mach64
Accelerator
BIOS Kit

SECOND DRAFT

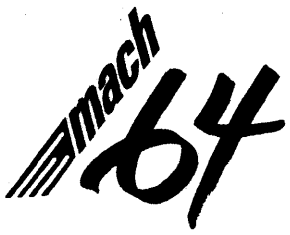
Information in this document is **proprietary** and **confidential.**

Technical Reference Manuals

P/N: BIO888GX0-02

ATI Technologies Inc.
33 Commerce Valley Drive East
Thornhill, Ontario
Canada L3T 7N6

User Support: 905-882-2626
Office: 905-882-2600
Fax: 905-882-2620
BBS: 905-764-9404



BIO888GX0-02

Preliminary Release: 0.3

The information contained in this document has been carefully checked and is believed to be entirely reliable. No responsibility is assumed for inaccuracies. ATI reserves the right to make changes at any time to improve design and supply the best product possible.

January1994

Table of Contents

Introduction **1-1**

About This Manual 1-1

Overview 1-1

Extended Video Modes 1-1

Supported BIOS Functions 1-2

Extended Graphics Accelerators 1-3

Customization 1-3

Getting Started **2-1**

Contents of the Kit 2-1

Other Tools 2-1

Copying Files 2-1

File Listings By Directories 2-2

Video BIOS **3-1**

Overview 3-1

VGA Compatibility 3-1

Symbolic Constants 3-1

88800GX RAMDAC Specific Constants 3-2

88800GX BUS Specific Constants 3-3

Clock Chip Specific Constants 3-3

EEPROM Specific Constants 3-4

VESA BIOS Extension Constants 3-4

Custom BIOS 3-4

BIOS Initialization 3-5

Sign-On Messages 3-5

Character Fonts 3-5

Monitor Support - ATI88800GX/CX 3-5
Internal and External Data Storage - EEPROM 3-6
8-/16-Bit ROM Selection 3-7
Clock Chip 3-7
Relocating ROM BIOS 3-8
Relocating the Video BIOS between C000:0 and E000:0 3-8
Paging Out the Initialization Code 3-8
PCI-Specific Implementation 3-8
VESA BIOS Extension Implementation 3-9
Making Non-Paged BIOS 3-11
Binary Files - Procedure Outline 3-11
Generating Binary Files - Parameters 3-11

BIOS Function Calls A-1
VGA Controller A-1
Extended ROM Services A-11
BIOS Support A-11
Calculating ROM Base Address A-11
Function Calls A-11
Query Structure A-16
Mode Table Structure A-18

EEPROM DATA B-1
ROM ENTRIES B-1
ATI88800CX EEPROM Data Structure B-2
ATI88800CX CRT Parameter Table B-5
VESA BIOS Extensions B-7
Status Information B-7
Function 00h - Return Super VGA information B-7
SUPER VGA Mode Numbers B-9
Supported VESA Modes B-9
Function 01h - Return Super VGA Mode Information B-9
Function 02h - Set Super VGA Video Mode B-14
Function 03h - Return Current Video Mode B-14
Function 05h - CPU Video Memory Window Control B-15
Function 06h - Set/Get Logical Scan Line Length B-16
Function 07h Set/Get Display Start B-17

00 - Report VBE/PM Capabilities B-18
01 - Set Display Power State B-18
02 - Get Display Power State B-19

Parameter Table Format C-1

Listing by Byte Numbers C-1

Dot Clocks D-1

ATI18811-1 Clock Chip Pixel Clocks D-1
ATI18818 Programmable Clock Chip D-2
ATI18818 Programmable Clock Chip D-3

Scratch Registers and Their Contents E-1

Index

1

Introduction

About This Manual

This manual is written for developers who wish to include a video BIOS and the **ATI88800GX/CX** in their system hardware. This manual explains how to generate a video BIOS, and provides example BIOS modules. Developers can modify the example modules in the kit to suit any types of monitors.

- *Chapter 1* provides an introduction to the organization of this manual. It includes an overview of the BIOS Kit contents.
- *Chapter 2* outlines the resources required to modify the source files and the object modules. It outlines the computer operating environment you require to get started. The supplied files and directory structures are listed.
- *Chapter 3* discusses compatibility, symbolic constant names, values and areas in the BIOS to be modified. It explains the use of the batch file MAKEROM.

Overview

The BIOS must be written for specific versions of controllers to provide support for specific video modes, monitor types, or extended functions as applicable. The specific versions of controllers are **ATI88800GX** and **ATI88800CX**.

Extended Video Modes

The following extended video modes are supported on standard PS/2 VGA monitors:

- 100x25 16 colors (Mode 21h)
- 100x30 16 colors (Mode 22h)
- 132x25 16 colors (Mode 23h)
- 132x44 16 colors (Mode 33h)

The following video modes require an analog monitor capable of displaying the stated resolutions:

- 800 x 600 16 colors (Mode 3Ah)
- 800x600 256 colors (Mode 63h)
- 1024 x 768 16 colors (Mode 55h)
- 1024x768 256 colors (Mode 64h)

Supported BIOS Functions

The following tested BIOS functions (Video Service INT 10h) are provided in source format for customization:

- Parameter AH=00h (Set Video Mode)
- Parameter AH=12h (Alternate Select)

The following tested BIOS functions (Video Service INT 10h) are provided in object format only. They need not be customized:

- Parameter AH=01h (Set Cursor Type)
- Parameter AH=02h (Set Cursor Position)
- Parameter AH=03h (Read Current Cursor Position)
- Parameter AH=04h (Read Light Pen Position)
- Parameter AH=05h (Select Active Display Page)
- Parameter AH=06h (Scroll Active Page Up)
- Parameter AH=07h (Scroll Active Page Down)
- Parameter AH=08h (Read Character/Attribute from Screen)
- Parameter AH=09h (Write Character/Attribute to Screen)
- Parameter AH=0Ah (Write Character Only to Screen)
- Parameter AH=0Bh (Set Color Palette)
- Parameter AH=0Ch (Write PEL)
- Parameter AH=0Dh (Read PEL)
- Parameter AH=0Eh (Write Teletype to Active Page)
- Parameter AH=0Fh (Read Current Video Status)
- Parameter AH=10h (Set Palette Registers)
- Parameter AH=11h (Load Character Generator)
- Parameter AH=13h (Write Strings: AL=00h-03h)
- Parameter AH=1Ah (Read/Write Display Combination Codes)
- Parameter AH=1Bh (Return Functionality/State Information)
- Parameter AH=1Ch (Save/Restore Video State)

Extended Graphics Accelerators

Source codes are provided to customize the BIOS for Graphics Accelerators. The source codes include the VGA parameter, coprocessor parameter conversion, extended video mode support, controller initialization, and extended function service call. Refer to the *Custom BIOS* section in Chapter 3 for details.

Customization

Source codes are also provided to customize the BIOS in the areas listed below. Refer to the *Custom BIOS* section in Chapter 3 for details.

- Sign-on messages
- Character fonts
- Add/Delete video mode
- Video mode support (AH=00h)
- 8-/16-bit ROM selection
- **ATI18820** mouse chip
- Integration with system BIOS
- Relocation of ROM BIOS to another address space
- Zero wait-state video RAM
- Zero wait-state video ROM
- **ATI88800GX/CX** extended function service call
- RAMDAC support

This page intentionally left blank.

2

Getting Started

Contents of the Kit

The BIOS kit consists of this manual and a diskette that contains the *source* and *object* files required to customize and generate a video BIOS for the **ATI88800GX/CX** graphics controller.

This kit requires 2M bytes of hard disk space for the BIOS files. Program tools for file preparation and generation are supplied. DOS 3.2 or a later operating system is recommended. You should specify 30 file handles in the CONFIG.SYS for efficient handling of files during the customization process. When generating the BIOS, the system must have 540K bytes or more of free conventional DOS memory.

- MS-DOS or PC-DOS Version 3.2 or later
- FILES=30
- 2M bytes free space in hard disk
- 540K bytes free conventional DOS memory

Other Tools

Not included in this kit are the following programs, which you will require:

- Microsoft MASM, version 5.1
- EXE2BIN.EXE
- LINK.EXE, version 3.65 or later
- EDLIN.EXE

Copying Files

To use the program tools provided, both *source* and *object* files must be organized in a directory structure identical to the one supplied on the diskette. You should copy the files and directories to the root directory or to a \BIOS directory on your hard disk.

Assuming XCOPY is in your path or the current directory, the commands shown below will copy all the files from diskette drive **A:** to hard disk drive **C:**. You may substitute any valid disk drive letters in this copy command.

XCOPY A:*.* C:*.* /S /V or

XCOPY A:*.* C:\BIOS*.* /S /V

Files are stored in separate directories by file types. These files will support the generation of a non-paged BIOS.

1. **\CXROM**
   mach64 BIOS source files.
2. **\CXROM\OBJS**
   mach64 BIOS object files.
3. **\CXROM\CONFIG**
   mach64 BIOS configuration files.
4. **\CXVESA**
   mach64 VESA BIOS source files.
5. **\VGACOMM**
   Macros and files of symbolic constants.
6. **\FONTS**
   ATI proprietary character generator in source format.
7. **\VGATOOLS**
   Tools that are required to generate video BIOS object codes.

File Listings By Directories

1. \CXROM - Source Codes

- **ATI$8$16.ASM** - Routine to enable 16-bit ROM for ISA configuration
- **ATI$MICE.ASM** - Routine to enable the ATI18820 mouse chip
- **CX$000.ASM** - Main routine to support the CX BIOS extensions
- **CX$001.ASM** - Routines to convert VGA CRTC parameters to Coprocessor CRTC parameters
- **CX$002.ASM** - Routines to program the Coprocessor CRT Controller
- **CX$003.ASM** - General routines to support the Coprocessor
- **CX$004.ASM** - General routines to support the Coprocessor, OEM specific
- **CX$005.AM** - Routines to locate the VGA CRTC parameters for conversion

- **CX$006.ASM** - Routines to support the query functions in the CX extended BIOS
- **CX$007.ASM** - Routines to load the Coprocessor parameters and set the active display mode
- **CX$008.ASM** - Routines and tables to support different clock chips and frequency tables
- **CX$2095.ASM** - Routines to program the ATI18818 Clock chip
- **CX$ADJ.ASM** - Routine to retrial CRTC parameters from EEPROM tables
- **CX$C01.ASM** - Symbolic constants by the CX ROM BIOS
- **CX$C02.ASM** - Symbolic constants for the VGA extended modes
- **CX$C03.ASM** - Symbolic constants used for the extended bits in the VGA CRTC table
- **CX$C04.ASM** - Symbolic constants used for the Graphics Coprocessor
- **CX$CLKS.ASM** - Routine to program the clock chip
- **CX$CTBL0.ASM** - Contains frequency table for ATI18811
- **CX$CTBL1.ASM** - Contains frequency table when PCLK_TABLE equals 1
- **CX$CTBL2.ASM** - Contains frequency table when PCLK_TABLE equals 2
- **CX$DAC.ASM** - General routines to initialize the RAMDAC
- **CX$DAC2.ASM** - Routines to initialize the TLC34075 and ATI68875 RAMDACs
- **CX$DAC4.ASM** - Routines to initialize the BT481, AT&T20C490/491/492/493 RAMDACs
- **CX$DAC5.ASM** - Routines to initialize the ATI68860 RAMDAC
- **CX$DAC6.ASM** - Routines to initialize the STG1700 and AT&T20C498 RAMDACs
- **CX$DAC7.ASM** - Routines to initialize the SC15021, STG1702, and ATI21G498 RAMDACs
- **CX$DEF.ASM** - Data structure for the EEPROM
- **CX$EE0.ASM** - Routines to read data from EEPROM
- **CX$EE1.ASM** - Routines to write data from EEPROM
- **CX$F00.ASM** - VGA set mode function (AH=0)
- **CX$F0F.ASM** - VGA functions AH=1, AH=2, AH=3, AH=4 and AH=0Fh
- **CX$F11.ASM** - Routines to load the character generator
- **CX$F11A.ASM** - Routines to load the character generator

- **CX$FNTS.ASM** - Includes information for fonts
- **CX$HEAD.ASM** - Extended header information for the video BIOS
- **CX$INIT.ASM** - Main routine to initialize the CX controller
- **CX$INIT0.ASM** - Routine to enable the controller and setup INT 10 vectors
- **CX$INIT1.ASM** - Routines to determine the total memory size installed
- **CX$INIT2.ASM** - Routines to set up the selected video mode at power-up
- **CX$INIT3.ASM** - Routines to initialize the extended VGA registers
- **CX$INIT4.ASM** - Routines to initialize the graphics coprocessor registers
- **CX$INIT5.ASM** - Routines to monitor detection (color/mono)
- **CX$INIT6.ASM** - Routines to detect individual RAMDAC, the detectable RAMDACs are BT481 and AT&T20C490/491/493, STG1700 and AT&T20C498
- **CX$JMP.ASM** - Jump table for the Video BIOS functions
- **CX$MSG.ASM** - Routines to print out the sign-on message for the video BIOS
- **CX$P00A.ASM** - CRTC parameter tables for 800x600 and 1024x768 SVGA modes
- **CX$P00B.ASM** - Tables identifying the supported refresh rate and color depth
- **CX$P00C.ASM** - CRTC parameter tables for VGA modes in high refresh rate
- **CX$P132.ASM** - CRTC parameter tables for 132 column mode
- **CX$PS2.ASM** - CRTC parameter tables for VGA modes
- **CX$REGS.ASM** - Routines to read and write extended VGA registers
- **CX$ROM.ASM** - Defines all the include files for the CX/GX BIOS when compiled with source code only
- **CX$TBLS.ASM** - Tables and data used by the Video BIOS
- **CX$V00.ASM** - Main routine to support the standard VGA BIOS
- **CX$V02.ASM** - Routine to select the frequency entry for VGA mode
- **CX$V03.ASM** - General routines used during power-up initialization
- **CX$V04.ASM** - Routine to return pointer to SVGA CRTC parameter table
- **CX$V05.ASM** - Routines to program the extended VGA registers during set mode
- **CX$V07.ASM** - Routine to return pointer to standard VGA CRTC parameter table
- **CX$V08.ASM** - Routine to support CGA emulation

