10

Ahead V5000
Ahead VGA
Wizard/Deluxe

Introduction

Ahead Systems, Inc. designed the V5000 VGA chip for use on their VGA Wizard/Deluxe display adapters. At this time, two versions of the chip have been made (versions A and B). As with most SuperVGAs, the Ahead V5000 VGA chips are fully IBM VGA-compatible, include register level compatibility for EGA, CGA, MDA and Hercules, and include extended high resolution text and graphics modes. High resolution applications software drivers are also available. Ahead Systems has captured the distinction of being the first company to ship a VGA product in volume that supports 1024x768 resolution with 256 colors. Wizard/Deluxe was selected as 1990 Video Board of the Year by InfoWorld magazine.

Version B of the V5000 VGA chip contains features that are not available in version A, which is no longer being produced. Information given in this chapter applies to version B only unless stated otherwise.

Chip Versions

Ahead V5000 VGA chips contain a version number that can be read from the least significant nibble of the Master Enable Register (I/O address 3CFh, index 0Fh). See section "Detection and Identification" for details on how the chip version can be determined.

New Display Modes

Table 10-1 lists the enhanced display modes that are supported by the Ahead VGA Wizard/Deluxe.

Memory Organization

For all extended display modes of the VGA Wizard/Deluxe, display memory organization is closely patterned after standard IBM VGA display modes.

For some extended modes, a memory paging mechanism is also used. Memory paging is described in detail in the programming examples.

High Resolution Text Modes

These modes utilize memory maps that are similar to those used in standard text modes (modes 0,1,2,3, and 7), except that the number of characters per line, or number of lines per screen, is increased. Display memory is organized as shown in Figure 5-1 (see Chapter 5).

Table 10-1 Enhanced display modes—Ahead VGA Wizard/Deluxe

Mode	Type	Resolution	Colors	Memory Required	Display Type
22h	Text	132 col x 44 rows	16	256 KB	EGA
23h	Text	132 col x 25 rows	16	256 KB	EGA
24h	Text	132 col x 28 rows	16	256 KB	EGA
2Fh	Text	160 col x 50 rows	16	256 KB	EGA
34h	Text	80 col x 66 rows	16	256 KB	Super VGA
50h	Text	132 col x 25 rows	Mono	256 KB	MDA
52h	Text	132 col x 44 rows	Mono	256 KB	MDA
25h,26h	Graphics	640x480	16	256 KB	VGA
60h	Graphics	640x400	256	256 KB	VGA
61h	Graphics	640x480	256	512 KB	VGA
62h	Graphics	800x600	256	512 KB	Super VGA
63h	Graphics	1024x768	256	1024 KB	8514
6Ah,71h	Graphics	800x600	16	256 KB	Super VGA
70h	Graphics	720x396	16	256 KB	Super VGA
74h	Graphics	1024x768	16	512 KB	8514
75h	Graphics	1024x768	4	512 KB	8514
76h	Graphics	1024x768	Mono	512 KB	8514

2-Color Graphics Mode

Memory organization for this mode resembles VGA mode 11h (640x350 2-color graphics) except that both the number of pixels per scan line and the number of scan lines are increased, and mode 76h requires paging.

4-Color Graphics Mode

Memory organization for this mode does not closely resemble any standard VGA modes; it somewhat resembles planar graphics mode 12h except that the memory planes are utilized differently. Planes 0 and 2 are used to store bytes at even host memory addresses. Planes 1 and 3 are used to store bytes at odd host memory addresses. See "Four Planes" in Chapter 9 to learn more about this type of memory organization.

16-Color Graphics Modes

Memory organization for these modes resembles VGA mode 12h (640x480 16-color graphics), except that both the number of pixels per scan line and the number of scan lines are increased. Mode 74h (1024x768 16-color graphics) requires display memory

paging. Display memory organization is shown in Figure 7-1. See Chapter 7 for programming examples.

