APPENDIX D

INTERRUPT CALLS AND LEGACY SOFTWARE

OVERVIEW

This appendix lists many of the interrupt calls for INT 21H, INT 10H, and so on, which are used primarily for input, output, and file and memory management.

SECTION D.1: 21H INTERRUPTS

AH Function of INT 21H

00 Terminate the program

Additional Call Registers	Result Registers
CS = segment address of PSP (program segment prefix)	None

Note: Files should be closed previously or data may be lost.

01 Keyboard input with echo

Additional Call Registers	Result Registers
None	AL = input character

Note: Checks for Ctrl-Break.

02 Output character to monitor

Additional Call Registers	Result Registers
DL = character to be displayed	None

03 Asynchronous input from auxiliary device (serial device)

Additional Call Registers	Result Registers
None	AL = input character

04 Asynchronous character output

Additional Call Registers	Result Registers
DL = character to be output	None

05 Output character to printer

Additional Call Registers	Result Registers
DL = character to be printed	None

06 Console I/O

Additional Call Registers	Result Registers
DL = OFFH if input or character to be displayed, if output	AL = 0H if no character available = character that was input, if input successful

Note: If input, ZF is cleared and AL will have the character. ZF is set if input and no character was available.

AH Function

07 Keyboard input without echo

Additional Call Registers	Result Registers
None	AL = input character

Note: Does not check for Ctrl-Break.

08 Keyboard input without echo

Additional Call Registers	Result Registers
None	AL = input character

Note: Checks for Ctrl-Break.

09 String output

Additional Call Registers	Result Registers
DS:DX = string address	None

Note: Displays characters beginning at address until a '$' (ASCII 36) is encountered.

0A String input

Additional Call Registers	Result Registers
DS:DX = address at which to store string	None

Note: Specify the maximum size of the string in byte 1 of the buffer. DOS will place the actual size of the string in byte 2. The string begins in byte 3.

0B Get keyboard status

Additional Call Registers	Result Registers
None	AL = 00 if no character waiting = 0FFH if character waiting

Note: Checks for Ctrl-Break.

0C Reset input buffer and call keyboard input function

Additional Call Registers	Result Registers
AL = keyboard function number 01H, 06H, 07H, 08H or 0AH	None

Note: This function waits until a character is typed in.

AH Function

0D Reset disk

Additional Call Registers	Result Registers
None	None

Note: Flushes DOS file buffers but does not close files.

0E Set default drive

Additional Call Registers	Result Registers
DL = code for drive (0 = A, 1 = B, 2 = C, etc.)	AL = number of logical drives in system

0F Open file

Additional Call Registers	Result Registers
DS:DX = address of FCB	AL = 00 if successful = 0FFH if file not found

Note: Searches current directory for file. If found, FCB is filled.

SECTION D.2: MOUSE INTERRUPTS 33H

AX Function of INT 33H

00 Initialize the mouse

Additional Call Registers	Result Registers
None	AX = 0H if mouse not available = FFFFH if mouse available BX = number of mouse buttons

Note: This function is called only once to initialize the mouse. If mouse support is present, AX = FFFFH, and the mouse driver is initialized, the mouse pointer is set to the center of the screen and concealed.

01 Display mouse pointer

Additional Call Registers	Result Registers
None	None

Note: This function displays the mouse pointer and cancels any exclusion area.

02 Conceal mouse pointer

Additional Call Registers	Result Registers
None	None

Note: This function hides the mouse pointer but the mouse driver monitors its position. Most programs issue this command before they terminate.

AX Function

03 Get mouse location and button status

Additional Call Registers	Result Registers
None	BX = mouse button status bit 0 -- left button bit 1 -- right button bit 2 -- center button = 0 if up; = 1 if down CX = horizontal position DX = vertical position

Note: The horizontal and vertical coordinates are returned in pixels.

04 Set mouse pointer location

Additional Call Registers	Result Registers
CX = horizontal position DX = vertical position	None

Note: The horizontal and vertical coordinates are in pixels. Will display the mouse pointer only within set limits; will not display in exclusion areas.

05 Get button press information

Additional Call Registers	Result Registers
BX = button: 0 for left; 1 for right; 2 for center	AX = button status bit 0 -- left button bit 1 -- right button bit 2 -- center button = 0 if up; = 1 if down BX = button press count CX = horizontal position DX = vertical position

Note: This returns the status of all buttons as well as the number of presses for the button indicated in BX when called. The position of the mouse pointer is given in pixels and represents the position at the last button press.