- **CX$V20.ASM** - Routines to program the Sequencer, Graphics and Attribute Controllers
- **CX$V21.ASM** - Routine to program the CRT controller
- **CX$VADJ.ASM** - Routine to adjust the centering of the display for data in EEPROM
- **O$C.ASM** - Symbolic constant used when compiling the object code
- **O$CX.ASM** - Main include file for supporting VGA enable configuration, this file will generate the binary from all source and compiled object code
- **O$INIT.ASM** - Defines all the include files for the CX/GX BIOS when compiling the initialization code
- **O$ROM.ASM** - Defines all the include files for the CX/GX BIOS when compiled with source and object code
- **O$XFCN.ASM** - Symbolic constants used to generate the object files
- **U$001.ASM** - Routine to load the coprocessor parameter for VGA disable configuration
- **U$CX.ASM** - Main include file for supporting VGA disable configuration for ATI88800CX
- **U$GX.ASM** - Main include file for supporting VGA disable configuration for ATI88800GX
- **U$INIT0.ASM** - Routines to initialize the controller in VGA disable configuration
- **U$INIT1.ASM** - Routines to intialize the controller in VGA disable configuration
- **U$MSG.ASM** - Routine to put out a text string in VGA disable configuration
- **U$ROM.ASM** - Main routine for VGA disabled configuration
- **V$CX.ASM** - Main include file for supporting VGA enable configuration for ATI88800CX
- **V$GX.ASM** - Main include file for supporting VGA enable configuration for ATI88800GX
- **VGA$8$16.ASM** - Routine to enable and disable 16-bit RAM for ISA configuration
- **VGA$WAIT.ASM** - Routine to test and enable zero wait state RAM for ISA configuration
- **MAKEROM.BAT** - Batch file to compile the BIOS from source and object code
- **MAKEROMS.BAT** - Batch file to compile the BIOS from source code
- **CX$MAP0.MAC** - Macro to support a mapped BIOS

- **CX$P00A.MAC** - Macro defining the 800x600, 1024x768 CRTC parameters for different refresh rate

2. \CXROM\OBJS

- **AH05.OBJ** - Select active display page function (AH=05h)
- **AH06.OBJ** - Scroll active page up function (AH=06h)
- **AH07.OBJ** - Scroll active page down function (AH=07h)
- **AH08.OBJ** - Read character/attribute from screen function (AH=08h)
- **AH09.OBJ** - Write character/attribute to screen (AH=09h) and Write character only to screen (AH=0AH)
- **AH0B.OBJ** - Set color palette
- **AH0C.OBJ** - Write PEL (AH=0Ch)
- **AH0D.OBJ** - Read PEL (AH=0Dh)
- **AH0E.OBJ** - Write TTY to active page (AH=0Eh)
- **AH0F.OBJ** - Read current video status (AH=0Fh)
- **AH10.OBJ** - Set palette registers (AH=10h)
- **AH11.OBJ** - Load character generator (AH=11h)
- **AH13.OBJ** - Write strings (AH=13h)
- **AH1A.OBJ** - Read/write display combination codes (AH=1Ah) and return functionality/state information functions (AH=1Bh)
- **V001.OBJ** - Print screen functions
- **V002.OBJ** - Scroll functions
- **V003.OBJ** - General purpose routines used in *VGAWONDER* BIOS
- **V004.OBJ** - Routines used in Mode 13 scrolling
- **V006.OBJ** - Routine to set the cursor type
- **V008.OBJ** - Routine to emulate an INT 10 call
- **V009.OBJ** - CGA/MDA emulation and routine to print a message
- **V010.OBJ** - A global variable used to define the ending location of the BIOS core area
- **VX001.OBJ** - Routine used for 256 colors scrolling
- **VX002.OBJ** - Routine used for Mode 55 to scroll up and down

3. \CXROM\CONFIG

- **CX.ISA** - Sample configuration file to support ISA
- **CX.VLB** - Sample configuration file to support VLB
- **CX.PCI** - Sample configuration file to support PCI
- **GX.ISA** - Sample configuration file to support ISA
- **GX.VLB** - Sample configuration file to support VLB
- **GX.PCI** - Sample configuration file to support PCI

4. \CXVESA\CONFIG

- **VESA$00.ASM** - Routines to support VESABIOS function AL=0, 1, and 2
- **VESA$01.ASM** - Routines to support the frame buffer windowing function
- **VESA$02.ASM** - Routines to support save and restore extended VGA states
- **VESA$03.ASM** - Routines to support get/set logical scan line length and get/set display start
- **VESA$10.ASM** - Routines to support DPMS
- **VESA$CNT.ASM** - Symbolic constants used by the VESA BIOS extension
- **VESA$DAT.ASM** - Data structure used by the VESA BIOS extension
- **VESA$JMP.ASM** - Jump table used by the VESA BIOS extension
- **VESA$XX.ASM** - Main include file for supporting the VESA BIOS extension

5. \VGACOMM - Source Codes

- **ATIU$06A.ASM** - Scrolling function to support ATI extended packed-pixel mode
- **ATIU$06C.ASM** - Scrolling function to support ATI extended 1024x768 4-plane planar mode
- **ATIU$07C.ASM** - Scrolling function to support ATI extended 1024x768 4-plane planar mode
- **ATIU$07D.ASM** - Scrolling function to support ATI extended 1024x768 4-color mode
- **EGADATA.ASM** - DOS data segment definition
- **EGAMAC.ASM** - Macros used in the video BIOS
- **M50$TAB.ASM** - Symbolic constants of parameter table offset
- **VGA$1AT.ASM** - Functionality support table
- **VGA$PAL.ASM** - Extended palette programming information
- **VGACONST.ASM** - Symbolic constants of VGA used in the BIOS
- **VGAF$12.ASM** - Alternate select subfunctions
- **VGAF$12A.ASM** - Alternate select subfunctions
- **VGAF$1B.ASM** - Routine to return VGA functionality and state information
- **VGAF$1C.ASM** - Routines to save and restore video states
- **VGAF$1CA.ASM** - Routines to save and restore video states

6. \FONTS - Source Codes

- **EGA8X8.ATI** - Include file definition for 8x8 font
- **EGA8X8A.ATI** - Upper 128 characters of 8x8 font
- **EGA8X8B.ATI** - Lower 128 characters of 8x8 font
- **EGA8X14.ATI** - Include file definition, 8x14 font
- **EGA8X14A.ATI** - Upper 128 characters of 8x14 font
- **EGA8X14B.ATI** - Lower 128 characters of 8x14 font
- **EGA8X14F.ATI** - 9x14 font supplement
- **EGA8X16.ATI** - Include file definition, 8x16 font
- **EGA8X16A.ATI** - Upper 128 characters of 8x16 font
- **EGA8X16B.ATI** - Lower 128 characters of 8x16 font
- **EGA8X16F.ATI** - 9x16 font supplement

7. \VGATOOLS - Program Tools

- **CHECKSUM.EXE** - program that places timestamp and checksum values in BIOS binary files (not used if video BIOS is integrated in system BIOS)
- **FIXPAGED.EXE** - program that places timestamp and checksum values in paged BIOS binary files (not used if video BIOS is integrated in system BIOS)
- **FIXE000.EXE** - program that places timestamp and checksum values in BIOS files, should be used for a BIOS that is placed in the E000:0 area and has 36K total size and 32K runtime size
- **FIXPCI.EXE** - program that places timestamp and checksum values in BIOS files, should be used for a BIOS generated to support PCI configuration
- **CX.EXE** - program that invokes Coprocessor functions in real and protected modes, supplied for testing the extended BIOS functions

3

Video BIOS

Overview

This chapter describes how to set up the video BIOS for integration with a system BIOS or a separate, customized video BIOS.

Most commercially available programs and screen drivers extract information from the video BIOS in order to set themselves up properly for the video subsystem (display adapter). To maintain compatibility with ATI's graphics controllers, specific information is required in the video BIOS header as indicated below:

VGA Compatibility

- Video BIOS starting segment address is adjusted so that the offset of the video BIOS begins at zero.
- For ATI BIOS extended function AH=12h: AL must hold the mode value and BX must be 5506h.
- The first 60h bytes of the video BIOS must not be altered (00h-5Fh). These bytes contain the product signature "761295520", product code "31" and other important data.
- Each BIOS is specific to a version of the graphics controller and mouse option. Files and parameters should be chosen accordingly when generating the BIOS binary file.
- A BIOS greater than 32KB will have pages mapped to the 32KB address.

Symbolic Constants

Symbolic constants are assigned values to indicate the type of hardware and software that the BIOS is to support. Once these values are declared, they will remain fixed (unchanged) in the program. In most cases, you should not re-define their values. You should use the default value that is already built into the BIOS kit. The following is a listing of symbolic constants that you may need to modify depending on your system configuration:

- BIOS_START_ADDR — The value 0C000h is the segment address of video BIOS
- BIOS_MSG — Log on message string
- HW_VER — **20h = ATI88800GX/CX** graphics controller
- IBMCG — **0** = Uses ATI fonts
- M_CHIP — **0** = Mouse chip not supported
  2 = **ATI18820** mouse chip or built-in mouse port supported
- MAJ_VER — Major version number
- MIN_VER — Minor version number
- PAGE_E000 — If defined, allows support of 36K total space and 32K runtime space in E000h:0 or C000:0

88800GX RAMDAC Specific Constants

- **ATT490_SUPPORT** - if set to 1, the BIOS generated will support AT&T20C490 RAMDAC. Default is **0**.
- **ATT491_SUPPORT** - if set to 1, the BIOS generated will support AT&T20C491 RAMDAC. Default is **0**.
- **ATT493_SUPPORT** - if set to 1, the BIOS generated will support AT&T20C493 RAMDAC. Default is **0**.
- **ATT498_SUPPORT** - if set to 1, the BIOS generated will support AT&T20C498 RAMDAC. Default is **0**.
- **ATI68860_SUPPORT** - if set to 1, the BIOS generated will support ATI68860 RAMDAC. Default is **0**.
- **BT481_SUPPORT** - if set to 1, the BIOS generated will support Brooktree BT481 RAMDAC. Default is **0**.
- **IMSG174_SUPPORT** - if set to 1, the BIOS generated will support Inmos IMSG174 RAMDAC. Default is **0**.
- **MU9C1880_SUPPORT** - if set to 1, the BIOS generated will support Music MU9C1880 RAMDAC. Default is **0**.
- **MU9C4910_SUPPORT** - if set to 1, the BIOS generated will support Music MU9C4910 RAMDAC. Default is **0**.
- **RAMDAC_AUTODETECT** - if set to 1, the BIOS will automatically detect BT481, AT&T20C491 and SCI5026. Default is **0**.
- **SC11486_SUPPORT** - if set to 1, the BIOS generated will support Sierra 11486 RAMDAC. Default is **0**.

- **SC15021_SUPPORT** - if set to 1, the BIOS generated will support Sierra 15021 RAMDAC. Default is **0**.
- **SC15026_SUPPORT** - if set to 1, the BIOS generated will support Sierra 15026/ 15025 RAMDAC. Default is **0**.
- **STG1700_SUPPORT** - if set to 1, the BIOS generated will support SGS-Thompson 1700 RAMDAC. Default is **0**.
- **STG1702_SUPPORT** - if set to 1, the BIOS generated will support SGS-Thompson 1702 RAMDAC. Default is **0**.
- **TLC34075_SUPPORT** - if set to 1, the BIOS generated will support Texas Instruments TLC 34075 RAMDAC. Default is **1**.

88800GX BUS Specific Constants

- **EISA** - if defined, the BIOS generated is EISA-specific.
- **LOCAL_BUS** - if defined, the BIOS generated is LOCAL BUS-specific.
- **PCI** - if defined, the BIOS generated is PCI-specific.
- If none of the above is defined, default is **ISA**.

Clock Chip Specific Constants

- **CLOCK_CHIP_TYPE** - specifies the type of clock chip used.
  = 0 ; ATI18811-1
  = 1 ; ATI18818 or compatible
- **ATI18818_MS1** - specifies the MS1 strapping for ATI18818.
  = 0 ; MS1 is strapped to ground
  = 1 ; MS1 is strapped to VCC
  Default is 0.
- **REF_FREQ** - specifies the reference frequency used. The unit is in KHz/10. Default is 1432.
- **PCLK_TABLE** - specifies the frequency table used by the programmable clock chip.
  = 0 ; ATI18811-1 clock chip is not programmable
  = 1 ; See Appendix D
  = 2 ; See Appendix D
  Default value depends on CLOCK_CHIP_TYPE and RAMDAC support. See "Clock Chip" in the Custom BIOS section.