256-Color Graphics Modes

Memory organization for these modes resembles VGA mode 13h (320x200 256-color graphics), except that both the number of pixels per scan line and the number of scan lines are increased, and extended modes require paging. Display memory organization is shown in Figure 8-1. See Chapter 8 for programming examples.

New Registers

Several new registers have been added to the V5000 chip to control display memory paging and CGA/EGA/MDA emulation modes. This extended register set resides in the address space of the Graphics Controller (I/O address 3CEh/3CFh) starting at index 0Ch. Table 10-2 contains a list of the registers in the extended register set; the programming examples in this chapter contain examples showing how to access the extended registers.

Table 10-2. Extended register set

Address	Index	Description	
3CEh/3CFh	0Ch	Mode	D7,D6 = Emulation mode 11 CGA 10 Hercules 01 EGA 00 VGA D5 = Enhanced mode enable D4 = 16 bit memory access enable D3 = High speed sequencer enable D2 = Reserved D1,D0 = Miscellaneous control 11 Reserved 10 Reserved 01 Enable 8 simultaneous fonts 00 Standard text mode
	0Dh	Segment	D4-D7 = Write page D0-D3 = Read page
	0Eh	Clock	D4-D7 = Divide input clock 0-3 by 2 D1-D3 = Reserved D0 = Clock 4 & 5 select enable
	0Fh	Master Enable	D5 = Extended register access enable D0-D3 = Chip revision (READ ONLY)

Table 10-2. Extended register set *(continued)*

Address	Index	Description	
	10h	Trap	D7 = Select 6845 as CRT controller
			D5 = Enable 3Cxh to cause traps
			D4 = Enable 3D8h, 3D9h to cause trap
			D3 = Enable 3B8h, 3BFh to cause trap
			D2 = Enable CRTC access to cause trap
			D1 = Enable 6845 access
			D0 = Enable CRTC access
	11h	Trap Source	D6-D7 = Reserved
			D5 = 3Cxh
			D4 = 3BFh
			D3 = 3D9h
			D2 = 3B8h, 3D8h
			D1 = 3B5h, 3D5h
			D0 = 3Dxh
	12h	Attribute	D7 = Enable CGA palette when in CGA mode
			D6 = Lock VGA internal palette
			D0-D5 = Reserved
	13h	Diagnostics	D0-D7 = Reserved
	14h	Lock	D7 = Lock clock select in 3C2h
			D6 = Lock CRTC index 13h
			D5 = Lock CRTC index 0Ah, 0Bh
			D4 = Lock CRTC index 9
			D3 = Lock CRTC index 9
			D2 = Lock CRTC vertical timing
			D1 = Lock CRTC horizontal timing
			D0 = Lock sync polarity in 3C2
	15h	3B8h, 3D8h Readback	
	16h	3BFh, 3D9h Readback	D0-D5 = 3D9h
			D6-D7 = 3BFh bits 0 & 1
	17h	Miscellaneous	D2-D7 = Reserved
			D1 = Must be 0
			D0 = Must be 1
	1Ch	CRTC Control	D6-D7 = Reserved
			D5 = Enable double scan
			D4 = Reserved
			D2-D3 = 00 Normal
			01 Reserved
			10 Reserved
			11 Interlaced mode
			D1 = Start address bit 17
			D0 = Start address bit 16
	1Dh	Control	D0-D7 = Reserved

Table 10-2. Extended register set *(continued)*

Address	Index	Description	
	1Eh	Scratch	Used by BIOS for flags
	1Fh	PowerUp (Read Only)	
			D4-D7 = Multiple Chip ID
			0000 - ID 0, BIOS enabled
			0001 - ID 1, BIOS enabled
			0002 - ID 2, BIOS disabled
			. . .
			1111 - ID 15, BIOS disabled
			D3 = 16-bit BIOS
			D2 = 0 for 24k BIOS, 1 for 32k BIOS
			D0-D1 = Memory type
			00 - 2 44256 DRAMs
			01 - 4 or 16 44256 DRAMs
			10 - 8/16 4464 DRAMs
			11 - 8 44256 DRAMs
46E8h		Setup Control register	
			D5-D7 = Reserved
			D4 = 0 for Setup Mode, 1 for Normal Mode
			D3 = 0 for VGA disabled, 0 for VGA disabled
103h		Multiple chip ID register	
			D0-D3 = Must match Power Up register bits 0-3
			V5000 allows up to 16 chips
			VGA Wizard/Delux allows up to 4 boards in one system

Note: Bits marked 'reserved' must be preserved when modifying register contents.

