB      000H,000H,03CH,066H,066H,066H,03CH,000H ; LC O D_6F
         3C 00
1DEE     00 00 6E 33 33 3E     DB      000H,000H,06EH,033H,033H,03EH,030H,078H ; LC P D_70
         30 78
1DF6     00 00 3A 66 66 3E     DB      000H,000H,03AH,066H,066H,03EH,006H,00FH ; LC Q D_71
         06 0F
1DFE     00 00 6E 3B 33 30     DB      000H,000H,06EH,03BH,033H,030H,078H,000H ; LC R D_72
         78 00
1E06     00 00 3E 70 3C 0E     DB      000H,000H,03EH,070H,03CH,00EH,07CH,000H ; LC S D_73
         7C 00
1E0E     08 18 3E 18 18 1A     DB      008H,018H,03EH,018H,018H,01AH,00CH,000H ; LC T D_74
         0C 00
1E16     00 00 66 66 66 66     DB      000H,000H,066H,066H,066H,066H,03BH,000H ; LC U D_75
         3B 00
1E1E     00 00 73 32 36 1C     DB      000H,000H,073H,032H,036H,01CH,008H,000H ; LC V D_76
         08 00
1E26     00 00 6B 6B 7F 36     DB      000H,000H,06BH,06BH,07FH,036H,022H,000H ; LC W D_77
```

```
     22 00
1E2E 00 00 73 36 1C 36    DB     000H,000H,073H,036H,01CH,036H,067H,000H ; LC X D_78
     67 00
1E36 00 00 77 33 1A 0C    DB     000H,000H,077H,033H,01AH,00CH,06CH,038H ; LC Y D_79
     6C 38
1E3E 00 00 7E 4C 18 32    DB     000H,000H,07EH,04CH,018H,032H,07EH,000H ; LC Z D_7A
     7E 00
1E46 0E 18 18 70 18 18    DB     00EH,018H,018H,070H,018H,018H,00EH,000H ; { D_7B
     0E 00
1E4E 18 18 18 00 18 18    DB     018H,018H,018H,000H,018H,018H,018H,000H ; | D_7C
     18 00
1E56 70 18 18 0E 18 18    DB     070H,018H,018H,00EH,018H,018H,070H,000H ; } D_7D
     70 00
1E5E 39 4E 00 00 00 00    DB     039H,04EH,000H,000H,000H,000H,000H,000H ; ~ D_7E
     00 00
1E66 08 1C 1C 36 26 63    DB     008H,01CH,01CH,036H,026H,063H,07FH,07FH ; DELTA D_7F
     7F 7F
```

Time of Day Entry (TIME_OF_DAY)

```
                    ;*************************************
                    ;   TIME_OF_DAY ROUTINE ENTRY POINT
                    ;*************************************

                    ;ORG      0FE6EH
1E6E                           ORG      01E6EH
= 1E6E                        TIME_OF_DAY      EQU      $
1E6E  E9 0000 E                JMP      TOD_PROC
```

Timer 0 Interrupt Handler (TIMER_INT)

```
                    ;*************************************
                    ; TIMER 0 INTERRUPT HANDLER ADDRESS
                    ;*************************************
                    ;ORG      0FEA5H
1EA5                           ORG      01EA5H
= 1EA5                        TIMER_INT        EQU      $
1EA5  E9 0000 E                JMP      TMRO_INT8
```

BIOS Vector Table (VECTOR_TABLE)

```
                    ;*************************************
                    ;  BIOS VECTOR TABLE
                    ;*************************************
                    ;ORG      0FEF3H
1EF3                           ORG      01EF3H
1EF3                          VECTOR_TABLE    LABEL   WORD

                    ;------- HARDWARE INTERRUPTS
```

```
                                                           ; VECTOR TABLE
1EF3  0000 E            DW      OFFSET TMR0_INT8         ; INTERRUPT 8  TIMER 0
1EF5  0000 E            DW      OFFSET KYBD_INT9         ; INTERRUPT 9  KEYBOARD
1EF7  1F23 R            DW      OFFSET  D11              ; INTERRUPT A
1EF9  1F23 R            DW      OFFSET  D11              ; INTERRUPT B
1EFB  1F23 R            DW      OFFSET  D11              ; INTERRUPT C
1EFD  1F23 R            DW      OFFSET  D11              ; INTERRUPT D
1EFF  0000 E            DW      OFFSET DSKT_INTE         ; INTERRUPT E  DISKETTE
1F01  1F23 R            DW      OFFSET  D11              ; INTERRUPT F