EEPROM Specific Constants

- **EE_TABLE1** - defines the location of the first CRTC table in the EEPROM. Default is 17h.
- **EE_TABLE_SIZE** - defines the size of the CRTC table. Default is 0Fh.
- **EE_CRTC_TABLE** - defines the number of CRTC tables in the EEPROM data structure. Default is 7.
- **STORAGE_LAST_ENTRY** - defines the last location of the internal and external storage. Default is 80h and there is no internal entries.

VESA BIOS Extension Constants

- **VESA_BIOS** - if set to 1, the BIOS generated will support VESA super VGA standard. Default is 0.
- **VESA_DPMS** - if set to 1, the BIOS generated will support VESA Display Power Management BIOS Extensions. Default is 0. (To set this to 1, VESA_BIOS must be set to 1).

Custom BIOS

The video parameter tables in the BIOS contain video mode data that controls proper monitor synchronization, screen refresh rates, screen sizing, and screen positioning. This data is programmed in the Sequencer registers, CRT Controller registers, Attribute Controller registers, and Graphics Controller registers. As a result of customizing the BIOS, proper monitor operation is assured.

In addition to customizing for specific monitor types, the BIOS also supports custom sign-on messages, mouse support, character font support, video modes and the like. The video BIOS may be integrated into the system BIOS or can be located at C000:0000h. See *BIOS Integration* later in this Chapter.

Supplied files for generating the video BIOS are organized in separate directories. The four directories containing shared files for both BIOS versions are as follows:

- \CXROM – Source Codes
- \CXVESA – Source Codes
- \VGACOMM – Source Codes
- \FONTS – Source Codes
- \VGATOOLS – Program Tools

BIOS Initialization

In order for the system BIOS to recognize the video BIOS, the first three bytes of the video ROM must be as follows: bytes one and two are 55h and AAh respectively; byte three is a number indicating the size of the BIOS in 2K byte blocks.

On power-up, the system BIOS calculates the checksum on the specified 2K byte blocks. If the last two digits of the checksum is 00h, the system BIOS executes a JMP instruction into the fourth location of the ROM BIOS for initialization.

Included in this kit is a batch file called **MAKEROM** which is used to automate the process of generating BIOS binary files.

Sign-On Messages

A sign-on message is displayed during power-up to indicate the BIOS version and mouse support. This message string is the symbolic constant BIOS_MSG. It is usually defined in the \CXROM\CONFIG\CX.* files.

Character Fonts

Three sets of character generators are provided in source format in the \FONTS directory. Generators support 8x8, 8x14, and 8x16 characters. Include file definitions are in the FONTS.ASM file which is in the \CXROM directory.

Customized fonts can be added. To indicate the presence of these fonts, symbolic constant IBMCG and file CX$FNTS.ASM should be updated.

Monitor Support - ATI88800GX/CX

The Graphics Accelerator BIOS supports a wide variety of popular monitors. The BIOS uses actual refresh rates for specific video mode and monitor combinations.

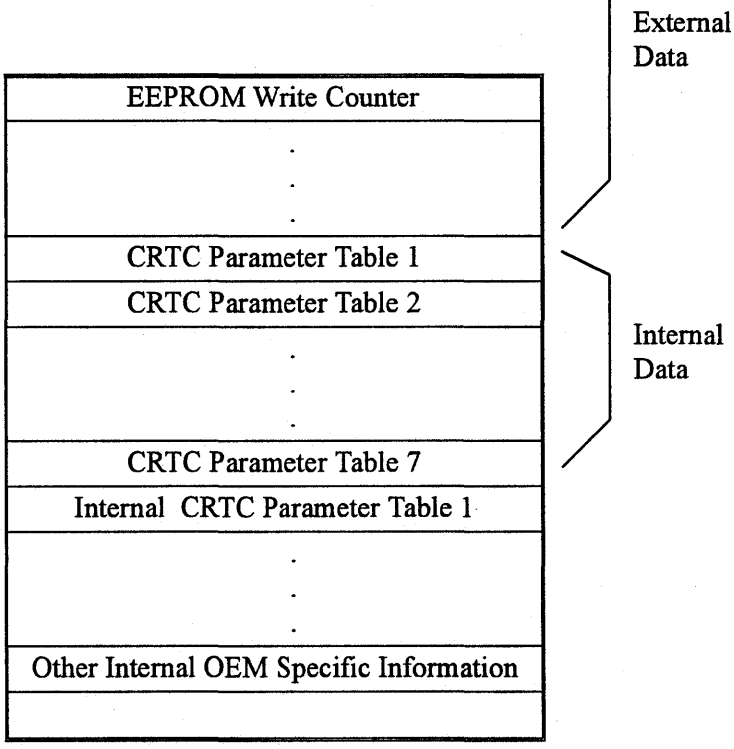
ATI's software installation/configuration program (INSTALL.EXE) translates the user-selected monitor types to refresh rate values and stores them in EEPROM entries 5, 6 and 7. These rates are then loaded into scratch registers during power-up. See Appendix E for the usage of scratch registers.

During *Set Mode*, the VGA initializes its controller to the specified refresh rate. If the Graphics Accelerator is enabled, the BIOS will automatically calculate the coprocessor CRTC parameters using VGA CRTC parameters and program the coprocessor accordingly.

If you wish to have higher refresh rates that only the coprocessor can support, you must load the CRTC into the EEPROM for programming the coprocessor using the installation/configuration program.

Internal and External Data Storage - EEPROM

The BIOS supports an external storage device, such as EEPROM, for storing the video configuration information. The device can be read and write through the extended BIOS function AL=3 and AL=4. The EEPROM can be replaced by CMOS or other storage device by replacing the READ_EE and WRITE_EE routines in the \CXROM\CX$EE0.ASM and \CXROM\CX$EE1.ASM files. These routines must be executable in protected mode. The layout of the storage device is in Appendix A. The starting location and size of the tables are returned in the extended function AL=11h. The implementation also supports an internal table which is attached to the end of the external storage device. The internal table is intended to store coprocessor parameters and the maximum number of the external and internal entries is 256. The layout of the external table and internal table can be viewed in the following table:



The symbolic constants EE_TABLE1 specifies the starting location of the CRTC parameter table, EE_TABLE_SIZE specifies the size of the CRTC parameter table, EE_LAST_ENTRY specifies the size of the external storage and STORAGE_LAST_ENTRY specifies the total size of the external and internal storage.

In the case when the internal table contains multiple CRTC parameter tables of the same resolution of different refresh rate, the routine GETREFRESHMASK in the file \CXROM\CX$004.ASM has to be modified to tell the other routines which table to use. For the Coprocessor Only mode, bit 7 of the refresh mask has to be 1 in order to have the algorithm working correctly. The code has been tested with a dummy internal CRTC table and is working as intended.

8-/16-Bit ROM Selection

If the graphics controller is integrated with the motherboard, 8-bit or 16-bit ROM operation is normally hard-wired. Thus, 8-/16-bit ROM switching is unnecessary. 16-bit ROM switching is disabled by setting the symbolic constant B16 to "0" in the user's customization file. The option is only available for ISA.

Clock Chip

This BIOS is structured to support different types of clock chips, both fixed and programmable ones. For the current release of the BIOS, both fixed and dynamic frequency tables are supported. For fixed frequency table, the BIOS will initialize a set of predefined frequencies and these will not be changed at run time. The predefined frequency table is returned through the BIOS extended function AL=0Ah. The BIOS assumed there is a maximum of 16 frequency entries and if there are less than 16 entries, the remaining entries are zeros.

The code has been implemented and tested to support dynamic programming. The benefit of having dynamic programming is that the BIOS can support clock chips with very little programmable entries. The BIOS will program the clock chip to the required frequency at the time when required. The drawback with this approach is that the SCO Unix like drivers will be more hardware dependent. The driver has to actually program the clock chip to a particular frequency rather than just selecting a frequency in the frequency table. An example has been explained when PCLK_TABLE = 2.

The current BIOS supports the ATI18818 programmable clock chip and the BIOS will initialize the two different frequency tables based on the hardware configuration. The symbolic constant PCLK_CHIP specifies which frequency table to use. See Appendix D for frequency table information.

For "PCLK_TABLE = 1", the frequency table is intended to use with all the RAMDACs, except ATI68860 and ATI68880. For "PCLK_TABLE = 2", the frequency table is intended to use with ATI68860 and ATI68880 RAMDACs.

It is recommended to use the same frequency table as suggested for product compatibility reason. For developers who want to support different programmable clock chips and frequency tables, they have to modify the following files. The file \CXROM\CX$CLKS contains routines to program the clock chip, whereas the files \CXROM\CX$008.ASM and \CXROM\CX$CTBL*.ASM define the frequency table. The symbolic constants VGA_PROG_CLK and CX_PROG_CLK specify the entry to be used for supporting dynamic programming in the VGA and Coprocessor modes. Both VGA_PROG_CLK and CX_PROG_CLK should be set to 0FFh if fixed frequency table is used.

There are two frequency tables defined by the BIOS. They are labelled as EXTCLK_ENTRIES and CLK_ENTRIES. The EXTCLK_ENTRIES are entries of the clock chip that are viewed by the application and are used to program the clock chip. The CLK_ENTRIES is the frequency table used internally by the BIOS. Under normal circumstances (when PCLK_TABLE = 1 or 2), the EXTCLK_ENTRIES and

CLK_ENTRIES are the same because the CRTC parameter tables in the BIOS are using the chosen frequency table. In the case when a different fixed frequency table is used, the user can define a table corresponding to its clock chip and the BIOS will automatically find the closest frequency when setting video modes.

Relocating ROM BIOS

Symbolic constant BIOS_START_ADDR specifies the starting segment address of the video BIOS. Default value is C000h. It can be re-defined in the user's configuration file to accommodate different designs. In addition, if the video BIOS is in the E000:0 area, it can be relocated to the C000:0 by setting RELOCATE_E000 to 1. For the system BIOS, it needs to initialize the video BIOS in the corresponding segment at least once by performing a call to X000:3.

Relocating the Video BIOS between C000:0 and E000:0

The video BIOS can be located in a different location by setting the corresponding value to BIOS_START_ADDR. In some cases, you may want to put the video BIOS in E000:0 area and later shadow it to the C000:0 area. Setting the symbolic constant RELOCATE_E000 to 1 allows the BIOS to generate all the necessary tables. To relocate a video BIOS from E000:0 to C000:0, the system BIOS has to do the following:

- Initialize the video BIOS by doing a call E000:3.
- Copy the first 32K from E000:0 to C000:0.
- Do a call C000:3 to have the video BIOS update the video interrupt vector and tables.

Paging Out the Initialization Code

The BIOS is organized such that the initialization is at the end of the BIOS and can be paged out after the video is initialized. There are also signature bytes used by the BIOS to determine if the initialization code is paged out before jumping into the code. The symbolic constant PAGE_E000 enables the code for size checking and paging out of the initialization code. The BIOS supporting this feature can be in C000:0 or E000:0. To use this paging mechanism, the FIXE000 has to be used to put in the correct checksum and signature bytes for 32K and 36K binary.

PCI-Specific Implementation

The PCI system BIOS has the capability to support the paging mechanism described in the last paragraph. In addition, the implemented PCI code can also be in C000:0 or E000:0 because the BIOS will patch all the pointers and checksum in the video BIOS at initialization. The BIOS assumed that the video BIOS is loaded into the shadow RAM and the shadow RAM is writeable when the video BIOS is initialized as outlined in the PCI specification 2.0. The PAGE_E000 has to be set to 1 and the FIXPCI.EXE has to be run to generate the correct checksum and signature bytes.

VESA BIOS Extension Implementation

The ATI8800GX/CX incorporates a VGA CRT controller and a graphics processor CRT controller for display. The VGA CRT controller is used for all standard VGA modes (0h — 13h). The graphics coprocessor CRTC controller is used for all accelerated and hicolor modes.

To maximize support of hicolor modes, the VESA BIOS extensions are implemented to support the graphics coprocessor CRT controller rather than the VGA CRT controller. All CRTC parameters are graphics-coprocessor-based. Because the translation of CRTC parameters is transparent to the application, this implementation should have little effect on existing applications that use VESA BIOS extensions.

Because the VESA BIOS extensions use the graphics coprocessor base, the VESA BIOS extended function, AL=4 (save and restore extended VGA states), is not implemented. In addition, extended 4 plane-planar mode is not supported in the VESA BIOS extension because the graphics coprocessor does not supor this mode.