Most registers in the extended register bank are generally not useful to the applications programmer. Listed below are the registers that we found useful enough to use in the programming examples.

Master Enable Register (I/O Address 3CFh Index 0Fh)

D7,D6 - reserved
D5 - Extended Register Access Enable (1 = enabled)
D4 - reserved
D3-D0 - Chip Revision (read only)

Extended Register Access Enable must be true before any other registers in the extended register bank can be accessed.

This bit is normally set for extended graphics modes by the BIOS mode select function.

Memory Page Select Register (I/O Address 3CFh Index 0Dh)

D7-D4 - Write page select
D3-D0 - Read page select

Programming Examples

Display Memory Paging

The Page Select register, located in the extended register bank at I/O address 3CFh Index 0Dh, selects which page of display memory is enabled. Two display memory pages may be selected simultaneously, one for reading and one for writing. Both pages reside at the same host memory address (normally A000:0). Dual page capability is useful when transferring data from one part of display memory to another, as for on-screen to on-screen BITBLT operations (see the BITBLT programming examples).

Figure 10-1 shows the format of the Page Select register. The read and write page may be set to the same value to achieve one memory page that is both readable and writable.

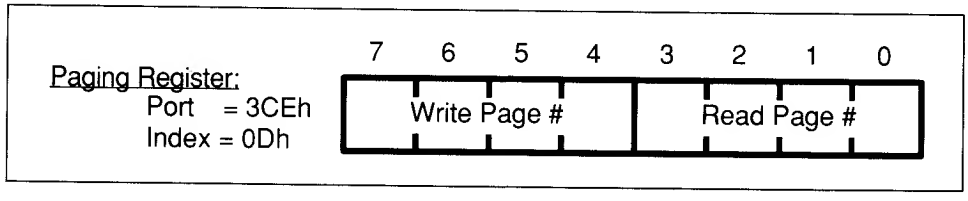

Figure 10-1. Page Select register format—V5000 Version B

Version A of the V5000 chip does not support dual memory pages; only one page is available. Page selection is not as straightforward as it is version B. A memory page is selected using bits 0-2 at 3CEh index 0Dh. The page can be enabled for writing using bit 5 at 3C2h and/or enabled for reading using register 3CCh. This is illustrated in Figure 10-2.