AX Function

06 Get button release information

Additional Call Registers	Result Registers
BX = button: 0 for left;	AX = button status
1 for right; 2 for center	bit 0 -- left button
	bit 1 -- right button
	bit 2 -- center button
	= 0 if up; = 1 if down
	BX = button release count
	CX = horizontal position
	DX = vertical position

Note: This returns the status of all buttons as well as the number of releases for the button indicated in BX when called. The position of the mouse pointer is given in pixels and represents the position at the last button release.

07 Set horizontal limits for mouse pointer

Additional Call Registers	Result Registers
CX = minimum horizontal position	None
DX = maximum horizontal position	

Note: This sets the horizontal limits (in pixels) for the mouse pointer. After this call, the mouse will be displayed within these limits.

08 Set vertical limits for mouse pointer

Additional Call Registers	Result Registers
CX = minimum vertical position	None
DX = maximum vertical position	

Note: This sets the vertical limits (in pixels) for the mouse pointer. After this call, the mouse will be displayed within these limits.

10 Set mouse pointer exclusion area

Additional Call Registers	Result Registers
CX = upper left horizontal coordinate	None
DX = upper left vertical coordinate	
SI = lower right horizontal coordinate	
DI = lower right vertical coordinate	

Note: This defines an area in which the mouse pointer will not display. An exclusion area can be cancelled by calling functions 00 or 01.

AX Function

24 Get mouse information

Additional Call Registers	Result Registers
None	BH = major version BL = minor version CH = mouse type CL = IRQ number

Note: This returns the version number (e.g., version 7.5: BH = 7, BL = 5). Mouse type: 1 for bus; 2 for serial; 3 for InPort; 4 for PS/2; 5 for HP; IRQ = 0 for PS/2; otherwise = 2, 3, 4, 5 or 7.

SECTION D.3: INT 10H

AH Function

00 Set video mode

Additional Call Registers	Result Registers
AL = video mode	None

See Table D-2 for a list of available video modes and their definitions.

01 Set cursor type

Additional Call Registers	Result Registers
CH = beginning line of cursor (bits 0–4) CL = ending line of cursor (bits 0–4)	None

Note: All other bits should be set to zero. The blinking of the cursor is hardware controlled.

02 Set cursor position

Additional Call Registers	Result Registers
BH = page number DH = row DL = column	None

Note: When using graphics modes, BH must be set to zero. Text coordinates of the upper left-hand corner will be (0,0).

AH Function

03 Read cursor position and size

Additional Call Registers	Result Registers
BH = page number	CH = beginning line of cursor CL = ending line of cursor DH = row DL = column

Note: When using graphics modes, BH must be set to zero.

04 Read light pen position

Additional Call Registers	Result Registers
None	AH = 0 if light pen not triggered = 1 if light pen triggered BX = pixel column CH = pixel row (modes 04H–06H) CX = pixel row (modes 0DH–13H) DH = character row DL = character column

05 Select active display page

Additional Call Registers	Result Registers
AL = page number (see Table D-1 below)	None

Table D-1: Display Pages for Different Modes and Adapters

Mode G	Pages	Adapters
00H	0–7	VGA
01H	0–7	VGA
02H	0–3	CGA
	0–7	VGA
03H	0–3	CGA
	0–7	VGA
07H	0–7	VGA
0DH	0–7	VGA
0EH	0–3	VGA
0FH	0–1	VGA
10H	0–1	VGA

All other mode-adapter combinations support only one page.

AH Function

06 Scroll window up

Additional Call Registers	Result Registers
AL = number of lines to scroll	None
BH = display attribute	
CH = *y* coordinate of top left	
CL = *x* coordinate of top left	
DH = *y* coordinate of lower right	
DL = *x* coordinate of lower right	

Note: If AL = 0, the entire window is blank. Otherwise, the screen will be scrolled upward by the number of lines in AL. Lines scrolling off the top of the screen are lost, and blank lines are scrolled in at the bottom according to the attribute in BH.

07 Scroll window down

Additional Call Registers	Result Registers
AL = number of lines to scroll	None
BH = display attribute	
CH = *y* coordinate of top left	
CL = *x* coordinate of top left	
DH = *y* coordinate of lower right	
DL = *x* coordinate of lower right	

Note: If AL = 0, the entire window is blank. Otherwise, the screen will be scrolled down by the number of lines in AL. Lines scrolling off the bottom of the screen are lost, and blank lines are scrolled in at the top according to the attribute in BH.