              ;------- SOFTWARE INTERRUPTS

1F03  1065 R            DW      OFFSET VIDEO_IO          ; INT 10H
1F05  184D R            DW      OFFSET EQUIPMENT         ; INT 11H
1F07  1841 R            DW      OFFSET MEMORY_SIZE_DET ; INT 12H
1F09  0000 E            DW      OFFSET DSKT_IO           ; INT 13H
1F0B  0000 E            DW      OFFSET COMMO_IO          ; INT 14H
1F0D  0000 E            DW      SYS_SERVICES             ; INT 15H
1F0F  0000 E            DW      OFFSET KYBD_IO           ; INT 16H
1F11  0000 E            DW      OFFSET PRT_IO            ; INT 17H
1F13  0000              DW      00000H                   ; INT 18H (RESIDENT BASIC ST)
              ;DW       0F600H                           ; SEGMENT (INT 18H FILLED IN
              ;                                      BY SYS_SETUP ROUTINE)
1F15  0000 E            DW      OFFSET SYS_BOOT          ; INT 19H
1F17  0000 E            DW      TOD_PROC                 ; INT 1AH -- TIME OF DAY
1F19  1F53 R            DW      DUMMY_RETURN             ; INT 1BH -- KYBD BREAK ADDR
1F1B  1F53 R            DW      DUMMY_RETURN             ; INT 1CH -- TIMER BREAK ADDR
1F1D  10A4 R            DW      VIDEO_PARMS              ; INT 1DH -- VIDEO PARAMETERS
1F1F  0FC7 R            DW      OFFSET DSKT_BASE         ; INT 1EH -- DSKT PARAMETERS
1F21  02C6 R            DW      CHAR_GEN_HI              ; INT 1FH -- PTR TO CHAR_GEN

              ;ORG      0FF23H
1F23                    ORG     01F23H
```

Default Interrupt Services (D11)

```
              ;*********************************************************************
              ;
              ; ROUTINE-NAME :  D11
              ;
              ; FUNCTION: THIS ROUTINE IS UNUSED INTERRUPT LEVEL HANDLER INSTALLED
              ;      BY POST FOR INTERRUPTS NOT USED BY BIOS. THIS ROUTINE RECORDS
              ;      THE OCCURRENCE OF AN INTERRUPT IN INTR_FLAG AND PERFORMS A
              ;      NON_SPECIFIC END_OF_INTERRUPT TO RESET THE INTERRUPT SERVICE.
              ;
              ; ENTRY CONDITIONS:
              ;      PURPOSE OF ENTRY: UNUSED INTERRUPT OCCURRED
              ;      INPUT CONDITIONS: NONE
              ;      RESTRICTIONS:     NONE
              ;
              ; EXIT CONDITIONS:
              ;      NORMAL EXIT CONDITIONS: INTR_FLAG = 00-07 TO INDICATE IRPT #
              ;                              INTR_FLAG = 0FFH WHEN NO HARDWARE IRPT
              ;                                          FOUND ACTIVE
              ;                   THE INTERRUPTING LEVEL WILL BE MASKED IN
              ;                   THE INTERRUPT CONTROLLER IF POST IS NOT ACTIVE.
              ;
              ;  ERROR EXIT CONDITIONS: NONE
              ;
              ;  REGISTERS MODIFIED:    NONE
              ;********************************************************************

1F23                   D11             PROC    NEAR
                        ASSUME DS:DATA
1F23  1E                PUSH    DS
1F24  52                PUSH    DX
1F25  50                PUSH    AX                       ; SAVE REG AX CONTENTS
1F26  E8 085C R         CALL    DDS
```

```
1F29  B0 0B                MOV     AL,0BH                ; READ IN-SERVICE REG
1F2B  E6 20                OUT     INTA00,AL             ; (FIND OUT WHAT LEVEL BEING
1F2D  90                   NOP                           ; SERVICED)
1F2E  E4 20                IN      AL,INTA00             ; GET LEVEL
1F30  0A C0                OR      AL,AL                 ; NO HARDWARE IN SERVICE?
1F32  75 05                JNZ     HW_INT