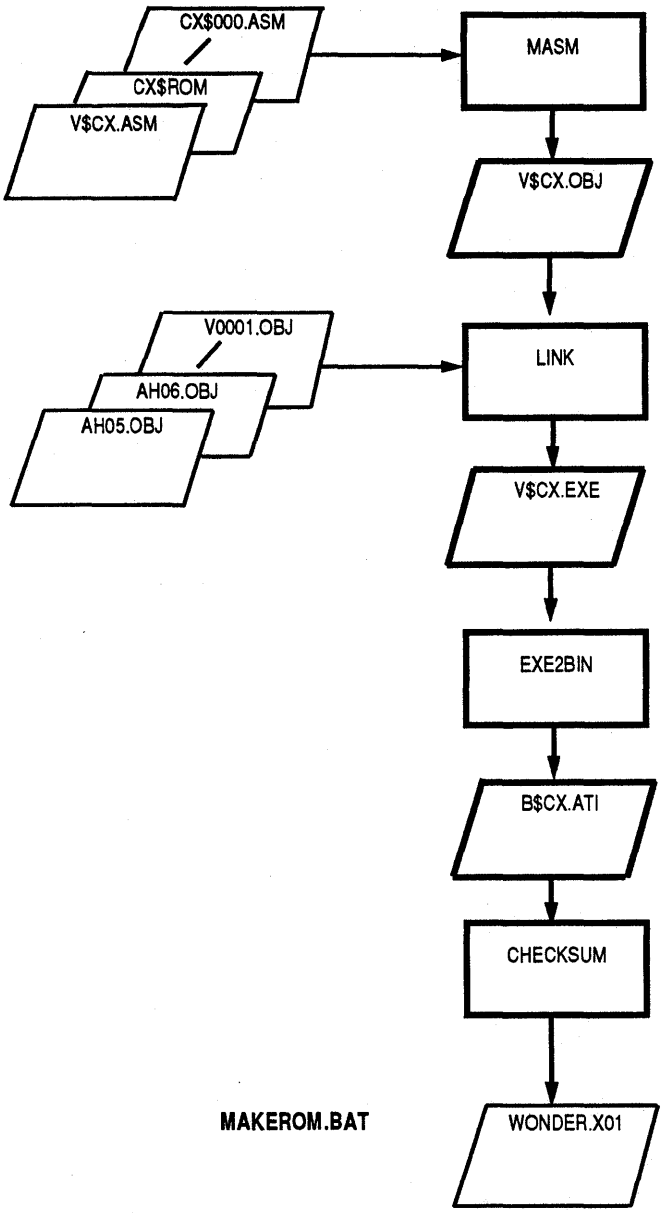


Figure 3-1. Non-Paged BIOS File Generation

Making Non-Paged BIOS

A batch file is provided in the BIOS kit for generating binary files, which in turn are used to program BIOS ROMs. The file is called **MAKEROM**. The output of the processing is a binary file named **WONDER.X01**, which is then used for programming the BIOS ROM.

Files and copying requirements for BIOS generation are provided in *Chapter 2*. In order to use **MAKEROM.BAT**, the following conditions must be satisfied:

- The directory structure for source and object files for the non-paged BIOS (in \VGAOBJ directory) must be identical to the directory structure of the provided diskette.
- Programs such as Microsoft Macro Assembler (MASM 5.1), Microsoft Object Linker (LINK), and EXE2BIN.EXE must be available.

Binary Files - Procedure Outline

1. Assemble your customized source modules using Microsoft Macro Assembler MASM.
2. Link the output from step 1 with the provided object modules to create an EXE file.
3. Convert the EXE file from step 2 to a binary file called B$<Par1><Mouse>.<Par2>. The parameters are explained in the following section.
4. Use the supplied checksum program, CHECKSUM, to place a correct checksum and timestamp (at offset 0X50h) in the binary file.
5. Rename this binary file to **WONDER.X01**.
6. Program the **WONDER.X01** file into a video ROM.

Generating Binary Files - Parameters

MAKEROM.BAT file is easy to use. Each file takes source modules and produces the required binary file **WONDER.X01**. You need supply only the required parameters.

<Par1>[M].<Par2> should be the name of the file containing the required object modules in the directory called \CXROM\CONFIG.

Sample files are provided in the most commonly used configurations. You can use them for reference and either customize or create files based upon their requirements. Batch file commands and parameters are as follows:

MAKEROM <Par1> <Par2> [M]

- <Par1> = **CX for ATI88800CX**
  GX for ATI88800GX
- <Par2> = ISA for ISA
  <Par2> = PCI for PCI
  <Par2> = VLB for VLB

Example 1:

If you use an **ATI88800CX** controller for ISA bus, the parameters for generating the BIOS file would be:

MAKEROM CX ISA

Example 2:

If you use an **ATI88800CX** controller for VLB bus, the command for generating the BIOS file would be:

MAKEROM CX VLB

Example 3:

If you use an **ATI88800CX** controller for PCI bus, the parameters for generating the BIOS file would be:

MAKEROM CX PCI

Example 4:

If you use an **ATI88800GX** controller for ISA bus, the parameters for generating the BIOS file would be:

MAKEROM GX ISA

Example 5:

If you use an **ATI88800GX** controller for VLB bus, the command for generating the BIOS file would be:

MAKEROM GX VLB

Example 6:

If you use an **ATI88800GX** controller for PCI bus, the parameters for generating the BIOS file would be:

MAKEROM GX PCI

This page intentionally left blank.

A

BIOS Function Calls

VGA Controller

AH = 0 **;set video mode** (AL = video mode)

AL	MODE/TYPE	RESOLUTION	DIM/COLOR	START ADDRESS
IBM Compatible Modes:				
00h	color/alpha	640x200	40x25/BW	B800h:0
01h	color/alpha	640x200	40x25/16	B800h:0
02h	color/alpha	640x200	80x25/BW	B800h:0
03h	color/alpha	640x200	80x25/16	B800h:0
04h	color/graphics	320x200	40x25/4	B800h:0
05h	color/graphics	320x200	40x25/BW	B800h:0
06h	color/graphics	320x200	80x25/BW	B800h:0
07h	mono/alpha	720x350	80x25/BW	B000h:0
0Dh	color/graphics	320x200	40x25/16	A000h:0
0Eh	color/graphics	640x200	80x25/16	A000h:0
0Fh	mono/graphics	640x350	80x25/BW	A000h:0
10h	color/graphics	640x350	80x25/16	A000h:0
11h	color/graphics	640x480	80x30/BW	A000h:0
12h	color/graphics	640x480	80x30/16	A000h:0
13h	color/graphics	320x200	80x25/256	A000h:0

AL	MODE/TYPE	RESOLUTION	DIM/COLOR	START ADDRESS
ATI Enhanced Modes:				
21h	color/alpha	100x25	800x400	B800h:0
22h	color/alpha	100x30	800x480	B800h:0
23h	color/alpha	1056x200	132x25/16	B800h:0
27h	mono/alpha	1056x200	132x25/BW	B000h:0
33h	color/alpha	1056x352	132x44/16	B800h:0
37h	mono/alpha	1056x352	132x44/BW	B000h:0
62h	color/graphics	640x480	80x30/256	A000h:0
63h	color/graphics	800x600	100x42/256	A000h:0
64h	color/graphics	1024x768	128x48/256	A000h:0

AH = 1 **;set cursor type**

CH = start line of cursor
CL = end line of cursor
CX = 1F00h to turn off cursor

AH = 2 **;set current cursor position**

BH = page number of the desired page
DH, DL = row and column of cursor

AH = 3 **;read current cursor position at the specified page**

BH = page number of the desired page
on exit:
CH, CL = cursor type
DH, DL = row, column of cursor at the specified page

AH = 4 **;read current light pen position** (VGA does not support light pen)

AH = 5 **;select active display page**

AL = page number to be active

AH = 6 **;scroll active page up**

AL = number of lines to be scrolled
= 0 ;blanks the whole window
BH = attribute of blanked line
CH, CL = row, column of upper left hand corner of scrolling window
DH, DL = row, column of lower right hand corner of scrolling window

AH = 7 **;scroll active page down**

AL = number of lines to be scrolled
= 0 ;blanks the whole window
BH = attribute of blanked line
CH, CL = row, column of upper left hand corner of scrolling window
DH, DL = row, column of lower right hand corner of scrolling window

AH = 8 **;read character/attribute at current active cursor position**

BH = page number of the desired page
on exit:
AL = character
AH = attribute (for text mode only)

AH = 9 **;write character/attribute at current cursor position of a specified page**

AL = character to be written
BL = attribute of character
BH = page number
CX = count of character to write

AH = 0Ah **;write character at current cursor position of a specified page**

AL = character to be written
BH = page number
CX = count of character to write

AH = 0Bh **;set color palette, valid for modes 4 and 5 only**

BH = 0 ;selects the background color
BL = color value used with that color id
= 1 ;selects the palette to be used
BL = 0 ;palette value is GREEN(1)/RED(2)/BROWN(3)
= 1 ;palette value is CYAN(1)/MAGENTA(2)/WHITE(3)

AH = 0Ch **;write dot (graphics mode)**

BH = page number
DX, CX = row, column of dot position
AL = color value of dot (if bit 7 of AL is ON, the color value will be XOR'd with the current value of the dot)

AH = 0Dh **;read dot (graphics mode)**

BH = page number
DX, CX = row, column of dot position
on exit
AL = color value of dot

AH = 0Eh **;write teletype to active page**

AL = character to write
BL = foreground color in graphics mode

AH = 0Fh **;return current video setting**

on exit:
AL = current mode
AH = number of column (in characters) on screen
BH = current active display page

AH = 10h **;set palette registers**

AL = 0 ;set individual palette register
BL = palette register
BH = palette value
AL = 1 ;set overscan register
BH = palette value
AL = 2 ;set all palette and overscan registers
ES:DX = pointer to palette value table (17 bytes long), bytes 0 - 15 are palette values for 16 palette registers, byte 16 is palette value for the overscan register
AL = 3 ;toggle between intensity/blinking bit
BL = 0 ;set intensity on
BL = 1 ;set blinking on
AL = 7 ;read individual palette register
BL = palette register
on exit:
BH = palette value
AL = 8 ;read overscan register
on exit:
BH = overscan value
AL = 9 ;read all palette and overscan registers
ES:DX = pointer to 17-byte buffer
on exit:
ES:DX = pointer to palette value table (17 bytes long), bytes 0 - 15 are palette values for 16 palette registers, byte 16 is palette value for the overscan register
AL = 10h ;set a color register
BX = color register
DH = red value
CH = green value
CL = blue value
AL = 12h ;set a block of color registers
BX = first color register to be set
CX = total number of color registers to be set
ES:DX = pointer to table of color register values in red, green, blue, red, green, blue ,... format
AL = 13h ;set color pages (only valid for 16 color modes)
BL = 0 ;select color page mode
BH = 0 ;select 4 pages of 64 color registers each
BH = 1 ;select 16 pages of 16 color registers each
BL = 1 ;select color page
BH = color page number
AL = 15h ;read a color register

```
                BX = color register
                on exit:
                DH = red value
                CH = green value
                CL = blue value
AL =17h         ;read a block of color registers
                BX = first color register to be set
                CX = total number of color registers to be set
                ES:DX = pointer to buffer to store the color register values
                on exit:
                ES:DX = pointer to table of color register values in red, green, blue, red, green,
                        blue, ..., format
AL = 1Ah        ;read current color page information
                BL = current color page mode
                BH = current color page
AL = 1Bh        ;change color values to gray shades
                BX = first color register to be changed
                CX = total number of color registers to be changed
```

AH=11h ;character generator routines

```
AL = 00         ;load user specified character set
                ES:BP = pointer to character table
                CX = number of characters to be stored
                DX = character of offset into current table
                BL = block to load
                BH = bytes per character
AL = 01         ;load 8x14 character set
                BL = block to load
AL = 02         ;load 8x8 character set
                BL = block to load
AL = 03         ;set block specifier
                BL = character generator block specifier
AL = 04         ;load 8x16 character set
                BL = block to load
```

The function AL = 1? is similar in function to AL = 0? except the number of rows on the screen is recalculated.

```
AL = 10h        ;load user specified character set
                ES:BP = pointer to character table
                CX = number of characters to be stored
                DX = character of offset into current table
                BL = block to load
                BH = bytes per character
AL = 11h        ;load 8x14 character set
                BL = block to load
AL = 12h        ;load 8x8 character set
                BL = block to load
AL = 14h        ;load 8x16 character set
                BL = block to load
```

```
        AL = 20h    ;update alternative character generator pointer (INT 1F)
                    ES:BP  =  pointer to table
        AL = 21h    ;update alternative character generator pointer (INT 43)
                    ES:BP  =  pointer to table
                    CX  =  bytes per character
                    BL  =  row specifier
                        =  0  ;DL = rows
                        =  1  ;rows = 14
                        =  2  ;rows = 25
                        =  3  ;rows = 43
        AL = 22h    ;update alternative character generator pointer (INT 43) with the 8x14 character
                    generator in ROM
        AL = 23h    ;update alternative character generator pointer (INT 43) with the 8x8 character
                    generator in ROM
        AL = 24h    ;update alternative character generator pointer (INT 43) with the 8x16 character
                    generator in ROM
        AL = 30h    ;return EGA character generator information
                    BH  =  0  ;return current INT 1F pointer
                        =  1  ;return current INT 43 pointer
                        =  2  ;return pointer to 8x14 character generator
                        =  3  ;return pointer to 8x8 character generator (lower)
                        =  4  ;return pointer to 8x8 character generator (upper)
                        =  5  ;return pointer to alternate 9x14 alpha
                        =  6  ;return pointer to 8x16 character generator
                        =  7  ;return pointer to alternate 9x16 alpha
                    on exit:
                    ES:BP  =  pointer to table as requested
                    CX  =  points (pixel column per char)
                    DL  =  rows (scan line per char)

AH = 12h            ;return current EGA settings/print screen routine selection

        BL = 10h    ;return EGA information
                    on exit:
                    BH  =  0  ;color mode in effect
                        =  1  ;monochrome mode in effect
                    BL  =  3  ;256k video memory installed (always return 3)
                    CH  =  simulated value of feature bits
                    CL  =  simulated EGA/VGA dip switch setting
        BL = 20h    ;select alternate print screen routine for EGA graphics mode
        BL = 30h    ;select number of scan lines for alpha modes
                    AL  =  0  ;200 scan lines
                        =  1  ;350 scan lines
                        =  2  ;400 scan lines
                    on exit:
                    AL = 12h  ;function supported
        BL = 31h    ;default palette loading during mode set
                    AH  =  0
                    AL  =  0  ;enable
                        =  1  ;disable
```

```
                on exit:
                AL = 12h  ;function supported
BL = 32h        ;video controller
                AL  =  0  ;enable video controller
                    =  1  ;disable video controller
                on exit:
                AL = 12h  ;function supported
BL = 33h        ;summing of color registers to gray shades
                AL  =  0  ;enable summing
                    =  1  ;disable summing
                on exit:
                AL = 12h  ;function supported
BL = 34h        ;cursor emulation
                AL  =  0  ;enable cursor emulation
                    =  1  ;disable cursor emulation
                on exit:
                AL = 12h  ;function supported
BL = 36h        ;video screen on/off
                AL  =  0  ;video screen on
                    =  1  ;video screen off
                on exit:
                AL = 12h  ;function supported
BX=5506h        ;VGAWONDER BIOS extension
                AL  =  video mode
                BP  =  0FFFFh
                DI  =  0
                SI  =  0
                on exit:
                if BP is not equal to 0FFFFh
                    then EP:BP = pointer to parameter table
                if SI is not equal to 0
                    then EP:SI = pointer to parameter table supplement
```