08 Read character and attribute at cursor position

Additional Call Registers	Result Registers
BH = display page	
AH = attribute byte	
AL = ASCII character code	

09 Write character and attribute at cursor position

Additional Call Registers	Result Registers
AL = ASCII character code	None
BH = display page	
BL = attribute	
CX = number of characters to write	

Note: Does not update cursor position. Use interrupt 10 Function 2 to set cursor position.

AH Function

0A Write character at cursor position

Additional Call Registers	Result Registers
AL = ASCII character code	None
BH = display page	
BL = graphic color	
CX = number of characters to write	

Note: Writes character(s) using existing video attribute. Does not update cursor position. Use interrupt 10 Function 2 to set cursor position.

0B Set color palette

Additional Call Registers	Result Registers
BH = 00H to set border or background colors	None
= 01H to set palette	
BL = palette/color	

Note: If BH = 00H and in text mode, this function will set the border color only. If BH = 00H and in graphics mode, this function will set background and border colors. If BH = 01H, this function will select the palette. In 320 × 200 four-color graphics, palettes 0 and 1 are available:

Pixel Colors for Palettes 0 and 1

Pixel	Palette 0	Palette 1
0	background	background
1	green	cyan
2	red	magenta
3	brown/yellow	white

0C Write pixel

Additional Call Registers	Result Registers
AL = pixel value	None
CX = pixel column	
DX = pixel row	
BH = page	

Note: Coordinates and pixel value depend on the current video mode. Setting bit 7 of AL causes the pixel value in AL to be XORed with the current value of the pixel.

0D Read pixel

Additional Call Registers	Result Registers
CX = pixel column	AL = pixel value
DX = pixel row	
BH = page	

0E TTY character output

Additional Call Registers	Result Registers
AL = character	None
BH = page	
BL = foreground color	

Note: Writes a character to the display and updates cursor position. TTY mode indicates minimal character processing. ASCII codes for bell, backspace, linefeed, and carriage return are translated into the appropriate actions.

AH AL Function

0F XX Get video mode

Additional Call Registers	Result Registers
None	AH = width of screen in characters AL = video mode BH = active display page

Note: See Table D-2 for a list of possible video modes.

10 00 Subfunction 00H: set palette register to color correspondence

Additional Call Registers	Result Registers
AL = 00H BH = color CL = palette register (00H to 0FH)	None

10 01 Subfunction 01H: set border color

Additional Call Registers	Result Registers
AL = 01H BH = border color	None

10 02 Subfunction 02H: set palette and border

Additional Call Registers	Result Registers
AL = 02H ES:DX = address of color list	None

13 Write string

Additional Call Registers	Result Registers
AL = write mode = 00H, attribute in BL, cursor not moved = 01H, attribute in BL, cursor moved = 02H, attributes follow char, cursor not moved = 03H, attributes follow char, cursor moved ES:BP = address of string CX = character count DH = initial row position DL = initial column position BH = page	None

Note: For AL = 00 and 01, the string consists of characters only, which will all be displayed with the attribute in BL. For AL = 02 and 03, the data is stored with the attributes (char, attrib, char, attrib, and so on).