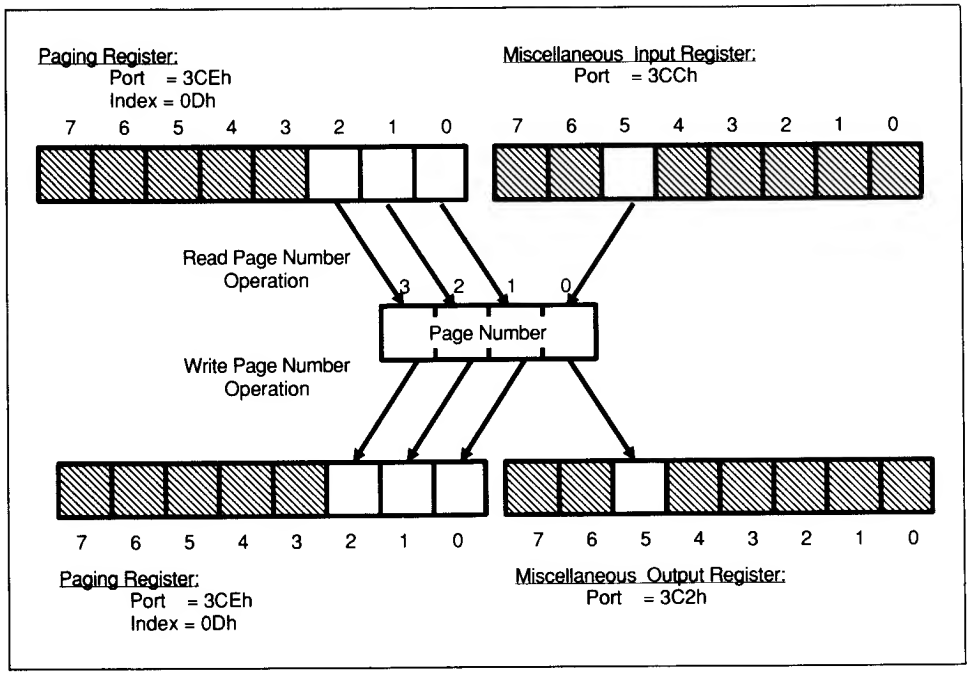

Figure 10-2. Page Select register format—V5000 Version A

For all graphics display modes except 256-color modes, a byte of display memory contains multiple pixels. Many drawing algorithms can be implemented efficiently by using a 'moving mask' to modify partial bytes. The BIT MASK register, index 8 of the Graphics Controller, is selected at the start of the algorithm:

```
MOV         DX,3CEh
MOV         AL,8
OUT         DX,AL
INC         DX
```

Inside the drawing loop of the algorithm, the mask data can be updated without rewriting the Index register:

```
MOV         AL,Mask
OUT         DX,AL
```

Since the Ahead Paging register resides at the same I/O address as the Graphics Controller, care must be taken to ensure that after a new page is selected, the BIT MASK register index is restored.

Display memory paging is illustrated in the following programming example. It includes a procedure _Select_Graphics to select mode (mode number is obtained from the include file MODE.INC), and three procedures for paging: _Select_Page, _Select_Read_Page and _Select_Write_Page. Note that all three page select procedures preserve the previous value of the Graphics Controller Index register to assure that drawing routines that preselect the Bit Mask register of the Graphics Controller operate properly.