AH = 13h **;write string to specified page**

```
ES:BP = pointer to string
CX  =  length of string
BH  =  page number
DH,DL =  starting row and column of cursor in which the string is placed
AL  =  0  ;cursor is not moved
          BL  =  attribute
          string = (char, char, char, char, ...)
AL  =  1  ;cursor is moved
          BL  =  attribute
          string = (char, char, char, char, ...)
AL  =  2  ;cursor is not moved
          string = (char, attr, char, attr, ...)
AL  =  3  ;cursor is moved
          string = (char, attr, char, attr, ...)
```

AH=1Ah ;read display combination code

AL = 0 ;read current display combination information
on exit
AL = 1Ah
BL = current active display code
BH = alternate display code
Display codes
00 - No display
01 - MDA mode
02 - CGA mode
04 - EGA in color mode
05 - EGA in monochrome mode
07 - VGA with analog monochrome monitor
08 - VGA with analog color monitor

AH=1Bh ;return VGA functionality and state information

BX = 0 ;
ES:DI = pointer to buffer used to store the functionality and state information (mininum 64 bytes)
on exit:
AL = 1Bh
ES:DI = pointer to buffer with functionality and state information
[DI+00h] word = offset to static functionality information
[DI+02h] word = segment to static functionality information
[DI+04h] byte = current video mode
[DI+05h] word = character columns on screen
[DI+07h] word = page size in number of bytes
[DI+09h] word = starting address of current page
[DI+0Bh] word = cursor position for eight display pages
[DI+1Bh] word = current cursor type
[DI+1Dh] byte = current active page
[DI+1Eh] word = current CRTC address
[DI+20h] byte = current 3x8 register setting
[DI+21h] byte = current 3x9 register setting
[DI+22h] byte = number of character rows on screen
[DI+23h] word = number of scan lines per character
[DI+25h] byte = active display combination code
[DI+26h] byte = alternate display combination code
[DI+27h] word = number of colors supported in current mode
[DI+29h] byte = number of pages supported in current mode
[DI+2Ah] byte = 0 ;200 scan lines in current mode
= 1 ;350 scan lines in current mode
= 2 ;400 scan lines in current mode
= 3 ;480 scan lines in current mode
[DI+2Bh] byte = reserved
[DI+2Ch] byte = reserved
[DI+2Dh] byte = miscellaneous state information
bits 7, 6 = reserved

bit 5 = 0 ;background intensity
= 1 ;blinking
bit 4 = 1 ;cursor emulation active
bit 3 = 1 ;mode set default palette loading disabled
bit 2 = 1 ;monochrome display attached
bit 1 = 1 ;summing active
bit 0 = 1 ;all modes on all display active

[DI+2Eh] byte = reserved
[DI+2Fh] byte = reserved
[DI+30h] byte = reserved
[DI+31h] byte = 3;256Kb of video memory available
[DI+32h] byte = save pointer information
bits 7, 6 = reserved
bit 5 = 1 ;DCC extension active
bit 4 = 1 ;palette override active
bit 3 = 1 ;graphics font override active
bit 2 = 1 ;alpha font override active
bit 1 = 1 ;dynamic save area active
bit 0 = 1 ;512 character set active
[DI+33h] 13 bytes = reserved

static functionality table format
0 - function not supported
1 - supported function

[00h] byte = supported video mode
bit 7 = mode 07h
bit 6 = mode 06h
bit 5 = mode 05h
bit 4 = mode 04h
bit 3 = mode 03h
bit 2 = mode 02h
bit 1 = mode 01h
bit 0 = mode 00h
[01h] byte = supported video mode
bit 7 = mode 0Fh
bit 6 = mode 0Eh
bit 5 = mode 0Dh
bit 4 = mode 0Ch
bit 3 = mode 0Bh
bit 2 = mode 0Ah
bit 1 = mode 09h
bit 0 = mode 08h
[02h] byte = supported video mode
bits 7 to 4= reserved
bit 3 = mode 13h
bit 2 = mode 12h
bit 1 = mode 11h
bit 0 = mode 10h
[03h] to [06h] = reserved

[07h] = scan lines availabe in text modes
bits 7 to 3 = reserved
bit 2 = 400 scan lines
bit 1 = 350 scan lines
bit 0 = 200 scan lines
[08h] = number of character fonts available in text modes
[09h] = maximun number of character fonts that can be active in text modes
[0Ah] byte = miscellaneous functions
bit 7 = color paging
bit 6 = color palette (color register)
bit 5 = EGA palette
bit 4 = cursor emulation
bit 3 = default palette loading when mode set
bit 2 = character font loading
bit 1 = color palette summing
bit 0 = all modes supported on all displays
[0Bh] = scan lines availabe in text modes
bits 7 to 4 = reserved
bit 3 = DCC supported
bit 2 = background intensity/blinking control
bit 1 = save/restore supported
bit 0 = light pen supported
[0Ch] to [0Dh] = reserved
[0Eh] = save pointer fucntions
bits 7 to 6 = reserved
bit 5 = DCC extension supported
bit 4 = palette override
bit 3 = graphics font override
bit 2 = alpha font override
bit 1 = dynamic save area
bit 0 = 512-character set
[0Fh] = reserved

AH=1Ch **;save and restore video state**

AL = 0 ;return video save state bufffer size requirement
CX = requested states
bit 0 = video hardware state
bit 1 = video BIOS data area
bit 2 = video DAC state and color registers
on exit:
AL = 1Ch
BX = number of 64 bytes block required for the states requested in CX
AL = 1 ;save video state
CX = requested states (see AL=0)
ES:BX=pointer to buffer to store the video states information
on exit:
AL = 1Ch
AL = 2 ;restore video state
CX = requested states (see AL=0)

ES:BX = pointer to buffer with previous saved video states information
on exit:
AL = 1Ch

Extended ROM Services

BIOS Support

The Graphics Accelerator BIOS has a special entry function to support *Set Mode* in the coprocessor mode. This function reduces the development efforts for programmers writing coprocessor mode screen drivers.

The benefits of usingfunction calls in the Graphics Accelerator are numerous:

- Using function calls reduces development time as well as the complexity of the driver.
- Can be used in protected mode, 16-bit only.
- The interface is upward-compatible and can be expanded to support 800x600, 1280x1024, and 1600x1200.
- Version-specific hardware code goes with the firmware.

Calculating ROM Base Address

The ROM base address is calculated as follows:

xxxx = (SCRATCH_REG1 & 0x7F) * 0x80 + 0xC000

where SCRATCH_REG1 is 046ECh.

Function Calls

Base ROM address is determined by the register SCRATCH_REG1 (46ECh) and the ROM services are accessible by absolute calls at this address with the following instructions:

CALL XXXX:64h

Another way to invoke the extended ROM service is by calling a INT 10h with AH = A0h. The support of INT 10h is also available with VGA disabled mode. The only requirement is that the primary adapter has to be a VGA. No CGA or monochrome card can be supported.

XXXX:64h

all functions return with error code in AH
ah = 0 ;no error

```
        ah = 1    ;function complete with error
        ah = 2    ;function not supported

    al = 0    ;load Coprocessor CRTC parameters
              cl[3-0]   = color depth
                        = 1     ;4bpp
                        = 2     ;8bpp
                        = 3     ;15bpp (555)
                        = 4     ;16bpp (565)
                        = 5     ;24bpp
                        = 6     ;32bpp
              cl[5]     = 1     ;enable gamma correction if 15bpp and above
                                ;set the RAMDAC to 8-bit if in 8bpp mode, for
                                support of 256 color greyscale
              cl[7-6]   = pitch size
                        = 0     ;1024
                        = 1     ;don't change
                        = 2     ;pitch size is the same as horizontal display
              ch        = resolution
                        = 12h   ;640x480
                        = 6ah   ;800x600
                        = 55h   ;1024x768
                        = 80h   ;load table from offset of external storage
                                (EEPROM) in bx
                        = 81h   ;load table according to data in dx:bx
                        = 82h   ;OEM specific mode
                        = 83h   ;1280x1024
              dx:bx     = pointer to parameter table if ch = 81h
                 bx     = offset into EEPROM table if ch = 80h

al = 1    ;set display mode
          cl[0] = 0     ;VGA and set the RAMDAC to 6 bits
                = 1     ;Coprocessor
          cl[7] = 1     ;enable 8-bit DAC or Gamma Correction
                        ;this bit is or with cl[5] in function AL=0

al = 2    ;load Coprocessor CRTC parameters and set display mode
          same agrument as al = 0

al = 3    ;read EEPROM data
          bx = index
          returns
          bx = data

al = 4    ;write EEPROM data
          bx = index
          dx = data

al = 5    ;memory aperture service
          cl = 0        ;disable memory aperture
          cl[0] = 1     ;enable memory aperture
          cl[2] = 1     ;enable VGA memory aperture
```

al = 6 ;short query function
al[5-0] = aperture configuration
= 0 ; disable
= 1 ; 4M
= 2 ; 8M
al[6] = 0 ;aperture address is user configurable
= 1 ;aperture address is predefined or hard coded in BIOS
al[7] = 0 ;aperture address is in 128M range
= 1 ;aperture address is in 4G range
bx = aperture address
ch = color deep support (see offset 13 in query structure)
cl = memory size
dx = ASIC identification, [7-0]=revision, [15-8]=type

al = 7 ;return hardware capability list
in return dx:bx = offset into a table specifying the maximum dot clock information, the table is terminated by a zero in the first column

al = format type
= 0

H_DISP	DACMASK	MEMREQ	MAXIMUM DOT CLOCK	PIXEL WIDTH
0				

H_DISP = horizontal resolution in number of characters
DACMASK = (1 shl dactype)
MEMREQ = the minimum memory required to support the specified resolution and color depth
(DRAM requirement shl 4) or (VRAM requirement)
MAX DOTCLOCK = maximum dot clock with the specified resolution and color depth in MHz
PIXEL WIDTH = color depth

al = 8 ;return query device data structure in bytes
on entry
cl[0] = 0 ;buffer size for header information only
= 1 ;buffer size for header information and mode tables
return
cx = number of bytes

al = 9 ;query device
dx:bx = pointer to buffer

```
                cl[0] = 0   ;return header information only
                      = 1   ;return header information and mode table

al = 0ah        ;return clock chip frequency table
                al = clock chip type
                dx:bx = offset pointing to the 16 words containing the pre-programmed
                        dot clock frequency, unit is in KHz/10 (4 significant digits)
                dx:cx = offset pointing to the table containing clock chip information in
                        the following format:
                            db      clock chip type
                            db      frequency table identification
                            db      user programmable entry if <>0ffh
                            db      reserved
                            dw      hardware dependent information
                            dw      minfreq, maxfreq (in KHz/10)

al = 0bh        ;program a specified clock entry
                ch = entry in the frequency table
                bx = unit is in KHz/10
                in return
                    al = clock chip type
                    bx = programming word depending on type

al = 0ch        ;DPMS service, set DPMS mode
                cl[2-0] = 0   ;active
                        = 1   ;stand-by
                        = 2   ;suspend
                        = 3   ;off
                        =     ;blanking the display only

al = 0dh        ;return current DPMS state in cl

al = 0eh        ;set Graphics Controller's Power Management state

al = 0fh        ;return current Graphics Controller's Power Management state

al = 10h        ;set the RAMDAC to different states
                cl = 80h      ;reserved
                cl[0] = 0     ;set RAMDAC to normal mode
                cl[1] = 1     ;set RAMDAC to sleep mode

al = 11h        ;return external storage device information, INSTALL should use this
                 information for dynamic configuration of the data structure
                cl = external data structure information
                cl[7] = 1     ;no external data storage can be used, everything is
                              pre-defined
                      = 0     ;external data storage available
                cl[6-4] = 000 ;external data is readable and writeable
                        = 001 ;external data storage is readable but not writeable
                        = 011 ;external data storage is not readable and writeable
                        = 100 ;external data storage is readable and writeable, the
                              writing has to be handled by the application program
                              based on device type in cl[3-0]
```

cl[3-0] = 0 ;device type
ch = number of 16 bit entry in the storage device
dh = number of 16 bit entry in the storage device, these entries are read only
bl = offset into the CRTC parameter table
bh = size of the CRTC parameter table, if the number is smaller than the one in the CRTC table, then discard the bottom ones

al = 12h ;short query
on return
ax = reserved
bx = reserved
cx = reserved
dx = I/O Base Address

al = 13h address of ADC implementation

Query Structure

Offset	Description
0 - 1	Size of structure in bytes
2	Revision of structure
3	Number of mode tables
4-5	Offset in bytes to mode tables
6	Size of each mode table in bytes
7	VGA Type 0 = disabled 1 = enabled
8 - 9	ASIC identification
0Ah	VGA Boundary 0 = full access 1 = 256K 2 = 512K 3 = 768K 4 = 1M 10h = no access through VGA
0Bh	Memory Size 0 = 512K 1 = 1M 2 = 2M 3 = 4M 4 = 6M 5 = 8M
0Ch	Bits 3-0, DAC Type 0 = reserved 1 = reserved 2 = TI 34075/ATI68875 3 = Brooktree BT476/8 4 = Brooktree BT481, AT&T20C490/491, SC15025/ 15026, IMS-G174, MU9C4910, MU9C1880 5 = ATI68860 6 = STG1700 7 = STG1702/SC15021 bits 7-4 = reserved

continued...

