Advance Information

This document contains information on a product under development. The parametric information contains target parameters that are subject to change.

BtV2115

MediaStream Controller

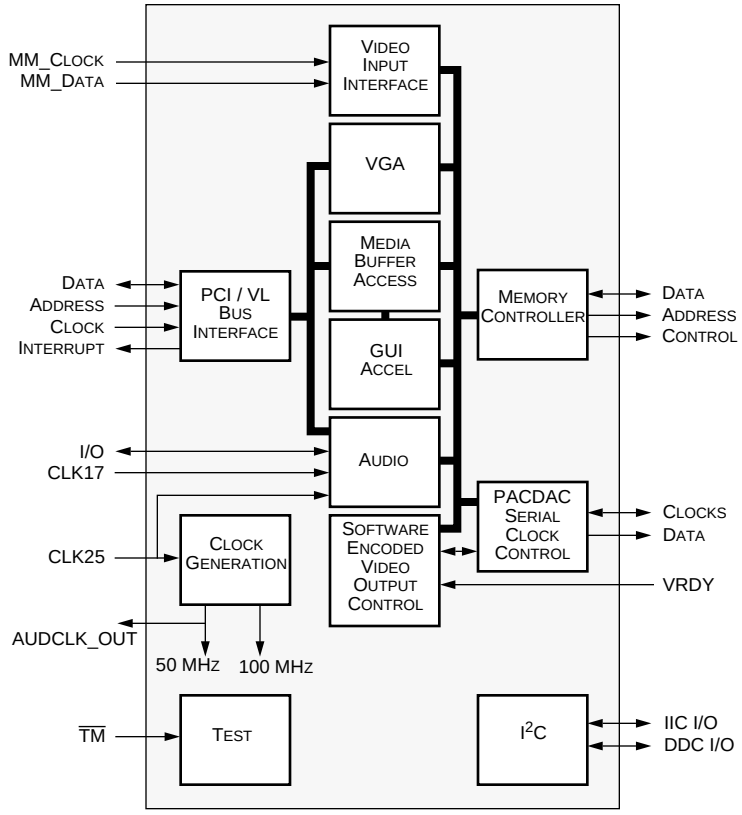
Applications

- Microsoft Windows™ 3.1, Win '95™, and NT™ GUI acceleration
- IBM OS/2™ GUI acceleration
- Xwindow System™ acceleration
- Video for Windows™ and Microsoft Windows MPC Audio™
- High Resolution Color Graphics
- Video-conferencing
- Multimedia Applications
- Video Playback Acceleration

Related Products

- BtV2487 PACDAC
- BtV2811A VideoStream Decoder
- BtV2300 AudioStream Interface

Functional Block Diagram



Distinguishing Features

- Local Bus Graphics Controller (glueless for):
  - PCI 2.0 bus
  - VESA VL 2.0 bus (32 bit)
- 32 bit VRAM (1MB to 4MB)
  - Samsung™ 256Kx8: -8,-7,-6
    128Kx16:-8,-7,-6
  - Texas Instruments™ 256Kx16:-8,-7,-6
  - NEC™, Toshiba™, IBM™ 256Kx16:-8,-7,-6
- Up to 1MB of Flash ROM
- Supports Yamaha 2 and 4 operator mode FM synthesizer family
- Resolutions:
  - 640x480: 4, 8, 16, 24, 32 b/pixel up to 76 Hz
  - 800x600: 4, 8, 16, 32 b/pixel up to 76 Hz
  - 1024x768: 4, 8, 16, 24 b/pixel up to 75 Hz
  - 1280x1024: 4, 8, 16 b/pixel up to 75 Hz
- VGA compatibility for modes 0x00 - 0x07 and 0x0D - 0x13
- Multimedia Solution
  - Glueless connection BtV2811A VideoStream Decoder
  - Acceleration of two video planes with hardware double buffering
  - Hardware-accelerated video playback
  - Stereo audio in/out
  - Software encoded NTSC/PAL video out to VCR
  - Stretch Blt and raster scaling
  - Legacy AudioPro™ compatibility
  - Digital Interface to consumer digital audio tape, minidisc, etc.
- Standard Interfaces
  - PCI bus 2.0 compatible
  - VESA VL bus 2.0
  - AES/EBU S/PDIF CP-340 (with full access to channel/user bits)
  - I^2C/ ACCESS.bus
  - VESA DDC1, DDC2A/B
  - VESA DPMS
- 208 pin Power Quad package

Brooktree® | *Brooktree Corporation • 9868 Scranton Road • San Diego, CA 92121-3707 • 619-452-7580*
1-800-2-BT-APPS • FAX: 619-452-1249 • Internet: apps@brooktree.com • L211501 Rev. C

Ordering Information

Model Number	Package	Ambient Temperature Range
BtV2115KSF	208 PQFP	0˚C to +70˚C
BtV2115AHF	208 plastic MQFP	0˚C to +50˚C

Trademarks

Brooktree and BtV are registered trademarks of Brooktree Corporation.

The following are trademarks of Brooktree Corporation: VIDEODAC, RAMDAC, Video CacheDAC, VideoCache, PACDAC, ByteStream, faSTICK, and MediaBuffer.

WinBench is a registered trademark and Winstone is a trademark of Ziff-Davis Publishing Company, L.P. In compliance with Ziff-Davis' licensing agreement: benchmark tests were obtained with WinBench 4.0 run on a Dell Dimension XPS P90 Minitower Pentium 90MHz, 256K SRAM cache, 16M system memory, and the BtV2115KSF PCI-bus graphics adapter.

Dell is a registered trademark and Dimension XPS P90 is a trademark of Dell Computer Corporation.

COMPAQ and DESKPRO are registered trademarks and EISA is a trademark of Compaq Computer Corporation.

Intel is a registered trademark and Pentium is a trademark of Intel Corporation.

Microsoft and MS-DOS are registered trademarks and Windows, Win '95, Windows NT, Plug and Play, Windows MPC Audio, and Video for Windows are trademarks of Microsoft Corporation.

Philips is a registered trademark of Philips International B.V.

ACCESS.bus is a trademark of Digital Equipment Corporation.

Tri-state is a trademark of National Semiconductor.

Micro Channel Architecture (MCA) and OS/2 are trademarks of IBM Corporation.

VESA is a registered trademark of and VL-Bus and VBE are trademarks of Video Electronics Standards Association.

Xwindow System is a trademark of Massachusetts Institute of Technology.

Yamaha OPL2/3/4 are trademarks of Yamaha Corporation.

Samsung is a trademark of Samsung Electronics Co., LTD

NEC is a trademark of NEC Corporation.

Toshiba is a trademark of Toshiba Corporation.

IBM is a trademark of IBM Corporation.

All other marks mentioned herein belong to their respective companies.

Print date: 08/30/95

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

What this Document Contains

This hardware specification contains the following chapters.

Product Description — Provides an overview of the BtV2115 MediaStream Controller, including a summary of the PCI and VL pin assignments. For a summary of the BtV MediaStream Family of chips, refer to the beginning of this Technical Reference binder.

VGA Implementation— Contains a description of the compatibility and operation of the VGA within the BtV2115 controller.

Register Definitions — Contains the definitions of or references to all of the BtV2115 registers, including both I/O mapped and memory mapped registers.

CPU Address Space Apertures — Describes the programming interface to the BtV2115 MediaStream accelerator, the Media Buffer, the audio subsystem, the flash ROM, and the Yamaha 2 and 4 operator mode FM synthesizer family.

Configuration Registers — Contains a summary of the registers that specify operating parameters for interaction of internal and external multimedia subsystem components.

I^2C Master and Slave Controllers — Specifies the programming interface between the BtV2115 and the I^2C master and slave modules.

GUI Accelerator — Describes the CPU addressing to the GUI and defines the GUI commands registers.

Softvideo Player— Describes the interface between the BtV2115 software encoded video module and the BtV2487 PACDAC via the VRAM serial port.

CPU Host Bus Interface — Describes the VESA Local bus and PCI bus interface between the BtV2115 and the host CPU. Defines the VESA and PCI signal pins and provides timing diagrams.

Video RAM Interface— Describes the interface between the BtV2115 and the VRAM bus which provides access to ROM, FM synthesizer, and multimedia data and control information. Includes timing diagrams and connection diagrams.

Video Input Subsystem — Describes the interfaces between the BtV2115 and the BtV2811A VideoStream Decoder, and between the BtV2115 and the memory controller and VRAM via the memory bus.

Audio Interface — Describes the interface between the BtV2115 and the AudioStream subsystem.

Operating Specifications — Provides the operating specifications for the BtV2115, including electrical, thermal, packaging, and signal pin layout.

Related Documents

BtV2300 AudioStream Specification

BtV2811A VideoStream Decoder Specification

BtV2487 PACDAC Specification

PCI Local Bus Specification, Revision 2.0

Video Electronics Standards Association (VESA) Local Bus Specification, Version 2.0

Yamaha YMF-262 Product Information bulletin

Philips Semiconductor, Data Sheet SAA7194, Digital Video Decoder and Scalar Circuit (DESC)

Philips I^2C-Bus and How to Use It

ACCESS.bus Specification - Version 2.0

Richard F. Ferraro, *Programmer's Guide to the EGA and VGA Cards*, Second Edition (Addison-Wesley Publishing Company, Inc., 1990)

Bradley Dyck Kliewer, *EGA/VGA: A Programmer's Reference Guide* (Intertext Publications, McGraw-Hill Publishing Company, New York, 1990)

Notation Conventions

Notation conventions used in this book are listed below.

Example	Description
$\overline{\text{VL\_RESET}}$	The overbar above a signal or pin (or a portion of the signal or pin) indicates active-LOW.
8’hAD	Hexadecimal notation used by hardware designers, in the format: ***<num bits>’h<value>*** Where: ***<num bits>*** = number of bits in hex value ***’h*** = hexadecimal ***<value>*** = hex value, each alphanumeric digit represents 4 bits
6’b01011	Binary notation, in the format: ***<num bits> ’b <value>*** Where: ***<num bits>*** = number of bits in binary value ***’b*** = binary ***<value>*** = binary value of each bit
AB31h	Hexadecimal notation used by Assembler programmers, in the format: ***<value>h*** Where: ***h*** = hexadecimal ***<value>*** = hex value, each alphanumeric digit represents 4 bits
0x000AC100	Hexadecimal notation used by C programmers, in the format: ***0x<value>*** Where: ***0x*** = hexadecimal ***<value>*** = hex value, each alphanumeric digit represents 4 bits

PRODUCT DESCRIPTION

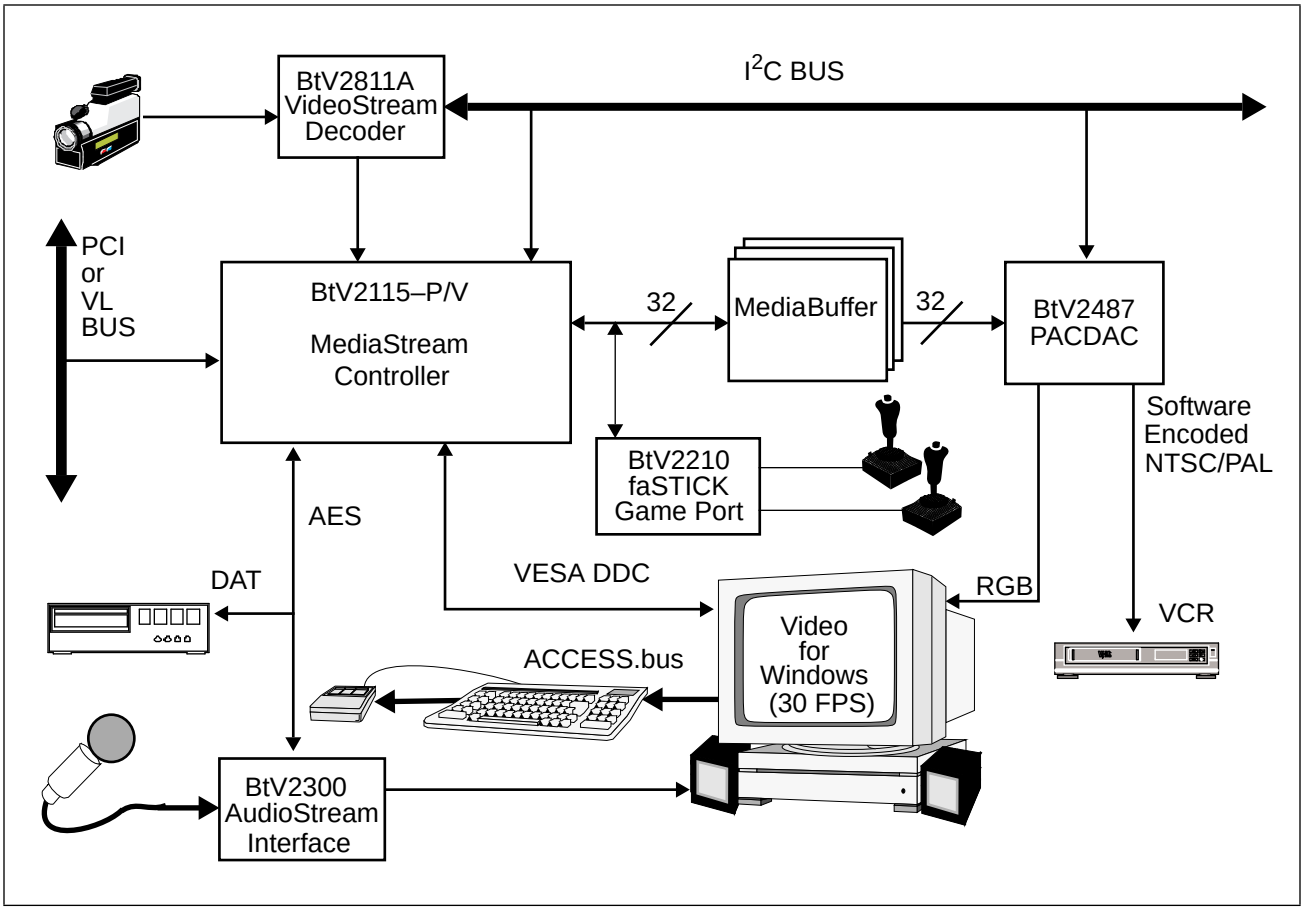
Introduction

The BtV2115 MediaStream Controller is designed specifically for low-cost, high-performance, high-resolution color graphics applications. It is designed to work with the BtV2487 PACDAC, the BtV2300 AudioStream interface, and the BtV2811A VideoStream Decoder in a high function multimedia user interface subsystem.

The complete BtV® MediaStream configuration is depicted in Figure 1 and consists of the following major components.

- BtV2115 MediaStream Controller
- BtV2300 AudioStream Interface providing stereo audio
- BtV2487 PACDAC providing a video cache for VFW acceleration and NTSC/PAL output
- BtV2811A VideoStream Decoder providing live video windows

Figure 1. BtV Configuration



The clock-quadrupled MediaStream controller offers very high performance GUI acceleration in a very compact form factor. A BtV2115 graphics subsystem can be constructed from as little as two VRAMs, one BtV2115, and one BtV2487. Such a one megabyte configuration can support 640x480x24 through 1280x1024x4. The 208 pin PQFP package allows glueless attachment to both 32 bit PCI™ bus and the VESA™ Local Bus. The VL™ address bus is fully decoded and demultiplexed, allowing the fastest VL bus cycles. The VRAM GUI FIFO can buffer up to 8192 commands permitting essentially full overlap of application and GUI command execution. This is the ideal graphics accelerator for medium to high performance motherboards.

This same four part configuration can support two accelerated video windows. Decompression acceleration allows 30 frames per second at full screen resolution for software decompressors. This configuration also supports two CD quality stereo output streams with on-chip audio decompression acceleration. The BtV2115/ BtV2487 system is the ideal graphics and multimedia system for Video For Windows™ applications and is small enough and inexpensive enough for mother-

boards. The BtV2115/BtV2487 system also supports a software encoded NTSC/PAL video output which can be run in parallel with RGB monitor refresh.

The BtV2811A single chip NTSC/PAL decoder can be gluelessly attached to the BtV2115 to turn the two video window system into an ideal platform for video teleconferencing. The video codec acceleration features in the BtV2115 allow software compressors and decompressors to function at much higher frame rates.

The BtV2115 controller is the first audio product designed for the PC environment that offers professional quality digital serial audio input and output. The BtV2115 and BtV2300 provide a completely self-contained windows audio subsystem. In addition, the chip set supports attachment to professional and consumer grade digital audio equipment, such as digital audio tape and digital mini disk units via AES/SPDIF or CP-340 interfaces with full program access to the AES channel and user data streams. Plus, the chip set can receive source material and master digital data streams. These features allow sophisticated, lossless, multipass, off-line edit of audio streams. The BtV2115 provides access to the digital subcode data which includes professional time code information, sample rate, etc. On a CD unit, the subcode conveys such information as track position, catalog, pre-emphasis, etc.

The BtV2115/BtV2300 support simultaneous play and record under Microsoft® Windows™. In addition, the chip set supports simultaneous play of two stereo streams so that application audio can be digitally mixed with Windows' system sounds. Also, external audio (analog or digital) can be digitally mixed with the Windows' playback audio streams so that application audio can be heard through the current background music selection. Sophisticated audio metering is provided in hardware for measuring average level, peak level and for detecting clipping in both playback and record.

In addition, the BtV2115 provides Legacy AudioPro emulation of the sound hardware normally used to support MS-DOS™ based games. Legacy AudioPro emulation has been designed to work with most popular DOS games and is the first PCI-based product to do so. All other implementations in the market today require ISA™ bus attachment. The BtV2115 provides decode and bus attachment support for the Yamaha™ 2 and 4 operator modes FM synthesizer family.

The BtV2115 provides a glueless attachment for the optional BtV2210 faSTICK game port controller (refer to the *BtV2210 Game Port/MIDI Controller spec*). This combination offers compatibility with today's analog PC joysticks and provides a very high speed digital interface to analog joysticks used in Windows 95™ gaming.

The serial bus controlling the BtV2487 and BtV2811A is I^2C compatible and available for controlling peripherals such as an add-in tuner and for querying monitor characteristics from a suitably equipped monitor, as well as for ACCESS.bus™ access to keyboards, mouse and other peripherals. The BtV2115 supports monitors equipped with all current VESA Display Data Channel (DDC) modes, including DDC1, DDC2A, and DDC2B. For monitors with DDC2B support, the BtV2115 provides ACCESS.bus support over the VGA monitor cable. The ACCESS.bus can be easily switched to a second ACCESS.bus port for older style monitors without DDC2B support. The BtV2115/BtV2487 chip set supports the VESA Display

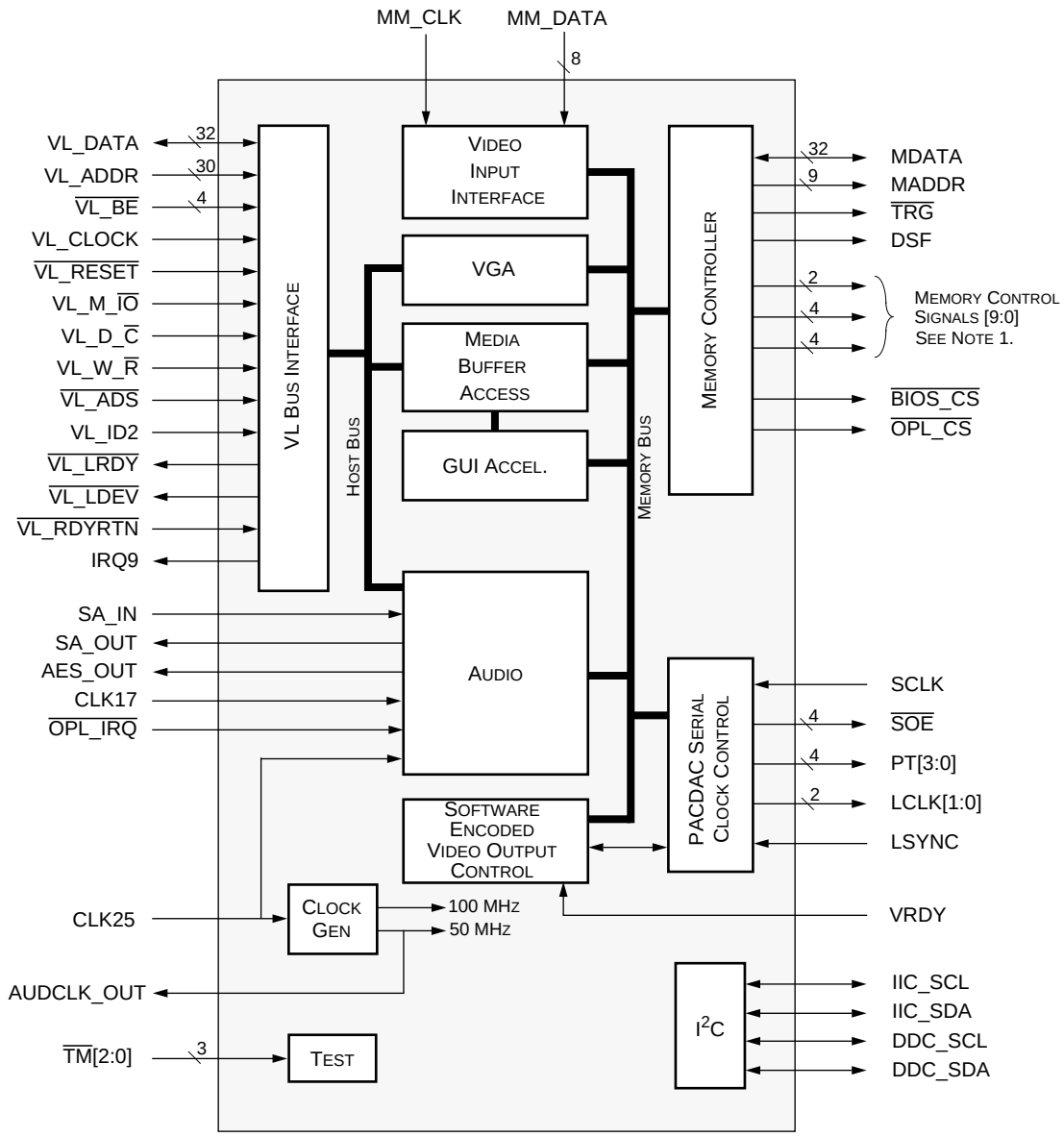
Power Management Signalling protocol for controlling monitor power consumption on suitably equipped monitors.

The BtV2115's maximum frame buffer size is 4MB. Two megabytes brings 1024x768x16 and 1280x1024x8 capability to a BtV2487-based system. The BtV2115 can support up to 1280x1024x16 and 1024x768x24 for VRAM configurations of greater than 2MB.

Use of such standard interfaces as VL, PCI, AES/EBU, I^2C, and ACCESS.bus greatly simplifies the system designer's task and offers many interesting opportunities for product differentiation.

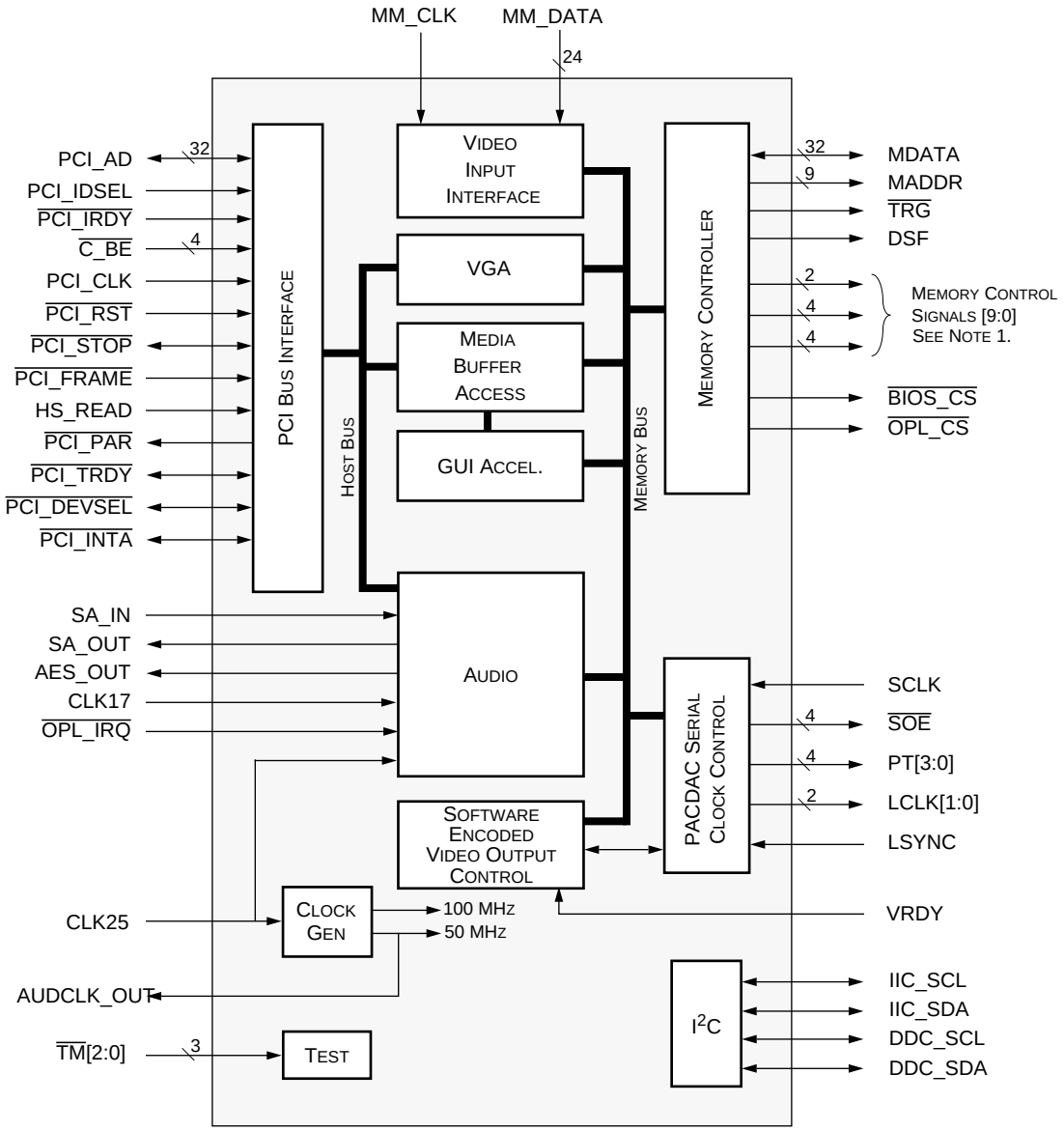
Figure 2 presents a block diagram of the VL-Bus™ BtV2115 configuration. Figure 3 presents a block diagram of the PCI-Bus™ BtV2115 configuration.