Listing 10-1. File: AHEAD\SELECT.ASM

```
File:  AHEADSELECT.ASM
;*************************************************************************
;* File:  SELECT.ASM                                                     *
;* Description: This module contains procedures to select mode and to    *
;*        select pages.  It also initializes global variables            *
;*        according to the values in the MODE.INC include file.          *
;* Entry Points:                                                         *
;*        _Select_Graphics   - Select a graphics mode                    *
;*        _Select_Text       - Set VGA adapter into text mode            *
;*        _Select_Page       - Set page for read and write               *
;*        _Select_Read_Page  - Select read page only                     *
;*        _Select_Write_Page - Select write page only                    *
;* Uses:                                                                 *
;*        MODE.INC           - Mode dependent constants                  *
;*        Following are modes and paths for Ahead boards:                *
;*   |-------- 256 colors ----------| |--16 colors --| 4 colors 2 col   *
;*   640x400 640x480 800x600 1024x768 800x600 1024x768 1024x768 1024x768*
;*Mode: 60h    61h  62h    63h    6Ah(71h)    74h   75h      76h         *
;*Path:256COL  256COL 256COL 256COL    16COL  16COL    4COL02   2COL    *
;*************************************************************************

     INCLUDE VGA.INC
     INCLUDE MODE.INC    ;Mode dependent constants

     PUBLIC    _Select_Graphics
     PUBLIC    _Select_Text
     PUBLIC    _Select_Page
     PUBLIC    _Select_Read_Page
     PUBLIC    _Select_Write_Page

     PUBLIC    Select_Page
     PUBLIC    Select_Read_Page
     PUBLIC    Select_Write_Page
     PUBLIC    Enable_Dual_Page
     PUBLIC    Disable_Dual_Page

     PUBLIC    Graf_Seg
     PUBLIC    Video_Height
     PUBLIC    Video_Width
     PUBLIC    Video_Pitch
     PUBLIC    Video_Pages
     PUBLIC    Ras_Buffer
     PUBLIC    Two_Pages

     PUBLIC    Last_Byte

;-------------------------------------------------------------------------
; Data segment variables
;-------------------------------------------------------------------------

;_DATA     SEGMENT WORD PUBLIC 'DATA'
;_DATA     ENDS
```

```
;---------------------------------------------------------------------------
; Constant definitions
;---------------------------------------------------------------------------


;---------------------------------------------------------------------------
; Code segment variables
;---------------------------------------------------------------------------

_TEXT     SEGMENT BYTE PUBLIC 'CODE'

Graf_Seg  DW   0A000h              ;Graphics segment addresses
          DW   0A000h
OffScreen_Seg  DW   0A000h         ;First byte beyond visible screen
Video_Pitch    DW   SCREEN_PITCH   ;Number of bytes in one raster
Video_Height   DW   SCREEN_HEIGHT  ;Number of rasters
Video_Width    DW   SCREEN_WIDTH   ;Number of pixels in a raster
Video_Pages    DW   SCREEN_PAGES   ;Number of pages in the screen
Ras_Buffer     DB   1024 DUP (0)   ;Working buffer
R_Page         DB   0FFh           ;Most recently selected page
W_Page         DB   0FFh
RW_Page        DB   0FFh
Two_Pages      DB   CAN_DO_RW      ;Indicate separate R & W capability

;*************************************************************************
;*                                                                       *
;* _Select_Graphics(HorizPtr, VertPtr, ColorsPtr)                        *
;*   Initialize VGA adapter to 640x400 mode with                         *
;*   256 colors.                                                         *
;*                                                                       *
;* Entry:                                                                *
;*   None                                                                *
;*                                                                       *
;* Returns:                                                              *
;*   VertPtr   - Vertical resolution                                     *
;*   HorizPtr  - Horizontal resolution                                   *
;*   ColorsPtr - Number of supported colors                              *
;*                                                                       *
;*************************************************************************

Arg_HorizPtr   EQU  WORD PTR [BP+4] ;Formal parameters
Arg_VertPtr    EQU  WORD PTR [BP+6] ;Formal parameters
Arg_ColorsPtr  EQU  WORD PTR [BP+8] ;Formal parameters

_Select_Graphics PROC NEAR
     PUSH BP                       ;Standard C entry point
     MOV  BP,SP

     PUSH DI                       ;Preserve segment registers
     PUSH SI
     PUSH DS
     PUSH ES

     ; Select graphics mode

     MOV  AX,GRAPHICS_MODE         ;Select graphics mode
     INT  10h

     ; Reset 'last selected page'

     MOV  AL,0FFh                  ;Use 'non-existent' page number
     MOV  CS:R_Page,AL             ;Set currently selected page
     MOV  CS:W_Page,AL
     MOV  CS:RW_Page,AL

     ; Set return parameters

     MOV  SI,Arg_VertPtr           ;Fetch pointer to vertical resolution
