In this appendix, we show some of the common BIOS and DOS interrupt calls. We begin with interrupt 10h; interrupts 0 to Fh are not normally used by application programs, their names are given in Table C.1.

**C.2 BIOS Interrupts**

**Interrupt 10h: Video**

**Function 0h: Select Display Mode**
Selects video display mode

- **Input:**
  - \( AH = 0h \)
  - \( AL = \) video mode

- **Output:**
  - none

**Function 1h: Change Cursor Size**
Selects the start and ending lines for the cursor.

- **Input:**
  - \( AH = 1h \)
  - \( Cl. (bits \ 0-4) = \) starting line for cursor
  - \( Cl. (bits \ 0-4) = \) ending line for cursor

- **Output:**
  - none
### Table C.1 Interrupts 0 to Df

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h</td>
<td></td>
<td>Divide by zero</td>
</tr>
<tr>
<td>1h</td>
<td></td>
<td>Single step</td>
</tr>
<tr>
<td>2h</td>
<td></td>
<td>NM</td>
</tr>
<tr>
<td>3h</td>
<td></td>
<td>Breakpoint</td>
</tr>
<tr>
<td>4h</td>
<td></td>
<td>Overflow</td>
</tr>
<tr>
<td>5h</td>
<td></td>
<td>PrintScreen</td>
</tr>
<tr>
<td>6h</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>7h</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>8h</td>
<td></td>
<td>Timer tick</td>
</tr>
<tr>
<td>9h</td>
<td></td>
<td>Keyboard</td>
</tr>
<tr>
<td>Ah</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0Ah</td>
<td></td>
<td>Serial communications (COM2)</td>
</tr>
<tr>
<td>0Ch</td>
<td></td>
<td>Serial communications (COM1)</td>
</tr>
<tr>
<td>0Dh</td>
<td></td>
<td>Fixed disk</td>
</tr>
<tr>
<td>0th</td>
<td></td>
<td>Floppy disk</td>
</tr>
<tr>
<td>ofh</td>
<td></td>
<td>Parallel minter</td>
</tr>
</tbody>
</table>

**Function 2h:**

**Move Cursor**

Positions the cursor.

**Input:**

- \( AH = 2h \)
- \( BH = \) page
- \( DH = \) row
- \( DL = \) column

**Output:** none

**Function 3h:**

**Get Cursor Position and Size**

Obtains the current position and size of the cursor.

**Input:**

- \( AH = 3h \)
- \( BL = \) page

**Output:**

- \( CH = \) starting line for cursor
- \( CL = \) ending line for cursor
- \( DH = \) row
- \( DL = \) column

**Function 5h:**

**Select Active Display Page**

**Input:**

- \( AH = 5h \)
- \( AL = \) page
- \( DH = \) row
- \( DL = \) column

**Output:** none

**Function 6h:**

**Scroll Window Up**

Scrolls the entire screen or a window up by a specified number of lines.

**Input:**

- \( AH = 6h \)
- \( AL = \) number of lines to scroll
- \( BH = \) attribute for blanked lines

- (If zero, entire window is blanked)
Function 7h: Scroll Window Down
Scrolls the entire screen or a window down by a specified number of lines.

Input:
- \( AH = 7h \)
- \( AL = \) number of lines to scroll
  - (If zero, entire window is blanked)
- \( AL-1 = \) attribute for blanked lines
- \( CH, CL = \) row, column of upper left corner of window
- \( DH, DL = \) row, column of lower right corner of window

Output: none

Function 8h: Read Character and Attribute at Cursor
Obtains the ASCII character and its attribute at the cursor position.

Input:
- \( AH = 8h \)
- \( BH = \) page

Output:
- \( AH = \) attribute
- \( AI = \) character

Function 9h: Write Character and Attribute at Cursor
Writes an ASCII character and its attribute at the cursor position.

Input:
- \( AH = 9h \)
- \( AL = \) character
- \( BH = \) page
- \( BL = \) attribute (text mode) or color (graphics mode)
- \( CX = \) count of characters to write

Output: none

Function 0Ah: Write Character at Cursor
Writes an ASCII character at the cursor position. The character receives the attribute of the previous character at that position.

Input:
- \( AH = 0Ah \)
- \( AL = \) character
- \( BH = \) page
- \( CX = \) count of characters to write

Output: none

Function 0Bh: Set Palette, Background, or Border
Selects a palette, background color, or border color.