Table D-2: Video Modes and Their Definition

AL	Pixels	Characters	Char box	Text/ graph	Colors	Adapter	Max pages	Buffer start
00H	320 × 200	40 × 25	8 × 8	text	16 *	CGA	8	B8000h
	320 × 350	40 × 25	8 × 14	text	16 *	EGA	8	B8000h
	360 × 400	40 × 25	9 × 16	text	16 *	VGA	8	B8000h
	320 × 400	40 × 25	8 × 16	text	16 *	MCGA	8	B8000h
01H	320 × 200	40 × 25	8 × 8	text	16	CGA	8	B8000h
	320 × 350	40 × 25	8 × 14	text	16	EGA	8	B8000h
	360 × 400	40 × 25	9 × 16	text	6	VGA	8	B8000h
	320 × 400	40 × 25	8 × 16	text	6	MCGA	8	B8000h
02H	640 × 200	80 × 25	8 × 8	text	16 *	CGA	8	B8000h
	640 × 350	80 × 25	8 × 14	text	16 *	EGA	8	B8000h
	720 × 400	80 × 25	9 × 16	text	16 *	VGA	8	B8000h
	640 × 400	80 × 25	8 × 16	text	16 *	MCGA	8	B8000h
03H	640 × 200	80 × 25	8 × 8	text	16	CGA	8	B8000h
	640 × 350	80 × 25	8 × 14	text	16	EGA	8	B8000h
	720 × 400	80 × 25	9 × 16	text	16	VGA	8	B8000h
	640 × 400	80 × 25	8 × 16	text	16	MCGA	8	B8000h
04H	320 × 200	40 × 25	8 × 8	graph	4	CGA	1	B8000h
	320 × 200	40 × 25	8 × 8	graph	4	EGA	1	B8000h
	320 × 200	40 × 25	8 × 8	graph	4	VGA	1	B8000h
	320 × 200	40 × 25	8 × 8	graph	4	MCGA	1	B8000h
05H	320 × 200	40 × 25	8 × 8	graph	4 *	CGA	1	B8000h
	320 × 200	40 × 25	8 × 8	graph	4 *	EGA	1	B8000h
	320 × 200	40 × 25	8 × 8	graph	4 *	VGA	1	B8000h
	320 × 200	40 × 25	8 × 8	graph	4 *	MCGA	1	B8000h
06H	640 × 200	80 × 25	8 × 8	graph	2	CGA	1	B8000h
	640 × 200	80 × 25	8 × 8	graph	2	EGA	1	B8000h
	640 × 200	80 × 25	8 × 8	graph	2	VGA	1	B8000h
	640 × 200	80 × 25	8 × 8	graph	2	MCGA	1	B8000h
07H	720 × 350	80 × 25	9 × 14	text	mono	MDA	8	B0000h
	720 × 350	80 × 25	9 × 14	text	mono	EGA	4	B0000h
	720 × 400	80 × 25	9 × 16	text	mono	VGA	8	B0000h
08H	reserved							
09H	reserved							
0AH	reserved							
0BH	reserved							
0CH	reserved							
0DH	320 × 200	40 × 25	8 × 8	graph	16	EGA	2/4	A0000h
	320 × 200	40 × 25	8 × 8	graph	16	VGA	8	A0000h
0EH	640 × 200	80 × 25	8 × 8	graph	16	EGA	1/2	A0000h
	640 × 200	80 × 25	8 × 8	graph	16	VGA	4	A0000h
0FH	640 × 350	80 × 25	9 × 14	graph	mono	EGA	1	A0000h
	640 × 350	80 × 25	8 × 14	graph	mono	VGA	2	A0000h
10H	640 × 350	80 × 25	8 × 14	graph	4	EGA	1/2	A0000h
	640 × 350	80 × 25	8 × 14	graph	16	VGA	2	A0000h
11H	640 × 480	80 × 30	8 × 16	graph	2	VGA	1	A0000h
	640 × 480	80 × 30	8 × 16	graph	2	MCGA	1	A0000h
12H	640 × 480	80 × 30	8 × 16	graph	16	VGA	1	A0000h
13H	320 × 200	40 × 25	8 × 8	graph	256	VGA	1	A0000h
	320 × 200	40 × 25	8 × 8	graph	256	MCGA	1	A0000h

* color burst off

SECTION D.4: INT 12H

Get conventional memory size

Call Registers	Result Registers
None	AX = memory size (KB)

Note: Returns amount of conventional memory.

SECTION D.5: INT 14H

AH Function

00 Initialize COM port

Additional Call Registers	Result Registers
AL = parameter (see below)	AH = port status (see below)
DX = port number (0 if COM1, 1 if COM2, etc.)	AL = modem status (see below)

Note 1: The parameter byte in AL is defined as follows:

```
7 6 5 4 3 2 1 0   Indicates
x x x             Baud rate (000=110, 001=150,
                  010=300, 011=600, 100=1200,
                  101=2400, 110=4800, 111=9600)
      x x         Parity (01=odd, 11=even, x0=none)
          x       Stop bits (0 = 1, 1 = 2)
            x x   Word length (10=7 bits, 11=8 bits)
```

Note 2: The port status returned in AH is defined as follows:

```
7 6 5 4 3 2 1 0   Indicates
1                 Timed-out
  1               Transmit shift register empty
    1             Transmit holding register empty
      1           Break detected
        1         Framing error detected
          1       Parity error detected
            1     Overrun error detected
              1   Received data ready
```