     MOV  WORD PTR [SI],SCREEN_HEIGHT   ;Set vertical resolution
     MOV  SI,Arg_HorizPtr          ;Fetch pointer to horizontal resolution
     MOV  WORD PTR [SI],SCREEN_WIDTH    ;Set horizontal resolution
```

```
     MOV  SI,Arg_ColorsPtr    ;Fetch pointer to number of colors
     MOV  WORD PTR [SI],SCREEN_COLORS   ;Set number of colors

     ; Clean up and return to caller

     POP  ES                  ;Restore segment registers
     POP  DS
     POP  SI
     POP  DI

     MOV  SP,BP               ;Standard C exit point
     POP  BP
     RET
_Select_Graphics ENDP

;*************************************************************************
;                                                                        *
; Select_Page                                                            *
; Entry:                                                                 *
;    AL - Page number                                                    *
;                                                                        *
;*************************************************************************

Select_Page    PROC NEAR
     CMP  AL,CS:RW_Page       ;Check if already selected
     JNE  SP_Go
     RET
SP_Go:
     PUSH AX
     PUSH BX
     PUSH DX
     ;Save currently selected page number
     AND  AL,0Fh              ;Force page number into range
     MOV  CS:RW_Page,AL       ;Save as most recent RW page
     MOV  CS:R_Page,AL        ;Invalidate R and W pages
     MOV  CS:W_Page,AL
     ;Fetch gr. ctrl. index (some drawing routines need it preserved)
     MOV  DX,3CEh             ;Fetch address of page select
     XCHG BL,AL               ;Save AL
     IN   AL,DX               ;Must save current gr. ctrl. index
     XCHG BL,AL
     ;Move page number into proper bits
     MOV  AH,AL               ;Copy page number into high nibble
     SHL  AL,1
     SHL  AL,1
     SHL  AL,1
     SHL  AL,1
     OR   AH,AL
     ;Select new page
     MOV  AL,0Dh              ;Fetch page register index
     OUT  DX,AX               ;Write out the new page select
     ;Restore gr. ctrl. index
     XCHG AL,BL               ;Restore gr. ctrl. index
     OUT  DX,AL

     POP  DX
     POP  BX
     POP  AX
     RET
Select_Page    ENDP

;*************************************************************************
;                                                                        *
; Select_Read_Page                                                       *
; Entry:                                                                 *
;    AL - Page number                                                    *
;                                                                        *
;*************************************************************************

Select_Read_Page PROC NEAR
     CMP  AL,CS:R_Page        ;Check if already selected
```

```
     JNE  SRP_Go
     RET
SRP_Go:
     PUSH AX
     PUSH BX
     PUSH DX
     ; Save new values
     MOV  CS:RW_Page,0FFh      ;Invalidate RW page value
     AND  AL,0Fh               ;Force page # into range
     MOV  CS:R_Page,AL
     MOV  AH,AL                ;Save page number
     ;Fetch gr. ctrl. index (some drawing routines need it preserved)
     MOV  DX,3CEh              ;Fetch address of page select
     IN   AL,DX                ;Must save current gr. ctrl. index
     MOV  BL,AL
     ;Move page number into proper bits and select new page
     MOV  AL,0Dh               ;Fetch page register index
     OUT  DX,AL                ;Select register
     INC  DX
     IN   AL,DX                ;Fetch previous value of page reg
     AND  AL,0F0h              ;Preserv write page
     OR   AL,AH                ;Move page number into ""read" bits
     OUT  DX,AL                ;Write out the new page select
     ;Restore graphics controller index
     MOV  AL,BL                ;Restore gr. ctrl. index
     DEC  DX
     OUT  DX,AL
     ; Clean up and return
     POP  DX
     POP  BX
     POP  AX
     RET
Select_Read_Page ENDP

;*************************************************************************
;                                                                        *
; Select_Write_Page                                                      *
; Entry:                                                                 *
;    AL - Page number                                                    *
;                                                                        *
;*************************************************************************

Select_Write_Page PROC NEAR
     CMP  AL,CS:W_Page         ;Check if already selected
     JNE  SWP_Go
     RET
SWP_Go:
     PUSH AX
     PUSH BX
     PUSH DX
     ; Save new values
     MOV  CS:RW_Page,0FFh      ;Invalidate RW page value
     MOV  CS:W_Page,AL         ;Save new write value
     MOV  AH,AL
     ;Fetch gr. ctrl. index (some drawing routines need it preserved)
     MOV  DX,3CEh              ;Fetch address of page select
     IN   AL,DX                ;Must save current gr. ctrl. index
     MOV  BL,AL
     ;Move page number into proper bits and select new page
     SHL  AH,1                 ;Copy page # into hi nibble of AH
     SHL  AH,1
     SHL  AH,1
     SHL  AH,1
     MOV  AL,0Dh               ;Fetch page register index