Offset	Description
0Dh	Memory Type 0 = DRAM 256Kx16 1 = VRAM 256Kx4 2 = VRAM 256Kx16 3 = DRAM 256Kx4 5 = VRAM 256Kx4 special 6 = VRAM 256Kx16 special
0Eh	Bus Type 0 = ISA 1 = EISA 2 = reserved 3 = reserved 4 = reserved 5 = reserved 6 = VLB 7 = PCI
0Fh	Bit 7 - enable composite sync Bit 6 - enable sync on green
10h-11h	Aperture address in megabytes (0-4095)
12h	Aperture Configuration (see extended BIOS function al=6)
13h	Color Deep Support Bit Definition 7 = 1 ;reserved 6 = 1 ;reserved 5 = 1 ;reserved 4 = 1 ;support 32bpp (unpack 24bpp) 3 = 1 ;support BGR in 24bpp 2 = 1 ;support RGB in 24bpp 1 = 1 ;support 16bpp, 555 0 = 1 ;support 16bpp, 565
14h	RAMDAC Support Feature Bit Definition 7 = 1 ;support sync on green 6 = 1 ;support gamma correction 5 = 1 ;support 256 greyscale 4 = 1 ;support sleep mode
15h	Reserved
16h - 17h	Offset into current mode table if non-zero (not implemented)
18h - 19h	I/O Base Address
1Ah - 1Fh	Reserved

Mode tables immediately follow the device status table. Use the forward pointer to reference mode tables, as the device status table may expand in the future. It is possible to have no modes installed. Typically, between 2 and 7 mode tables will be returned.

Mode Table Structure

Offset	Description
Installed Mode Table 1	
0-1	Horizontal display resolution in pixels
2-3	Vertical display resolution in scanlines
4	Maximum pixel depth (see extended function AL=0, CL[3-0] for interpretation)
5	Mode number (see extended function AL=0, Ch)
6-7	Offset into EEPROM = 0 ;table is generated from VGA parameters <> 0 ;offset into EEPROM table
8-9	Reserved
0Ah-0Bh	Reserved
0Ch-0Dh	Bits 15-14 = reserved Bit 13 = enable MUX mode Bits 12-10 = reserved Bit 9 = enable interlace Bit 8 = enable double scan Bits 7-0 = reserved
0Eh	CRTC_H_TOTAL
0Fh	CRTC_H_DISP
10h	CRTC_H_SYNC_STRT
11h	CRTC_H_SYNC_WID
12h-13h	CRTC_V_TOTAL
14h-15h	CRTC_V_DISP
16h-17h	CRTC_V_SYNC_STRT
18h	CRTC_V_SYNC_WID
19h	CLOCK_CNTL
1Ah-1Bh	Dot clock for coprocessor mode, for programmable clock chip

Byte	Description
1Ch-1Dh	Bits 15-12 = Reserved Bits 11-8 = CRTC_H_SYNC_DLY Bits 7-4 = OVR_WID_RIGHT Bits 3-0 = OVR_WID_LEFT
1Eh-1Fh	OVR_WID_TOP, OVR_WID_BOTTOM
20h-21h	OVR_CLR_B, OVR_CLR_8
22h-23h	OVR_CLR_G, OVR_CLR_R
Installed Mode Table 2	
24h-47h	Entries definition same as mode table 1.
. . .	
Installed Mode Table n	
N*24h- (N*24+23h)	Entries definition same as mode table 1.

This page intentionally left blank.

B

EEPROM DATA

ROM ENTRIES

Information required for programming the graphics controller is stored in the video ROM. With the ATI BIOS, extra sets of values called parameter tables are stored in the EEPROM. For your convenience, EEPROM entries are explained in this appendix as functional units, for example, parameter tables.

Using a configuration program, users can customize the size and position of the extended video mode displays on their monitors and store the values in parameter tables. The BIOS will then use them instead of the values from standard tables defined for a selected monitor type. This feature allows users to change monitors and still have perfect screen alignment without changing the existing BIOS ROM.

ATI88800CX EEPROM Data Structure

Offset	Bits	Description
0	15 - 0	EEPROM Write Counter.
1	15 - 8 7 - 0	Reserved. EEPROM checksum, modular 8 of 8-bit data, the summation of all the entries in the EEPROM must be 0.
2	15 - 0	Reserved. No application program should touch this entry. Factory default should set field to 0.
3	15 - 4 3 - 0	Reserved. EEPROM table revision.
4	15 - 0	Custom monitor indices.
5	15 - 8 7 6 5 - 2 1 0	1280x1024 refresh rate information = 1; use stored 640x480 coprocessor parameters for coprocessor mode = 1; enable 640x480 72Hz Reserved Enable sync on green Enable composite sync
6	15 - 8 7 6 5 4 3 2 1 0	Reserved = 1; use stored 800x600 coprocessor parameters for coprocessor mode Reserved = 1; select 800x600 in 72Hz = 1; select 800x600 in 70Hz = 1; select 800x600 in 60Hz = 1; select 800x600 in 56Hz = 1; select 800x600 in 89Hz interlaced = 1; select 800x600 in 85Hz interlaced
7	15 - 8 7 6 - 4 3 2 1 0	Reserved = 1; use stored 1024x768 coprocessor parameters for coprocessor mode Reserved = 1; select 1024x768 in 72Hz = 1; select 1024x768 in 70Hz = 1; select 1024x768 in 60Hz = 1; select 1024x768 in 87Hz interlaced

continued ...

Offset	Bits	Description
8	15 - 8	Power Up Video Mode 03h = VGA color - secondary 05h = VGA monochrome - secondary 09h = VGA color - primary 0Bh = VGA monochrome - primary
	7 - 6	Monochrome Mode Color Select 0 = white 1 = green 2 = amber
	5	Dual Monitor Enable
	4	Font Selection at Power Up 0 = 8x14 or 9x14 1 = 8x16 or 9x16
	3	VGA Bus I/O 0 = 8 bits 1 = 16 bits
	2	Zero Waite State RAM 0 = disable 1 = enable
	1	Zero Waite State ROM 0 = disable 1 = enable
	0	16 bits ROM 0 = disable 1 = enable
9	15 - 14	Host Data Transfer Width 0 = auto select 1 = 16-bit 2 = 8-bit 3 = 8-bit host/16-bit other
	13 - 8	Monitor Code
	7 - 6	Reserved
	5 - 4	VGA Boundary 0 = no boundary 1 = 512K 2 = 1M
	3	Monitor Alias Enable
	2 - 0	Monitor Alias
A	15 - 4	Aperture Location (in MByte)
	3 - 0	Aperture Size (will not be used by the BIOS if Aperture Location is non-zero, assume the aperture will be enabled, the aperture size will be based on video memory size)

continued ...

Offset	Bits	Description
B	15 - 8	Mouse Address 00h = mouse disable 08h = secondary address selected 18h = primary address selected
	7 - 0	Interrupt Level 20h = IRQ 5 28h = IRQ 4 30h = IRQ 3 38h = IRQ 2
0Ch - 16h		Reserved
17h - 25h		CRT Parameter Table 1
26h - 34h		CRT Parameter Table 2
35h - 43h		CRT Parameter Table 3
44h - 52h		CRT Parameter Table 4
53h - 61h		CRT Parameter Table 5
62h - 70h		CRT Parameter Table 6
71h - 7Fh		CRT Parameter Table 7

ATI88800CX CRT Parameter Table

Offset	Bits	Description
0	15 - 8 7 - 0	Video Mode Select 1 / Reserved Video Mode Select 2 / Reserved
1	15 - 8 7 - 0	Video Mode Select 3 / Video Mode Select CRT refresh rate bit mask / (set to 0x80 for coprocessor mode)
2	15 - 14 13 12-10 9 8 7 6 5 4 3-0	Reserved Enable MUX mode Reserved Enable interlace Enable double scan Vertical Sync Polarity Horizontal Sync Polarity Reserved CRT Usage (VGA only) 0 = use sync polarities only 1 = use all CRT parameters Reserved
3	15 - 8 7 - 0	MAX_SCAN_LINE (CRT09) / CRTC_H_DISP H_TOTAL (CRT00) / CRTC_H_TOTAL
4	15 - 8 7 - 0	H_RETRACE_END (CRT05) / CRTC_H_SYNC_WID H_RETRACE_STRT (CRT04)/CRTC_H_SYNC_STRT
5	15 - 8 7 - 0	V_RETRACE_END (CRT11) / CRTC_V_TOTAL (15 - 8) V_RETRACE_STRT (CRT10) / CRTC_V_TOTAL (7 - 0)
6	15 - 8 7 - 0	H_BLANK_END (CRT03) / CRTC_V_DISP (15 - 8) H_BLANK_STRT (CRT02) /CRTC_V_DISP (7 - 0)
7	15 - 8 7 - 0	V_BLANK_END (CRT16) / CRTC_V_SYNC_STRT (15 - 8) V_BLANK_STRT (CRT15) /CRTC_V_SYNC_STRT (7 - 0)
8	15 - 8 7 - 0	CRTC_OVERFLOW (CRT07) / CLOCK_CNTL if == 0ffh or == programmable entry in clock chip, use dot clock in entry 9 and programmable entry in dot clock V_TOTAL (CRT06) / CRTC_V_SYNC_WIDTH
9	15 - 8 7 - 0	V_DISP_END (CRT 12) / DOT CLOCK (15 - 8) CRT_MODE (CRT17) / DOT CLOCK (7 - 0)

continued ...

Offset	Bits	Description
A	15 - 0	Bits 15 - 12 = CRTC_H_TOTAL_DLY Bits 11 - 8 = CRTC_H_SYNC_DLY Bits 7 - 4 = OVR_WID_RIGHT Bits 3 - 0 = OVR_WID_LEFT
B	15 - 0	OVR_WID_TOP, OVR_WID_BOTTOM
C	15 - 0	OVR_CLR_B, OVR_CLR_8
D	15 - 0	OVR_CLR_G, OVR_CLR_R
E	15 - 0	Reserved

VESA BIOS EXTENSIONS

The *mach64* product family has the VESA BIOS extension VP911922 implemented in the ROM. The VESA BIOS supports 256 color and hicolor modes through the extension. A brief description of the VESA BIOS functions is included for completeness. For detailed information, please refer to the original, published documents.

Status Information

Every function returns status information in the AX register. The format of the status word is as follows:

AL= = 4Fh:Function is supported
AL != 4Fh:Function is not supported
AH= = 00h:Function call successful
AH= = 01h:Function call failed

Software should treat a non-zero value in the AH register as a general failure condition. In later versions of the *VESA BIOS Extensions*, new error codes may be defined.

Function 00h - Return Super VGA information

Input: AH=4Fh Super VGA support
AL=02h **Return Super VGA information**
ES:DI = Pointer to 256-byte buffer

Output: AX= Status
All other registers are preserved.