Note 3: The modem status returned in AL is defined as follows:

```
7 6 5 4 3 2 1 0   Indicates
1                 Received line signal detect
  1               Ring indicator
    1             DSR (data set ready)
      1           CTS (clear to send)
        1         Change in receive line signal detect
          1       Trailing edge ring indicator
            1     Change in DSR status
              1   Change in CTS status
```

AH Function

01 Write character to COM port

Additional Call Registers	Result Registers
AL = character	AH bit 7 = 0 if successful, 1 if not
DX = port number (0 if COM1,	AH bits 0–6 = status if successful
1 if COM2, etc.)	AL = character

Note: The status byte in AH, bits 0–6, after the call is as follows:

```
6 5 4 3 2 1 0     Indicates
1                 Transmit shift register empty
  1               Transmit holding register empty
    1             Break detected
      1           Framing error detected
        1         Parity error detected
          1       Overrun error detected
            1     Receive data ready
```

02 Read character from COM port

Additional Call Registers	Result Registers
DX = port number (0 if COM1,	AH bit 7 = 0 if successful, 1 if not
1 if COM2, etc.)	AH bits 0–6 = status if successful
	AL = character read

Note: The status byte in AH, bits 1–4, after the call is as follows:

```
4 3 2 1     Indicates
1           Break detected
  1         Framing error detected
    1       Parity error detected
      1     Overrun error detected
```

03 Read COM port status

Additional Call Registers	Result Registers
DX = port number (0 if COM1,	AH = port status
1 if COM2, etc.)	AL = modem status

Note: The port status and modem status returned in AH and AL are the same format as INT 14H function 00H, described above.

04 Extended initialize COM port

Additional Call Registers	Result Registers
AL = 00H (break),	AH = port status (see function AH = 0)
01H (no break)	AL = modem status (see function AH = 0)
DX = port number (0 if COM1,	
1 if COM2, etc.)	
BH = parity = 00H none	
= 01H odd	
= 02H even	

= 03H stick parity odd
= 04H stick parity even
BL = stop bits = 00H (one stop bit)
= 01H (1.5 bits for 5-bit word)
= 01H (2 bits for > 5-bit word)
CH = word length = 00H 5-bit
= 01H 6-bit
= 02H 7-bit
= 03H 8-bit
CL = baud rate = 00H 110 baud
= 01H 150 baud
= 02H 300 baud
= 03H 600 baud
= 04H 1200 baud
= 05H 2400 baud
= 06H 4800 baud
= 07H 9600 baud
= 08H 19200 baud

05 Extended COM port control

Additional Call Registers	Result Registers
AL = 00H (read control register),	If read subfunction,
= 01H (write to control register)	BL = modem control register
DX = port number (0 if COM1, 1 if COM2, etc.)	If write subfunction,
	AL = modem status
BL = Modem control register	(see Figure 9-16)
(see Figure 9-14)	AH = line status
bits 7–5: reserved	(see Figure 9-15)
bit 4: loop	
bit 3: out2	
bit 2: out1	
bit 1: RTS	
bit 0: DTR	

Note: Subfunction AL = 00H returns the modem control register contents in BL. Subfunction AL = 01H writes the contents of BL into the modem control register and returns modem and line status register contents in AL and AH.

SECTION D.6: INT 16H -- KEYBOARD

AH Function

00H Keyboard read

Additional Call Registers	Result Registers
None	AH = key scan code
	AL = ASCII char

Note: Reads one character from the keyboard buffer and updates the head pointer.

AH Function

01H Get keyboard status

Additional Call Registers	Result Registers
None	If no key waiting, ZF = 1. If key waiting, ZF = 0, AH = key scan code, AL = ASCII char.

Note: If a key is waiting, the scan code and character are returned in AH and AL, but the head pointer of the keyboard buffer is not updated.

02H Get shift status

Additional Call Registers	Result Registers
None	AL = status byte bit 7: Insert pressed bit 6: Caps Lock pressed bit 5: Num Lock pressed bit 4: Scroll Lock pressed bit 3: Alt pressed bit 2: Ctrl pressed bit 1: Left Shift pressed bit 0: Right Shift pressed

Note: The keyboard status byte returned in AL indicates whether certain keys have been pressed. If the bit = 1, the key has been pressed.