Figure 2. BtV2115 VL-Bus Block Diagram



Note 1. Refer to Table 1 on page 13, column “VRAM Interface Pins” for details on the RAS, WE, and CAS memory control pins.

Figure 3. BtV2115 PCI-Bus Block Diagram



Note 1. Refer to Table 2 on page 17, column “VRAM Interface Pins” for details on the RAS, WE, and CAS memory control pins.

Interface Description

The BtV2115 provides seven interfaces to the graphics subsystem:

- CPU or Host bus interface
- VRAM interface
- PACDAC™ interface
- BtV2811A/multimedia interface
- I^2C interface
- Serial Audio interface
- Test interface

These interfaces are described briefly here; however, for more detail refer to subsequent chapters of this document.

CPU/HOST Interface

The interface between the CPU and BtV2115 is via a VL-32 or PCI bus. The actual bus or protocol in use is selected by configuration strapping resistors on the VRAM data bus, see below. The VL-32 bus is fully decoded and fully demultiplexed and provides a glueless VL-slave interface that runs as fast as the VESA VL spec permits. The PCI bus interface uses only 46 pins (60% of VL-32). The VL-32 pins are redefined to provide a PCI bus interface when the configuration resistors are suitably strapped. The PCI interface is compliant with PCI specifications as of revision 2.0. The VL-32 interface is compliant with version 2.0 of the VESA specification. For more information refer to "CPU Host Bus Interface" on page 211.

The BtV2115 does not support VL-16 (e.g. 386 microprocessors), EISA®, ISA, or MCA® buses.

VRAM Interface

The VRAM interface of BtV2115 supports many different form factors and speed grades of VRAM, including TI256Kx16 in 70ns and 60ns speed grades as well as various two megabit VRAMs, (refer to the "Video RAM Interface" starting on page 225 and "VRAM Timing Parameters" on page 317 for complete details). The VRAM interface supports up to four megabytes of VRAM. Special VRAM features are utilized to optimize performance, e.g. nonpersistent write, block write, etc.

The VRAM data bus is used to support additional graphics/multimedia subsystem components like Flash ROM devices and a Yamaha 2 or 4 operator mode FM synthesizer chip.

PACDAC Interface

The PACDAC interface is derived internally from the serial clock controller. The SEV out, and CRT controller all share the serial clock controller to drive the PACDAC. This interface includes the VRAM serial clocks, VRAM serial output enables, PACDAC tag lines and synchronization signals. For more information refer to "PACDAC Controller" on page 195.

Multimedia Interface

The BtV2811A VideoStream Decoder chip attaches to BtV2115 via the multimedia interface. All transfers are packetized over an eight bit bus. For PCI bus configurations, a 24bit parallel decoder interface for 16 bit YCrCb video data is available in addition to the BtV2811A. For details on parallel interface, refer to "Parallel Decoder Interface" on page 266.

I^2C Interface

BtV2115 contains support for two I^2C interfaces. Either one can be used to support the ACCESS.bus protocol. One interface is intended to support VESA DDC1, DDC2A, and DDC2B monitor signal protocols. The other is intended to support communication with the BtV2487 and BtV2811A even if an older monitor without DDC signalling is attached to the VGA connector. For more information refer to "I^2C Master and Slave Controllers" on page 137.

Serial Audio Interface

Audio I/O to or from the BtV2300 is transferred via the AES/EBU format on the serial audio in and serial audio output pins. This interface transfers a serial stereo bit stream at up to 48KHz in both directions. The AES_OUT pin provides an isolated version of the serial audio stream for consumer audio equipment access. For more information refer to "Audio Interface" on page 271.

Test Interface

The test interface includes the test mode pins $\overline{TM}$[2:0] and the AUDCLK_OUT pin. In addition, scan data is shifted into the MM_DATA[7:0] lines and out the lower eight bits of the MADDR lines.

Distinguishing Features

The BtV2115 provides the following features.

- Local Bus Graphics Controller (glueless for):
  - PCI 2.0 bus
  - VESA VL 2.0 bus (32 bit)
- 32 bit VRAM (1MB to 4MB)
  - Samsung™ 256Kx8: -8,-7,-6
  - Hitachi™ 256Kx8:-8,-7
  - Texas Instruments™ 256Kx16:-8,-7,-6
  - IBM™ 256Kx16:-8,-7,-6
  - NEC™ 256Kx8:-8,-7
- Up to 1MB of Flash ROM
- Supports Yamaha 2 and 4 operator modes FM synthesizer family
- Resolutions:
  - 640x480: 4,8,16,24,32 bits/pixel up to 76Hz
  - 800x600: 4,8,16,32 bits/pixel up to 76Hz
  - 1024x768: 4,8,16,24 bits/pixel up to 75Hz
  - 1280x1024: 4,8,16 bits/pixel up to 75Hz
- VGA compatibility for modes 00h - 07h and 0Dh - 13h
- Multimedia Solution
  - Glueless connection BtV2811A VideoStream Decoder
  - Acceleration of 2 video windows with hardware double buffering
  - Hardware assisted video playback acceleration
  - Stereo audio in/out
  - Software encoded NTSC/PAL video out to VCR
  - Big and little endian support
  - Stretch Blt and raster scaling
  - Legacy AudioPro™
  - Digital Interface to consumer Digital Audio Tape, Minidisc, etc.
- Standard Interfaces
  - PCI bus 2.0 compatible
  - VESA VL bus 2.0
  - AES/EBU SPDIF CP-340 (with full access to channel/user bits)
  - I^2C/ ACCESS.bus
  - VESA DDC1, DDC2A/B
  - VESA DPMS

Applications and Related Products

The BtV2115 supports the following applications.

- Microsoft Windows™ 3.1, Win ‘95™, and NT™ GUI acceleration
- IBM OS/2™ GUI acceleration
- Xwindow System™ acceleration
- Video for Windows™ and Microsoft Windows MPC Audio™
- High Resolution Color Graphics
- Video-conferencing
- Multimedia Applications
- Video Playback Acceleration

The BtV related products are as follows.

- BtV2487 PACDAC
- BtV2811A VideoStream Decoder
- BtV2300 AudioStream Interface

BtV2115 VL-Bus Pin List Summary

Below, Table 1 provides a summary of the BtV2115 pin list for the VL-Bus. Table 2 provides a summary of the BtV2115 pin list for the PCI-Bus. The overbar above a signal indicates active-LOW.

For a VL-Bus pin layout diagram, see Figure 75 on page 343.

Table 1. BtV2115 VL-Bus Pin List Summary *(1 of 4)*

	Pin Count	Signal Name	I/O	Description
Total VL pins = 77	32	VL_DATA[31:0]	IO	“VL Data Bus” on page 211
	30	VL_ADDR[31:2]	I	“VL Address Bus” on page 211
	4	$\overline{\text{VL\_BE}}$[3:0]	I	“VL Byte Enables” on page 211
	1	$\overline{\text{VL\_ADS}}$	I	“VL Address Strobe” on page 211
	1	VL_D_$\overline{\text{C}}$	I	“VL Data or Code Status” on page 212
	1	VL_M_$\overline{\text{IO}}$	I	“VL Memory or IO Status” on page 212
	1	VL_W_$\overline{\text{R}}$	I	“VL Read or Write Status” on page 212
	1	VL_CLOCK	I	VL bus clock (50MHz max). see “VL Clock” on page 211
	1	$\overline{\text{VL\_RESET}}$	I	“VL Reset” on page 211
	1	$\overline{\text{VL\_RDYRTN}}$	I	“VL Ready Return” on page 212
	1	$\overline{\text{VL\_LDEV}}$	O	“VL Local Device Select” on page 212
	1	$\overline{\text{VL\_LRDY}}$	O	“VL Local Ready” on page 212
	1	VL_ID2	I	“VL Identifier Pin” on page 212
	1	IRQ9	O	“Interrupt Request Line 9” on page 212

Table 1. BtV2115 VL-Bus Pin List Summary *(2 of 4)*

	Pin Count	Signal Name	I/O	Description
Total PACDAC and serial controller pins = 15	4	PT[3:0]	O	PACDAC tag to identify data type “PACDAC Data Types” on page 196
	4	$\overline{\text{SOE}}$ [3:0]	O	VRAM serial port serial output enable
	2	LCLK[1:0]	O	Serial Clock to VRAM, Load clock to PACDAC
	1	LSYNC	I	Linesync signal from PACDAC indicates that BtV2115 can begin to load the graphics, video1, video2, cursor and timing FIFOs.
	1	VRDY	I	SEV signal from PACDAC indicates that the highwater mark on the SEV out FIFO has not been reached and that BtV2115 might send further SEV FIFO packets.
	1	SCLK	I	Serial clock input from PACDAC.
	1	CLK17	I	Audio clock input from PACDAC.
	1	CLK25	I	Main clock input from PACDAC. From this signal, a 100MHz and a 50MHz clock are derived for the chip core (VGA, GUI, audio, etc).

Table 1. BtV2115 VL-Bus Pin List Summary *(3 of 4)*

	Pin Count	Signal Name	I/O	Description
Total VRAM interface pins = 57	24	MDATA[31:8]	I/O	Bidirectional tri-state VRAM memory data. Due to address line loading and pin limitations, address lines for Flash ROM and register-selects for Yamaha FM synthesizer are multiplexed on MDATA[27:8].
	8	MDATA[7:0]	I/O	Bidirectional tri-state VRAM, ROM, and Yamaha data.
	9	MADDR[8:0]	O	Multiplexed memory address lines to VRAM capable of driving up to 8 VRAMs. In testmode, MADDR[7:0] conveys the internal scan chain data out of the chip.
	4	$\overline{\text{RAS}}$[3:0]	O	VRAM Row Address Strobe (RAS), 1 per bank, up to 4MB
	1	$\overline{\text{CAS0}}$	O	Dual CAS: VRAM Column Address Strobe (CAS) for the least significant byte of the 32 bit data bus, byte0.
				Dual WE: Write Enable ($\overline{\text{WE}}$) for byte0.
	1	$\overline{\text{CAS1}}$	O	Dual CAS: CAS for byte1.
				Dual WE: CAS for least significant 16 data bits (byte1 and byte0).
	1	$\overline{\text{WE0}}$	O	Dual CAS: WE for least significant 16 data bits (byte1 and byte0).
				Dual WE: Write Enable (WE) for byte1. Yamaha and Flash ROM write pins connected here or identically to $\overline{\text{WE3}}$
	1	$\overline{\text{CAS2}}$	O	Dual CAS: VRAM Column Address Strobe (CAS) for byte2.
				Dual WE: Write Enable (WE) for byte2.
	1	$\overline{\text{CAS3}}$	O	Dual CAS: CAS for byte3.
				Dual WE: CAS for most significant 16 data bits (byte3 and byte2).
	1	$\overline{\text{WE1}}$	O	Dual CAS: WE for most significant 16 data bits (byte3 and byte2).
				Dual WE: Write Enable (WE) for byte3. Yamaha and Flash ROM write pins connected here or identically to $\overline{\text{WE0}}$
	1	$\overline{\text{TRG}}$	O	VRAM Transfer Enable, also sampled at $\overline{\text{RAS}}$ fall during non-refresh cycles, to select between DRAM external access or DRAM to SAM Register Transfer (SAM to DRAM is not supported by BtV2115). $\overline{\text{TRG}}$ is also used as Read/Write for Flash ROM and Yamaha
	1	DSF	O	VRAM Special Function Select. Sampled at $\overline{\text{RAS}}$ fall, DSF=0 during refresh cycles resets Masked (write-per-bit) Mode to non-persistent. During Masked Writes, DSF must be low (=0). During Non-Masked Writes, DSF=0 selects normal DRAM, and DSF=0 selects Register LOAD. Sampled at CAS fall of DRAM cycles, DSF selects BLOCK Mode.
	1	$\overline{\text{BIOS\_CS}}$	O	Flash ROM chip select
	1	$\overline{\text{OPL\_CS}}$	O	Yamaha FM synthesizer chip select. The synthesizer data bus is attached to VRAM_DATA[7:0].
	1	$\overline{\text{OPL\_IRQ}}$	I	Yamaha interrupt input (with an internal pull-up).
	1	$\overline{\text{RESET\_OUT}}$	O	Reset to supported peripherals: BtV2487, BtV2811A, BtV2300, Yamaha, etc

Table 1. BtV2115 VL-Bus Pin List Summary *(4 of 4)*

	Pin Count	Signal Name	I/O	Description
Total serial interface pins = 7	1	AES_OUT	O	Output to AES compatible consumer electronics (such as DAT tape, etc.)
	1	SA_OUT	O	Serial audio output to compatible audio stream interface such as the BtV2300.
	1	SA_IN	I	Serial audio input from the audio ADC. This bit serial input supports left and right channel 16 bit input and is asynchronous to MCLK. This signal is internally pulled up.
	2	IIC_SCL, IIC_SDA	I/O	I^2C interface pins, used for I^2C bus. Optional for ACCESS.bus for pre-DDC monitor
	2	DDC_SCL, DDC_SDA	I/O	I^2C interface pins, used for I^2C bus for support of the VESA DDC1 and DDC2 signalling modes as well as for ACCESS.bus when presented through the monitor VGA cable, as in VESA DDC2B signalling.
Total test pins = 4	3	$\overline{TM}$[2:0]	I	Test mode, signals, 111= normal operation, 000 = tri-state, all outputs. These signals are internally pulled up.
	1	AUDCLK_OUT	O	Audio clock for driving the AudioStream Interface clock-in. For test purposes, the 50MHz internal clock of BtV2115 is output when $\overline{TM}$[2:0] = 011. When $\overline{TM}$[2:0] = 010, a delayed version of the 50MHz internal clock of BtV2115 is output. For normal mode ($\overline{TM}$[2:0] = 11x), audio clock is output
Total multimedia bus pins = 9	8	MM_DATA[7:0]	I	Multimedia data input. In test modes, scan chain inputs enter the chip through these pins. These pins are pulled up internally.
	1	MM_CLK	I	Multimedia clock, this clock is used to transfer data on the multimedia bus. This pin is pulled up internally
Total VL-Bus Signal Pins = 169				
Total non-signal pins = 39	19	GND		Ground
	17	VDD		± 5V Power Supply
	1	G_PLL		Ground for internal PLL
	1	V_PLL		Power for internal PLL
	1	N_C		No connect. PCI_INTA in PCI mode.
Total BtV2115 VL-Bus Pins = 208				

BtV2115 PCI-Bus Pin List Summary

Below, Table 2 provides a summary of the BtV2115 pin list for the PCI-Bus. Table 1 provides a summary of the BtV2115 pin list for the VL-Bus. The overbar above a signal indicates active-LOW.

For a PCI-Bus pin layout diagram, see Figure 76 on page 344.

Table 2. BtV2115 PCI-Bus Pin LIst Summary *(1 of 4)*

	Pin Count	Signal Name	I/O	Description
Total PCI pins =47	32	PCI_AD[31:0]	I/O-3S	“PCI Address and Data Bus” on page 216
	4	$\overline{\text{C\_BE}}$[3:0]	I	“PCI Bus Command and Byte Enables” on page 217
	1	$\overline{\text{PCI\_STOP}}$	I/O-3S	“PCI Stop” on page 218
	1	$\overline{\text{PCI\_PAR}}$	O-3S	“PCI Parity” on page 217
	1	$\overline{\text{PCI\_TRDY}}$	I/O-3S	“PCI Target Ready” on page 218
	1	$\overline{\text{PCI\_DEVSEL}}$	I/O-3S	“PCI Device Select” on page 218
	1	$\overline{\text{PCI\_IRDY}}$	I	“PCI Initiator Ready” on page 217
	1	$\overline{\text{PCI\_FRAME}}$	I	“PCI Cycle Frame” on page 217
	1	PCI_IDSEL	I	“PCI Initialization Device Select” on page 218
	1	HSREAD	I	“PCI High Speed Read” on page 218
	1	$\overline{\text{PCI\_INTA}}$	I/O	“PCI Interrupt A” on page 218
	1	PCI_CLK	I	“PCI Clock” on page 217
	1	$\overline{\text{PCI\_RST}}$	I	“PCI Reset” on page 217

Table 2. BtV2115 PCI-Bus Pin LIst Summary *(2 of 4)*

	Pin Count	Signal Name	I/O	Description
Total PACDAC and serial controller pins = 15	4	PT[3:0]	O	PACDAC tag
	4	$\overline{\text{SOE}}$ [3:0]	O	VRAM serial port serial output enable
	2	LCLK[1:0]	O	Serial Clock to VRAM, Load clock to PACDAC
	1	LSYNC	I	Linesync signal from PACDAC indicates that BtV2115 can begin to load the graphics, video1, video2, cursor and timing FIFOs.
	1	VRDY	I	SEV signal from PACDAC indicates that the highwater mark on the SEV out FIFO has not been reached and that BtV2115 might send further SEV FIFO packets.
	1	SCLK	I	Serial clock input from PACDAC.
	1	CLK17	I	Audio clock input from PACDAC.
	1	CLK25	I	Main clock input from PACDAC. From this signal, a 100MHz and a 50MHz clock are derived for the chip core (VGA, GUI, audio, etc).
Total multimedia bus pins =25	8	MM_DATA[23:16]	I	[a]·parallel decoder interface
	8	MM_DATA[15:8]	I	[a]·parallel decoder interface, [b]·CHROMA[7:0] mode
	8	MM_DATA[7:0]	I	Multimedia data input. In test modes, scan chain inputs enter the chip through these pins. These pins are pulled up internally. [a]·parallel decoder interface
	1	MM_CLK	I	Multimedia clock, this clock is used to transfer data on the multimedia bus. This pin is pulled up internally

a. To enable parallel decoder interface, refer to bit 2 on Table 141 on page 256.
b. To enable chroma keying, refer to the CHROMA_MODE field in Table 146 on page 259.

Table 2. BtV2115 PCI-Bus Pin LIst Summary *(3 of 4)*

	Pin Count	Signal Name	I/O	Description
Total VRAM interface pins = 57	24	MDATA[31:8]	I/O	Bidirectional tri-state VRAM memory data. Due to address line loading and pin limitations, address lines for Flash ROM and register selects for Yamaha FM synthesizer are multiplexed on MDATA[27:8].
	8	MDATA[7:0]	I/O	Bidirectional tri-state VRAM, ROM, and Yamaha data.
	9	MADDR[8:0]	O	Multiplexed memory address lines to VRAM capable of driving up to 8 VRAMs. In testmode, MADDR[7:0] conveys the internal scan chain data out of the chip.
	4	$\overline{\text{RAS}}$[3:0]	O	VRAM Row Address Strobe (RAS), 1 per bank, up to 4MB
	1	$\overline{\text{CAS0}}$	O	Dual CAS: VRAM Column Address Strobe (CAS) for the least significant byte of the 32 bit data bus, byte0.
				Dual WE: Write Enable (WE) for byte0.
	1	$\overline{\text{CAS1}}$	O	Dual CAS: CAS for byte1.
				Dual WE: CAS for least significant 16 data bits (byte1 and byte0).
	1	$\overline{\text{W\_W1E0}}$	O	Dual CAS: WE for least significant 16 data bits (byte1 and byte0).
				Dual WE: Write Enable (WE) for byte1. Yamaha and Flash ROM write pins connected here or identically to $\overline{\text{WE1}}$
	1	$\overline{\text{CAS2}}$	O	Dual CAS: VRAM Column Address Strobe (CAS) for byte2.
				Dual WE: Write Enable (WE) for byte2.
	1	$\overline{\text{CAS3}}$	O	Dual CAS: CAS for byte3.
				Dual WE: CAS for most significant 16 data bits (byte3 and byte2).
	1	$\overline{\text{WE1}}$	O	Dual CAS: WE for most significant 16 data bits (byte3 and byte2).
				Dual WE: Write Enable (WE) for byte3. Yamaha and Flash ROM write pins connected here or identically to $\overline{\text{WE0}}$
	1	$\overline{\text{TRG}}$	O	VRAM Transfer Enable, also sampled at RAS fall during non-refresh cycles, to select between DRAM external access or DRAM to SAM Register Transfer (SAM to DRAM is not supported by BtV2115). $\overline{\text{TRG}}$ is also used as Read/Write for Flash ROM and Yamaha
	1	DSF	O	VRAM Special Function Select. Sampled at $\overline{\text{RAS}}$ fall, DSF=0 during refresh cycles resets Masked (write-per-bit) Mode to non-persistent. During Masked Writes, DSF must be low (=0). During Non-Masked Writes, DSF=0 selects normal DRAM, and DSF=0 selects Register LOAD. Sampled at CAS fall of DRAM cycles, DSF selects BLOCK Mode.
	1	$\overline{\text{BIOS\_CS}}$	O	Flash ROM chip select
	1	$\overline{\text{OPL\_CS}}$	O	Yamaha FM synthesizer chip select. The synthesizer data bus is attached to VRAM_DATA[7:0].
	1	$\overline{\text{OPL\_IRQ}}$	I	Yamaha interrupt input (with an internal pull-up).
	1	$\overline{\text{RESET\_OUT}}$	O	Reset to supported peripherals: BtV2487, BtV2811A, BtV2300, Yamaha, etc

Table 2. BtV2115 PCI-Bus Pin LIst Summary *(4 of 4)*

	Pin Count	Signal Name	I/O	Description
Total serial interface pins = 7	1	AES_OUT	O	Output to AES compatible consumer electronics (such as DAT tape, etc.)
	1	SA_OUT	O	Serial audio output to compatible audio stream interface such as the BtV2300.
	1	SA_IN	I	Serial audio input from the audio ADC. This bit serial input supports left and right channel 16 bit input and is asynchronous to MCLK. This signal is internally pulled up.
	2	IIC_SCL, IIC_SDA	I/O	I^2C interface pins, used for I^2C bus. Optional for ACCESS.bus for pre-DDC monitor
	2	DDC_SCL, DDC_SDA	I/O	I^2C interface pins, used for I^2C bus for support of the VESA DDC1 and DDC2 signalling modes as well as for ACCESS.bus when presented through the monitor VGA cable, as in VESA DDC2B signalling.
Total test pins = 4	3	$\overline{TM}$[2:0]	I	Test mode, signals, 111= normal operation, 000 = tri-state, all outputs. These signals are internally pulled up.
	1	AUDCLK_OUT	O	Audio clock for driving the AudioStream Interface clock-in. For test purposes, the 50MHz internal clock of BtV2115 is output when $\overline{TM}$[2:0] = 011. When $\overline{TM}$[2:0] = 010, a delayed version of the 50MHz internal clock of BtV2115 is output. For normal mode ($\overline{TM}$[2:0] = 11x), audio clock is output
Total PCI-Bus Signal Pins = 155				
Total non-signal pins = 53	19	GND		Ground
	17	VDD		± 5V Power Supply
	1	G_PLL		Ground for internal PLL
	1	V_PLL		Power for internal PLL
	12	PCI_PDN[11:0]	I	PCI pulldown pins
	3	PCI_NC[2:0]		No connect
Total BtV2115 PCI-Bus Pins = 208				

VGA IMPLEMENTATION

VGA Compatibility

The BtV2115 is fully compatible with VGA applications. Upon configuring and enabling the BtV2115 VGA sub-module, the chip emulates standard VGA hardware without the aid of software. (For PC platforms, BtV-supplied video BIOS properly initializes the chip for VGA operation.) Once the chip is configured, applications may take full advantage of the VGA-defined hardware capability and switch VGA modes either through BIOS calls or by programming the VGA register set directly.

The BtV2115 also supports VESA BIOS extensions to standard VGA modes (Super VGA modes). However, for Super VGA, applications must switch modes through BIOS calls. This is reasonable (and standard practice) since the selection of Super VGA pixel clock frequencies is non-standard in industry VGA implementations. However, in the majority of VESA BIOS modes, applications can freely program those VGA registers which don't determine the basic monitor timing.

Within a register definition, the functional definition is implemented for all of the fields required for VGA compatibility. If the BtV2115 does not support a field's functionality that was present in previous VGA implementations, it is only because that particular functionality was required by the previous implementation, but not required by current implementations. Nevertheless, any software that was written to access that field, may write and read the field as before. The written values are stored in the VGA register for later retrieval. The only difference is that value in the register will have no effect on VGA operation. An example of such fields are the Asynchronous Reset and Synchronous Reset bits in the Sequencer Reset register.