1F34  B4 FF                MOV     AH,0FFH               ; SET NO LEVEL IN SERVICE FLAG
1F36  EB 08 90             JMP     D11_EXIT              ; SET FLAG TO FF IF NON-HDWARE
              ;
              ; HARDWARE INTERRUPT OCCURRED
              ;
1F39                      HW_INT:
1F39  E8 1F57 R            CALL    MASK_LEVEL            ; GO MASK LEVEL, RETURN WITH
                                                         ; AH INDICATING LEVEL MASKED
1F3C  B0 20                MOV     AL,EOI                ; SEND END OF INTERRUPT
1F3E  E6 20                OUT     INTA00,AL
              ;
              ; SAVE INTR_FLAG AND EXIT
              ;
1F40                      D11_EXIT:
1F40  88 26 006B R         MOV     INTR_FLAG,AH          ; SET FLAG
1F44  58                   POP     AX                    ; RESTORE REG AX CONTENTS
1F45  5A                   POP     DX
1F46  1F                   POP     DS
1F47  CF                   IRET
1F48                      D11             ENDP

              ;ORG      0FF53H
1F53                       ORG     01F53H
              ;*****************************************************************
              ; DUMMY INTERRUPT RETURN
              ;*****************************************************************
1F53                      DUMMY_RETURN    PROC    FAR
1F53  CF                   IRET
1F54                      DUMMY_RETURN    ENDP

              ;*************************************
              ;  PRINT SCREEN ROUTINE ENTRY ADDRESS
              ;*************************************
              ;ORG      0FF54H
1F54                       ORG     01F54H
= 1F54                    PRINT_SCREEN    EQU     $
1F54  E9 0000 E            JMP     PRT_SCRN

              ;******************************************************************
              ; ROUTINE USED BY D11 HANDLER TO MASK THE INTERRUPT LVL BEING SERVICED
              ;*****************************************************************
1F57                      MASK_LEVEL      PROC    NEAR
1F57  51                   PUSH    CX
1F58  B4 01                MOV     AH,01                 ; START AT LEVEL 0
1F5A  E4 A0                IN      AL,IONMI_CNTL         ; READ HIGHEST LVL IN SERVICE
1F5C  24 07                AND     AL,07                 ; JUST KEEP LEVEL BITS
1F5E  8A C8                MOV     CL,AL                 ; GET LEVEL AS SHIFT COUNT
1F60  D2 E4                SHL     AH,CL                 ; AH CONTAINS MASK FOR LEVEL
1F62  F6 06 0012 R 01      TEST    POST_STATUS,POST_ACTIVE ; POWER_ON_SELF_TEST ACTIVE?
1F67  75 06                JNZ     MASK_EXIT             ; JUMP IF POST ACTIVE

1F69  E4 21                IN      AL,INTA01             ; GET MASK VALUE
1F6B  0A C4                OR      AL,AH                 ; MASK OFF LVL BEING SERVICED
1F6D  E6 21                OUT     INTA01,AL
1F6F                      MASK_EXIT:
1F6F  59                   POP     CX
1F70  C3                   RET
1F71                      MASK_LEVEL      ENDP
```

Hardware Power-on Reset Address

```
              ;****************************************
              ;  POWER ON RESET VECTOR                :
              ;****************************************
              ;ORG     OFFFOH
1FFO                     ORG      01FFOH
1FFO                    P_O_R            LABEL  FAR
1FFO  EA                 DB       OEAH                   ; HARD CODE JUMP FAR
1FF1  0000 E             DW       OFFSET  POSTMAIN       ; OFFSET
1FF3  F000               DW       OF000H                 ; SEGMENT

1FF5                    ROMCODE          ENDS
                         END
```

Release Date Marker

```
                         TITLE  BLDMARK.ASM

0000                    ROMCODE SEGMENT BYTE PUBLIC
               ;*****************************************
               ;  MODULE RELEASE DATE MARKER            :
               ;****************************************

0000  30 39 2F 31 33 2F  DB      '09/13/85'           ;RELEASE MARKER
      38 35
0008  4C                 DB      'L'                  ;BUILD NUMBER
```

System Model Byte

```
               ;*****************************************
               ;  P.C. MODEL BYTE FOR THIS MODEL        :
               ;****************************************
               ;ORG     OFFFEH
0009  F9                 DB      OF9H                 ;THIS PC'S ID
000A                    ROMCODE         ENDS
                         END
```

Notes:

Notes:

Notes:

Notes:

IBM

Hardware Reference
Library

Reader's Comment Form

IBM PC Convertible **55X8817**
Technical Reference
Volume 2

Your comments assist us in improving our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for questions regarding setup, operation, or program support or for requests for additional publications. Instead, contact your authorized IBM PC Convertible dealer in your area.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM CORPORATION
DEPARTMENT 95H
11400 BURNET ROAD
AUSTIN, TEXAS 78758

Fold here

Tape Please do not staple Ta

International Business
Machines Corporation
P.O. Box 1328-W
Boca Raton,
Florida 33429-1328

Printed in the
United States of America

55X8817