Input:
- To select the background color and border color
  - \( AH = 0Bh \)
  - \( BH = 0 \)
  - \( BL = \) color
- To select palette (320 x 200 four-color mode)
  - \( AH = 0Bh \)
  - \( BH = 1 \)
  - \( BL = \) palette

Output: none
Function 0Ch:
Write Graphics Pixel
Input: AH = 0Ch
AL = pixel value
BH = page
CX = column
DX = row
output: none

Function 0Dh:
Read Graphics Pixel
Obtains a pixel value.
Input: AH = 0Dh
BH = page
CX = column
DX = row
output: AL = pixel value

Function 0Eh:
Write Character in Teletype Mode
Writes an ASCII character at the cursor position, then increments cursor position.
Input: AH = 0Eh
AL = character
BH = page
BL = color (graphics mode)
Output: none
Note: the attribute of the character cannot be specified.

Function 0Fh:
Get Video Mode
Obtains current display mode.
Input: AH = 0Fh
output: AH = number of character columns
AL = display mode
BH = active display page

Function 10h, Subfunction 10h:
Set Color Register
Sets individual VGA color register.
Input: AH = 10h
AL = 10h
BX = color register
Cl = green value
Cl = blue value
DH = red value
Output: none

Function 10h, Subfunction 12h:
Set Block of Color Registers
Sets a group of VGA color registers.
Input: AH = 10h
AL = 12h
BX = firstcolor register
CX = number of color registers
ES:DX = segment:offset of color table
Output: none
Note: the table consists of a group of three-byte entries corresponding to red, green, and blue values for each color register.

Function 10h, Subfunction 15h: Get Color Register
Obtains the red, green, and blue values of a VGA color register.
Input: AH = 10h
AL = 15h
RX = color register
Output: CL = green value
CL = blue value
DH = red value

Function 10h, Subfunction 17h: Get Block of Color Registers
Obtains the red, green, and blue values of a group of VGA color registers.
Input: AH = 10h
AL = 17h
RX = first color register
CX = number of color registers
ES:DX = segment:offset of buffer to receive color list
Output: ES:DX = segment:offset of buffer
Note: the color list consists of a group of three-byte entries corresponding to red, green, and blue values for each color register.

Interrupt 11h: Get Equipment Configuration
Obtains the equipment list code word.
Input: none
Output: AX = equipment list code word
(bits 14-15 = number of printers installed,
13 = internal modem,
12 = game adapter,
9-11 = number of serial ports,
8 is reserved,
6-7 = number of floppy disk drives,
4-5 = initial video mode,
2-3 = system board RAM size, original PC used by PS/2,
1 = math coprocessor,
0 = floppy disk installed)

Interrupt 12h: Get Conventional Memory Size
Returns the amount of conventional memory.
Input: none
Output: AX = memory size (in KB)

Interrupt 13h: Disk I/O
Function 2h: Read Sector
Reads one or more sectors.
Input: AH = 2h
AL = number of sectors
CH = cylinder
CL = sector
DH = head
DL = drive (0-7Fh = floppy disk, 80h-FFh = fixed disk)
ES:BX = segment:offset of buffer

output:
If function successful
CF = clear
AH = 0
AL = number of sectors transferred
If function unsuccessful
CF = set
AH = error status

Function 3h: Write Sector
Writes one or more sectors.
Input: AH = 3h
AL = number of sectors
BX = firstcolor register
CH = cylinder
CL = sector
DH = head
DL = drive (0-7Fh = floppy disk, 80h-FFh = fixed disk)
ES:BX = segment:offset of buffer
Output:
If function successful
CF = clear
AH = 0
AL = number of sectors transferred
If function unsuccessful
CF = set
AH = error status

Interrupt 15h: Cassette I/O and Advanced features
for AT, PS/2

Function 87h: Move Extended Memory Block
Transfers data between conventional memory and extended memory.
Input: AH = 87h
CX = number of words to move
ES:SI = segment:offset of Global Descriptor Table
output:
If function successful
CF = clear
AH = 0
AL = number of sectors transferred
If function unsuccessful
CF = set
AH = error status

Function 88h: Get Extended Memory Size
Obtains amount of extended memory
Input: AH = 88h
Output: AX = extended memory size (in KB)
Interrupt 16h: Keyboard

Function 0h: Read Character from Keyboard
Input: \(AH = 0h\)
Output: \(AH = \) keyboard scan code
\(AL = \) ASCII character

Function 2h: Get Keyboard Flags
Obtains key flags that describe the status of the function keys.
Input: \(AH = 2h\)
Output: \(AL = \) flags

<table>
<thead>
<tr>
<th>Bit</th>
<th>If Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Insert on</td>
</tr>
<tr>
<td>6</td>
<td>Caps Lock on</td>
</tr>
<tr>
<td>5</td>
<td>Num Lock on</td>
</tr>
<tr>
<td>4</td>
<td>Scroll Lock on</td>
</tr>
<tr>
<td>3</td>
<td>(Alt) key is down</td>
</tr>
<tr>
<td>2</td>
<td>Ctrl key is down</td>
</tr>
<tr>
<td>1</td>
<td>left shift key is down</td>
</tr>
<tr>
<td>0</td>
<td>right shift key is down</td>
</tr>
</tbody>
</table>

Function 10h: Read Character from Enhanced Keyboard
Input: \(AH = 0h\)
Output: \(AH = \) keyboard scan code
\(AL = \) ASCII character

Note: this function can be used to return scan codes for control keys such as \(F1\) and \(F12\).