VGA Operation

This specification does not intend to define the VGA architecture and functionality; application programmers should consult a book on the subject. However, the chapter "Register Definitions" provides a list of VGA registers, including I/O addresses, read/write capability, and a list of the fields within that register.

Simultaneous Operation with Other BtV2115 Modules

The BtV hardware does not preclude the simultaneous operation of the VGA and GUI or VGA and Video modules. However, such operations are usually not feasible because VGA applications typically do not include a driver-based interface to GUI or Video functionality. BtV2115 assumes that GUI and Video applications (such as Microsoft Windows or Video for Windows) will run with VGA hardware disabled, and provides drivers accordingly. A GUI application could run in that situation using a standard VGA driver, but the GUI block would be disabled and only the VGA module enabled. Whether motion video applications work with a VGA graphics driver is a design choice in the coding of the BtV2115 low-level video driver.

A full-screen DOS window within Windows can also take advantage of the VGA hardware. However, when switching to a full-screen DOS window, the GUI hardware is again disabled so that GUI and VGA hardware are not actually enabled concurrently.

The BtV product provides hardware, firmware, and software to run VGA applications which access audio hardware through standard I/O locations.

VGA and PACDAC Controller

The VGA definition originally assumed that the VGA hardware directly drove the monitor timing signals and directly sequenced pixel data to the DAC. This was done through a VGA sub-module called the CRT Controller. Since the BtV2115 instead implements a packet-based interface to a multi-media PACDAC device, the BtV2115 VGA module implemented some unique mechanisms to emulate the VGA CRT Controller functionality. Basically, the VGA module must generate data structures in the VRAM media buffer based on the contents of VGA I/O registers and the VGA memory-mapped frame buffer. The BtV2115 PACDAC Controller will then process these data structure to send packets to the PACDAC device.

The VGA frame buffer is stored in a 256KB region starting at address zero of the VRAM media buffer, and the PACDAC data structures are stored at a configurable region outside of this 256KB region.

Here is a brief review of the packet-based architecture. (For more detail consult the chapter "PACDAC Controller".) Rather than sending pixel and monitor control signals to the DAC in real time, packets describing the screen are burst to a PACDAC device on a line-by-line basis. The PACDAC buffers these packets in FIFOs,

and then constructs the scan line on the monitor screen by unloading the contents of these FIFOs in real time.

Several packet types are defined on the packet bus, but for VGA, the important packet types are graphics, timing, and DAC register (or I/O) packets. The graphics packets contain the pixel data, the timing packets contain the monitor timing information, and the DAC register packets contain values which will be programmed into configuration registers in the PACDAC device.

Graphics packets are only sent during scan lines which contain active pixels. Timing packet types are sent on every line. They are required to maintain the monitor sync timing. They also control when the graphics pixel data is unloaded from the FIFO for the active portion of the screen. DAC register (or I/O) packet types are sent at the start of every frame during vertical blanking.

The VGA uses these I/O packets to program the DAC palette (based on the VGA Color Registers), the border color (based on the VGA Attribute Overscan Register), and the pixel clock rate (based on the Miscellaneous Output Register). Consequently, until the CPU writes a new value to either the VGA DAC Color Registers, the Attribute Overscan register, or the pixel clock select field in the Miscellaneous Output register, these packets will program the same values into the PACDAC registers at the beginning of every frame.

Therefore, rather than sending the VGA pixels to a standard DAC, the VGA module must write the pixel data back into the VRAM media buffer at a location and format expected by the BtV2115 PACDAC Controller. There is a data structure for each packet type: graphics display (the pixel values, which are 8-bit palette indexes for VGA), timing, and I/O (or DAC Register). Thus, the VGA module is responsible for generating a graphics display structure based on the VGA frame buffer contents, generating a timing structure in VRAM based on the contents of the VGA CRT Registers and Sequencer Clocking Mode Register, and generating parts of the DAC register structure based on VGA Color Registers, the VGA attribute Overscan Register, and the clock select bit of the VGA Miscellaneous Output Register.

There is also a Master Structure required by the BtV2115. The Master Structure contains generic configuration information as well as pointers to the other packet-specific data structures described above. The Master Structure also contains other fields for each packet type which tell it how to send the packets. Some of these fields depend on the VGA mode, and consequently the VGA module will modify these fields too. These fields are shown in Table 3.

Table 3. VGA Modified Fields

Master Structure Field	Purpose
GD_X	Controls the number of graphics packets to send on each line. Depends on VGA horizontal resolution.
GD_Y	Controls which scan lines contain active video. Depends on VGA Vertical Display Enable, Vertical Sync Start, and Vertical Total Registers.
TS_LENGTH	Controls the number of timing atoms in the timing structure. Depends on any VGA CRT register which affects the timing structure, which is most of them.

Once again, the VGA frame buffer is always mapped into the first 256 KB of the media buffer's addressing space, and the other data structures are stored anywhere else, depending on how the Master Structure is set up. In turn, the Master Structure's location depends on the PDC_MSPTR registers.

The VGA module initiates writes to the timing structure, DAC Register data structures, and the Master Structure only upon detecting modifications to the corresponding VGA registers. However, it continually cycles through the VGA frame buffer to translate its contents into the graphics data structure

The VGA reads the master structure to determine where the graphics and timing data structures are to be stored. (It determines where the master structure is located from the PDC_MSPTR registers.)

On the other hand, to determine where to store the DAC Register values, it reads internal registers, PDAC_PTR1 and PDAC_PTR0 (see "DAC Alias Pointer Registers" on page 46). This method was chosen over reading the DAC register pointer from the master structure because the VGA does not generate the whole DAC register structure, it just modifies a part of it. Software or BIOS must build the rest of the DAC register structure around the VGA-generated data. This register gives the code flexibility as to where to locate the VGA-generated data within the structure. The VGA always stores DAC register values that it modifies at fixed positions relative to the location indicated by the PDAC_PTR registers. As shown in Table 4, the writes to the DAC Register Structure in the media buffer are triggered by writes to the corresponding VGA registers.

Table 4. DAC Register Offset

DAC Register Type	Offset from PDAC_PTR in Dwords	Modification Triggered by:
DAC Palette	0-255	Palette writes to 3c9 DAC Data Register
Border Color Index (Overscan)	257	Writes to VGA Attribute Overscan Register
Pixel Clock PLL	258	Writes to Clock Select field of VGA Miscellaneous Output Register

See the *BtV2487 PACDAC Specification* for details about the format of the DAC register data for each register type.

Another important aspect of the interaction between the VGA and PACDAC Controller modules is mode switching. A hardware mechanism allows dynamic mode switching. Here, “dynamic mode switching” means that an application can directly modify the values of the VGA CRT Registers without having to disable the VGA module. The mechanism allows the VGA module to specify to the PACDAC Controller when to switch between the two Master Structures which are defined as part of the PACDAC Controller architecture. This special mechanism is necessary to prevent the CRT Controller from modifying the structures while the PACDAC Controller is in the middle of generating the structures. Modifying any VGA register that affects either the Timing Structure or Master Structure would cause a system failure if this mechanism were not implemented.

In general, the PACDAC Controller provides dual Master Structures to allow mode switching. This allows software (or in this case the VGA hardware) to modify one set of structures while the PACDAC Controller continues to process the active set. When the modifications are done, software (or VGA hardware) requests the PACDAC Controller to switch the active master structure at the end of the current frame. The VGA module is aware of the two Master Structures. It modifies the inactive structure (and associated data structures pointed to by the Master Structure). When done, it asserts an internal hardware signal requesting the PACDAC Controller to switch the active structure. For this mechanism to work, the timing structure pointer of each Master Structure (TS_PNTR), must be different. All other values in the both Master Structures can be identical.

The algorithm for switching modes is as follows. First, the procedure is kicked off by any modification to any of the bit fields which affect either the Timing or Master Structure. The VGA regenerates the entire Timing Structure and the Master Structure entries listed in Table 3, regardless of which bit was modified. The fields that will kick off a mode switch include bits 3 and 0 of the VGA Sequencer Clock Mode Register and most of the VGA CRT Registers. The VGA module reads the location of the inactive Master Structure from an internal register. It then reads the Timing Structure pointer from the Master Structure itself. Next it starts processing the VGA registers to generate a new Timing Structure and some Master Structure entries. Upon completing the modifications, the VGA module asserts a hardware

signal which requests that the BtV2487 switch Master Structures at the end of the frame.

The VGA module keeps track of whether changes to VGA CRT registers were made while it was generating structures in the media buffer. Therefore, no modifications will be missed. The CRTC continues to generate the inactive structure and the PACDAC controller does not switch structures until structure generation completes with no further pending modifications. Assuming that the software or BIOS does a series of consecutive writes to VGA CRT registers, the first register modification will kick off a mode switching procedure. While this is happening, the VGA module sets an internal flag which indicates that modifications are pending. The BtV2115 architecture does not require that VGA software modify the VGA CRT Registers in any specific order. Intermediate frames may not have the desired monitor timing, but they will not cause system failure.

VGA Extension Registers

Some BtV2115 implementation-specific register fields are unique to the VGA. Some are provided to configure how the VGA backend logic writes data structures into the media buffer. Other fields provide status of the VGA modules. These are implementation-specific registers which are located in the VGA Graphics Controller Extension space. However, not all registers in that space are related to VGA, and hence they are not all listed below. For a complete list of all registers, consult the "Register Definitions" chapter starting on page 33.

The VGA fields are summarized below.

Enable VGA CRT Controller Mechanism

The ENABLE_VGA_CRTC field in the GRP_CFG4 register (bit 2, index 44h) enables the VGA module to write to any PACDAC structure (see Table 22 on page 50). If disabled, neither the graphics, timing, or master structure data will be modified. If enabled, the ENABLE_VGA_CRTC_GEN bit must also be enabled in order to allow VGA generation of the Timing Structure and parts of the Master Structure. Otherwise, the VGA will only generate the graphics data. This bit will not disable access to VGA I/O registers or the VGA memory-mapped frame buffer.

Enable VGA CRT Controller Structure Generation

The ENABLE_VGA_CRTC_GEN field in the GRP_CFG5 register (bit 7, index 45h) enables the VGA module to write the to the Timing Structure and Master Structure areas (see Table 21 on page 49). This bit is only effective if the ENABLE_VGA_CRTC bit is set also. When set to zero, this bit allows the VGA to run with software-generated timing structures. Note that this bit does not disable the generation of the Graphics Data structure.

PACDAC Pointer

The PDAC_PTR field in the GRP_PDAC1 register (bits 2:0, index 27h) and GRP_PDAC0 register (bits 7:0, index 26h) determines where in the media buffer the VGA will store DAC Register values (see"DAC Alias Pointer Registers" on page 46). These registers are mapped into the bits 21:11 of the pointer, so that the pointer can only be set in 2KB increments. The least significant bits are always set to zero.

25 MHz PLL Select

Registers GRP_25PLL1 (index 1Bh) and GRP_25PLL0 (index 1Ah) specify the 16-bit PLL value to write to the DAC Register Structure when the VGA Miscellaneous Output Register is set for a 25 MHz pixel clock. Refer to "PLL25 Select Registers" on page 44. For definition of the PLL format, see *BtV2487 PACDAC Specification*.

28 MHz PLL Select

Registers GRP_28PLL1 (index 1Eh) and GRP_28PLL0 (index 1Dh) specify the 16-bit PLL value to write to the DAC Register Structure when the VGA Miscellaneous Output Register is set for a 28 MHz pixel clock. Refer to "PLL28 Select Registers" on page 45. For definition of the PLL format, see *BtV2487 PACDAC Specification*.

Attribute Index Status

Bit 0 of the GRP_VGASTAT register (index 2Ch) reflects the status of the Attribute Register port (see "VGA Status Register" on page 47). If a zero, writing to 3C0h will write the Attribute Index Register. If a one, writing to 3C0h will write the indexed Attribute Data register.

VGA CRT Generation Status

When bit 1 of the GRP_VGASTAT register (index 2Ch) is set to 1, the VGA is busy performing a mode switch. This means it is generating a new Timing Structure and Master Structure and when done will request the BtV2487 to switch structures. When this bit is zero, it means that the structures have been completed. This bit can be used by software to manage mode switching, but it is not necessary for standard VGA operation. Refer to "VGA Status Register" on page 47.

VGA Vertical Sync Select

The VGA_SYNC_SEL field of the GRP_CFG4 register (bit 3, index 44h) selects the source of the VGA vertical sync status (see Table 22 on page 50). The internal VGA sync status is used to set the Vertical Retrace Interrupt (status is shown in Bit 7 of the VGA Input Status 0 Register) and the Vertical Sync Pulse (Bit 3 of the VGA Input Status 1 Register.)

VGA Read Latches

Registers GRP_RDLAT3 (index 2Bh) through GRP_RDLAT0 (index 28h) allow access to the 32 bit VGA Read Latch. The VGA Read Latch (defined by the VGA criteria) is an entity which is loaded with a 32-bit frame buffer value for each VGA memory read. Read Latch Register [0] is the least significant byte of the read latch. See "Read Latch Registers" on page 46.

VGA DAC Mode

Bit 0 of the GRP_VGACFG register enables the VGA DAC 8-bit mode (see "VGA Configuration Register" on page 44). The default for the VGA DAC logic is the 6-bit mode. The 8-bit mode means that VGA DAC writes and reads to 3C9h are assumed to be 8-bit DAC values, and are therefore transferred between the DAC register block of the media buffer and the CPU bus unmodified. The 6-bit mode means that DAC VGA access to 3C9h are 6-bit DAC values, and therefore a translation is done between the 6-bit value on the CPU bus and the 8-bit value in the DAC register block of the media buffer. This is done for both reads and writes.

Configuration for VGA

The BtV2115 requires the initialization of several registers and media buffer locations for VGA operation. The BtV2115 provides drivers and BIOS routines to do this for common applications and environments. Below is an outline of the required steps.

1. Decide how to allocate the media buffer. The VGA frame buffer image will occupy the first 256KB of the media buffer. In the remainder, the programmer must allocate locations for two Master Structures, one Graphics Data structure, two Timing Structures, and one DAC Register structure.
2. Set Master Pointer Registers. Write to the both Master Pointer A (PDC_MSPTRA)and Master Pointer B (PDC_MSPTRB) registers so that they point to Master Structures A and B.
3. Create Master Structure Skeleton. Initialize Master Structure A and B. The programmer must initialize the PKT_ORDR, GD_PNTR, TS_PNTR, DR_PNTR, DR_END, and CLK_MOD entries. The GD_Y, GD_X, and TS_LENG double words can be zeroed out if the VGA Structure Generation bit is enabled (ENABLE_VGA_CRTC_GEN). Other unused fields such as the video fields can also be zeroed out. Table 5 provides recommendations for Master Structure fields.

Table 5. Recommended Master Structure Fields

Field	Comments
PKT_ORDR	Set the order to Timing, Graphics, DAC Register (IO). Thus PKT_ORDR = EEEEE526h.
GD_PNTR	Different GD structures can be allocated for each structure A and B, but it is unnecessary.
TS_PNTR	The pointer in each structure must point to different locations; there must be two Timing Structures.
DR_PNTR	The pointer in each structure must point to the SAME location. The VGA module does not generate new DAC Register values when it requests a master pointer switch. It only generates new values when corresponding VGA registers are modified.
DR_END	Configure the DR_BLK sub-field for no more than about 100
CLK_MOD	A value of 0000B00Bh works well for standard VGA modes, assuming a serial clock of 35MHz. Some Super VGA modes might need different values.

The VGA was designed for a packet order of Timing, Graphics, and DAC Register (PKT_ORDR = EEEEE526h). A CLK_MOD value of 0000C00Ch has worked well for standard VGA modes.

4 Create DAC Register structure skeleton. The code must build a DAC Register structure around the areas which will be written to by the VGA module. Below, Table 6 provides a typical strategy for creating a DAC structure.

Table 6. Typical DAC Structure

Offset from DR_PNTR (Dwords)	Contents
0	Write an entry which loads PACDAC device address register to 100h. This is the address of the palette.
1 - 256	PDAC_PTR points here. This is where the VGA stores the palette.
257	Write an entry which loads PACDAC device address register to 0h. This is the address of the border color index. The VGA module does not write to this address.
258	The VGA module writes the border color index value here.
259	The VGA module writes the pixel PLL value here.
260	Write the Serial Clock PLL value here. The VGA design is somewhat flexible, but 25 MHz works well for standard VGA modes.
261 - (n-1)	Write PACDAC register values for PACDAC device addresses 03h through FFh. Stop at FFh or when there are no more defined registers. Most register values in this range do not affect VGA functionality and can be set to zero.
n	Write an entry which loads PACDAC device address register to 200h.
n - end	Write PACDAC register values for any remaining addresses above 200h.

5 Write Pixel Clock PLL registers. (For definition of the PLL format, see *BtV2487 PACDAC Specification*.) If the mode is not a standard VGA mode (i.e. a SuperVGA mode), write the PLL value to the location in the DAC Register structure which contains the Pixel Clock PLL value.

6 Write PDAC Pointer Register. This register should be set to point within the DAC Register structure. Typically, it will point to the second double word within the structure.

7 Enable VGA and VGA Structure generation. Set the ENABLE_VGA_CRTC and the ENABLE_VGA_CRTC_GEN bits. However, if the mode uses software generated timing structures, set the ENABLE_VGA_CRTC_GEN bit to zero.

Support for VESA BIOS Extension Modes

The VESA BIOS Extension specification defines standard BIOS function calls which provide software access to resolutions, color depths, and frame buffer organizations beyond the VGA hardware standard. This allows software developers to take advantage of many unique implementations of Super VGA (SVGA) functionality.

VBE Modes

The BtV2115 hardware supports VESA BIOS Extension (VBE) functions and most modes. The VESA modes that use VGA-compatible memory models support the use of VGA hardware. Those that use 256-color non-chain4 and direct color memory models are supported, but without VGA hardware. In these modes the BtV2115 behaves as a dumb frame buffer controller supporting 8-bit color depths. VGA IO operations will effect neither the frame buffer accesses nor the display format. Status and control functions should be done through calls to the VBE functions.

Frame Buffer Models

The VBE specification defines two ways to implement a frame buffer aperture: VGA window or linear/flat model. The set mode function includes an extra bit which specifies which model to use. The BtV2115 supports both.

For the VGA windowing frame buffer model, the BtV2115 provides dual real mode apertures via VGA address space. Each port may be configured for a 32K or 64K window size. The 32K and 64K mode aperture addresses are defined in Table 75 on page 124. Each port includes read/write capability and a paging mechanism which allows an application to “window” into the frame buffer.

For the linear/flat buffer model, the BtV2115 chip provides a protected mode aperture which can be located anywhere within the CPU’s 32-bit address space. For more information, refer to “Real Mode Aperture” on page 127.

Timing Structures and High Resolution Modes

The VGA timing structure generation mechanism is used to support applications which change modes by writing directly to VGA CRT Controller registers, rather than making a BIOS call. However, a few issues arise with this mechanism in higher resolutions modes such that applications have to use BIOS for non-standard VGA modes. There is no standard mechanism for switching to high resolution modes independently of making a BIOS call. Therefore, it is wiser for BIOS or application drivers to load the timing structure into the media buffer rather than rely upon the VGA hardware to do it. This also means that the VGA structure generation mechanism must be disabled by clearing the ENABLE_VGA_CRT_GEN bit (bit 7) in the GRP_CFG5 register.

The biggest issue that arises with VGA hardware generated timing structures is the fixed position of the Lsync pulse, which works for standard VGA resolutions but could fail for some SVGA resolutions.

Mode Setting Procedure

The steps required to initialize a VBE mode depend on a couple of factors:

- Does the mode use a VGA compatible memory model? If not, the VGA hardware must be disabled through the ENABLE_VGA_CRT bit.
- Can the mode use VGA-generated timing structures? If not, the ENABLE_VGA_CRT_GEN should be set to '0' to disable timing data generation. BIOS must load the timing structure.

In general, here is the procedure:

1. Initialize PACDAC Controller structures in the media buffer, if not already initialized. Initialize the register fields which configure the VGA module.
2. If the mode does not use a VGA-compatible memory model (VBE models 0 through 4), then disable VGA CRT Controller functionality by clearing the ENABLE_VGA_CRT bit (bit 2 of GRP_CFG4). In this case, enable the real-mode or protected-mode aperture.
3. If the resolution exceeds standard VGA modes (such as 720 X 400 or 640 X 480), disable generation of the timing structure. Load the mode's timing structure into the media buffer.
4. Modify the Clock Modulation field (CLK_MOD) of the master structure. Like the timing structure, the clock modulation rate must be tailored for each mode. (See the "PACDAC Controller" chapter starting on page 195.)
5. Modify the Pixel Clock fields and Serial Clock fields of the DAC Register Structure. If the 25 MHz or 28 MHz pixel clock will be selected through the VGA Miscellaneous Output Register, then the pixel clock field does not need to be modified. The VGA will generate the Pixel Clock PLL value automatically.
6. Follow appropriate mode switching procedure. (See the "PACDAC Controller" chapter starting on page 195.)

Logical Window Control

The VBE functions include logical window setup and control (such as setting the logical line width and adjusting the display start address). These functions can be used for horizontal panning, vertical scrolling, and display buffer switching for animation.

The BtV2115 chip provides the capability to support these functions in hardware, although horizontal panning must be used with caution. The display start address can be changed by modifying the GD_PNTR field of the master structure. The logical scan line length can be adjusted by modifying the PTCH field of the GD_X entry in the master structure. (See the "PACDAC Controller" chapter starting on page 195.)

When implementing horizontal panning, keep in mind the following. First, the start address has a resolution of 32 bits. Thus in an 8-bit mode, the screen can be panned in only four-pixel increments. (BIOS can theoretically get around this limitation by modifying the timing structure to pan on a pixel basis.) Second, when the start address is not aligned to 256-byte boundaries, the performance of the serial packet bus will be affected. Each scan line could require more Read Transfer operations to the VRAM. In some modes, this slowdown will cause system failure. As a result, horizontal panning should only be supported if the resolution is feasible and if the mode can support the performance degradation on the serial port.

DAC Palette Format

The VBE includes a function to set/get the number of bits per palette primary color. For palette access via the VGA 3C9h port, the BtV2115 controller supports both 6-bit and 8-bit formats, which are the most common formats. The selection is made through bit 0 of the GRP_VGACFG register. A '1' indicates an 8-bit mode, while a '0' indicates a 6-bit mode (refer to "VGA Configuration Register" on page 44).

This bit only affects accesses via the VGA 3C9h port. Palette operations via this address can be translated between 6-bit and 8-bit formats. When configured for 6-bit modes, palette operations via 3C9h are translated between 6-bit and 8-bit formats. On the other hand, any palette values stored directly into the frame buffer are always in the 8-bit format. However, BIOS can still perform the translation if the palette is accessed through the Load/Unload Palette Data function call.

REGISTER DEFINITIONS

Introduction

The BtV2115 presents both an I/O space interface and a memory interface to its internal registers. In addition, PCI bus implementations of the BtV2115 present a PCI configuration space interface to certain registers (see “PCI Configuration Space” on page 219). Not all registers are accessible from I/O space nor are all registers accessible from memory space. I/O space registers include the I/O addresses used by VGA controllers as well as the I/O space registers used by DOS-based game audio. Some registers such as the GUI accelerator registers and the windows-based audio/video registers are only available through CPU memory instructions. Some registers, such as bits 31:25 of the GUI_BASE register are accessible through I/O space and through the PCI bus configuration space.

Table 7 shows a complete list of the I/O port addresses that are implemented within the BtV2115 and their corresponding alternate decode addresses (see “Parallel Decoder Interface” on page 266). Notice that all I/O port addresses are present for software compatibility with either VGA or DOS based game audio, i.e. the BtV Legacy Audio emulation. Since these I/O spaces were required for software compatibility, they were extended to control most new BtV2115 functions. In a PCI environment, access to these exact I/O addresses are requested by defining two configuration function spaces and declaring a VGA compatible class code and a Legacy sound device class code. For information on the PCI interface refer to “PCI Bus Interface” on page 216.