03H Set typematic rate

Additional Call Registers	Result Registers
AL = 05H BH = repeat delay (see below) BL = repeat rate (see below)	None

Note: Sets the rate at which repeated keystrokes are accepted.

The delay value in BH can be 00H (for 250), 01H (for 500), 02H (for 750), or 03H (for 1000). All values are in milliseconds. The repeat rate in BL represents the number of characters per second. Options are:

```
00H: 30.0   0BH: 10.9   16H: 4.3
01H: 26.7   0CH: 10.0   17H: 4.0
02H: 24.0   0DH: 9.2    18H: 3.7
03H: 21.8   0EH: 8.6    19H: 3.3
04H: 20.0   0FH: 8.0    1AH: 3.0
05H: 18.5   10H: 7.5    1BH: 2.7
06H: 17.1   11H: 6.7    1CH: 2.5
07H: 16.0   12H: 6.0    1DH: 2.3
08H: 15.0   13H: 5.5    1EH: 2.1
09H: 13.3   14H: 5.0    1FH: 2.0
0AH: 12.0   15H: 4.6    20H to FFH - reserved
```

AH Function

10H Extended keyboard read

Additional Call Registers	Result Registers
None	AH = key scan code AL = ASCII char

Note: Used in place of INT 16H function 00H to allow program to detect F11, F12, and other keys of the extended keyboard. After the read, the head pointer of the keyboard buffer is updated.

11H Extended keyboard status

Additional Call Registers	Result Registers
None	If no key waiting, ZF = 1. If key waiting, ZF = 0, AH = key scan code, AL = ASCII char.

Note: This function is used instead of INT 16H function 01H so that programs can detect keys of the extended keyboard such as F11 and F12. If a key is waiting, the scan code and character are returned in AH and AL, but the head pointer of the keyboard buffer is not updated.

12H Extended shift status

Additional Call Registers	Result Registers
None	AL = shift status bit 7: Insert locked bit 6: Caps Lock locked bit 5: Num Lock locked bit 4: Scroll Lock locked bit 3: Alt pressed bit 2: Ctrl pressed bit 1: Left Shift pressed bit 0: Right Shift pressed AH = extended shift status bit 7: SysRq pressed bit 6: Caps Lock pressed bit 5: Num Lock pressed bit 4: Scroll Lock pressed bit 3: Right Alt pressed bit 2: Right Ctrl pressed bit 1: Left Alt pressed bit 0: Left Ctrl pressed

Note: The keyboard status bytes returned in AL and AH indicate whether certain keys have been pressed. If the bit = 1, the key has been pressed.

SECTION D.7: INT 1AH

AH Function

00H Read system-timer time counter

Additional Call Registers	Result Registers
None	CX = high portion of count
	DX = low portion of count
	AL = 0 if 24 hours have not passed since last read
	> 0 if 24 hours have passed since last read

Note: This function returns the number of ticks since midnight. A second is about 18.2 ticks. When the number of ticks indicates that 24 hours have passed, AL is incremented and the tick count is reset to zero. Calling this function resets AL so that whether 24 hours have passed can only be determined once a day.

01H Set system-timer time counter

Additional Call Registers	Result Registers
CX = high portion of tick count	None
DX = low portion of tick count	

Note: Calling this function will cause the timer overflow flag to be reset.

02H Read real-time clock time

Additional Call Registers	Result Registers
None	CH = hours
	CL = minutes
	DH = seconds
	DL = 01 for daylight savings option
	= 00 for no option
	CF = 0 if clock operating, otherwise = 1

Note: Hours, minutes, and seconds are returned in BCD format. This function is used to get the time in the CMOS time/date chip.

03H Set real-time clock time

Additional Call Registers	Result Registers
CH = hours	None
CL = minutes	
DH = seconds	
DL = 01 for daylight savings option	
= 00 for no option	

Note: Hours, minutes, and seconds are in BCD format. This function is used to set the time in the CMOS time/date chip.

AH Function

04H Read real-time clock date

Additional Call Registers	Result Registers
None	CH = century (19 or 20) CL = year DH = month DL = day CF = 0 if clock operating, otherwise = 1

Note: Century, year, month, and day are in BCD format. This function is used to get the date in the CMOS time/date chip.

05H Set real-time clock date

Additional Call Registers	Result Registers
CH = century (19 or 20) CL = year DH = month DL = day	None

Note: Century, year, month, and day are in BCD format. This function is used to set the date in the CMOS time/date chip.