Interrupt 17h: Printer

Function 0h: Write Character to Printer
Input: \(AL = 0\)
\(AL = \) character
\(DX = \) printer number
Output: \(AH = \) status

<table>
<thead>
<tr>
<th>Bit</th>
<th>If Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>printer not busy</td>
</tr>
<tr>
<td>6</td>
<td>printer acknowledge</td>
</tr>
<tr>
<td>5</td>
<td>out of paper</td>
</tr>
<tr>
<td>4</td>
<td>printer selected</td>
</tr>
<tr>
<td>3</td>
<td>(VO) error</td>
</tr>
<tr>
<td>2</td>
<td>unused</td>
</tr>
<tr>
<td>1</td>
<td>unused</td>
</tr>
<tr>
<td>0</td>
<td>printer timed out</td>
</tr>
</tbody>
</table>
c.3
DOS Interrupts

Interrupt 21h

Function Oh:
Program Terminate
Terminates the execution of a program.
Input: \( AH = 0h \)
\( CS = \) segment of PSP
Output: none

Function 1h:
Keyboard Input
Waits for a character to be read at the standard input device (unless one is ready), then echoes the character to the standard output device and returns the ASCII code in AL.
Input: \( AH = 01h \)
Output: \( AL \) = character from the standard input device

Function 2h:
Display Output
Outputs the character in DL to the standard output device.
Input: \( AH = 02h \)
\( DL \) = character
Output: none

Function 5h:
Printer Output
Outputs the character in DL to the standard printer device.
Input: \( AH = 05h \)
\( DL \) = character
Output: none

Function 09h:
Print String
Outputs the characters in the print string to the standard output device.
Input: \( AH = 09h \)
\( DS:DX \) = pointer to the character string ending with 'S'
Output: none

Function 2Ah:
Get Date
Returns the day of the week, year, month and date.
Input: \( AH = 2Ah \)
Output: \( AL \) = Day of the week (0=SUN, 6=SAT)
\( CX \) = Year (1980-2099)
\( DH \) = Month (1-12)
\( DL \) = Day (1-31)

Function 2Bh:
Set Date
Sets the date.
Input: \( AH = 2Bh \)
\( CX \) = year (1980-2099)
\( DL \) = month (1-12)
\( DL \) = day (1-31)
Output:  
AL = 00h, if the date is valid  
FFh, if the date is not valid  

**Function 2Ch:**  
**Get time**  
Returns the time: hours, minutes, seconds and hundredths of seconds.  
Input:  
AH = 2Ch  
output:  
CH = hours (0-23)  
CL = minutes (0-59)  
DH = seconds (0-59)  
DL = hundredths (0-99)  

**Function 2Dh:**  
**Set Time**  
Sets the time.  
Input:  
AH = 2Dh  
CH = Hours (0-23)  
DH = Seconds (0-59)  
CL = Minutes (0-59)  
DL = Hundredths (0-99)  
Output:  
AL = 00h if the time is valid  
FFh if the time is not valid  

**Function 30h:**  
**Get DOS Version Number**  
Returns the DOS version number.  
Input:  
AH = 30h  
output:  
BX = 0000h  
cx = 0000h  
AL = major version number  
AL1 = minor version number  

**Function 31h:**  
**Terminate Process and Remain Resident**  
Terminates the current process and attempts to set the initial allocation block to the memory size in paragraphs.  
Input:  
AH = 31h  
AL = return code  
output: none  

**Function 33h:**  
**Ctrl-break Check**  
Set or get the state of BREAK (Ctrl-break checking).  
Input:  
AH = 33h  
AL = 00h, to request current state  
01h, to set the current state  
00h, to set current state OFF  
01h, to set current state ON  
Output:  
DL = The current state (00h=OFF, 01h=ON)  

**Function 35h:**  
**Get Vector**  
Obtains the address in an interrupt vector.  
Input:  
AH = 35h
AL = interrupt number
output: ES:BX = pointer to the interrupt handling routine.