Table 7. BtV2115 I/O Address Map *(1 of 3)*

I/O Port	Alternate Decoder Port	Write	Read
0102	0102	MCA pos 102	MCA pos 102
03B4	33C4	VGA CRT index (mono)	CRT index (mono)
03B5	33C5	VGA CRT data (mono)	CRT data (mono)
03BA	33CA	VGA feature control (mono)	Input status reg 1 (mono)
03C0	23C0	VGA attribute index/data	Attribute index
03C1	23C1	Cycle accepted from bus but no operation performed.	Attribute data
03C2	23C2	VGA miscellaneous output	Input status reg 0
03C3	23C3	Mother board sleep	Mother board sleep
03C4	23C4	VGA Sequencer index	VGA Sequencer index
03C5	23C5	VGA Sequencer data	VGA Sequencer data
03C6	23C6	VGA DAC pixel mask	VGA DAC pixel mask
03C7	23C7	VGA DAC address read mode	VGA DAC state
03C8	23C8	VGA Pixel address write mode	VGA Pixel address write mode
03C9	23C9	VGA Pixel data	VGA Pixel data
03CA	23CA	Cycle accepted from bus but no operation performed	VGA Feature control read
03CC	23CC	Cycle accepted from bus but no operation performed	VGA Miscellaneous output read
03CE	23CE	VGA Graphics controller index	VGA Graphics controller index
03CF	23CF	VGA Graphics controller data	VGA Graphics controller data
03D4	23D4	VGA CRT index (color)	VGA CRT index (color)
03D5	23D5	VGA CRT data (color)	VGA CRT data (color)
03DA	23DA	VGA Feature control (color)	VGA Input status reg 1 (color)
46E8	46E8	Adapter sleep	Adapter sleep
0220	0220	Bank0 address	Bank0 status
0221	0221	Bank0 data	Reserved
0222	0222	Bank1 address	Bank1 status
0223	0223	Bank1 data	Reserved
0224	0224	Mixer address	Reserved
0225	0225	Mixer data	Mixer data
0226	0226	DSP reset	Reserved

Table 7. BtV2115 I/O Address Map *(2 of 3)*

I/O Port	Alternate Decoder Port	Write	Read
0227	0227	Reserved	Reserved
0228	0228	Bank address	Bank status
0229	0229	Bank data	Bank data
022A	022A	Reserved	DSP read data
022B	022B	Reserved	Reserved
022C	022C	DSP cmd-data	DSP write status
022D	022D	Reserved	Reserved
022E	022E	Reserved	DSP read data status
022F	022F	Reserved	Reserved
0240	0240	Bank0 address	Bank0 left status
0241	0241	Bank0 data	Reserved
0242	0242	Bank1 address	Bank1 status
0243	0243	Bank1 data	Reserved
0244	0244	Mixer address	Reserved
0245	0245	Mixer data	Mixer data
0246	0246	DSP reset	Reserved
0247	0247	Reserved	Reserved
0248	0248	Bank address	Bank status
0249	0249	Bank data	Bank data
024A	024A	Reserved	DSP read data
024B	024B	Reserved	Reserved
024C	024C	DSP cmd-data	DSP write status
024D	024D	Reserved	Reserved
024E	024E	Reserved	DSP read data status
024F	024F	Reserved	Reserved
0388	0388	Bank0 address	Bank0 status
0389	0389	Bank0 data	Reserved
038A	038A	Bank1 address	Bank1 status
038B	038B	Bank1 data	Reserved

Table 7. BtV2115 I/O Address Map *(3 of 3)*

I/O Port	Alternate Decoder Port	Write	Read
3F8-3FF	3F8-3FF	COMM PORT 0	COMM PORT 0
2F8-2FF	2F8-2FF	COMM PORT 1	COMM PORT 1
3E8-3EF	3E8-3EF	COMM PORT 2	COMM PORT 2
2E8-2EF	2E8-2EF	COMM PORT 3	COMM PORT 3
0201	0201	JOY STICK/GAME PORT	JOY STICK/GAME PORT
0330-0331	0330-0331	MPU PORT	MPU PORT

The VGA emulation supports six distinct sets of I/O space addresses: VGA Graphics registers, VGA Sequencer registers, VGA CRTC registers, VGA Attribute registers, VGA Color registers, and VGA External registers.

The graphics registers are accessed by loading the index of the desired register into the Graphics Index register at I/O port address 3CEh followed by reading or writing the desired data from the register through the Graphics Data register at I/O port address 3CFh. See ‘‘VGA Graphics Registers (I/O Mapped)’’ on page 38 for more detail.

Similarly, VGA Sequencer registers are accessed by loading the index of the desired register into the Sequencer Index register at I/O port address 3C4h followed by reading or writing the desired data from the register through the Sequencer Data register at I/O port address 3C5h. See ‘‘VGA Sequencer Registers (I/O Mapped)’’ on page 62 for more detail.

The VGA CRT Controller registers are accessed by a similar mechanism with the CRTC Index registers at either I/O port address 3B4h or 3D4h and the CRTC Data register at either I/O port address 3B5h or 3D5h. See ‘‘VGA CRT Controller (I/O Mapped)’’ on page 69 for more detail.

The VGA Attribute registers are also indexed, but the mechanism has the standard VGA characteristic that both the index and data values pass through the same I/O port address at 3C0h. Reading 3C0h returns the contents of the indexed attribute register. Reading 3C1h returns the contents of the Attribute Index register. See ‘‘VGA Attribute Controller (I/O Mapped)’’ on page 79 for more detail.

The VGA External registers are accessed at their specified I/O port addresses without further Indexing. See ‘‘VGA General / External Registers (I/O Mapped)’’ on page 84 for more detail.

The VGA Color registers have their own, standard index mechanism involving I/O port addresses 3C6h, 3C7h, 3C8h, and 3C9h. See ‘‘VGA Color Registers (I/O Mapped)’’ on page 89 for more detail.

The BtV2115 Legacy Audio emulation supports three sets of I/O space addresses. 388h through 38Bh provide access to the Yamaha OPL3 registers. The bank of I/O port addresses from 220h to 24Fh is duplicated in the bank of I/O port addresses from 240h to 22Fh. Only one of these banks can be active at a time. Only one DOS-based game audio subsystem can be active at a time so that if another is installed, the BtV2115 Legacy Audio emulation should be disabled, see ''Legacy Audio FM Registers (I/O Mapped)'' on page 91, ''Audio Registers (Memory Mapped)'' on page 96, and "Audio Interface" chapter starting on page 271.

VGA Graphics Registers (I/O Mapped)

The BtV2115 MediaStream Controller uses indexed addressing to access the 149 control registers in this block. The first nine registers are standard VGA registers and the remainder are BtV2115 extensions. The offset into the VGA data space is provided through 3CFh.

Graphics Index Port

name: GRP_INDEX
port: 3CEh, read/write
size: 8 bit
function: The value in this register determines which register in the graphics controller block is written or read when performing an I/O cycle to the GRP_DATA address. Only the lower 4 bits of the Index Register are valid when the BtV2115 extensions are locked. When the BtV2115 extensions are unlocked, all 8 bits of the Index Register are used. Bit 7:4 are held at zero, when locked.

Graphics Data Port

name: GRP_DATA
port: 3CFh, read/write
size: 8 bit
function: The various VGA GRAPHICS registers are accessed through this I/O whenever their corresponding index has been loaded into GRP_INDEX.

Set/Reset Register

name: GRP_SR
index: 3CFh.00h, read/write
size: 8 bit
function: This register is loaded with the pattern to be written to the display planes in Write Mode 0 or Write Mode 3. There is a 1 to 1 correspondence between the bits in this register and the display planes. The bit contents of the Set/Reset Register are shown in Table 8.

Table 8. Set/Reset Register

Bit	Function
7:4	Reserved
3	Set/Reset Plane 3
2	Set/Reset Plane 2
1	Set/Reset Plane 1
0	Set/Reset Plane 0

Enable Set/Reset Register

name: GRP_ESR
index: 3CFh.01h, read/write
size: 8 bit
function: This register enables or disables the individual planes affected by the Set/Reset Register. This register is only used in Write Mode 0. The bit contents of the Enable Set/Reset Register are shown in Table 9.

Table 9. Enable Set/Reset Register

Bit	Function
7:4	Reserved - hardwired to 0h
3	Enable Set/Reset Plane 3
2	Enable Set/Reset Plane 2
1	Enable Set/Reset Plane 1
0	Enable Set/Reset Plane 0

Color Compare Register

name: GRP_CC
index: 3CFh.02h, read/write
size: 8 bit
function: The color compare bits contain the value to which all eight bits of the corresponding memory plane are compared. The Color Don't Care register establishes the mask for the memory planes. This register is only used in Read Mode 1. A '1' is returned for the plane positions where the bits of all 4 planes equal the Color Compare Register. The bit contents of the Color Compare Register are shown in Table 10.

Table 10. Color Compare Register

Bit	Function
7:4	Reserved - hardwired to 0h
3	Color Compare Plane 3
2	Color Compare Plane 2
1	Color Compare Plane 1
0	Color Compare Plane 0

Data Rotate Register

name: GRP_ROT
index: 3CFh.03h, read/write
size: 8 bit
function: This register contains the Rotate Count and the Function Select fields. The Rotate Count field specifies the number of bit positions to rotate data right by the CPU in Write Mode 0 or 3. The Function Select field selects a Boolean between the write data and the data contained in the Read Latches. The logical functions are enabled for Write Mode 0, 2, and 3. The bit contents of the Data Rotate Register are shown in Table 11.

Table 11. Data Rotate Register

Bit	Function	Detail
7:5	Reserved	Hardwired to b'000
4:3	Function Select 1:0	00 = data unmodified 01 = data ANDed with data in latches 10 = data ORed with data in latches 11 = data XORed with data in latches
2:0	Rotate Count 2:0	Amount to rotate right CPU data

Read Map Select Register

name: GRP_RDPLN
index: 3CFh.04h, read/write
size: 8 bit
function: This register selects which planes are read in Read Mode 0. The bit contents of the Read Map Select Register are shown in Table 12.

Table 12. Read Map Select Register

Bit	Function	Detail
7:2	Reserved	Hardwired to 00h
1:0	Map Select 1:0	00 = Plane 0 01 = Plane 1 10 = Plane 2 11 = Plane 3

Mode Register

name: GRP_MODE
index: 3CFh.05h, read/write
size: 8 bit
function: This register controls the VGA Read and Write modes, data shift registers, and other addressing modes. The bit contents of the Mode Register are shown in Table 13.

Table 13. Mode Register

Bit	Function	Detail
7	Reserved	Hardwired to 0
6	256 Color Mode	0 = Configure shift registers for 2, 4, or 16 colors 1 = Configure shift registers for 256 colors
5	Shift Register	0 = Configure shift registers for EGA/VGA compatibility 1 = Configure shift registers for CGA compatibility
4	Odd/Even	0 = Normal frame buffer read addressing 1 = Even/odd addressing - use CPU address bit 0 to select odd or even planes on reads
3	Read Mode	0 = Read Mode 0 - Read data from planes selected by Read Map Select register (has no effect if bit 3 of the Sequencer Memory Mode Register = 1) 1 = Read Mode 1 - Read comparison of the planes and Color Compare register
2	Reserved	Hardwired to 0
1:0	Write Mode 1:0	The Write Mode field controls the configuration of the datapath for VGA writes. It controls how the Set/Reset, Enable Set/Rest, Logic Unit, Data Rotator, and Bit Mask Logic are interconnected. Consult a referenced VGA publication for more detailed information. 00 = Write Mode 0 01 = Write Mode 1 10 = Write Mode 2 11 = Write Mode 3.

Graphics Miscellaneous Register

name: GRP_MISC
index: 3CFh.06h, read/write
size: 8 bit
function: This register controls the VGA display mode, monochrome graphics emulation, and odd/even addressing. The bit contents for this register are shown in Table 14.

Table 14. Graphics Controller: Miscellaneous Register

Bit	Function	Detail
7:4	Reserved	Hardwired to 0h
3:2	Memory Map 1:0	00 = 128KB A0000-BFFFFh 01 = 64KB A0000-AFFFFh 10 = 32KB B0000-B7FFFh 11 = 32KB B8000-BFFFFh
1	Chain Odd/Even	This bit controls whether CPU address A0 is used for memory address bit 0, or replaced with another value (chaining). Consult a VGA text for more detail. 0 = Normal addressing 1 = Chain Odd/Even function is enabled
0	Graphics Mode	0 = Text mode 1 = Graphics mode

Color Don't Care Register

name: GRP_CCX
index: 3CFh.07h, read/write
size: 8 bit
function: This register selects which planes will be used in a color comparison between the values in the Graphics Controller Color Compare register and the data read from video memory. This comparison is only done while in Read Mode 1.If the map bit = 1, color comparison is enabled for that plane or map. The bit contents of the Color Don't Care Register are shown in Table 15.

Table 15. Color Don't Care Register

Bit	Function
7:4	Reserved - hardwired to 0h
3	Memory Map 3
2	Memory Map 2
1	Memory Map 1
0	Memory Map 0

Bit Mask Register

name: GRP_BITMASK
index: 3CFh.08h, read/write
size: 8 bit
function: This register is a mask for modifying displayed pixels. A bit set to a 1 allows the corresponding pixel to be changed. The LSB corresponds to the rightmost pixel in the group. This register is disabled with Write Mode 1. The bit contents of the Bit Mask Register are shown in Table 16.

Table 16. Bit Mask Register

Bit	Function
7:0	Bit Mask

Extension Registers Accessed Via VGA Space (I/O Mapped)

This section describes general purpose I/O registers accessed through the VGA space. The offset into the VGA data space is through 3CFh.

VGA Configuration Register

name: GRP_VGACFG
index: 3CFh.16h, read/write
size: 8 bit
function: Writing 01h to this register enables the VGA DAC 8-bit mode. The default for the VGA DAC logic is the 6-bit mode. The 8-bit mode means that VGA DAC writes and reads to 3C9h are assumed to be 8-bit DAC values, and are therefore transferred between the DAC register block of the media buffer and the CPU bus unmodified. The 6-bit mode means that DAC VGA access to 3C9h are 6-bit DAC values, and therefore a translation is done between the 6-bit value on the CPU bus and the 8-bit value in the DAC register block of the media buffer. This is done for both reads and writes.

Chip ID Registers

name: GRP_CID[1:0]
index: 3CFh.18h:17h, read only
size: 16 bit
function: These registers contain the 16 bit BtV2115 chip ID value. If the chip design were released in June 1995, the chip ID would be 9506h.

ROM Page Register

name: GRP_ROMPAGE
index: 3CFh.19h, read/write
size: 8 bit
function: For VL mode, the 5 least significant bits of this register are appended to the VGA_ROM aperture CPU address to select any 32KB page in the 1MB Flash ROM. For PCI mode, this register has no effect since CPU address bits [19:2] are passed to the ROM (refer to “VGA ROM Aperture” on page 129).

PLL25 Select Registers

name: GRP_25PLL[2:0]
index: 3CFh.1Ch:1Ah, read/write
size: 24 bit
function: When the VGA Miscellaneous Output Register is programmed to select a 25MHz pixel clock for the VGA, this register contains the PLL register value that is programmed into the PACDAC chip. The VGA CRTC block writes this value into the DAC register block of the media

buffer so that it will be sent to the PACDAC on the packet interface. The bit contents for this register are shown in Table 17.

For information on the pixel clock, refer to the *BtV2487 PACDAC Specification*, "Pixel Clock PLL" section.

Table 17. PLL 25MHz and 28MHz Select Registers

Bits	Function
23:16	Reserved
15	Source crystal reference selection 0= XTAL25 (24.576 MHz) 1= XTAL17 (16.934 MHz)
14:12	Reserved
11:10	Post scalar factor (variable L) 00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8
9:6	Value of N, range: 1 to 15 (0h to Fh)
5:0	Value of M, range: 1 to 63 (0h to 3Fh)
The Pixel Rate is defined as: (M/(NxL)) x crystal frequency	

PLL28 Select Registers

name: GRP_28PLL[2:0]
index: 3CFh.1Fh:1Dh, read/write
size: 24 bit
function: When the VGA Miscellaneous Output Register is programmed to select a 28MHz pixel clock for the VGA, this register contains the PLL register value that is programmed into the PACDAC chip. The VGA CRTC block writes this value into the DAC register block of the media buffer so that it will be sent to the PACDAC on the packet interface. The bit contents for this register are shown in Table 17.

For information on the pixel clock, refer to the *BtV2487 PACDAC Specification*, "Pixel Clock PLL" section.

GUI Base Registers

name: GRP_GUI_BASE[3:0]
index: 3CFh.23h:20h, read/write
size: 32 bit
function: These registers select the base addresses for the four 8MB windows of the Graphics Users Interface/Media Buffer Aperture. For detail on the usage and bit definitions of these registers refer to "GUI Base Address Register" on page 123 of the "CPU Address Space Apertures" chapter.

DAC Alias Pointer Registers

name: GRP_PDAC[1:0]
index: 3CFh.27h:26h, read/write
size: 16 bit
function: When the VGA module needs to update values in the DAC register block of the media buffer, this register determines the base address of the region where it will make its modifications. If we call the media buffer pointer PDAC_PTR, it is generated as follows:

PDAC_PTR[21:19] = GRP_PDAC1[2:0]
PDAC_PTR[18:11] = GRP_PDAC0[7:0]
PDAC_PTR [10:0] = 0

The bit contents for the GRP_PDAC register are shown in .

Table 18. GRP_PDAC Registers

Register	Bits	Description
GRP_PDAC1	7:3	Reserved - all zeroes
	2:0	Three bits of the 22 bit media buffer pointer - PDAC_PTR[21:19]
GRP_PDAC0	7:0	Eight bits of the 22 bit media buffer pointer - PDAC_PTR[18:11]

The VGA block is responsible for updating the DAC palette, the border color index value, and the pixel clock PLL based on values written to VGA I/O registers. Therefore writes to the media buffer are triggered by writes to DAC_DATA (“DAC Data Register” on page 90), MISC_OUT_W (“Write Miscellaneous Output Register” on page 87), and ATT_OVRS (“Overscan Color Register” on page 81). Data retrieval reads from the media buffer are triggered by reads from the DAC_DATA register only.

The VGA block always puts the DAC palette at locations PDAC_PTR + 0 through PDAC_PTR + 1023. (The RGB values for color index 0 go at PDAC_PTR + 0 to PDAC_PTR + 2. Color index 1 values go from PDAC_PTR + 4 to PDAC_PTR + 6, and so on.) It puts the Border Index value at PDAC_PTR + 1028, and the Pixel Clock PLL value at PDAC_PTR + 1032.

Refer to “DAC Data” on page 206 for additional information on the DAC.

Read Latch Registers

name: GRP_RDLAT[3:0]
index: 3CFh.2Bh:28h, read only
size: 32 bit
function: These four registers allow access to the 32 bit VGA Read Latch. The VGA Read Latch is an entity defined by VGA criteria which is loaded with a 32-bit frame buffer value for each VGA memory read. Read Latch Register [0] is the least significant byte of the read latch.

VGA Status Register

name: GRP_VGASTAT
index: 3CFh.2Ch, read/write
size: 8 bit
function: This register returns a '1' in bit position zero if pointing to the Attribute Data register and a '0' if pointing to the Attribute Index register, reflecting the direction of a write to 3C0h as either the Attribute Index or Attribute Data register.

This register also reports the status of the VGA CRT Controller. Bit 1of this register reflects CRT generator status. When bit 1 = 1, the CRT controller is busy generating the CRTC data structures. When bit 1 = 0, the CRT controller is finished generating data structures.

Configuration Registers

name: GRP_CFG[7:0]
index: 3CFh.47h:40h, read/write
size: 64 bit
function: These eight registers allow operating parameters to be set at power-on for the BtV2115 and modified later for desired chip operation through video BIOS. For more information see the "Configuration Registers" starting on page 133.

Table 19 through Table 26 define the contents of the configuration registers.

Table 19. Configuration Register <7> (GRP_CFG7)

Bits	Function
7	PCI_PREF_BASE (VRAM data bus 25) - For PCI bus only. 1 = Enable PCI Prefetch Base register. Must also be enabled by the PCI configuration mechanism. Allows the 32MB aperture to also appear as a prefetchable address range. Refer to "PCI Prefetchable Base Address Registers" on page 222 0 = Hide 32MB prefetchable base address
6	Wake Up State at Power On (VRAM data bus 24) 1 = VGA awake at power on 0 = VGA asleep at power on
5:0	VRAM Type: Read only field, indicates the type of VRAM attached to the BtV2115. Contains the resistor strapped state of VRAM data bus [15:10]. See "Supported VRAM Types," Table 139 on page 229.

Table 20. Configuration Register <6> (GRP_CFG6)

Bits	Function
7	1 = PCI two clock write enable. The cycle must be a memory write cycle and the address must match the non-zero value in GRP_GUI_BASE[31:25] (see "GUI Base Registers" on page 45). (default) 0 = PCI two clock write not enabled.
6	1 = Burst memory write enable (default) 0 = For all cycle types, attempts to burst are disconnected with a Type C Disconnect/Retry mechanism (see the *PCI Local Bus Specification, Revision 2)* $\overline{\text{PCI\_STOP}}$ is asserted with $\overline{\text{PCI\_TRDY}}$ deasserted (see "PCI Bus Interface" on page 216). The BtV2115 takes the first data phase and disconnects the second.
5	Reserved
4	Interrupt Request Polarity 1 = IRQ active high (default) 0 = IRQ active low
3	Audio insert filter clock select: Selects the filter clock rate in use for the switched cap filter. This bit is only read by software and is set by VRAM data bus 23. 1= 100X external filter 0 = 50X external filter
2	Enable save/restore location in SEQ_UNLOCK register (see "Unlock Register" on page 67) 1 = Enable context save/restore 0 = Disable (default)
1	Memory burst size max 4 for GUI and video bursts 1 = max of 4 0 = max of 8 (default)
0	Block write enable for GUI / GUI BIST control 1 = GUI will use block write where appropriate (default) 0 = GUI will use byte enable controlled writes even for cases where block write would offer significant performance improvement. - or- 0 = Run GUI BIST If set to zero when GRP_CFG4[0] (GUI Enable) transitions to high (is set to 1), the GUI RAM Built-In Test (BIST) will run on the GUI result FIFO RAMs. Bit 19 of the GUIREG_DEPTH register will reflect GUI_BUSY status while BIST is running. BIST pass/fail and completion status is reflected in the GUIREG_CFG register (see Table 107 on page 177). To disable block writes without running BIST, set GRP_CFG4[0] = 1 before writing block write enable to zero.

Table 21. Configuration Register <5> (GRP_CFG5)

Bits	Function
7	Enable VGA CRTC Generator (ENABLE_VGA_CRT_GEN) 1 = CRTC structure generation is enabled so that certain VGA I/O cycles will modify PACDAC control structures in the VRAM. This includes Master Structure and the Timing Structure. This bit must be set in conjunction with GRP_CFG4[2], the Enable VGA CRTC bit. This bit does not affect the generation of graphics data nor DAC Register data. 0 = No VRAM cycles issued in response to VGA I/O cycles (default).
6	Clock Modulator Reset: 1 = Force pixel tag RST to the BtV2487 from the PACDAC controller until cleared 0 = Normal operation (default)
5:0	Monitor ID Bits: Allow monitors to be identified by reading their unique ID. These bits are loaded at the end of reset from a ‘244 which buffers the monitor id pins from the VGA connector to the VRAM data bus. Bits [5:4] are used as board revision information for BIOS and drivers. The BIOS Init Sequence (see Table 23) saves these bits to the BIOS Save area (top 128 bytes of VRAM) and overwrites bits [1:0] with the VRAM size as follows: Bits[1:0] -- 00 = 1M 01 = 2M 10 = 3M 11 = 4M

Table 22. Configuration Register <4> (GRP_CFG4) *(1 of 2)*

Bits	Function
7	DAC Snoop: Allows the CPU bus to contain an additional write-only palette DAC which share address space with the BtV2115 VGA Color Registers. This includes 03C6-03C9h. The BtV2115 will capture VGA Color Register writes, but will not drive any CPU bus handshaking signals. This is called snooping the writes. Reads are handled normally. 1 = VGA DAC writes are snooped, while VGA DAC reads sourced from the BtV2115 as normal. 0 = Both reads and writes are handled by BtV2115(default).
6	ROM Write Enable 1 = Enable Flash ROM write mode 0 = Disable Flash ROM write cycles (default)
5	BIOS ROM Enable for VGA ROM aperture at 000C000h -000C7FFFh. 1 = Enable ROM access 0 = Disable ROM aperture. ROM cycles on CPU bus are ignored.
4	This bit is valid for VL bus only. ROM Hole from 000C6000h-000C67FFh used to be a compatibility problem for some network cards. 1 = Enable ROM hole in memory address map, i.e. the BtV2115 will not respond to ROM reads within the ROM HOLE. 0 = BtV2115 responds to ROM reads within the ROM HOLE, if BIOS ROM is enabled by bit 5.

Table 22. Configuration Register <4> (GRP_CFG4) *(2 of 2)*

Bits	Function
3	VGA_SYNC_SEL - Select the source of the VGA vertical sync status. The internal VGA sync status is used to set the Vertical Retrace Interrupt (status is shown in Bit 7 of the VGA Input Status 0 Register) and the Vertical Sync Pulse (Bit 3 of the VGA Input Status 1 Register.) 1 = Pseudo VGA sync. The sync status is asserted when the VGA CRTC finishes drawing to display buffer. 0 = Real VGA sync. VGA vertical sync status is based on monitor vertical sync (default)
2	Enable VGA CRTC block (ENABLE_VGA_CRTC). The VGA CRTC block maps the VGA frame buffer into the display buffer, according to the current VGA register settings. This bit must be set to enable generation of graphics data, parts of the Master Structure, and the Timing Structure. However, if the ENABLE_VGA_CRTC_GEN bit (GRP_CFG5, bit 7) is not set, then the VGA will only be able to generate the graphics data. 1 = Enable the VGA CRT controller. 0 = Disable the VGA CRT controller, force CRTC into idle state (default).
1	Enable PACDAC Controller to send timing, cursor, video and graphics packets to the BtV2487. 1 = PACDAC controller enabled 0 = No monitor refresh functions (default).
0	GUI Enable 1 = Enable GUI and GUI portions of FBA logic 0 = Reset all of GUI plus GUI portions of FBA logic (default) “GUI portions” are defined as: Anything having to do with the function or access of the GUI VRAM queue. Anything having to do with reading or writing queued or non-queued registers, including registers that control bit flipping in the Flippin Map.