The information block has the following structure:

```
VgaInfoBlock struc
    VESASignature   db   'VESA'      ;4 signature bytes
    VESAVersion     da   ?102h       ;VESA version number
    OEMStringPtr    dd   ?           ;Pointer to OEM string
    Capabilities    db   4 dup (?)   ;Capabilities of the video
                                     environment
    VideoModePtr    dd   ?           ;Pointer to supported Super VGA
                                     modes
    TotalMemory     dw   ?           ;Number of 64Kb memory blocks
                                     on board
    Reserved        db   242 dup (?) ;Remainder of VGAInfoBlock
VgaInfoBlock ends
```

- The **VESASignature** field contains the characters **VESA** if this is a valid block.
- **VESAVersion** is a binary field that specifies what level of the VESA standard the Super VGA BIOS conforms to.
- **OEMStringPtr** is a far pointer to a null-terminated, OEM-defined string that currently points to **ATI MACH64**.
- The **Capabilities** field describes the general features supported in the video environment. The bits are defined as follows:

```
D0      =  DAC is switchable
           0 =  DAC is fixed-width, with 6 bits per primary color
           1 =  DAC width is switchable
D1 -31  =  Reserved
```

- The **VideoModePtr** points to a list of supported Super VGA (VESA-defined as well as OEM-specific) mode numbers. Each mode number occupies one word (16 bits). The list of mode numbers is terminated by a -1 (0FFFh) The pointer could point into either ROM or RAM, depending on the specific implementation. Either the list would be a static string stored in ROM, or the list would be generated at run-time in the information block (see above in RAM). It is the application's responsibility to verify the current availability of any mode returned by this function, through the **Return Super VGA mode information** (Function 1) call. Some returned modes may not be available, due to the video board's current memory and monitor configuration.
- The **TotalMemory** field indicates the amount of memory installed on the VGA board. Its value represents the number of 64Kb blocks of memory currently installed.

SUPER VGA MODE NUMBERS

Supported VESA Modes

The following VESA modes are supported:

Graphics

15-bit Mode Number	7-bit Mode Number	Resolution	Colors
101h	—	640x480	256
103h	—	800x600	256
105h	—	1024x768	256
110h	—	640x480	32K (5:5:5)
111h	—	640x480	64K (5:6:5)
112h	—	640x480	16.8M (8:8:8)
113h	—	800x600	32K (5:5:5)
114h	—	800x600	64K (5:6:5)
115h	—	800x600	16.8M (8:8:8)
116h	—	1024x768	32K (5:5:5)
117h	—	1024x768	64K (5:6:5)
118h	—	1024x768	16.8M (8:8:8)
119h	—	1280x1024	32K (5:5:5)
11Ah	—	1280x1024	64K (5:6:5)
11Bh	—	1280x1024	16.8M (8:8:8)

Function 01h - Return Super VGA Mode Information

This function returns information about a specific Super VGA video mode.

Input:	AH=4Fh	
	AL=01h	
	CX=	Super VGA video mode*
	ES:DI=	Pointer to 256-byte buffer
Output:		AX= Status
	All other registers are preserved	

*. Mode number must be one of those returned by Function 0

The mode information block has the folowing structure:

```
ModeInfoBlock struc

;mandatory information
        ModeAttributes      dw  ?   ;mode attributes
        WinAAttributes      db  ?   ;window A attributes
        WinBAttributes      db  ?   ;window B attributes
        WinGranularity      dw  ?   ;window granularity
        WinSize             dw  ?   ;window size
        WinASegment         dw  ?   ;window A start segment
        WinBSegment         dw  ?   ;window B start segment
        WinFuncPtr          dd  ?   ;pointer to window function
        BytesPerScanLine    dw  ?   ;bytes per scan line
```

```
;formerly optional information (now mandatory)
        XResolution         dw  ?   ;horizontal resolution
        YResolution         dw  ?   ;vertical resolution
        XCharSize           db  ?   character cell width
        YCharSize           db  ?   character cell height
        NumberOfPlanes      db  ?   number of memory planes
        BitsPerPixel        db  ?   bits per pixel
        NumberOfBanks       db  ?   number of banks
        MemoryModel         db  ?   memory model type
        BankSize            db  ?   bank size, in Kb
        NumberOfImagePages  db  ?   number of images
        Reserved            db  1   reserved for page function
```

```
;New Direct Color Fields
        RedMaskSize             db    ?       ;size of direct color red mask, in bits
        RedFieldPosition        db    ?       ;bit position of lsb of red mask
        GreenMaskSize           db    ?       ;size of direct color green mask, in
                                                bits
        GreenFieldPosition      db    ?       ;bit position of lsb of green mask
        BlueMaskSize            db    ?       ;size of direct color blue mask, in bits
        BlueFieldPosition       db    ?       ;bit position of lsb of blue mask
        RsvdMaskSize            db    ?       ;size of direct color reserved mask, in
                                                bits
        RsvdFieldPosition       db    ?       ;bit position of lsb of reserved mask
        DirectColorModeInfo     db    ?       direct color mode attributes

        Reserved                db    216     ;remainder of ModeInfoBlock
                                      dup
                                      (?)

ModeInfoBlock ends
```

- The **ModeAttributes** field describes certain important characteristics of the video mode.

The field is defined as follows:

```
D0=     Mode supported in hardware:
        0=  mode not supported in hardware
        1=  mode supported in hardware

D1=1    (Reserved)
D2=     Output functions supported by BIOS:
        0=  output functions not supported by BIOS
        1=  output functions supported by BIOS
D3=     Monochrome/color mode (see note below):
        0=  monochrome mode
        1=  Color mode
D4=     Mode type:
        0=  text mode
        1=  graphics mode
D5-D15=Reserved
```

- The **BytesPerScanline** field specifies the number of bytes in each logical scanline. The logical scanline could be equal to or larger than the displayed scanline.
- **WinAAttributes** and **WinBAttributes** describe the characteristics of the CPU windowing scheme, such as whether the windows exist and are read/writeable, as follows:

D0=	Window supported:
	0= window is not supported
	1= window is supported
D1	Window readable:
	0= window is not readable
	1= window is readable
D2	Window writeable:
	0= window is not writeable
	1= window is writeable
D3-D7=	Reserved

If windowing is not supported (bit **D0** = 0) for both Window A and Window B, an application can assume that the display memory buffer resides at the standard CPU address appropriate for the **MemoryModel** of the mode.

- **WinGranularity** specifies the smallest boundary, in KB, on which the window can be placed in the video memory. The value of this field is undefined if Bit D0 of the appropriate **WinAttributes** field is not set.
- **WinSize** specifies the size of the window, in KB.
- **WinASegment** and **WinBSegment** addresses specify the segment addresses where the windows are located in the CPU address space.
- **WinFuncAddr** specifies the address of the CPU video memory windowing function. The windowing function can be invoked either through **VESA BIOS function 05h** or by calling the function directly. A direct call will provide faster access to the hardware paging registers than using Int 10h, and is intended tp be used by high-performance applications. If this field is Null, Function 05h must be used to set the memory window, if paging is supported.
- **XResolution** and **YResolution** specify the height and width of the video mode, in pixels.
- **XCharCellSize** and **YCharCellSize** specify the size of the character cell, in pixels.
- The **NumberOfPlanes** field specifies the number of memory planes available to software in that mode. For standard 16-color VGA graphics, this would be set to 4. For standard packed pixel modes, the field would be set to 1.
- The **BitsPerPixel** field specifies the total number of bits that define the color of one pixel. For example, a standard VGA 4-plane, 16-color graphics mode would have a 4 in his field, and a packed-pixel, 256-color graphics mode would specify

8 in this field. The number of bits per pixel *per plane* can normally be derived by dividing the **BitsPerPixel** field by the **NumberOfPlanes** field.

- The **MemoryModel** field specifies the general type of memory organization used in this mode. The following models have been defined:

00h=	**Text mode**
01h=	**CGA graphics**
02h=	**Hercules graphics**
03h=	**4-plane planar**
04h=	**Packed pixel**
05h=	**Non-chain 4, 256 color**
06h=	**Direct Color**
07h=	**YUV**
08h-0fh=	**Reserved, to be defined by VESA**
10h-ffh=	**To be defined by OEM**

In version 1.1 and earlier of the VESA Super VGA BIOS Extension, OEM-defined Direct Color video modes with pixel formats 1:5:5:5 and 8:8:8:8 were described as a **Packed Pixel** model with 16, 24, and 32 bits per pixel, respectively.

- **NumberOfBanks** is the number of banks in which the scan lines are grouped. This field is set to 1.
- The **BankSize** field specifies the size of a bank, in units of 1KB. This field is set to 0.
- The **NumberOfImagePages** field specifies the number of additional, complete display images that will fit into the memory, at one time, in this mode. The application may load more than one image into the memory if this field is non-zero, and flip the display between
- the **Reserved** field has been defined to support a future VESA BIOS extension feature, and will always be set to 1 in this version.
- The **RedMaskSize**, **GreenMaskSize**, **BlueMaskSize**, and **RsvdMaskSize** fields define the size, in bits, of the red, green, and vlue components of a direct color pixel. A bit mask can be constructed from the MaskSize fields, using simple shift arithmetic. For example, the MaskSize values for a Direct Color 5:6:5 mode would be 5, 6, 5, and 0, for the red, green, blue, and reserved fields, respectively.
- The **RedFieldPosition**, **GreenFieldPosition**, **BlueFieldPosition**, and **RsvdFieldPosition** fields define the bit position within the direct color pixel or YUV pixel of the lsb of the respective color component. A color value can be aligned with its pixel field by shifting the value left by the FieldPosition. For example, the FieldPosition values for a Direct Color 5:6:5 mode would be 11, 5, and 0, for the red, green, blue, and reserved fields, respectively.

- The **DirectColorModeInfo** field describes important characteristics of direct color modes. **Bit D0** specifies whether the color ramp of the DAC is fixed or programmable. If the color ramp is fixed, it cannot be changed. If the color ramp is programmable, it is assumed that the red, green, and blue lookup tables can be loaded using a standard VGA DAC color registers BIOS call (AX=1012h). **Bit D1** specifies whether the bits in the **Rsvd** field of the direct color pixel can be used by the application, or are reserved, and thus unusable.

```
D0=  Color ramp is fixed/programmable:
     0=  color ramp is fixed
     1=  color ramp is programmable

D1=  Bits in Rsvd field are usable/reserved
     0=  bits in Rsvd field are reserved
     1=  bits in Rsvd field are usable by the application
```

Function 02h - Set Super VGA Video Mode

This function initializes a video mode. The BX register contains the mode to set.

```
Input:   AH=4Fh              Super VGA support
         AL=02h              Set Super VGA video mode
         BX=      Video mode
                  D0-D14= Video mode
                  D15= Clear memory flag:
                       0= clear video memory
                       1= don't clear video memory
Output:           AX=      Status
         All other registers are preserved
```

Function 03h - Return Current Video Mode

this function returns the current video mode in BX.

```
Input:   AH=4Fh          Super VGA support
         AL=03h          Return current video mode
Output:           AX=    Status
         BX=      Current video mode
         All other registers are preserved
```

Function 05h - CPU Video Memory Window Control

This function sets or gets the position of the specified window in the video memory. The function allows direct access to the hardware paging registers. To use this function properly, the software should use **VESA BIOS Function 01h** (Return Super VGA mode information) to determine the size, location, and granularity of the windows.

Input:	AH= 4Fh	Super VGA support
	AL= 05h	**Super VGA video memory window control**
	BH= 00h	**Select Super VGA video memory window**
	BL=	Window number:
		0= Window A
		1= Window B
	DX=	Window position in video memory (in window granularity units)
Output:		AX= Status

(See notes below.)

Input:	AH= 4Fh	Super VGA support
	AL= 05h	**Super VGA video memory window control**
	BH= 01h	**Return Super VGA video memory window**
	BL=	Window number:
		0= Window A
		1= Window B
Output:		AX= Status
	DX=	Window position in video memory (in window granularity units)

(See notes below.)

Notes:

- This function is also directly accessible through a far call from the application. The address of the BIOS function may be directly obtained by using VESA BIOS function 01h (return Super VGA mode information). A field in the ModeInfoBlock contains the address of this function. Note that this function may be different among video modes in a particular BIOS implementation, so the function pointer should be obtained after each set mode.
- In the far call version, no status information is returned to the application. Also, in the far call version, the AX and DX registers will be destroyed. Therefore, if AX and/or DX must be preserved, the application must do so before making the call.
- The application must load the input arguments in BH, BL, and DX (for set window), but does not need to load either AH or AL in order to use the far call version of this function.

Function 06h - Set/Get Logical Scan Line Length

This function sets or gets the length of a logical scan line. It allows an application to set up a logical video memory buffer that is wider than the displayed area. Function 07h then allows the application to set the starting position that is to be displayed.

Input:	AH = 4Fh	Super VGA support
	AL = 06h	**Logial scan line length**
	BL = 00h	**Select scan line length**
	CX =	Desired width, in pixels
Output:	AX =	Status
	BX =	Bytes per scan line
	CX =	Actual pixels per scan line
	DX =	Maximum number of scan lines
Input:	AH = 4Fh	Super VGA support
	AL = 06h	**Logical scan line length**
	BL = 01h	**Return scan line length**
Output:	AX =	Status
	BX =	Bytes per scan line
	CX =	Actual pixels per scan line
	DX =	Maximum number of scan lines

Notes:

- The desired width, in pixels, may not be achievable because of hardware limitations. The next-larger value that will acccomodate the desired number of pixels will be selected , and the actual number of pixels will be returned in CX. BX returns a value, which when added to a pointer into video memory, will point to the next scan line.
- The *mach64* implementation only supports the extended modes.

Function 07h Set/Get Display Start

This function selects the pixel to be displayed in the upper left corner of the display from the logical page. This function can be used to pan and scroll around logical screens that are larger than the displayed screen. This function can also be used to rapdly switch between two, different displayed screens for double-buffered animation effects.