Function 36h:
Get Disk Free Space
Returns the disk free space (available clusters, clusters/drive, bytes/sector).
Input:  
\[ AH = 36h \]
\[ DL = \text{drive @default, } 1=\text{A} \]
Output:  
\[ BX = \text{Available clusters} \]
\[ DX = \text{clusters/drive} \]
\[ CX = \text{bytes/sector} \]
\[ AX = \text{FFFFh if the drive in DL is invalid, otherwise the number of sectors per cluster} \]

Function 39h:
Create Subdirectory (MKDIR)
Creates the specified directory.
Input:  
\[ AH = 39h \]
\[ DS:DX = \text{pointer to an ASCII string} \]
Output:  
\[ AX = \text{error codes if carry flag is set} \]

Function 3Ah:
Remove Subdirectory (RMDIR)
Removes the specified directory.
Input:  
\[ AH = 3Ah \]
\[ DS:DX = \text{pointer to an ASCII string} \]
Output:  
\[ AX = \text{error codes if carry flag is set} \]

Function 3Bh:
Change the Current Directory(CHDIR)
Changes the current directory to the specified directory.
Input:  
\[ AH = 3Bh \]
\[ DS:DX = \text{pointer to an ASCII string} \]
Output:  
\[ AX = \text{error codes if carry flag is set} \]

Function 3Ch:
Create a File (CREAT)
Creates a new file or truncates an old file to zero length in preparation for writing.
Input:  
\[ AH = 3Ch \]
\[ DS:DX = \text{pointer to an ASCII string} \]
\[ CX = \text{attribute of the file} \]
Output:  
\[ AX = \text{error codes if carry flag is set} \]
\[ 16\text{-bit handle if carry flag not set} \]

Function 3Dh:
Open a File
Opens the specified file.
Input:  
\[ AH = 3Dh \]
\[ DS:DX = \text{pointer to an ASCII path name} \]
\[ AL = \text{access Code} \]
Output:  
\[ AX = \text{error codes if carry flag is set} \]
\[ 16\text{-bit handle if carry flag not set} \]
Function 3Eh:
Close a File Handle
Closes the specified file handle.
Input:  \text{AH} = 3Eh
RX = file handle returned by open or create
Output: AX = error codes if carry flag set
none if carry flag not set

Function 3Fh:
Read from a File or Device
Transfers the specified number of bytes from a file into a buffer location.
Input:  \text{AH} = 3Fh
BX = file handle
DS:DX = buffer address
cx = number of bytes to be read
Output: AX = number of bytes read
error codes if carry flag set

Function 40h:
Write to a File or Device
Transfers the specified number of bytes from a buffer into a specified file.
Input:  \text{AH} = 40h
BX = file handle
DS:DX = address of the data to write
cx = number of bytes to write
Output: AX = number of bytes written
error codes if carry flag set

Function 41h:
Delete a File from a Specified Directory (UNLINK)
Removes a directory entry associated with a file name.
Input:  \text{AH} = 41h
DS:DX = address of an ASCII string
Output: AX = error codes if carry flag set
none if carry flag not set

Function 42h:
Move File Read Write Pointer (LSEEK)
Moves the read/write pointer according to the method specified.
Input:  \text{AH} = 4211
CS:DX = distance (offset) to move in bytes
AI = method of moving (0,1,2)
RX = file handle
Output: AX = error codes if carry flag set
DX:AX = new pointer location if carry flag not set

Function 47h:
Get Current Directory
Places the full path name (starting from the root directory) of the current directory for the specified drive in the area pointed to by DS:SI.
Input:  \text{AH} = 47h
DS:SI = pointer to a 64-byte user memory area
DI = drive number (0=default, 1=A, etc.)
error codes if carry flag set
Function 48h: 
Allocate Memory
Allocates the requested number of paragraphs of memory.
Input: Al = 48h
RX = number of paragraphs of memory requested
output: AX:0 = points to the allocated memory block
AX = error codes if carry flag is set
BX = size of the largest block of memory available (in paragraphs) if the allocation fails

Function 49h: 
Free Allocated Memory
Frees the specified allocated memory.
Input: Al = 49h
SX = segment of the block to be returned
output: AX = error codes if carry flag set
none if carry flag not set

Function 4Ch: 
Terminate a Process (EXIT)
Terminates the current process and transfers control to the invoking process.
Input: Al = 4Ch
AL = return code
Output: none

Interrupt 25h: Absolute Disk Read
Input: Al = drive number
CX = number of sectors to read
DX = beginning logical sector number
DS:BX = transfer address
output: If successful CF = 0
If unsuccessful CF = 1 and AX contains error code

Interrupt 26h: Absolute Disk Write
Input: Al = drive number
CX = number of sectors to read
DX = beginning logical sector number
DS:BX = transfer address
Output: If successful CF = 0
If unsuccessful CF = 1 and AX contains error code

Interrupt 27h: Terminate but Stay Resident
Input: DX = offset of beginning of free space, segment is with respect to PSI
output: none