Table 23. Configuration Register <3> (GRP_CFG3)

Bits	Function
7	BIOS_RESET_STATE Indicates BIOS Init Sequence has been executed one or more times. 0 = BIOS Init Sequence not executed (default) 1 = BIOS Init Sequence has been executed
6	Decode Test: Enable alternate decode address space so that the VGA real mode aperture A0000-BFFFF becomes 860A0000-860BFFFFh, the VGA ROM aperture at C0000-C7FFF becomes 860C0000h-860C7FFFh. The VGA I/O's at 03Cxh become 23Cxh. Both mono and color VGA registers at 03Bxh and 03Dxh become 33Cxh. Audio I/O registers remain unchanged as does the 46E8h adapter sleep address. 1 = Enable alternate decodes. 0 = Enable normal decodes.
5	Sleep Bit: The BtV2115 can be configured as a VGA motherboard device which responds to sleep address 03C3h or as an adapter card VGA which responds to sleep address 046E8h and the MCA position address at 0102h. 1 = Adapter sleep at 46E8h and 0102h 0 = Motherboard sleep at 03C3h
4:3	These read only bits indicate which CPU bus type is in use by the BtV2115. The BtV2115/V version can initialize to any valid combination of bits, while the BtV2115/P can only initialize to 0. 00 = VL bus 01 = PCI bus
2	Fast GUI: This read-only bit indicates whether the VRAM GUI queue is present in the BtV2115 1 = Fast 0 = Slow
1	Reserved read-only. Always = 1
0	Reserved read-only. Always = 1

Table 24. Configuration Register <2> (GRP_CFG2)

Bits	Function
7:0	Refresh Rate Count. This value is used as a down count to determine when to issue refresh cycles to VRAM. Units are MCLK/8. Time between successive refresh cycles is ($Period_{MCLK}$ • 8) • GRP_CFG2). One row in each four VRAM bank is refreshed per refresh cycle. Initial value is set to 0xB4.

Table 25. Configuration Register <1> (GRP_CFG1)

Bits	Function
7	Reserved - write 0
6	Block Mode. 0 = 4X8 block write mode (default) 1 = TI 4Mbit block write mode or 2MBit block write mode (4X4)X4
5	ROM Speed: This bit selects the speed of the ROM interfaced to the BtV2115. 0 = 120ns 1 = 70ns
4	SAM Length: This bit sets the length of the serial shift register in VRAM. 0 = 256 entries (default) 1 = 512 entries
3	Communications decode enable: enables PCI/VL decoder to capture communications port addresses and pass the I/O cycles to the VRAM data bus as rom cycles above 512K of Flash ROM space. 1= accept one of four comm port I/O register bank decodes 0= ignore communications port I/O
2:1	Communications port bank select when bit 3 =1 00= comm port 0 @ 03F8-03FF maps to 07803F8-F 01= comm port 1 @ 02F8-02FF maps to 07802F8-F 10= comm port 2 @ 03E8-03EF maps to 07803E8-F 11= comm port 3 @ 02E8-02EF maps to 07802E8-F
0	MIDI port decode enable: enables PCI/VL decoder to capture MPU401 I/O cycles and map them to the Flash ROM space. 1 = accept I/Os to 0330-0331 and map to 07A0330-1 0= ignore MPU 401 I/O cycles

Table 26. Configuration Register <0> (GRP_CFG0)

Bits	Function
7	Joy stick decode enable: enables PCI/VL decoder to capture game port I/O cycles and map them to the FLASH ROM space. 1 = Accept I/Os to 0201 and map to 07C0201 0 = Ignore game port I/O cycles
6:5	RAM Speed: This field sets the access rate of the VRAM. 00 = 80 ns (default) 01 = 70 ns 10 = 60 ns 11 = default 80 ns, writes 00
4	Dual CAS: This bit determines whether a VRAM requires dual CAS support. Configure BtV2115 for dual CAS VRAMs. 1 = Dual CAS 0 = Dual WE
3	Reserved for Spare B
2	Disable Refresh: 1 = disable VRAM refresh cycles
1	Reserved for Spare A
0	2 Mbit refresh. 0 = Force DSF low during CBRR cycles 1 = Force DSF high during CBRR cycles DSF is Device Special Function select. CBRR is CAS Before RAS Refresh with Reset.

I^2C Control Register

name: GRP_I2C_CTRL
index: 3CFh.4Bh, read/write
size: 8 bit
function: The I^2C serial control interface specifies the use of either the master or slave A bus or B bus. For more information on this register, refer to "I^2C Control WRITE Register" on page 138.

I^2C Slave Control Register

name: GRP_I2C_SCTRLR, GRP_I2C_SCTRLW
index: 3CFh.4Ch, read/write
size: 8 bit
function: This register is the slave device I^2C control bits. The slave control registers are defined differently for read and write operations. For bit definitions of these registers, refer to "I2C Slave Control Read Register" on page 152.

I²C Slave Data Register

name: GRP_I2C_SDATA
index: 3CFh.4Dh, read/write
size: 8 bit
function: This register contains the slave device I²C data. When this register is read it contains the contents of the slave module I²C input data buffer. System software should poll the DONE bit in the I²C Slave Control Register to verify completion of a transaction before reading to avoid corruption of incoming data.

When this register is written it contains data to be transmitted over the I²C interface. System software should poll the DONE bit in the I²C Slave Control Register to verify completion of a transaction before writing to avoid corruption of outgoing data.

For bit definitions of this register, refer to "I2C Slave Data Register" on page 151.

I²C Master Control Register

name: GRP_I2C_MCTRLR, GRP_I2C_MCTRLW
index: 3CFh.4Eh, read/write
size: 8 bit
function: This register contains the master device I²C control bits. For bit definitions of this register, refer to "I²C Master Control Read Register" on page 142 and "I²C Master Control Write Register" on page 144.

I²C Master Data Register

name: GRP_I2C_MDATA
index: 3CFh.4Fh, read/write
size: 8 bit
function: This register contains the master device I²C data. When this register is read, it contains the contents of the master I²C input data buffer. System software should poll the DONE bit in the I²C Master Control Register to verify completion of a transaction before reading to avoid corruption of incoming data.

When this register is written it contains data to be transmitted over the I²C interface. System software should poll the DONE bit in the I²C Master Control Register to verify completion of a transaction before writing to avoid corruption of outgoing data.

For more bit contents information on this register, refer to "I²C Master Data Register" on page 141.

Master Structure A Address Register

name: PDC_MSPTRA[2:0]
index: 3CFh.59h:57h, read/write
size: 24 bit
function: These three registers contain the dword address for the VRAM starting location of Master Structure A. The registers are concatenated to form the 20 bit pointer field; refer to Table 27. Only one Master Structure (A or B) can be active at a time.

For more information, refer to “Master Structure” on page 197 of the “PACDAC Controller” chapter.

Table 27. Master Structures A and B Address Registers

Bits	Function
24:23	Not used
22:02	Dword address for the VRAM starting address
01:00	Not used

Master Structure B Address Register

name: PDC_MSPTRB[2:0]
index: 3CFh.5Ch:5Ah, read/write
size: 24 bit
function: These three registers contain the dword address for the VRAM location of Master Structure B. The registers are concatenated to form the 20 bit pointer field; refer to Table 27. Only one Master Structure (A or B) can be active at a time.

For more information, refer to “Master Structure” on page 197 of the “PACDAC Controller” chapter.

PACDAC Controller Status Register

name: PDC_STAT
index: 3CFh.5Dh, read only
size: 8 bit
function: This register monitors the PACDAC Controller status. The bit contents of the PACDAC Controller Status Register are shown in Table 28.

PDC_STAT[0] reports which Master Structure (A or B) is in use; and PDC_CNTL[0] is used to request a switch in Master Structure pointer. For example, a 1 in PDC_CNTL[0] and a 0 in PDC_STAT[0] indicates that a switch from A to B has been requested, but has not been executed. The PACDAC Controller updates the Master Structure pointers at the start of a frame, after all timing structure data for the frame is sent and all data for the last line is sent to the PACDAC.

Dual pointers for each video window are supported to increase the quality of displayed video images. These pointers, DV1_PNTRA, DV1_PNTRB, DV2_PNTRA, DV2_PNTRB are loaded via the Master Structure initialization mechanism at the start of each frame. Each pointer indicates the locations of buffered video data with two buffers allocated for each video window. Switching from A to B and B to A is controlled by the video module which is outside the PACDAC Controller module. The video module sets or clears flags for DV1 and DV2 to tell the PACDAC Controller which pointer (A or B) to use for the two video windows. The effects of these flags occurs at the start of each frame.

Table 28. PACDAC Controller Status Register

Bits	Function	Detail
7		Reserved
6	VERT_DISP_EN	0 = disabled 1 = enabled
5	LSYNC	0 = disabled 1 = enabled
4	DV2_PNTR Status	0 = DV2_PNTRA in use (default) 1 = DV2_PNTRB in use
3	DV1_PNTR Status	0 = DV1_PNTRA in use (default) 1 = DV1_PNTRB in use
2	Reserved	
1	Frame Sync status bit	0 = Frame active 1 = Starting new frame
0	MSPNTR status	0 = MSPNTRA in use (default) 1 = MSPNTRB in use

PACDAC Controller Control Register

name: PDC_CNTL
index: 3CFh.5Eh, read/write
size: 8 bit
function: This register controls the PACDAC Controller functions for switching PDC_MSPNTR and halting the sequencer. The bit contents of the PACDAC Controller Control Register are shown in Table 29.

PDC_CNTL[0] is used to request a switch in Master Structure pointer; and PDC_STAT[0] reports which Master Structure (A or B) is in use. For example, a 1 in PDC_CNTL[0] and a 0 in PDC_STAT[0] indicates that a switch from Master Structure A to Master Structure B has been requested, but has not been executed. The PACDAC Controller updates the Master Structure pointers at the start of a frame, after all timing structure data for the frame is sent and all data for the last line is sent to the PACDAC.

Table 29. PACDAC Controller Control Register

Bits	Function	Detail
7:5		Reserved
4	Screen mode	1 = Interlaced screen 0 = Non-interlaced screen
3		Reserved - write 0
2	EOF halt mode	End of frame disable. 0 = Normal operation 1 = Disable PACDAC at end of current frame and clear GRP_CFG4[1] (PACDAC enable)
1	EOL halt mode	End of line disable. 0 = Normal operation 1 = Disable PACDAC at end of current line and clear GRP_CFG4[1] (PACDAC enable)
0	MSPNTR usage command bit	0 = Use MSPNTRA at next EOF 1 = Use MSPNTRB at next EOF

VGA PCI Configuration Space (I/O Mapped)

The BtV2115 Graphics Controller uses the following registers to access the PCI Configuration Space from the VGA IO space. For additional information, refer to "PCI Configuration Space" on page 219.

The offset into the VGA data space is 3CFh.

Read PCI Prefetchable Base Address Register

name: GRP_PBASE
index: 3CFh.24h, read only
size: 8 bit
function: This register provides read access to the PCI prefetchable base address register. The bits are defined in Table 30.

Table 30. Read PCI Prefetch Base Address

Bits	Description
7:1	PCI_PREF_BASE0[31:25] prefetch address, refer to "PCI Prefetchable Base Address Registers" on page 222
0	Always 0

Read PCI ROM Base Address Register

name: GRP_ROMBASE
index: 3CFh.25h, read only
size: 8 bit
function: This register provides read access to the PCI ROM base address register. The bits are defined in Table 31.

Table 31. Read PCI ROM Base Address

Bits	Description
7:0	PCI_ROM_BASE0[31:24] ROM address, refer to "PCI ROM Base Address Register" on page 223

Read PCI Enable Registers

name: GRP_PCI_EN
index: 3CFh.2Dh, read only
size: 8 bit
function: This register provides read access to various PCI enable register values. The bits are defined in Table 32.

Table 32. Read PCI Enable Registers

Bits	Description
7:5	Always 0
4	PCI enable IO cycles in the audio range. PCI_IO1_EN value specified in PCI_COMMAND1[0]; refer to “PCI Command Registers” on page 220
3	PCI enable snoop on DAC. PCI_SNOOP_DAC value specified in PCI_COMMAND0[5]; refer to “PCI Command Registers” on page 220
2	PCI enable memory cycles. PCI_MEM_EN value specified in PCI_COMMAND0[1]; refer to “PCI Command Registers” on page 220
1	PCI enable IO cycles in the VGA address space. PCI_IO0_EN value specified in PCI_COMMAND0[0]; refer to “PCI Command Registers” on page 220
0	PCI enable ROM cycles. PCI_ROM_EN value specified in PCI_ROM_BASE0[0]; refer to “PCI ROM Base Address Register” on page 223

SEV Registers (I/O Mapped)

The SEV Player registers are defined in detail starting on page 185 of “Software Encoded Video (SEV) Player” chapter. Below are brief summaries of the SEV registers.

Softvideo Controller Write Register

name: GRP_SV_CTRLW[3:0]
index: B4h:B1h, write only
size: 32 bit
function: Refer to “SEV Controller Write Register” on page 189.

Softvideo Controller Read Register

name: GRP_SV_CTRLR[3:0]
index: B4h:B1h, read only
size: 32 bit
function: Refer to “SEV Controller Read Register” on page 191.

VGA Sequencer Registers (I/O Mapped)

The Sequencer uses indexed addressing to access seven different control registers. The first six registers are standard VGA registers. The seventh register is a BtV2115 extension. The offset into the VGA sequencer space is 3C5h.

Sequencer Index Register

name: SEQ_INDEX
port: 3C4h, read/write
size: 8 bit
function: This register specifies the register in the sequencer block to be accessed by the next I/O read or write to the SEQ_DATA address. Only bits 2:0 are writable. Bits 7:3 are hardwired to zero.

Sequencer Data Port

name: SEQ_DATA
port: 3C5h, read/write
size: 8 bit
function: This port accesses the sequencer data registers. The register accessed depends on the value of SEQ_INDEX.

Reset Register

name: SEQ_RST
index: 3C5h.00h, read/write
size: 8 bit
function: This register has no effect on the behavior of BtV2115. A 00h should be written to this register during mode set; bits 7:2 are hardwired to zero.

Clocking Mode Register

name: SEQ_CLK
index: 3C5h.01h, read/write
size: 8 bit
function: This register contains the fields necessary to establish the video clock timing and control display refresh. The bit contents of the Clocking Mode Register are shown in Table 33. The options for the shift register control are shown in Table 34.

Table 33. Clocking Mode Register

Bits	Function	Detail
7:6	Reserved	Hardwired to 00
5	Display refresh	0 = Normal operation 1 = Display refresh stops
4	Shift & Load 32	Control display shift registers
3	Dotclk	0 = Normal operation 1 = Divide by 2
2	Shift and Load 16	Control display shift register
1	Reserved	Hardwired to 0
0	8/9 Dot clocks	0 = 9 dot wide character clock 1 = 8 dot wide character clock

Table 34. Shift & Load Control Bits

Bits 4 &2	Function
00	Every character clock
01	Every 2nd character clock
10	Every 4th character clock
11	Every 4th character clock

Map Mask Register

name: SEQ_WPMASK
index: 3C5h.02h, read/write
size: 8 bit
function: This register masks write access to the video buffer planes by the host CPU. The bit contents of the Map Mask Register are shown in Table 35.

Table 35. Map Mask Register *(1 of 2)*

Bit	Function	Detail
7:4	Reserved	Hardwired to 0h

Table 35. Map Mask Register *(2 of 2)*

Bit	Function	Detail
3	Map3 enable	0 = no write 1 = write allowed
2	Map2 enable	0 = no write 1 = write allowed
1	Map1enable	0 = no write 1 = write allowed
0	Map0 enable	0 = no write 1 = write allowed

Character Map Select Register

name: SEQ_CFS
index: 3C5h.03h, read/write
size: 8 bit
function: This register determine which character font will be used for display output. Bit 7 of the text attribute byte determines whether the primary or secondary font is used. The bit contents of the Character Map Select Register are shown in Table 36.

Table 36. Character Map Select Register

Bit	Function
7:6	Reserved - hardwired to 00
5	Secondary 0
4	Primary 0
3	Secondary 2
2	Secondary 1
1	Primary 2
0	Primary 1

Table 37 and Table 38 define the Primary and Secondary bits used to select font tables. This is **only** valid for **Text** modes. Text fonts are normally loaded into video buffer planes 2 or 3. The offset in the tables below refer to the offset from the base of plane 2 or 3.

Table 37. Secondary Font Selection

Bits 5,3,2	Map Selected	Plane Offset
000	0	0K
001	1	16K
010	2	32K
011	3	48K
100	4	8K
101	5	24K
110	6	40K
111	7	56K

Table 38. Primary Font Selection

Bits 4,1,0	Map Selected	Plane Offset
000	0	0K
001	1	16K
010	2	32K
011	3	48K
100	4	8K
101	5	24K
110	6	40K
111	7	56K

Memory Mode Register

name: SEQ_MMODE
index: 3C5h.04h, read/write
size: 8 bit
function: This register controls miscellaneous setup features related to the video buffer and sequencer control. The bit contents of the Memory Mode Register are shown in Table 39.

Table 39. Memory Mode Register

Bit	Function	Detail
7:4	Reserved	Hardwired to 0h
3	Chain 4	0 = Normal addressing 1 = Packed pixel mode; CPU address bits A1 and A0 determine which plane is accessed, and buffer address bits A3, A2 are replaced by CPU address bits A15, A14.
2	Odd/Even	0 = Even CPU addresses access Map0 and Map2, odd CPU addresses access Map1 and Map3 1 = Normal addressing
1	Extended memory	This bit only affects the odd/even addressing functionality controlled by Chain Odd/Even bit in the VGA Graphics Controller Miscellaneous register. This bit never needs to be set to zero for BtV2115 systems. 0 = 64KB of video memory 1 = 256KB of video memory for VGA modes
0	Reserved	Hardwired to 0

VGA Sequencer Extension Registers (I/O Mapped)

Unlock Register

name: SEQ_UNLOCK
index: 3C5h.06h, read/write
size: 8 bit
function: This register enables or disables access to the BtV2115 extension registers. If the extensions are unlocked, reading this register will return a status value. If the extensions are locked, reading this register will return 0Fh. The bit contents of the Unlock Register are shown in Table 40.

Table 40. Sequencer Unlock Register

Bit	Function	Detail
7:5	Reserved	
4:0	Unlock	Writing 12h to this field unlocks the extension registers. Reads a 12h status. Writing 14h to this field saves VGA context and unlocks the VGA at interrupt service routine entry. Reads a 14h status. Writing 18h to this field restores the VGA context after an interrupt. Reads back the previous status value (never contains 18h)

The BtV2115 implements a context save restore mechanism in the VGA block, that drastically reduces the I/O overhead for entering and leaving the interrupt-service routine.

When 14h is written to the sequencer unlock register (3C5h.6), the current state is saved as follows:

1. Copy current sequencer lock register value to a shadow byte4.
2. Copy current graphics index register (3CE) to a shadow byte0.
3. Copy GUI_BASE0 to a shadow byte1.
4. Copy GUI_BASE1 to a shadow byte2.
5. Copy GUI_BASE2 to a shadow byte3.
6. Force unlock state in sequencer 6 register

When 18h is written to the sequencer unlock register, the following occurs:

1. Copy shadow byte 4 back to sequencer lock register
2. Copy shadow byte 0 back to graphics index register
3. Copy shadow byte 1 back to GUI_BASE0
4. Copy shadow byte 2 back to GUI_BASE1
5. Copy shadow byte 3 back to GUI_BASE2

The ISR input is as follows:

1. Save sequencer index register 3C4h
2. Write sequencer index to address the unlock and unlock, 6->3C4h
3. Write save value (14h) to 3C5h (2Ah and 2Bh are one out16)
4. Write to graphics index reg and GUI_BASE0 (3CEh,3CFh)
5. Write to graphics index reg and GUI_BASE1 (3CEh,3CFh)
6. Write to graphics index reg and GUI_BASE2 (3CEh,3CFh)
7. Read interrupt status through 000A0000h and process

The return from ISR is as follows:

1. Write restore value (18h) to sequencer unlock register
2. Restores unlock register and those shadowed graphics registers
3. Restore sequencer index register 3C4h

VGA CRT Controller (I/O Mapped)

The CRT Controller (CRTC) uses indexed addressing to access the control registers. The offset into the VGA CRTC data space is 3?5h, where "?" is either B (3B5h) or D (3D5h) depending on whether the CRT is monochrome or color, respectively.

CRTC Color Index

name: CRT_INDEX_C
port: 3D4h, read/write
size: 8 bit
function: This address accesses the CRT_INDEX register when VGA Miscellaneous Output Register bit 0 is set for a color display. This register contains the location of the register in the CRTC block to be accessed by the next I/O read or write to the CRT_DATA port.

CRTC Monochrome Index

name: CRT_INDEX_M
port: 3B4h, read/write
size: 8 bit
function: This address accesses the CRT_INDEX register when VGA Miscellaneous Output Register bit 0 is set for a monochrome display. This register contains the location of the register in the CRTC block to be accessed by the next I/O read or write to the CRT_DATA port.

CRTC Color Data

name: CRT_DATA_C
port: 3D5h, read/write
size: 8 bit
function: This port allows access to one of the CRT data registers, according to the value of the CRT_INDEX register. This register is accessible only for color, MISC_OUT_R[0] = 1 (see Table 64 on page 88).

CRTC Monochrome Data

name: CRT_DATA_M
port: 3B5h, read/write
size: 8 bit
function: This port allows access to one of the CRT data registers, according to the value of the CRT_INDEX register. This register is accessible only for monochrome, MISC_OUT_R[0] = 0 (see Table 64 on page 88).

Horizontal Total Register

name: CRT_HTOT
index: 3?5h.00h, read/write
size: 8 bit
function: This register sets the horizontal scan time and is loaded in terms of character time. It includes left and right borders, displayed characters, and horizontal retrace time (front porch, back porch, and sync pulse). If 'n' is the total line character count, then this register should be loaded with 'n-5'.

Horizontal Display End Register

name: CRT_HDSP_END
index: 3?5h.01h, read/write
size: 8 bit
function: This register contains the number of displayed characters on a horizontal line, and is loaded in terms of character clocks. If 'n' is the character count, then this register should be loaded with 'n-1'.

Start Horizontal Blank Register

name: CRT_HBLANK_ST
index: 3?5h.02h, read/write
size: 8 bit
function: This register contains the count at which horizontal blanking should begin. It is in terms of character clocks. If 'n' is the character count, then this register should be loaded with 'n-1'.

End Horizontal Blank Register

name: CRT_HBLANK_END
index: 3?5h.03h, read/write
size: 8 bit
function: This register contains the count at which horizontal blanking should end, in terms of character clocks. If 'n' is the character count, then this register should be loaded with 'n-1'. The MSB of the End Horizontal Blank field is in the End Horizontal Sync register. Together, these two registers specify only the least significant 6 bits of the count at which blanking will end. The actual 8-bit count at which blanking will end is the first time after the start blanking count that the least significant 6 bits of the character counter match these 6 bits.

The bit contents of the End Horizontal Blank Register are shown in Table 41.

Table 41. End Horizontal Blank Register

Bit	Function	Detail
7	Compatible Read	This field has no effect on BtV2115 operation.
6:5	Display Enable Skew	This field has no effect on BtV2115 operation.
4:0	HBLANK <4:0>	End Horizontal Blank count. The MSB is in the End Horizontal Sync register.

Start Horizontal Sync Register

name: CRT_HSYNC_ST
index: 3?5h.04h, read/write
size: 8 bit
function: This register contains the count at which the horizontal sync pulse should begin. It is in terms of character clocks. If ‘n’ is the character count, then this register should be loaded with ‘n-1’.

End Horizontal Sync Register

name: CRT_HSYNC_END
index: 3?5h.05h, read/write
size: 8 bit
function: This register contains the count at which the horizontal sync pulse should end. It is in terms of character clocks. If ‘n’ is the character count, then this register should be loaded with ‘n-1’. This register specifies only the least significant 5 bits of the count at which the sync pulse will end. The actual 8-bit count at which it will end is the first time after the start horizontal sync count that the least significant 5 bits of the character counter match these 5 bits. The bit contents of the End Horizontal Sync Register are shown in Table 42.

Table 42. End Horizontal Sync Register

Bit	Function	Detail
7	HBLANK 5	MSB of HBLANK
6:5	Reserved	
4:0	HSYNC <4:0>	Least significant 5 bits of Start Horizontal Sync register plus the width of HSYNC in character clocks.

Vertical Total Register

name: CRT_VTOT
index: 3?5h.06h, read/write
size: 8 bit
function: This register sets the number of scan lines in the frame. It contains the eight least significant bits. The most significant two bits are in the Overflow register. If 'n' is the scan line count, then this register should be loaded with 'n-2'.

Overflow Register

name: CRT_OVERFLOW
index: 3?5h.07h, read only
size: 8 bit
function: This register contains miscellaneous overflow bits associated with other vertical timing registers. The bit contents of the Overflow Register are shown in Table 43.

Table 43. Overflow Register

Bit	Function
7	Start VSYNC 9
6	End Vertical Display Enable 9
5	Vertical Total 9
4	Line Compare 8
3	Start VBLANK 8
2	Start VSYNC 8
1	End Vertical Display Enable 8
0	Vertical Total 8

Preset Row Scan Register

name: CRT_PRE_RS
index: 3?5h.08h, read/write
size: 8 bit
function: This register is used for line scrolling. The bit contents of the Preset Row Scan Register are shown in Table 44.

Table 44. Preset Row Scan Register

Bit	Function	Detail
7	Reserved	Hardwired to 0
6:5	Byte Pan	This field has no effect on BtV2115 operation.
4:0	Preset Row Scan Count	Specify scan line for 1st character row

Character Height Register

name: CRT_CHEIGHT
index: 3?5h.09h, read/write
size: 8 bit
function: This register specifies the cell height for characters and also contains two overflow bits and the scan line double bit. The bit contents of the Character Height Register are shown in Table 45.