Input:	AH = 4Fh	Super VGA support
	AL = 07h	**Display start control**
	BH = 00h	*Reserved, and must be 0*
	CX =	First displayed pixel in scan line
	DX =	First displayed scan line
Output:	AX =	Status
Input:	AH = 4Fh	Super VGA support
	AL = 07h	**Display start control**
	BL = 01h	**Return display start**
Output:	AX =	Status
	BH =	*00h Reserved, and will be 0*
	CX =	First displayed pixel in scan line
	DX =	First displayed scan line

Notes:

- The *mach64* implementation only supports this function in extended mode.
- This function is also valid in text modes. In text modes, the application should determine the current character cell width through normal BIOS functions, multiply that by the desired starting character column, and pass that value in the CX register. It should also multiply the current character cell height by the desired starting character row, and pass that value in the DX register.

00 - Report VBE/PM Capabilities

Input:	AH = 4Fh	VESA Extension
	AL = 10h	**VBE/PM Services**
	BL = 00h	**Report VBE/PM Capabilities**
	ES:DI	Null pointer; must be 0000:0000h in version 1.0. Reserved for future use
Output:	AX =	Status
	BH =	Power saving state signals supported by the controller: 1 = supported, 0 = not supported
	bit 0	STANDBY
	bit 1	SUSPEND
	bit 2	OFF
	BL =	VBE/PM Version number (0001 0000b for this version)
	bits 0-3	Minor Version number
	bits 4-7	Major Version number
	ES:DI	Unchanged

01 - Set Display Power State

Input:	AH = 4Fh	VESA Extension
	AL = 10h	**VBE/PM Services**
	BL = 01h	**Set Display Power State**
	BH =	Requested Power state:
	00h	ON
	01h	STANDBY
	02h	SUSPEND
	04h	OFF
Output:	AX =	Status
	BH =	Unchanged

02 - Get Display Power State

Input:	AH = 4Fh	VESA Extension
	AL = 10h	**VBE/PM Services**
	BL = 02h	**Get Display Power State**
Output:	AX =	Status
	BH =	Power state currently requested by the controller:
	00h	ON
	01h	STANDBY
	02h	SUSPEND
	04h	OFF

C

Parameter Table Format

Listing by Byte Numbers

Byte	Description
0	Number of text columns
1	Number of text rows
2	Character height (in pixels)
3	Display page length (LSB Byte)
4	Display page length (MSB Byte)
5	SEQ01 - Clocking Mode Register
6	SEQ02 - Map Mask Register
7	SEQ03 - Character Map Select Register
8	SEQ04 - Memory Mode Register
9	GENMO - Miscellaneous Output Register
0Ah	CRT00 - Horizontal Total Register
0Bh	CRT01 - Horizontal Display End Register
0Ch	CRT02 - Start Horizontal Blanking Register
0Dh	CRT03 - End Horizontal Blanking Register
0Eh	CRT04 - Start Horizontal Retrace Register
0Fh	CRT05 - End Horizontal Retrace Register
10h	CRT06 - Vertical Total Register

Byte	Description
11h	CRT07 - Overflow Register
12h	CRT08 - Preset Row Scan Register
13h	CRT09- Maximum Scan Line Register
14h	CRT0A- Cursor Start
15h	CRT0B - Cursor End
16h	CRT0C - Start Address High
17h	CRT0D - Start Address Low
18h	CRT0E - Cursor Location High
19h	CRT0F - Cursor Location Low
1Ah	CRT10 - Start Vertical Retrace Register
1Bh	CRT11 - End Vertical Retrace Register
1Ch	CRT12- Vertical Display Enable End Register
1Dh	CRT13 - Offset Register
1Eh	CRT14 - Underline Location Register
1Fh	CRT15 - Start Vertical Blanking Register
20h	CRT16 - End Vertical Blanking Register
21h	CRT17 - Mode Register
22h	CRT18 - Line Compare Register
23h	ATTR00 - Palette Register 0
24h	ATTR01 - Palette Register 1
25h	ATTR02 - Palette Register 2
26h	ATTR03 - Palette Register 3
27h	ATTR04 - Palette Register 4
28h	ATTR05 - Palette Register 5

Byte	Description
29h	ATTR06 - Palette Register 6
2Ah	ATTR07 - Palette Register 7
2Bh	ATTR08 - Palette Register 8
2Ch	ATTR09 - Palette Register 9
2Dh	ATTR0A -Palette Register A
2Eh	ATTR0B -Palette Register B
2Fh	ATTR0C - Palette Register C
30h	ATTR0D - Palette Register D
31h	ATTR0E - Palette Register E
32h	ATTR0F - Palette Register F
33h	ATTR10 - Mode Control Register
34h	ATTR11 - Overscan Color Register
35h	ATTR12 - Color Map Enable Register
36h	ATTR13 - Horizontal PEL Panning Register
37h	GRA00 - Set/Reset Register
38h	GRA01 - Enable Set/Reset Register
39h	GRA02 - Color Compare Register
3Ah	GRA03 - Data Rotate Register
3Bh	GRA04 - Read Map Select Register
3Ch	GRA05 - Mode Register
3Dh	GRA06 - Miscellaneous Register
3Eh	GRA07 - Color Don't Care Register
3Fh	GRA08 - Bit Mask Register

In addition, the VIDEO BIOS is using some of the unused bits in the parameter table to store the extended registers programming information. This information defines the video memory model, DAC programming information, CRTC and Dot Clock selection.

Parameter Table Entry 5

Bit 7	0 = Set ATI38[7] to 0 1 = Set ATI38[7] to 1
Bit 6	0 = Set ATI38[6] to 0 1 = Set ATI38[6] to 1

Parameter Table Entry 7

Bit 8	0 = Set ATI31[6] to 0 1 = Set ATI31[6] to 1
Bit 7	0 = Set ATI3E[1] to 0 1 = Set ATI3E[1] to 1

Parameter Table Entry 8

Bit 7	0 = Set ATI39[1] to 1 1 = Set ATI39[1] to 0
Bit 6	0 = Set ATI3E[4] to 1 1 = Set ATI3E[4] to 0
Bit 5	0 = Set ATI38[7,6] to 0,1 1 = Set ATI38[7,6] to 0,0
Bit 4	0 = Set ATI30[0] to 0 1 = Set ATI30[0] to 1

D

Dot Clocks

ATI18811-1 Clock Chip Pixel Clocks

Frequency Output (MHz)	Select Bits			
	ATI3E[4]	ATI39[1]	GENMO[3]	GENMO[2]
100.00	0	0	0	0
126.00	0	0	0	1
92.40	0	0	1	0
36.00	0	0	1	1
50.35	0	1	0	0
56.64	0	1	0	1
External Frequency	0	1	1	0
44.90	0	1	1	1
135.00	1	0	0	0
32.00	1	0	0	1
110.00	1	0	1	0
80.00	1	0	1	1
39.91	1	1	0	0
44.90	1	1	0	1
75.00	1	1	1	0
65.00	1	1	1	1

ATI18818 Programmable Clock Chip

PCLK_TABLE = 1

Frequency Output (MHz)	Select Bits			
	ATI3E[4]	ATI39[1]	GENMO[3]	GENMO[2]
50.35	0	0	0	0
56.64	0	0	0	1
63.00	0	0	1	0
72.00	0	0	1	1
40.00	0	1	0	0
44.90	0	1	0	1
49.50	0	1	1	0
50.00	0	1	1	1
100.00	1	0	0	0
110.00	1	0	0	1
126.00	1	0	1	0
135.00	1	0	1	1
78.25	1	1	0	0
80.00	1	1	0	1
75.00	1	1	1	0
65.00	1	1	1	1

ATI18818 Programmable Clock Chip

PCLK_TABLE = 2

Frequency Output (MHz)	Select Bits			
	ATI3E[4]	ATI39[1]	GENMO[3]	GENMO[2]
25.18	0	0	0	0
28.32	0	0	0	1
31.50	0	0	1	0
36.00	0	0	1	1
40.00	0	1	0	0
44.90	0	1	0	1
49.50	0	1	1	0
50.00	0	1	1	1
100.00	1	0	0	0
110.00	1	0	0	1
126.00	1	0	1	0
135.00	1	0	1	1
78.25	1	1	0	0
80.00	1	1	0	1
75.00	1	1	1	0
65.00	1	1	1	1

This page intentionally left blank.

E

Scratch Registers and Their Contents

SCRATCH_REG0+1	800x600 refresh rate information
SCRATCH_REG0+3	1024x768 refresh rate information
SCRATCH_REG1	ROM location
SCRATCH_REG1+1	
bits 7-4	RAMDAC subtype
bits 0	Sync on green enable
SCRATCH_REG1+2	
bits 7-6	CRTC pitch size
bit 5	MUX mode
bit 4	Enable gamma correction or 256 color greyscale
bit 3	TI output clock select information
bit 0	Current gamma correction or 256 color state
SCRATCH_REG1+3	Programmable dot clock information
1CE/BB	
bits 7-6	640x480 refresh rate information
bits 5-4	Monochrome mode, color information
bit 0	Set to VGA display if int10 is called

This page intentionally left blank.

A
ATI Enhanced Modes A-2
ATI68800CX EEPROM Data Structure B-2

B
Binary Files
  Parameters 3-11
  Procedure Outline 3-11
BIOS
  Customization 3-4
  customization 1-3
  function calls A-1, A-11
  Initialization 3-5
  support A-11
  Supported Functions 1-2
Bus
  EISA 3-3
  ISA 3-3
  Local Bus 3-3
  PCI 3-3
Bus - ISA 3-12
Bus - PCI 3-12
Bus - VLB 3-12

C
Calculating ROM Base Address A-11
Character Fonts 3-5
Character generator routines A-5
Clock Chip 3-7, D-1
Contents of the Kit 2-1
Copying Files 2-1
CRT Parameter
  table B-5
  table entry C-4
  table format C-1
Custom BIOS 3-4

D
Dot Clocks D-1
Dot clocks D-1

E
EDLIN.EXE 2-1
EEPROM 3-6
  data structure B-2
  entries B-1
  internal and external data storage 3-6
EEPROM data B-1, E-1
EXE2BIN.EXE 2-1
Extended Graphics Accelerators 1-3
Extended ROM Services A-11
Extended Video Modes 1-1

F
File Listings By Directories 2-2
File Listings by Directories
  CXROM 2-2
  CXROMCONFIG 2-7
  CXROMŒBJS 2-6
  CXVESACONFIG 2-7
  FONTS 2-8
  VGACOMM 2-8
  VGATOOLS 2-9
Frequency Tables D-1

G
Get Display Power State B-19

I
IBM Compatible Modes A-1

L
LINK 3-11
LINK.EXE 2-1

M
MAKEROM.BAT 3-5, 3-10, 3-11, 3-12
Making Non-Paged BIOS 3-11
Microsoft MASM 2-1

Mode Table Structure A-18
Monitor Support 3-5

N
Non-Paged BIOS 2-2
Non-paged BIOS 3-11

P
Paging Out the Initialization Code 3-8
Parameter table format C-1
PCI-Specific Implementation 3-8
Print screen routine selection A-6
Program Tools - VGATOOLS 2-9

Q
Query Structure A-16

R
Read character/attribute A-3
Read current cursor position A-2
Read current light pen position A-2
Read display combination code A-8
Read dot A-3
Report VBE/PM Capabilities B-18
Return current EGA settings A-6
Return current video setting A-4
Return Super VGA Information B-7
Return VGA functionality and state information A-8
ROM BIOS Relocation 3-8
ROM Selection 3-7

S
Save and restore video state A-10
Scratch Registers 3-5, A-11, E-1
Scroll active page down A-3
Scroll active page up A-2
Select active display page A-2
Set color palette A-3
Set current cursor position A-2
Set cursor type A-2
Set Display Power State B-18
Set Mode 3-5, A-11
Set palette registers A-4
Set video mode A-1
Sign-on Messages 1-3, 3-5
Source Codes 1-3
  CXROM 2-2
  FONTS 2-8
  VGACOMM 2-8
  VGAROM 2-2
Super VGA Mode Numbers B-9
  Supported VESA Modes B-9
Supported VESA Modes B-9
  Function 01h - Return Super VGA Mode Information B-9
  Function 02h - Set Super VGA Video Mode B-14
  Function 03h - Return Current Video Mode B-14
  Function 05h - CPU Video Memory Window Control B-15
  Function 06h - Set/Get Logical Scan Line Length B-16
  Function 07h - Set/Get Display Start B-17
Symbolic Constants 3-1, 3-6, 3-7, 3-8
  88800GX BUS 3-3
  88800GX RAMDAC 3-2
  Clock Chip 3-3
  EEPROM 3-4
  VESA BIOS Extension 3-4
System BIOS 3-1, 3-5, 3-8

V
VESA BIOS Extension 3-9
VESA BIOS Extension Constants 3-4
VESA BIOS extensions B-7
  Status information B-7
VGA Compatibility 3-1
VGA Controller A-1
Video BIOS 3-1, 3-5, 3-8
Video BIOS Relocation 3-8

W
WONDER.X01 3-11
Write character at current cursor position A-3
Write character/attribute A-3
Write dot A-3
Write string to specified page A-7
Write teletype to active page A-3

X
XCOPY 2-2

Perfecting the PC

Printed in Canada