     OUT  DX,AL                ;Select register
     INC  DX
     IN   AL,DX                ;Get current values
     AND  AL,0Fh               ;Preserve read page number
     OR   AL,AH                ;Move page number into ""write" bits
     OUT  DX,AL                ;Write out the new page select
     ;Restore graphics controller index
```

```
     MOV  AL,BL              ;Restore gr. ctrl. index
     DEC  DX
     OUT  DX,AL
     ; Clean up and return
     POP  DX
     POP  BX
     POP  AX
     RET
Select_Write_Page ENDP

;*************************************************************************
;*                                                                       *
;* Enable_Dual_Page                                                      *
;* Disable_Dual_Page                                                     *
;*   Not supported by Ahead based boards                                 *
;*                                                                       *
;*************************************************************************

Enable_Dual_Page    PROC NEAR
     RET
Enable_Dual_Page    ENDP

Disable_Dual_Page   PROC NEAR
     RET
Disable_Dual_Page   ENDP

;*************************************************************************
;                                                                        *
; _Select_Page(PageNumber)                                              *
; Entry:                                                                *
;    PageNumber - Page number                                           *
;                                                                       *
;*************************************************************************

Arg_PageNumber EQU  BYTE PTR [BP+4]

_Select_Page   PROC NEAR
     PUSH BP                 ;Setup frame pointer
     MOV  SP,BP
     MOV  AL,Arg_PageNumber  ;Fetch argument
     POP  BP                 ;Restore BP
     JMP  Select_Page
_Select_Page   ENDP

;*************************************************************************
;                                                                        *
; _Select_Read_Page(PageNumber)                                         *
; Entry:                                                                *
;    PageNumber- Page number for read                                   *
;                                                                       *
;*************************************************************************

Arg_PageNumber EQU  BYTE PTR [BP+4]

_Select_Read_Page   PROC NEAR
     PUSH BP                 ;Setup frame pointer
     MOV  SP,BP
     MOV  AL,Arg_PageNumber  ;Fetch argument
     POP  BP                 ;Restore BP
     JMP  Select_Read_Page
_Select_Read_Page   ENDP

;*************************************************************************
;                                                                        *
; _Select_Write_Page(PageNumber)                                        *
; Entry:                                                                *
;    PageNumber - Page number for write                                 *
;                                                                       *
;*************************************************************************

Arg_PageNumber EQU  BYTE PTR [BP+4]
```

```
_Select_Write_Page  PROC NEAR
     PUSH BP                  ;Setup frame pointer
     MOV  SP,BP
     MOV  AL,Arg_PageNumber   ;Fetch argument
     POP  BP                  ;Restore BP
     JMP  Select_Write_Page
_Select_Write_Page  ENDP

;*************************************************************************
;*                                                                       *
;* _Select_Text                                                          *
;*   Set VGA adapter to text mode                                        *
;*                                                                       *
;*************************************************************************

_Select_Text   PROC NEAR
     MOV       AX,TEXT_MODE        ;Select mode 3
     INT       10h                 ;Use BIOS to reset mode
     RET
_Select_Text   ENDP

Last_Byte:
_Text     ENDS
          END
```

Detection and Identification

Ahead VGA cards can be detected by a signature field located in the Ahead BIOS ROMs at location C000:0025h, containing the ASCII characters 'AHEAD'. Chip version can be obtained from register index 20h in the extended register set. Version A chips return a value of 20h, and version B chips return a value of 21h. For example:

```
          MOV          DX,3CEh              ;Fetch I/O Address
          MOV          AL,0Fh               ;Fetch index of 'Enable' reg
          OUT          DX,AL                ;Select 'Master Enable' register
          INC          DX                   ;I/O address of data
          MOV          AL,20h               ;Fetch ENABLE value
          OUT          DX,AL                ;Enable extended register set
          JMP          $+2                  ;Wait for I/O to complete
          IN           AL,DX                ;Fetch chip version
          TEST         AL,1                 ;Test for version B
          JNZ          VersionB
VersionA:
          ...
VersionB:
          ...
```