Table 45. Character Height Register

Bit	Function	Detail
7	Scan Double	0 = normal operation 1 = activate line doubling
6	Line Compare 9	Overflow bit
5	Start VBLANK 9	Overflow bit
4:0	Cell Height	Number of scan lines / character

Cursor Start Register

name: CRT_CUR_ST
index: 3?5h.0Ah, read/write
size: 8 bit
function: This register sets the first scan line to display the cursor within the character cell which contains the cursor. It also turns the cursor off and on. The bit contents of the Cursor Start Register are shown in Table 46.

Table 46. Cursor Start Register

Bit	Function	Detail
7:6	Reserved	Hardwired to 00
5	Cursor Toggle	0 = cursor on 1 = cursor off
4:0	Top of Cursor	Starting scan line for cursor

Cursor End Register

name: CRT_CUR_END
index: 3?5h.0Bh, read/write
size: 8 bit
function: This register sets the last scan line to display the cursor within the character cell which contains the cursor. The bit contents of the Cursor End Register are shown in Table 47.

Table 47. Cursor End Register

Bit	Function	Detail
7	Reserved	Hardwired to 0
6:5	Cursor Skew	This field has no effect on BtV2115 operation.
4:0	Bottom of Cursor	Ending scanline for cursor

Start Address High Register

name: CRT_SCREEN_STH
index: 3?5h.0Ch, read/write
size: 8 bit
function: This register sets the high order 8 bits of the 16-bit quantity which determines the frame buffer address of the upper left corner of the display screen. The start address is in terms of character positions. The actual frame buffer starting address is scaled by a factor of 1, 2, or 4 depending on whether the byte, word, or double-word CRT addressing mode is selected. See the CRT Mode Control and CRT Underline register definitions.

Start Address Low Register

name: CRT_SCREEN_STL
index: 3?5h.0Dh, read/write
size: 8 bit
function: This register sets the low order 8 bits of the 16-bit quantity which determines the frame buffer address of the upper left corner of the display screen. The start address is in terms of character positions. The actual frame buffer starting address is scaled by a factor of 1, 2, or 4 depending on whether the byte, word, or double-word CRT addressing mode is selected. See the CRT Mode Control and CRT Underline register definitions.

Cursor Location High Register

name: CRT_CUR_LOCH
index: 3?5h.0Eh, read/write
size: 8 bit
function: This register sets the high order 8 bits of the 16-bit quantity that controls the cursor location. The location is in terms of character positions. It points to the character in the frame buffer which will contain the cursor. The actual frame buffer cursor address is scaled by a factor of 1, 2, or 4 depending on whether the byte, word, or double-word CRT addressing mode is selected. See the CRT Mode Control and CRT Underline register definitions.

Cursor Location Low Register

name: CRT_CUR_LOCL
index: 3?5h.0Fh, read/write
size: 8 bit
function: This register sets the low order 8 bits of the 16-bit quantity that controls the cursor location. The location is in terms of character positions. It points to the character in the frame buffer which will contain the cursor. The actual frame buffer cursor address is scaled by a factor of 1, 2, or 4 depending on whether the byte, word, or double-word CRT addressing mode is selected. See the CRT Mode Control and CRT Underline register definitions.

Start Vertical Sync Register

name: CRT_VSYNC_ST
index: 3?5h.10h, read/write
size: 8 bit
function: This register determines the start of the vertical sync pulse. The scan line counter is compared to this register at each horizontal sync time. This register contains the lower 8 bits of this value. Bits 8 and 9 are in the Overflow register. If ‘n’ is the scan line count, then this register should be loaded with ‘n-1’.

End Vertical Sync Register

name: CRT_VSYNC_END
index: 3?5h.11h, read/write
size: 8 bit
function: This register determines the end of the vertical sync pulse. The horizontal scan line count is compared to the Start Vertical Sync Register. If the values are equal, a vertical sync pulse is ended. This register specifies only the least significant 4 bits of the 10-bit count at which the sync pulse will end. The actual count at which it will end is the first time after the start vertical sync count that the least significant 4 bits of the line counter match these 4 bits. If ‘n’ is the scan line count, then this register should be loaded with ‘n-1’. Miscellaneous control fields are also supported in this register. The bit contents of the End Vertical Sync Register are shown in Table 48.

Table 48. End Vertical Sync Register

Bit	Function	Detail
7	CRTC [7:0] Write Protect	0 = Enable writes to CRT index registers 7:0h 1 = Disable writes to CRT index registers 7:0h, the Line Compare bit in the Overflow register is not affected
6	Reserved	
5	Enable Vertical Sync Interrupt	0 = Enable vertical sync interrupt 1 = Disable vertical sync interrupt
4	Clear Vertical Sync Interrupt	0 = Clear vertical sync interrupt 1 = Allows vertical sync interrupt generation
3:0	Vertical Sync End	Scan line count to end VSYNC

Vertical Display End Register

name: CRT_VDSP_END
index: 3?5h.12h, read/write
size: 8 bit
function: This register determines the number of scan lines displayed with frame buffer data. This register contains the lower 8 bits of the display end value. Bits 8 and 9 are stored in the Overflow register. If ‘n’ is the scan line count, then this register should be loaded with ‘n-1’.

Offset Register

name: CRT_OFFSET
index: 3?5h.13h, read/write
size: 8 bit
function: This register specifies the pitch in bytes between adjacent character row or scan lines. The next row start address (dword address) equals the current row start address plus (K * value in Offset register), where K has a value of 2 in byte mode, 4 in word mode, and 8 in dword mode. Byte or word addressing is set by the CRTC Mode Control register.

Underline Register

name: CRT_UNDERLINE
index: 3?5h.14h, read/write
size: 8 bit
function: This register includes the scan line location for underlining characters and the double word addressing control bits. The bit contents of the Underline Register are shown in Table 49.

Table 49. Underline Register

Bit	Function	Detail
7	Reserved	hardwired to 0
6	Dword Mode	0 = CRT addresses frame buffer in byte or word mode 1 = CRT addresses frame buffer in dword mode.
5	Count by Four	0 = CRT increments frame buffer pointer on every character clock unless Count-by-Two bit in the CRT Mode Control Register is set. 1 = CRT increments frame buffer pointer once every four character clocks
4:0	Underline Location	Scan line value within a character cell where the underline character is displayed. Lines 1 through 'n' are specified by values 0 through 'n-1'.

Start Vertical Blank Register

name: CRT_VBLANK_ST
index: 3?5h.15h, read/write
size: 8 bit
function: This register determines the start of vertical blanking. The scan line counter is compared to this register. When the values are equal, vertical blank begins. The lower 8 bits are loaded in this register. Bit 8 is stored in the Overflow Register and bit 9 is stored in the Character Height Register. If 'n' is the scan line count, then this register should be loaded with 'n-1'.

End Vertical Blank Register

name: CRT_VBLANK_END
index: 3?5h.16h, read/write
size: 8 bit
function: This register determines the end of vertical blanking. The scan line counter is compared to this register. When the values are equal, vertical blank ends. This register specifies only the least significant 8 bits of the 10-bit count at which the blanking period will end. The actual count at which it will end is the first time after the start vertical blank count that the least significant 8 bits of the line counter match these 8 bits. If 'n' is the scan line count, then this register should be loaded with 'n-1'.

Mode Control Register

name: CRT_MODE_CTL
index: 3?5h.17h, read/write
size: 8 bit
function: This register controls several aspects of the display setup. The bit contents of the Mode Control Register are shown in Table 50.

Table 50. Mode Control Register

Bit	Function	Detail
7	Sync enable	0 = HSYNC and VSYNC inactive 1 = HSYNC and VSYNC active
6	Word or Byte Mode	This bit is overridden by the Double-Word Mode bit in CRT Underline register. 0 = Word mode - Frame Buffer Address is incremented by two 1 = Byte mode - Frame Buffer Address is incremented by one
5	Address Wrap	0 = In word mode, memory address 14 appears at memory address 0 1 = Select memory address 16 for odd/even mode when 256KB of video memory is present
4	Reserved	
3	Count by Two	0 = Character clock increments memory address counter 1 = Character clock/2 increments memory address counter
2	Vertical Count Double	0 = Vertical line counter increments once per hsync 1 = Vertical line counter increments once every two hsync pulses, effectively doubling all vertical counts.
1	Select Row Scan Counter	0 = Row scan counter bit 1 is output at memory address 14 1 = Bit 14 of the CRTC address counter is output at memory address 14
0	Compatibility Mode	0 = Row scan counter bit 0 is output for memory address 13 1 = Memory address bit 13 unchanged

Line Compare Register

name: CRT_LINECMP
index: 3?5h.18h, read/write
size: 8 bit
function: This register is used for split screen operation. When the scan line counter equals the line compare value, the CRT's frame buffer pointer is cleared causing the lines beneath the compare scan line to be updated from video buffer display location 0. The lower 8 bits are loaded in the Line Compare register. Bit 8 is in the Overflow register and bit 9 is located in the Character Height register. If 'n' is the scan line count, then this register should be loaded with 'n-1'.

VGA Attribute Controller (I/O Mapped)

The Attribute Controller uses indexed addressing to access twenty control registers. Sixteen of these registers are Palette registers. The attribute controller uses a single I/O port at 3C0h to write both the index register and the data registers. Reading the 3C0h address will always return the value of the index register. A second port is available at 3C1h to read the value of the indexed data register.

The VGA contains internal state which keeps track of whether the index or data registers will be accessed on the next write. Each write access toggles the state between index and data registers. The Attribute Controller register write mechanism can be reset to point to a known state. This is done by reading the Input Status #1 register (3xAh). This resets the 3C0h write port to point to the index register.

Attribute Index and Data Port

name: ATT_ADDR
port: 3C0h, read/write
size: 8 bit
function: This register can always be read at 3C0h. However, when writing to 3C0h, the state of the port toggles between writing to the index register and the data registers. The data register written to is determined by the value last written to the index register.

If the state is unknown, issue a read of the Input Status #1 register (3xAh) to reset the port for writing the index register.

Bit 5 of this register must be a zero to allow the CPU to read or write the palette registers. Otherwise, writes are ineffectual and reads return 3Fh. Also, if bit 0 is not set to one upon completing the palette modifications, then the VGA refresh logic will not be able to access the palette and the screen will be all one color.

The bit contents of the Attribute Index and Data Port register are shown in Table 51.

Table 51. Attribute Index/Data Port Register

Bit	Function	Detail
7:6	Reserved	Hardwired to 00
5	Video On	0 = CPU drives the Palette address 1 = Refresh logic drives the Palette address
4:0	Attribute Index	0-Fh palette registers 10h mode control 11h overscan color 12h color plane enable 13h horizontal panning 14h color select

Read Data Port

name: ATT_RD
port: 3C1h, read only
size: 8 bit
function: Data registers are always read from the Read Data Port. Reading the data register does not affect the Index register. The bit contents of the Read Data Port register are shown in Table 51.

Table 52. Read Data Port Register

Bit	Function
7:5	reserved
4:0	Index to the data registers in the Attribute Controller block.

Palette Registers

name: ATT_PAL_REG[15:0]
index: 3C0h.0Fh-00h, read/write
size: 8 bits each
function: These 16 registers comprise the EGA palette. In all BIOS modes, frame buffer color indices are translated to an output color index through this palette. The bit contents of the Palette Register are shown in Table 53.

Table 53. Palette Registers

Bit	Function
7:6	Reserved - hardwired to 00
5	Secondary Red
4	Secondary Green/Intensity
3	Secondary Blue/Mono
2	Red
1	Green
0	Blue

Mode Control Register

name: ATT_MODE
index: 3C0h.10h, read/write
size: 8 bit
function: This register controls various attribute modes. The bit contents of the Mode Control Register are shown in Table 54.

Table 54. Mode Control Register

Bit	Function	Detail
7	Video Select	0 = Bits 4 and 5 of the Palette registers are used as address bits 1 = Color Select register <1:0> are used as address bits
6	PEL Width	This bit determines how pixels are assembled at the output of the EGA palette. When zero, the six-bit output of the EGA palette is sent on every pixel clock. When one, the least significant 4-bits of EGA palette output from two consecutive pixel clocks are assembled into one 8-bit quantity. This 8-bit value is then sent to the DAC for two consecutive clocks. 0 = Normal mode 1 = 256 color mode
5	Pixel Pan	0 = A line compare has no effect on pixel panning 1 = A line compare terminates pixel panning for remainder of frame.
4	Reserved	hardwired to 0
3	Blink/Intensity	0 = Blink inactive, set background intensity 1 = Blink active
2	Line Graphics Enable	0 = The 9th bit of a 9 pixel character will be the same as the background 1 = The 9th bit of a 9 pixel character will be the same as the 8th bit for character code between C0-DFh
1	Display Type	This bit only effects graphics mode blinking. 0 = Color display attributes 1 = Monochrome attributes
0	Mode Enable	0 = Text mode 1 = Graphics mode

Overscan Color Register

name: ATT_OVRS
index: 3C0h.11h, read/write
size: 8 bit
function: The overscan color register sets the color for the border area of the display. For monochrome displays this register is set to 0.

Color Plane Enable Register

name: ATT_CPE
index: 3C0h.12h, read/write
size: 8 bit
function: This register enables the four planes into the Palette Registers. This register also sets the inputs for the diagnostics bits in the Input Status #1 register. The bit contents of the Color Plane Register are shown in Table 55.

Table 55. Color Plane Enable Register

Bit	Function	Detail
7:6	Reserved	Hardwired to 00
5:4	Video Status Mux	00 = Input Status #1[5:4] = P2-Red, P0-Blue 01 = Input Status #1[5:4] = P5-Secondary Red, P4-Secondary Blue 10 = Input Status #1[5:4] = P3-Secondary Blue, P1-Green 11 = Input Status #1[5:4] = P7, P6 (256-color modes only) See Table 63 on page 87.
3	Enable Plane 3	0 = Disable plane, force pixel bit to 0 before addressing palette 1 = Enable color plane
2	Enable Plane 2	See bit 3
1	Enable Plane 1	See bit 3
0	Enable Plane 0	See bit 3

Horizontal Panning Register

name: ATT_HPAN
index: 3C0h.13h, read/write
size: 8 bit
function: This register is available for text or graphics modes and specifies the number of pixels the display data will be shifted left horizontally. In text mode, characters can be shifted left one pixel less than the cell width. In 256 color modes, up to 3 position pixel shifts may occur. The bit contents of the Horizontal Panning Register are shown in Table 56 and the allowable pixel pans are shown in Table 57.

Table 56. Horizontal Panning Register

Bit	Function
7:4	Reserved - hardwired to 0h
3:0	Pixel Pan Amount

Table 57. Allowable Pixel Pans

Pixel Pan Amount	9 Bit Character	8 Bit Character	256 Color Mode
0	1	0	0
1	2	1	--
2	3	2	1
3	4	3	--
4	5	4	2
5	6	5	--
6	7	6	3
7	8	7	--
8-Fh	0	--	--

Color Select Register

name: ATT_CLRSEL
index: 3C0h.14h, read/write
size: 8 bit
function: This register contains two fields used to select locations in the video DAC. The bit contents of the Color Select Register are shown in Table 58.

Table 58. Color Select Register

Bit	Function	Detail
7:4	Reserved	Hardwired to 0h
3:2	Color Bits[7:6]	These bits are concatenated with the lower 6 bits from the Palette Registers to form an index into the video DAC. These bits are ignored in 256 color modes.
1:0	Color Bits[5:4]	If Attribute Controller Mode Register <7> =1, these bits replace the corresponding bits in the Palette Registers to form an index into the video DAC. Otherwise, these bits are ignored. These bits are ignored in 256-color modes. ese bits are ineffective in 8, 16, and 24 bpp modes.

VGA General / External Registers (I/O Mapped)

The BtV2115 has seven general registers. The fields in this register block includes motherboard/adapter handshake during boot or switching of video display, clock frequency selection, sync signal polarity selection, and general-purpose status. All of these registers are directly accessible from the I/O bus.

POS102 Register

name: MCA_POS
port: 102h, read/write
size: 8 bit
function: The POS102 register contains a bit which wakes up the VGA in conjunction with the Adapter Sleep Register enable bit. When the VGA is in the wakeup state, the BtV2115 may respond to bus cycles to VGA address space. This register is accessible only if the BtV2115 is configured as an adapter and the setup bit is set in the Adapter Sleep register. The bit contents of the POS102 Register are shown in Table 59.

See the Configuration Registers section for information about how to configure the chip as an adapter versus a motherboard and how to configure the reset state of the sleep registers.

Table 59. POS102 Register

Bit	Function	Detail
7:1	Reserved	hardwired to 0
0	Wakeup VGA	0 = VGA is disabled 1 = VGA is enabled to respond to VGA bus cycles if the Enable bit is set in the Adapter Sleep Register

Adapter Sleep Register

name: ADAPT_SLEEP
port: 46E8h, read/write
size: 8 bit
function: The ADAPT_SLEEP register contains a bit which wakes up the VGA in conjunction with the POS102 Wakeup VGA bit. When the VGA is in the wakeup state, the BtV2115 may respond to bus cycles to VGA address space. This register is accessible only if the BtV2115 is configured as an adapter VGA. The setup bit is used to enable access to the POS102 register. The bit contents of the Adapter Sleep Register are shown in Table 60.

The protocol for enabling an adapter VGA is the following:

1. Enable POS102 Setup. Write 18h to Adapter Sleep.
2. Set Wakeup VGA bit in POS102 register. Write 01h to POS102.
3. Turn off POS102 Setup. Write 08h to Adapter Sleep.

Refer to GRP_CFG3[5] in Table 23 on page 52 for information about how to configure the chip as an adapter versus a motherboard and how to configure the reset state of the sleep registers.

Table 60. Adapter Sleep Register

Bit	Function	Detail
7:5	Reserved	Hardwired to b'000
4	Setup	0 = POS102 write access is disabled 1 = POS102 write access is enabled, VGA access is disabled.
3	Enable Subsystem	0 = BtV2115 is disabled 1 = BtV2115 is enabled to accept VGA bus cycles if the POS102 register Wakeup VGA bit is set.
2:0	Reserved	Hardwired to b'000

Motherboard Sleep Register

name: MBD_SLEEP
port: 3C3h, read/write
size: 8 bit
function: This register can be accessed only if the BtV2115 is configured as a motherboard VGA. The bit contents of the Motherboard Sleep Register are shown in Table 61.

Refer to GRP_CFG3[5] in Table 23 on page 52 for information about how to configure the chip as an adapter versus a motherboard and how to configure the reset state of the sleep registers.

Table 61. Motherboard Sleep Register

Bit	Function	Detail
7:1	Reserved	hardwired to 0
0	Enable Subsystem	0 = BtV2115 is disabled from accepting VGA bus cycles 1 = BtV2115 is enabled to accept VGA bus cycles

Input Status #0 Register

name: INP_STAT0
port: 3C2h, read only
size: 8 bit
function: The Input Status #0 register contains the vertical retrace interrupt bit. The bit contents of the Input Status #0 Register are shown in Table 62.

Table 62. Input Status #0 Register

Bit	Function	Detail
7	Vertical Retrace Interrupt	0 = interrupt cleared 1 = interrupt pending
6:5	Reserved	Hardwired to b'00
4	Monitor Detection	This field is undefined. Monitor status is available by accessing registers on the PACDAC device.
3:0	Reserved	Hardwired to 0h

Color Input Status #1 Register

name: INP_STAT_C
port: 3DAh color, read only
size: 8 bit
function: This register contains assorted status bits. Reading this register resets the state of the Attribute Controller index/write port for color systems to 3C0h. The bit contents of the Input Status #1 Register (for both color and monochrome) are shown in Table 63. This register is accessible only if MISC_OUT_R[0] = 1 (see Table 64 on page 88).

Monochrome Input Status #1 Register

name: INP_STAT_M
port: 3BAh, read only
size: 8 bit
function: This register contains assorted status bits. Reading this register resets the state of the Attribute Controller index/write port for monochrome systems to 3C0h. The bit contents of the Input Status #1 Register (for both color and monochrome) are shown in Table 63. This register is accessible only if MISC_OUT_R[0] = 0 (see Table 64 on page 88).

Table 63. Input Status #1 Register

Bit	Function	Detail
7:6	Reserved	Hardwired to b'00
5:4	Video status mux	Multiplexed Video Bits as selected by bits 5:4 of the Attribute Controller Color Plane Enable register. 00 = P2, P0 01 = P5, P4 10 = P3, P1 11 = P7, P6 See Table 55 on page 82
3	Vertical Retrace Status	0 = Vertical Retrace inactive 1 = Vertical Retrace active
2:1	Reserved	Hardwired to b'10
0	Display Enable Status	0 = Display active 1 = Display inactive-horizontal or vertical retrace active

Write Miscellaneous Output Register

name: MISC_OUT_W
port: 3C2h, write only
size: 8 bit
function: This register contains write only fields related to setting up the VGA environment. The bit contents of the Miscellaneous Output Registers (for both write and read) are shown in Table 64. What is written here is read at 3CCh.

Read Miscellaneous Output Register

name: MISC_OUT_R
port: 3CCh, read only
size: 8 bit
function: This register contains several read only fields related to setting up the VGA environment. The bit contents of the Miscellaneous Output Registers (for both write and read) are shown in Table 64.

Table 64. Miscellaneous Output Register (Read and Write)

Bit	Function	Detail
7	VSYNC Polarity [a]	0 = Active high 1 = Active low
6	HSYNC Polarity[a]	0 = Active high 1 = Active low
5	Page Select	This bit determines affects the value used as the least significant bit in Odd/Even addressing modes. It is for diagnostic use, and should be set to one for normal use.
4	Reserved	Hardwired to 0
3:2	Clock Select [1:0]	00 = Select 25.180 MHz Pixel Clock 01 = Select 28.325 MHz Pixel Clock 10 = Pixel clock rate determined by PLL value written to media buffer 11 = reserved
1	Enable Frame Buffer Access	0 = Disable CPU access to VGA frame buffer 1 = Enable CPU access to VGA frame buffer
0	CRTC I/O Address	0 = Select monochrome - 3Bh 1 = Select color - 3Dh

a. Monitors select their vertical scan rate and gain based on HSYNC and VSYNC polarity, which is set by Bits <7:6> . The selection is as follows:
0 = Reserved
1 = 400 line mode
2 = 350 line mode
3 = 480 line mode

Feature Control Register

name: FC
port: 3CAh read
3xAh write: 3BAh if MISC_OUT_R[0] = 0 (mono), or 3DAh if MISC_OUT_R[0] = 1 (color)
size: 8 bit
function: The feature control functionality does not add any capability to BtV2115 system and is not supported.

VGA Color Registers (I/O Mapped)

The BtV2115 supports the 5 VGA DAC registers (color registers) in addition to the DAC Alias Registers in the Graphics Controller extended address space. Following is a description of the VGA DAC registers.

DAC Mask Register

name: DAC_MASK
port: 3C6, read/write
size: 8 bit
function: This 8-bit register contains a bit mask of the pixel value going to the DAC lookup table (VGA palette, or Color Registers). A '1' enables the corresponding bit, while a '0' forces the bit to zero before it addresses the color lookup table. This register does not affect access to the color palette through the 3Cxh CPU ports. It only affects the pixel value coming from the Attribute logic for display on the screen. This register is normally not written by application code but initialized by BIOS or device drivers to establish the color lookup table.

DAC State Register

name: DAC_STATE
port: 3C7h, read only
size: 8 bit
function: This register shows whether the DAC Data read/write port is in a read or write mode, depending on whether the DAC Read Address register or DAC Write Address register was written to most recently. The bit contents of this register are shown in Table 65.

Table 65. Color Registers: DAC State Register

Bit	Function	Detail
7:2	Reserved	Hardwired to b'000000
1:0	State	00 = Write mode 11 = Read mode

DAC Read Address Register

name: DAC_RDADDR
port: 3C7h, write only
size: 8 bit
function: This register contains an 8 bit address used to select one of 256 video DAC color registers during a read operation. The next three reads of the DAC Data register will return three 6-bit values for RED, GREEN, and BLUE settings, respectively. After the BLUE value is read, the val-

ue in the DAC Read Address register is incremented to point to the next DAC color register. This feature allows the programmer to read all 768 values from the DAC without updating the index value in the DAC Write Address register. This value will also wrap from 255 to 0. The DAC Read Address can be written with another value at any time between each set of three reads to the DAC Data register.

DAC Write Address Register

name: DAC_WRADDR
port: 3C8h, read/write
size: 8 bit
function: This register contains an 8 bit address used to select one of 256 video DAC color registers during a write operation. The next three writes to the DAC Data register will output three 6-bit values for RED, GREEN, and BLUE settings, respectively. After the BLUE value is output, the value in the DAC Write Address register is incremented to point to the next DAC color register. This feature allows the programmer to output all 768 values to the DAC without updating the index value in the DAC Write Address register. This value will also wrap from 255 to 0. The DAC Write Address register can be written with another value at any time between each set of three writes to the DAC Data register

DAC Data Register

name: DAC_DATA
port: 3C9h, read/write
size: 8 bit
function: This register is used to read or write 18 bits of palette data depending on the mode of operation. Writes to the DAC Read Address register put this register in the read mode, while writes to the DAC Write Address register put this register in the write mode.

Data access to and from this register is in groups of three bytes. Each byte contains the 6-bit palette value for the DAC output signal. The first byte contains the red signal value, the second byte contains the green signal value, and the third byte contains the blue signal value. On reads, the most significant two bits of each byte are always set to ‘0’.

Legacy Audio FM Registers (I/O Mapped)

The BtV2115 provides I/O mapping for the Legacy Audio FM register space. Provided specifically to support the Yamaha OPL3 FM Synthesizer, this I/O interface indirectly addresses the OPL3's internal registers. For specific register descriptions, refer to the *Yamaha YMF-262 Product Information* bulletin.

NOTE

Legacy Audio supports the Yamaha 2 and 4 operator mode FM synthesizer chip family.

The OPL3 I/O address space is a superset of the OPL2's address space; thus OPL2 (YM3812) interface support is implied.

The I/O decoder is enabled by setting bit 7 of the Legacy Audio Emulation register. Select either I/O base address 022Xh or 024Xh by setting bit 6 of the Legacy Audio Emulation register (default is 022Xh).

Additionally, a memory mapped interface supports OPL3 and OPL4 (YMF278-F) device communications. Refer to "Yamaha Support" on page 120.

This section provides register addressing information for the OPL3 device. Below, Table 66 summarizes the Legacy Audio FM register addresses.

Table 66. Legacy Audio FM Register Addresses

Legacy Audio Emulation Register Bit 6 = 0		Legacy Audio Emulation Register Bit 6 = 1	
I/O Addr	**Register Type**	**I/O Addr**	**Register Type**
388	Bank0 Address	388	Bank0 Address
389	Bank0 Data	389	Bank0 Data
38A	Bank1 Address	38A	Bank1 Address
38B	Bank1 Data	38B	Bank1 Data
220	Bank0 Address	240	Bank0 Address
221	Bank0 Data	241	Bank0 Data
222	Bank1 Address	242	Bank1 Address
223	Bank1 Data	243	Bank1 Data
228	Bank0 Address	248	Bank0 Address
229	Bank0 Data	249	Bank0 Data

Notes:
1. The OPL3 uses both Bank 0 and Bank 1, OPL2 uses only Bank 0

OPL Bank 0 Address

name: ADLIB_BNK0_ADDR
port: 0388h, read/write

name: OPL3_BNK0_ADDR2
port: 0220h, read/write

name: OPL2_BNK_ADDR2
port: 0228h, read/write

name: OPL3_BNK0_ADDR4
port: 0240h, read/write

name: OPL2_BNK_ADDR4
port: 0248h, read/write
size: 8 bit
function: This register is the Yamaha OPL3 bank 0 address port. The value written into this register provides an offset into the OPL3 register space. Additionally, this register provides OPL3 status information when read. This register does not auto-increment.

OPL Bank 0 Data

name: ADLIB_BNK0_DATA
port: 0389h, write only

name: OPL3_BNK0_DATA2
port: 0221h, write only

name: OPL2_BNK_DATA2
port: 0229h, write only

name: OPL3_BNK0_DATA4
port: 0241h, write only

name: OPL2_BNK_DATA4
port: 0249h, write only
size: 8 bit
function: This register is the Yamaha OPL3 bank 0 data port. The value written into this register is placed into an OPL3 internal register whose offset is determined by the current bank 0 address port value. Read-back is not available.

OPL Bank 1 Address

name: ADLIB_BNK1_ADDR
port: 038Ah, read/write

name: OPL3_BNK1_ADDR2
port: 0222h, read/write

name: OPL3_BNK1_ADDR4
port: 0242h, read/write
size: 8 bit
function: This register is the Yamaha OPL3 bank 1address port. The value written into this register provides an offset into the OPL3 register space. Additionally, this register provides OPL3 status information when read. This register does not auto-increment.

OPL Bank 1 Data

name: ADLIB_BNK1_DATA
port: 038Bh, write only

name: OPL3_BNK1_DATA2
port: 0223h, write only

name: OPL3_BNK1_DATA4
port: 0243h, write only
size: 8 bit
function: This register is the Yamaha OPL3 bank 0 data port. The value written into this register is placed into an OPL3 internal register whose offset is determined by the current bank 0 address port value. Read-back is not available.

Legacy Audio Mixer Registers (I/O Mapped)

The BtV2115 provides I/O mapping for the Legacy Audio Mixer registers. The mixer is controlled by a series of indirectly addressed 8-bit registers. These registers are accessed by the following Mixer Address/ Mixer Data ports. The I/O decoder is enabled by setting bit 7 of the Legacy Audio Emulation register. Select either I/O base address 022Xh or 024Xh by setting bit 6 of the Legacy Audio Emulation register.

Mixer Address

name: LA_MIX_ADDR2
port: 0224h, write only

name: LA_MIX_ADDR4
port: 0244h, write only
size: 8 bit
function: This write only register provides an offset into the internal register mapping of the Legacy Audio Mixer registers.

Mixer Data

name: LA_MIX_DATA2
port: 0225h, read/write

name: LA_MIX_DATA4
port: 0245h, read/write
size: 8 bit
function: The value written into this register is placed into a Legacy Audio Mixer internal register whose offset is determined by the current Mixer Address port value.

Legacy Audio DSP Registers (I/O Mapped)

The BtV2115 provides I/O mapping for the Legacy Audio DSP registers used for digitized voice audio playback and record. These registers are comprised of a reset register, read data port, command data port/command status port, and a read data status port. The I/O decoder is enabled by setting bit 7 of the Legacy Audio Emulation register. Select either I/O base address 022Xh or 024Xh by setting bit 6 of the Legacy Audio Emulation register.

DSP Reset

name: LA_DSP_RESET2
port: 0226h, write only

name: LA_DSP_RESET4
port: 0246h, write only
size: 8 bit
function: This write only register is provided for resetting the Legacy Audio subsystem.

DSP Read Data

name: LA_DSP_READ2
port: 022Ah, read only

name: LA_DSP_READ4
port: 024Ah, read only
size: 8 bit
function: This read only register provides either command response data or recorded audio data. The DSP Read Status register is polled for valid data ready.

DSP Command-Data / Command Status

name: LA_DSP_CMD2
port: 022Ch, read/write

name: LA_DSP_CMD4
port: 024Ch, read/write
size: 8 bit
function: This register provides digitized voice control commands or digitized voice data when written. The read-back function of this register is polled for the command interface ready to write condition.

DSP Read Status

name: LA_READY2
port: 022Eh, read only

name: LA_READY4
port: 024Eh, read only
size: 8 bit
function: This read only register provides the status of the DSP Read Data port.

Audio Registers (Memory Mapped)

The audio registers are defined starting on page 276 of “Audio Subsystem” in the “Audio Interface” chapter.

GUI Aperture Registers (Memory Mapped)

The GUI aperture space registers are defined as follows:

GUIREG_FIFO register: “GUI FIFO Register” on page 112.
GUIREG_DEPTH register: “GUI FIFO Depth Register” on page 114.
GUIREG_MBA register: “MBA Control Register” on page 125.

GUI Accelerator Registers (Memory Mapped)

The GUI accelerator registers are defined starting in“GUI Registers” on page 172.

Video Input Registers (Memory Mapped)

The video input registers are defined starting on in “Buffer Management Hardware” on page 256 in the “Video Input Subsystem” chapter.

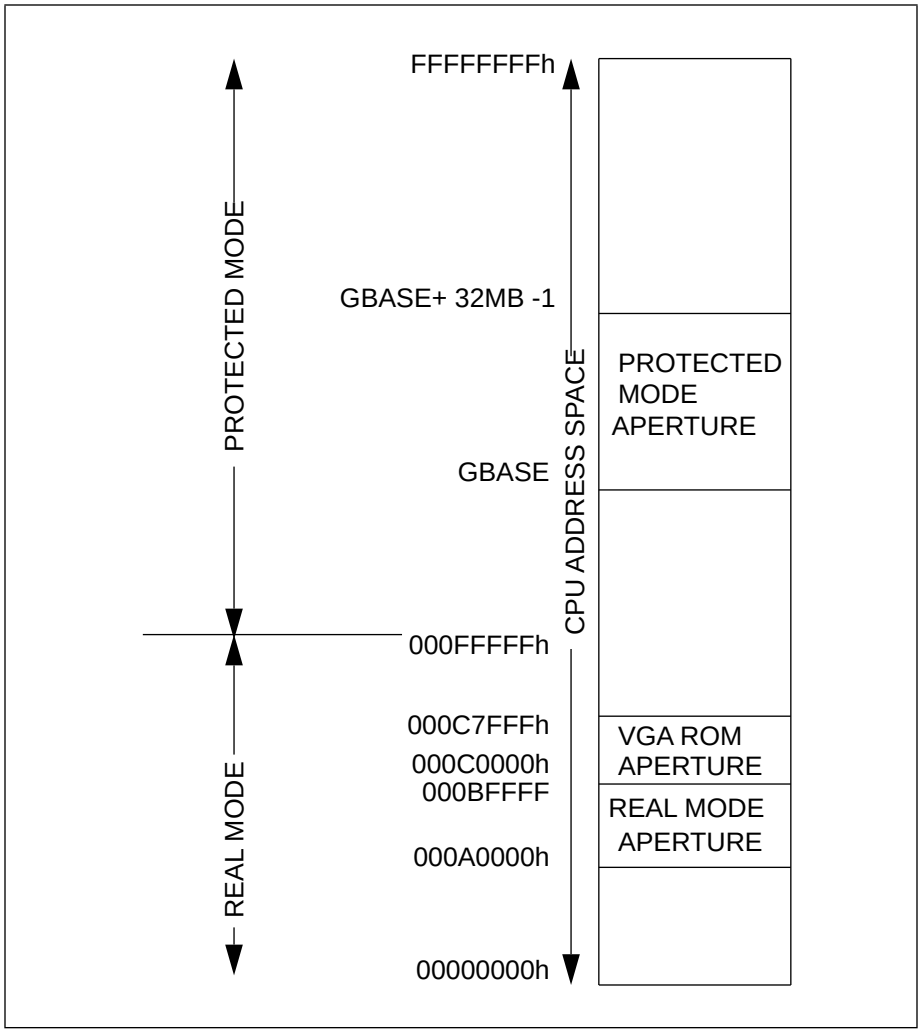
CPU ADDRESS SPACE APERTURES

Introduction

This section describes the programming interface to the BtV2115 media accelerator, the Media Buffer Access (MBA) unit, the audio subsystem, the Flash ROM and the Yamaha OPL FM synthesizer. As in conventional VGA systems, there is a CPU bus aperture at 000A0000h through 000BFFFFh through which the VGA frame buffer is accessed, which we shall call the real mode aperture. In addition, in VL-bus configurations, there is an optional aperture at 000C0000h which can be used for VGA ROM access, known as the VGAROM aperture. In PCI configurations, the VGAROM aperture becomes the PCI Expansion ROM Aperture and can be moved to any 16MB boundary. The design assumes that the BtV2115 is attached to the CPU via either a VL-bus or PCI-bus. Furthermore it is assumed that the GUI and MBA are accessed from protected mode well above the base memory of the system through the protected mode aperture.

The protected mode aperture occupies 32MB of VL or PCI bus address space beginning at the address specified in the GUI base address register, as shown in Figure 4.

Figure 4. CPU Apertures on a VL System



Bits 31:25 of the hardware register GUI_BASE, determine where the 32MB protected mode aperture resides in CPU address space. These seven bits, known as the protected mode base field (PM_BASE), are matched against bits 31:25 of the VL/PCI CPU address to determine when the protected mode aperture is accessed. The protected mode aperture must be offset from zero to prevent the protected mode aperture from overwriting any part of the 1MB of main memory reserved for the operating system from 00000000h to 000FFFFFh. To do this, at least one bit in the PM_BASE must be set to ”1” in order to specify a valid protected mode aperture address. In addition, GUI_BASE[24] (PM_ENABLE field) is an overall enable for the protected mode aperture and must be set to 1 to enable the protected mode aperture. Consequently, GUI_BASE[31:24] must be between 03h and FFh and must be odd so that the PM_BASE will contain 7’b0000001 through 7’b1111111 thus allowing the protected mode aperture to be placed at any one of 127 possible 32MB blocks of CPU address space.

Assume for the purposes of this discussion that the protected mode aperture starts at AC000000h so that the first byte in the aperture is accessed at AC000000h. Thus the value to load into GUI_BASE[31:24] is ADh. Therefore the placement of the aperture is constrained on 32MB boundaries. Notice that the GUI base address is specified in a VGA style register. Since two VGAs can be supported in the system one can have two GUI accelerators residing at two different base addresses.

In essence, the 32MB aperture can be placed at address space A where 32MB≤A≤4GB-32MB such that A is evenly divisible by 32M. In this manual, the offset to the protected mode is indicated by ORing the specified addresses with an address constant, GBASE, which is defined as follows: GBASE[31:25] are set exactly equal to the PM_BASE in GUI_BASE[31:25] while GBASE[24:0] are set to zero. Thus if GUI_BASE[31:24] are set to 8'hAD then GBASE would have the value 32'hAC000000. To form a CPU bus address, the programmer must ensure that segments and base registers of the CPU are set so that bits 31:25 of the CPU bus will match GBASE[31:25] for accesses destined for the protected mode aperture. In C this would be coded:

```
#define GBASE 0xAC000000L
```

because the hardware will match CPU address bits [31:25] against the PM_BASE to detect a protected mode aperture access.

One must distinguish GBASE which is a programming constant or variable from the PM_BASE, which is a 7 bit field in the 32-bit hardware GUI_BASE register. These two values must be distinguished from GRP_GUI_BASE[3:0], which are the four Extended VGA Graphics Register indices placed in GRP_INDEX to read or write bytes within the GUI_BASE register.

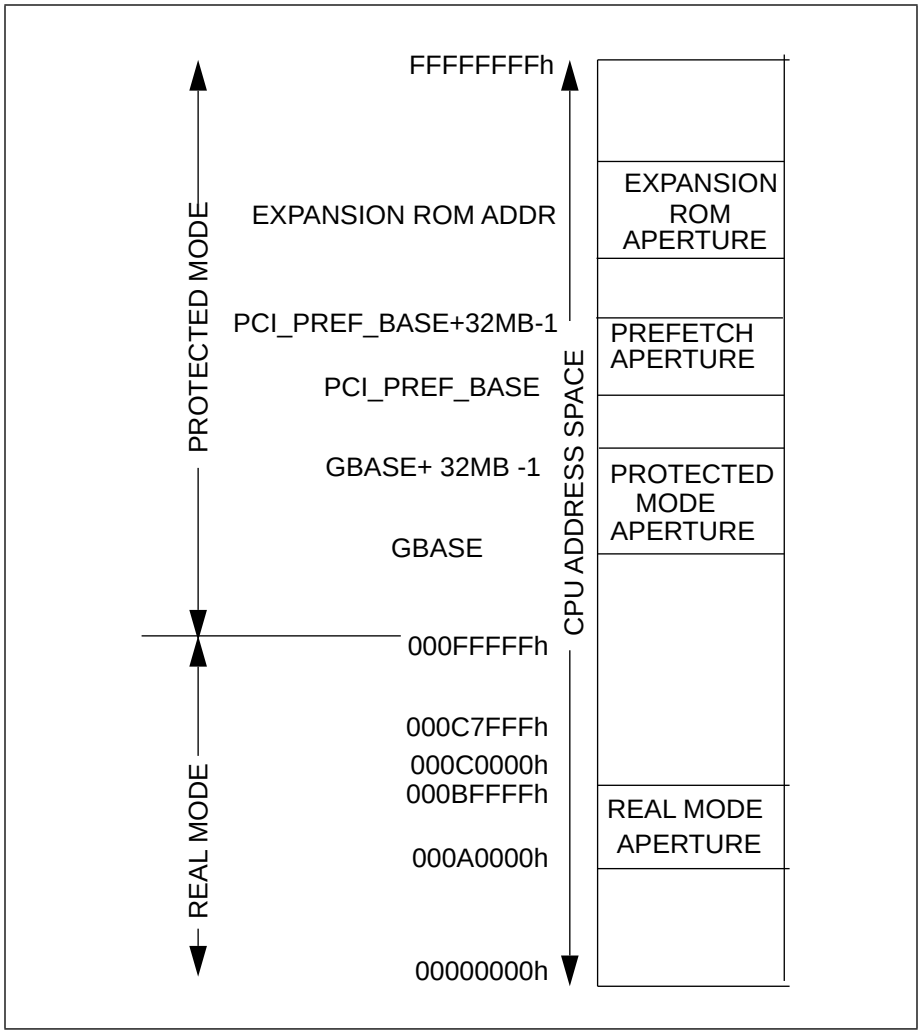
REMINDER

The identifiers GUI_BASE, GRP_GUI_BASE, and GBASE refer to different entities (as stated above).

Notice that some differences exist between the apertures on a VL system versus a PCI system. As shown in Figure 5, for PCI-bus the ROM aperture is relocated to high memory (protected mode). It is the system BIOS' responsibility to copy the VGA BIOS from this high memory location to system RAM and map it to 000C0000h to 000C7FFFh. PCI systems should never enable the ROM HOLE.

In PCI configurations, the GUI_BASE register is actually loaded via PCI configuration space writes to the Base Address Register 0 of configuration space zero. See "CPU Apertures In a PCI-Bus Environment" on page 130. Notice that PCI BIOS places some restrictions on GBASE. See "PCI Constraints on GBASE" on page 130.

Figure 5. CPU Apertures on a PCI System



Protected Mode Aperture

CPU accesses are applied to the host interface module whether via VESA VL bus or PCI bus. The host interface gives rise to an internal 3-state bus called the HBUS. A number of chip subsystems or functional "agents" are attached to the HBUS, such as the VGA controller, the audio subsystem, the GUI accelerator, etc., as shown in Figure 6. The protected mode aperture provides access to any of the functional agents *except* the VGA. For this reason, devices on the VGA extension bus are not accessible through the protected mode aperture.

Figure 6. HBUS Agents

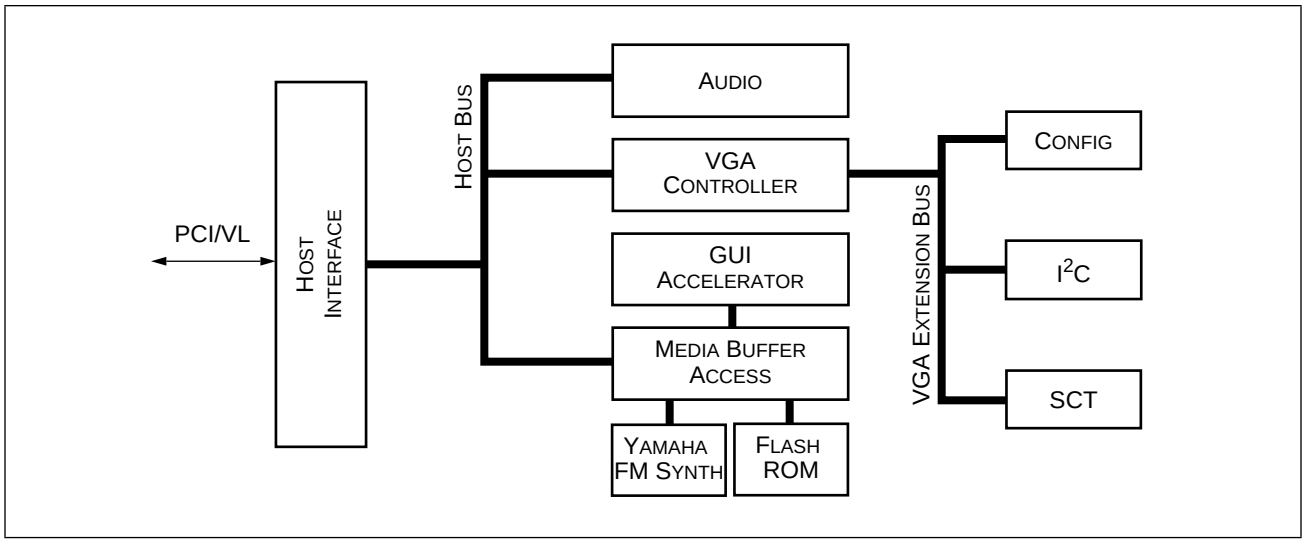
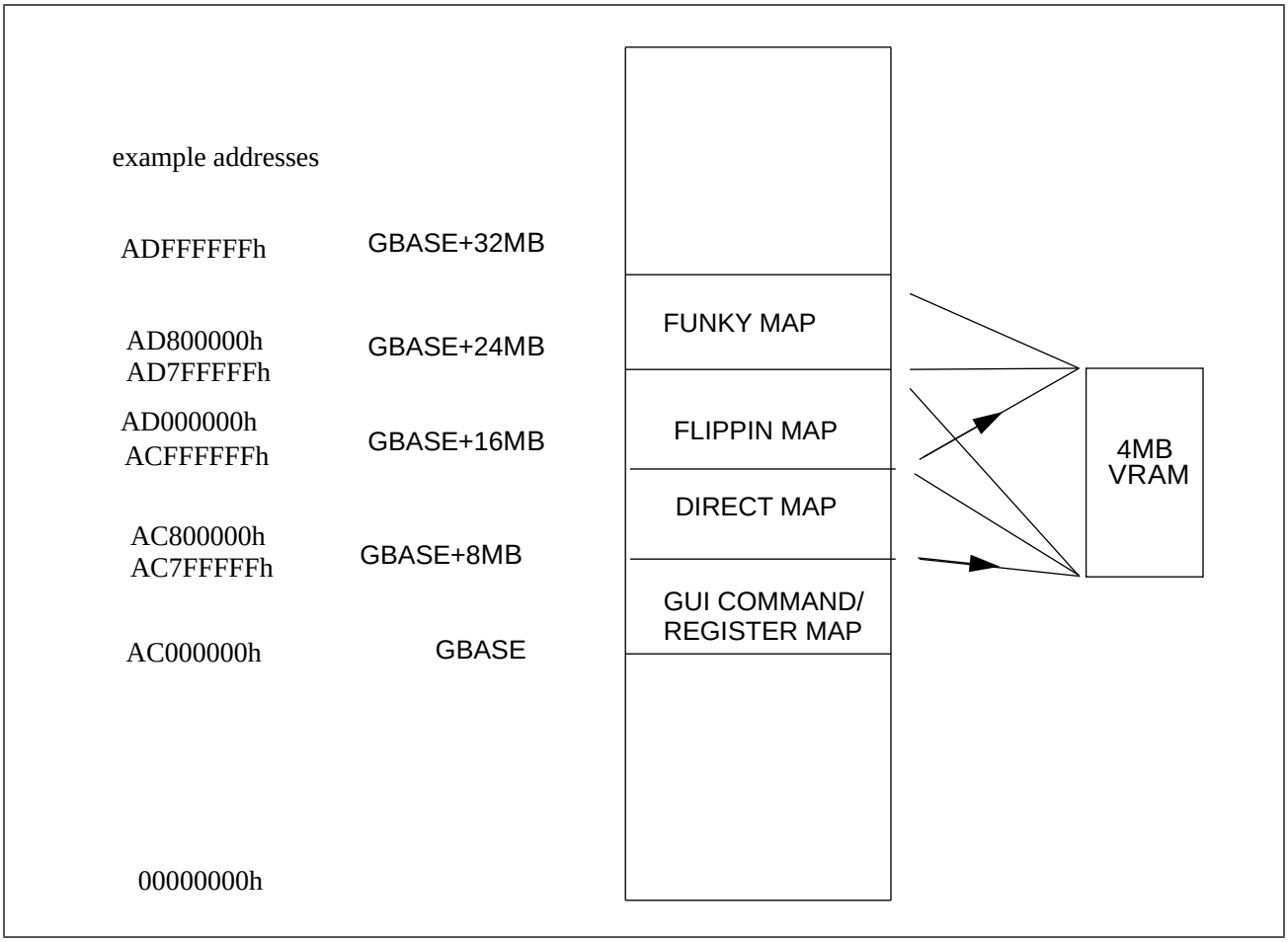


Figure 7 shows how the 32MB protected mode aperture is broken into four 8MB maps. Through the lowest addressed map passes all commands and data destined for the GUI accelerator. The other three maps provide different views of the 4MB Media Buffer. BtV2115 supports up to 4MB of VRAM Media Buffer memory. Each map could have been specified at 4MB, however to do 1280x1024x32 requires 8MB of VRAM, therefore BtV2115 has been architected for an upgrade path to 8MB Media Buffers.

Looking at the example addresses in Figure 7, the map from AC800000h to ACFFFFFFh is a direct mapping of the Media Buffer. The map from AD000000h to AD7FFFFFh, called the Flippin Map, also completely maps the Media Buffer; however, bit, byte, and word data swapping is allowed in this region. The map from AD800000h to ADFFFFFFh is known as the Funky Map. The Funky Map can provide many different views of the Media Buffer.

Figure 7. Protected Mode Aperture Mapping To VRAM



Three Media Buffer maps are a distinct advantage over other GUI accelerators, particularly when multimedia audio and video are supported in the Media Buffer. Having the ability to support data bus steering for word, byte, and bit flipping is a distinct advantage for supporting arbitrary image formats. Such steering is controlled from deep within the windows device driver. Such modes make supporting both *little endian* and *big endian* systems practical from a single controller design.

Direct Map

The Direct Map covers all of the Media Buffer memory and gives a clean, unmodified view of all of the VRAM Media Buffer. Addresses beyond 4MB in the 8MB map wrap back onto the actual 4MB (or less) VRAM. Figure 7 depicts the Direct Map covering the VRAM. Bytes, bits, and words are in their natural 486 little endian order with the least significant byte in the 486 EAX register mapping to the left most pixel position when a dword move instruction is used. No word, byte, or bit steering is performed on this Direct Map so regardless of settings in the windows driver for the Flippin Map devices and modules that wish to see an unmodifiable little endian view of the VRAM can *always* do so through the Direct Map.

Figure 8 shows that the CPU address bits 31:25 must match GUI_BASE bits 31:25 (PM_BASE) for a protected mode aperture access to be accepted. Once accepted, CPU address bits 24:2 are passed to the internal HBUS, shown in Figure 6. In the Media Buffer access module HBUS address bits 24:20 are decoded to determine the type of access to perform. When the Direct Map is selected, CPU address bits 21:2 are passed directly to the internal MBUS for use as a VRAM address. Thus, CPU address bits 24:23 must equal 2'b01 for the Direct Map to be selected. Since only a maximum 4MB VRAM is supported in the BtV2115 notice that CPU address bit 22 is a don't care.

Finally notice that CPU address bits 21:2, in conjunction with the CPU byte enables determine which byte(s) of the VRAM Media Buffer are accessed through the Direct Map. Because the entire VRAM memory is accessible through the protected mode aperture without having to resort to any page register mechanism, it is referred to as having a "flat view" of the Media Buffer.

If we define a C constant to reference the Direct Map it could be coded:

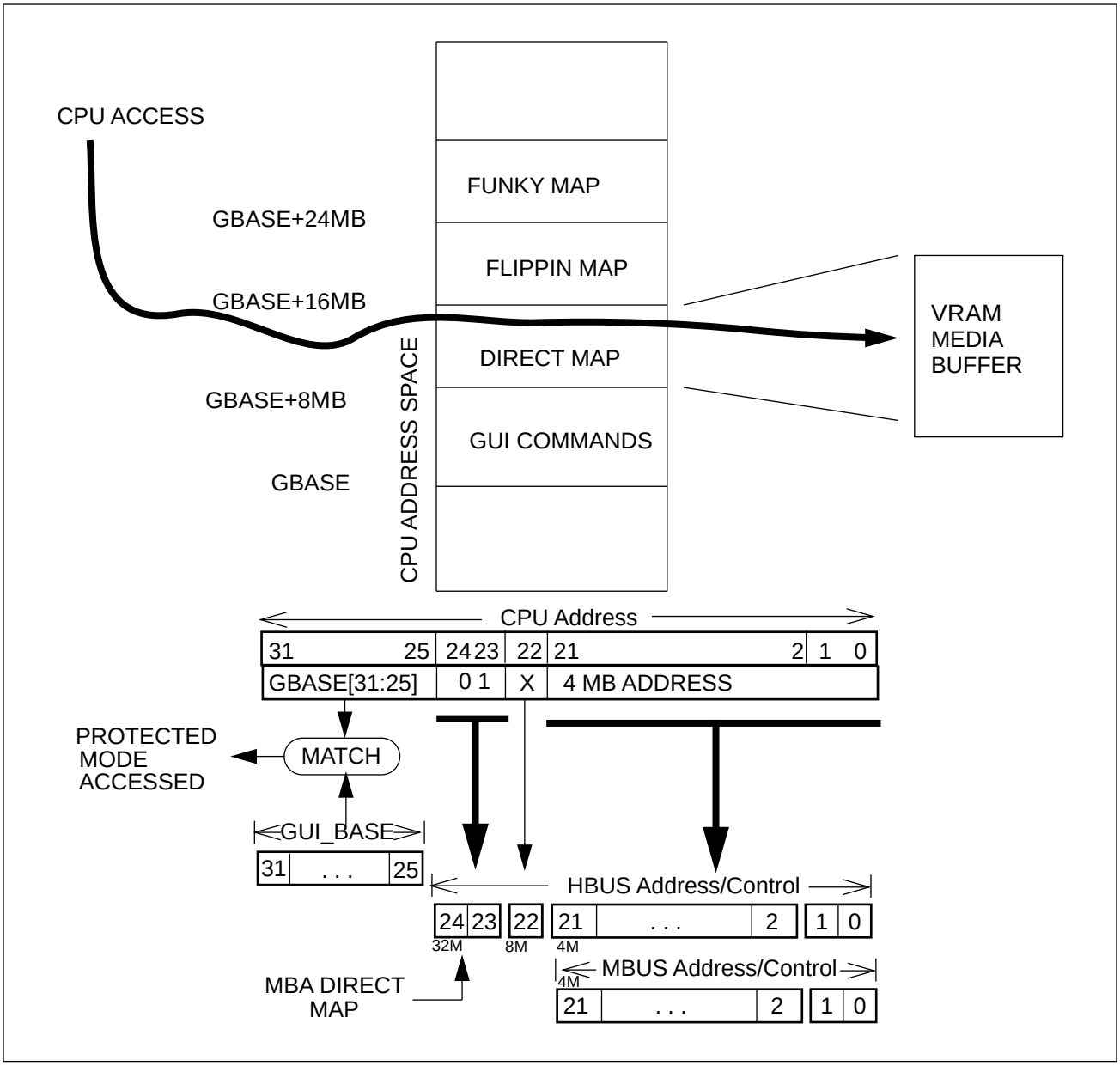
```
#define MBA_DIRECT 0x00800000L
```

To access byte 100 hex within the Direct Map we could form an address such as:

```
((char *)(GBASE | MBA_DIRECT | 0x00000100L)).
```

Having a separate map that is unchangeable is useful for multimedia accelerators that reside on the PCI; such audio DSPs need "unmutilated" access to the audio in and out FIFOs in the Media Buffer. Similarly a PCI-resident hardware video codec can get "unmutilated" access to the video in and video out buffers in the Media Buffer. Notice that address bit 22 is unconnected in BtV2115 and is reserved for future expansion.

Figure 8. Media Buffer Access Via Direct Map



Software video codecs reside in different driver modules from the windows driver code. By offering both a Funky Map and a Flippin Map then these two code modules can be written to be totally oblivious to the internal drawing state of each other.

Flippin Map

The Flippin Map covers all of the Media Buffer memory and gives a flat view of all of the VRAM Media Buffer. However, unlike the Direct Map, the windows driver can control the data path steering in BtV2115 so that bits can be swapped within a byte, bytes can be swapped within a word, or words can be swapped within a dword. Addresses beyond 4MB in the 8MB map wrap back onto the actual 4MB (or less) VRAM.

Figure 9 depicts the Flippin Map covering the VRAM. In a fashion similar to the Direct Map, notice that the CPU address bits 31:25 must match the GUI_BASE register bits 31:25 (PM_BASE) for a protected mode aperture access to be detected. Similarly, CPU address bit 22 is a don't care. Notice that CPU address bits 24:23 must equal 2'b10 for the Flippin Map to be selected.

CPU address bits 21:2 are passed to the MBus and ultimately to the memory controller in an unmodified way. The Flippin Map word and byte flip bits in GUIREG_MBA can cause the data path byte lanes and the byte enables to be reordered to effect a byte or word swap within a dword.

Finally, the GUIREG_MBA Flippin Map bit flip bit can cause the data path to bit reverse the word being written or read from the VRAM. If we define a C constant to reference the Flippin Map it could be coded:

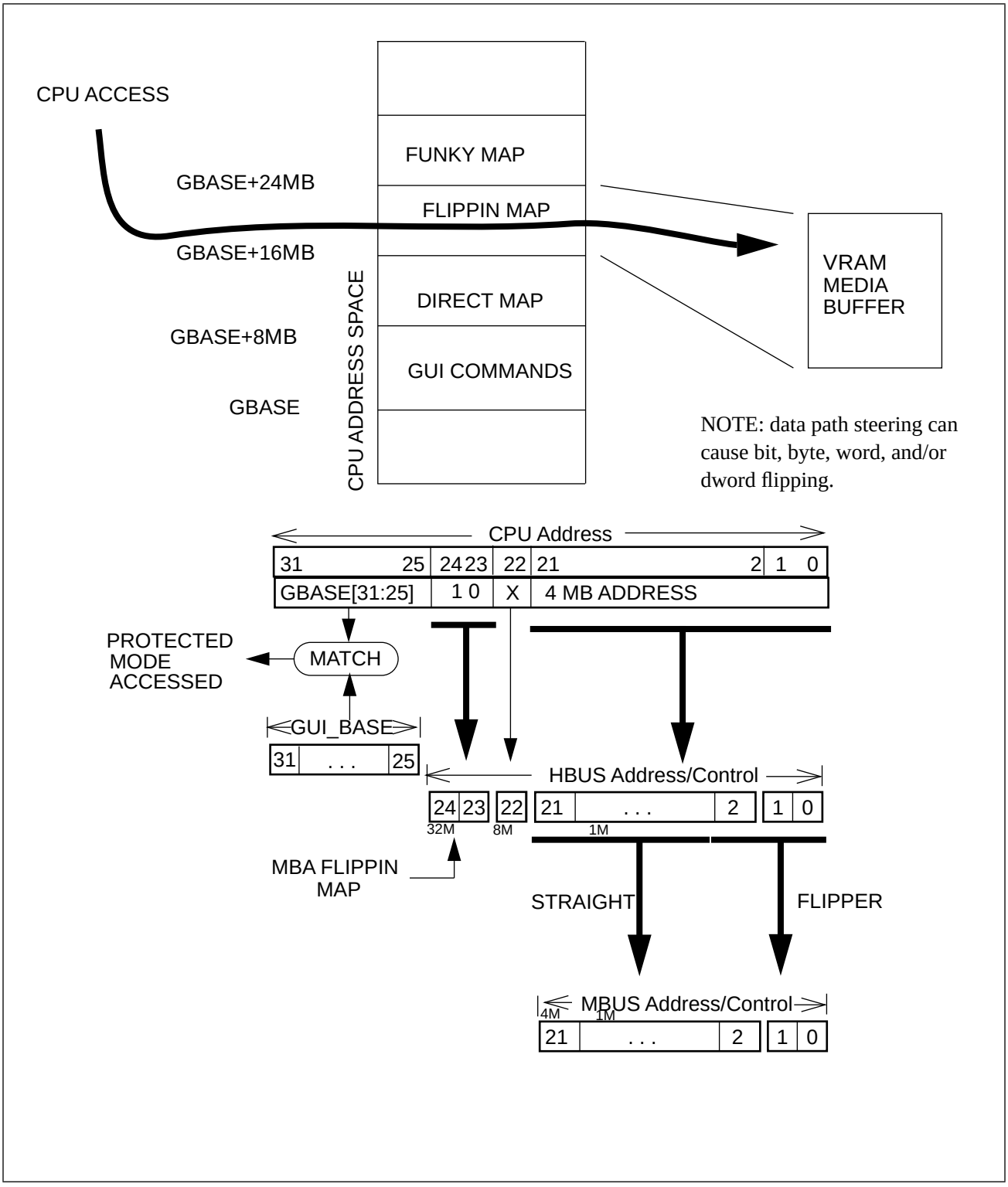
```
#define MBA_FLIPPIN 0x01000000L
```

To access byte 200 hex within the Flippin Map we could form an address such as:

```
((char *)(GBASE | MBA_FLIPPIN | 0x00000200L))
```

The same byte of VRAM can be accessed through either the Direct Map or the Flippin Map; however, the dwords viewed through the Flippin Map may have different orderings of bits, bytes, words, within a dword than the same dword viewed through the Direct Map. Notice that address bit 22 is unconnected in BtV2115 and is reserved for future expansion.

Figure 9. Protected Mode Aperture Accessing VRAM Via Flippin Map



Funky Map

The Funky Map covers all of the Media Buffer memory and gives an alternate view of all of the VRAM Media Buffer. If we define a C constant to reference the Funky Map it could be coded:

```
#define MBA_FUNKY 0x01800000L
```

To access byte 100 hex within the Funky Map we could form an address such as:

```
((char *)(GBASE | MBA_FUNKY | 0x00000100L))
```

The Funky Map provides some hardware assistance for soft video encoding. The video output driver supplies Y and UV components to the Funky Map in this mode and the Funky Map modulates them with a quadrature sin/cos table to produce a portion of a composite NTSC or PAL PCM (Pulse Code Modulated) waveform.

The Funky Map functionality is controlled by registers within the non-queued GUI address space, see section“GUI Command/Register Map” on page 112 for further information. The programming interface includes several registers accessible from the non-queued GUI Command/Register Map. These registers include “GUI FIFO Register” on page 112, “GUI FIFO Depth Register” on page 114 and the “MBA Control Register” on page 125.

Funky Map Support for SEV

To enable the Funky Map support of software encoded video (SEV), GUIREG_MBA[23] (MODUL_EN) must be set to “1.” The Funky Map implements the NTSC and PAL quadrature modulation function to speed up the process of encoding a SEV signal. In essence, the Funky Map implements the expressions: composite = Y + Usin(i) + Vcos(i) and composite = Y -Usin(i) - Vcos(i). The hardware stores these results as 16 composite samples for each UV sample and four composite samples for each Y value presented. The Y values are interpolated. Hardware maintains an i counter for addressing the UsinVcos look-up table. All modulation CPU bus cycles are write cycles for maximum through-put. CPU read cycles are treated as component access reads regardless of the setting of MODUL_EN.

In the SEV mode, software writes to the Funky Map are as shown in Figure 10. For each 32 bit composite plus or composite minus write cycle the CPU generates, a total of four dwords in VRAM are modified with quadrature modulation data. For each 32 bit UV write the CPU generates in NTSC mode the CPU should supply two 32 bit composite plus and two 32 bit composite minus operations. The UV write operation causes 8 dwords of Usin/Vcos data to be burst read from VRAM. Thus for each UV write operation a total of 32 bytes of Usin/Vcos data will be read.

Figure 10. Funky Map SEV

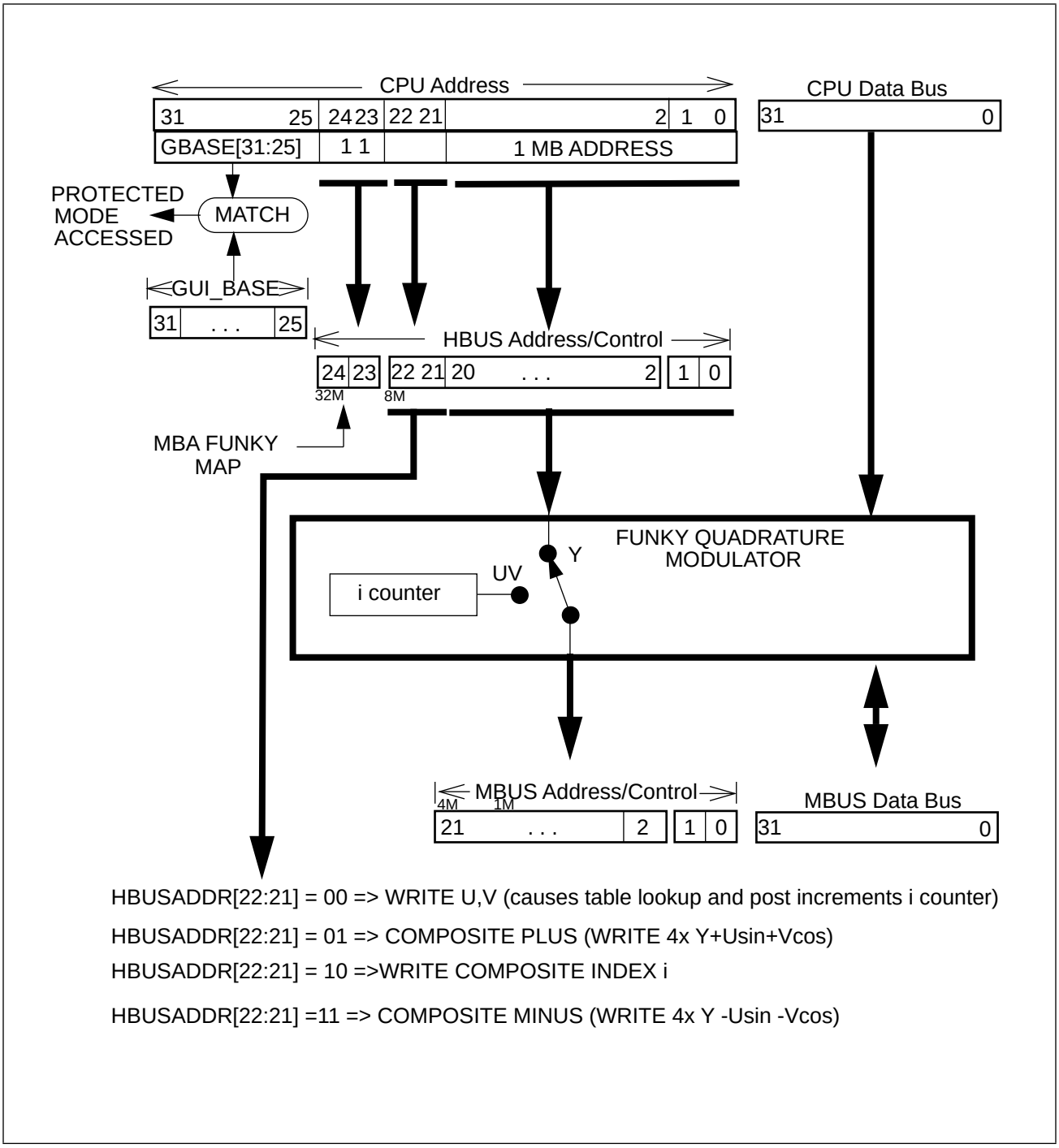


Table 67 shows that when writing U/V values, U goes in VL/PCI data bus [7:0] and V goes into VL/PCI data bus [15:8]. The composite index register is incremented after a WRITE U/V cycle. This is a post increment, occurring after the $16U_i$ and $16V_i$ values have been fetched.

Table 67. WRITE U/V ([22:21]=00)

Bits	Description
31:16	unused
15:8	V sample in 8 bit UV mode, 13:8 for 6 bit UV mode.
7:0	U sample in 8 bit UV mode, 13:8 for 6 bit UV mode.

The MBUS address for each Usin sample in 6 bit UV mode is generated as follows:

```
MBUS_ADDR[21:2] = {GUIREG_MBA[30:28], i_value[7:0],
u_value[7:2], 3'b000}
```

Where i_value[7:0] are the upper bits of the i counter shown in Figure 10. The lower bits of the i counter are supplied by a state machine as the 4 dwords are accessed. The u_value is supplied on the CPU data bus on the WRITE U/V cycle; notice that in 6 bit UV mode, only bits 7:2 are used. In 8 bit UV mode, the address is as follows:

```
MBUS_ADDR[21:2] = {GUIREG_MBA[30], i_value[7:0],
u_value[7:0], 3'b000}.
```

The MBUS address for each Vcos sample in 6 bit UV mode is generated as follows:

```
MBUS_ADDR[21:2] = {GUIREG_MBA[30:28], i_value[7:0],
v_value[7:2], 3'b100}
```

The v_value is supplied on the CPU data bus on the WRITE U/V cycle, notice that in 6 bit UV mode only bits 7:2 are used. In 8 bit UV mode the address is:

```
 MBUS_ADDR[21:2] = {GUIREG_MBA[30], i_value[7:0],
v_value[7:0], 3'b100}
```

For all Usin and Vcos address the bottom 4 bits of the address, i.e. [3:0], are effectively incremented as the 16 bytes of Usin and 16 bytes of Vcos data is fetched.

Table 68 shows that when writing composite plus and composite minus, 16 pixels are sent per Funky Map write.

Table 68. Composite Plus ([22:21]=01)

Bits	Description
31:24	Y3 sample, causes fourth 32 bit write to frame buffer.
23:16	Y2 sample, causes third 32 bit write to frame buffer.
15:8	Y1 sample, causes second 32 bit write to frame buffer
7:0	Y0 sample, causes first 32 bit write to frame buffer interpolated either from previous interpolation from the first y value on the line sent with the write composite index CPU cycle.

The MBUS address for each dword written as a result of the composite plus or composite minus write cycle is generated as follows:

```
MBUS_ADDR[21:2] = {GUIREG_MBA[26],HBUS_ADDR[20:2]}
```

Table 69 shows the bits of the WRITE composite index. No VRAM cycles are performed in response to this operation. Instead the i counter is initialized and the first Y value for the line is put into the Y interpolator.

Table 69. WRITE Composite Index ([22:21]=10)

Bits	Description
31:24	Reserved
23:16	Composite index for sin/cos look up.
15:8	Reserved
7:0	First Y_{old} value for interpolation on a line

Table 70 shows the bits of the Composite Minus field. The difference between this cycle and that shown in Table 68, “Composite Plus ([22:21]=01),” on page 109 is that the modulation performed is Y–u_value–v_value instead of Y+u_value+v_value.

Table 70. Composite Minus ([22:21]=11)

Bits	Description
31:24	Y3 sample, causes fourth 32 bit write to frame buffer.
23:16	Y2 sample, causes third 32 bit write to frame buffer.
15:8	Y1 sample, causes second 32 bit write to frame buffer
7:0	Y0 sample, causes 32 write to frame buffer

Quadrature modulation of NTSC follows the following general sequence shown in Table 71 for one 4:1:1 super pixel consisting of a U and V component and four Y samples. This super pixel results in the writing of 16 PCM samples on each of two even lines and 16 PCM samples on each of two odd lines, for a total of 64 samples written to the media buffer.

Table 71. Quadrature Modulator CPU bus Example

Cycle Type	Address	Data [31:24]	Data [23:16]	Data[15:8]	Data[7:0]	comments
WRITECOMPOSITE INDEX and FIRST Y VALUE	any Funky Map addr w/ [22:21]=10	00h	composite index	reserved	Y_{n-1}	Written once per four lines of output
WRITE U/V	any Funky Map addr w/ [22:21]=00	00h	00h	Vn	Un	no write
COMPOSITE PLUS	PCM w/ [22:21]=01	Y_{n+3}	Y_{n+2}	Y_{n+1}	Y_n	write16 PCM samples
COMPOSITE MINUS	PCM w/ [22:21]=11	Y_{n+3}	Y_{n+2}	Y_{n+1}	Y_n	write16 PCM samples
WRITE U/V	any Funky Map addr w/ [22:21]=00	0000h		V_{n+1}	U_{n+1}	no write
continue with next Y samples						

GUI Command/Register Map

The command/data interface to the GUI accelerator is mediated by a large queue which is implemented as an on-chip FIFO (the GQUE) with an off-chip overflow FIFO in VRAM (VQUE). The GQUE can hold up to 8 entries; the VQUE can be up to 64K entries in length.

The VQUE is controlled by the GUIREG_FIFO register and its start location and length in VRAM vary based on the driver. To prevent wrap-around and corruption of data in the GQUE/VQUE, the driver must limit the addition of items to the queues. The depth of the queues is accessed in the GUI_QUEUE_DEPTH field of the GUIREG_DEPTH register (see Table 73).

GUI FIFO Register

name: GUIREG_FIFO
address: GBASE | 004000F8h, read/write
size: 32 bit
function: This register is accessed *only* via the non-queued interface and controls various aspects of the VQUE, including its location and size. Table 72 shows the bit contents of the GUI FIFO register. Upon reset, all bits are cleared. A write to this register will reset all GUI queues to an initial empty state, except when bit [30] is set to allow masking.

NOTE

The GUI enable bit in configuration register (GRP_CFG4[0]) must be set before this register can be accessed.

Table 72. GUI FIFO Register (GUIREG_FIFO)

Bit(s)	Name	Description
31	VFIFO_ENABLE	1 = Enable VFIFO. Resets status bits in GUIREG_DEPTH.
30	EN_FIFO_WR	Write-only. Always reads back as zero. For diagnostics only. 1= Enable a write only to GUI_STEP and GUI_FREEZE bits without affecting the other bits 0 = Enable write to VFIFO_BASE or VFIFO_ADDR_MASK. Resets GQUE and VQUE to an initial empty condition.
29:28	unused	Must be zero
27:16	VFIFO_BASE	Base address of VFIFO. Actual address interface as 8 zeroes appended to the LSB of the address to form an MBUS address.
15	GUI_FREEZE	1= Stop sending commands from GQUE to the GUI for diagnostic purposes. EN_FIFO_WR must be set. 0 = Normal operation
14	GUI_STEP	0->1 = step next command from GQUE to the GUI into registers. Must only be used when GUI_FREEZE is a one.
13:6	Reserved	Must be zero
5:0	VFIFO_ADDR_MASK	Bits 5:0 mask bits 15:10 of all VFIFO address calculations. 0 means that the corresponding address is not modified as the VFIFO address pointer is incremented. Thus GUI VFIFOs are allocated in powers of 2 dwords (starting at 256). A 6 bit mask value of 01h enables a 512 dword VFIFO while a value of 0Fh enables 4096 dword VFIFOs. VFIFO start addresses are aligned on power of two boundaries at least as large as the VFIFO size. dwords — dwords 000000 - 256 — 001111 - 4k 000001 - 512 — 011111 - 8k 000011 - 1k — 111111 - 16k 000111 - 2k

GUI FIFO Depth Register

name: GUIREG_DEPTH
address: GBASE | 004000F4h, read only
size: 32 bit
function: This register is *only* accessed via the non-queued interface. The purpose of this register is to provide information about the GUI queues, such as the depth (total number of entries) in the GUI queues (GQUE and VQUE) as well as giving software a way to determine whether the GUI subsystem has completed all requested actions. Table 73 shows the bit contents of the GUIREG_DEPTH register. Upon reset, all bits are cleared.

NOTE

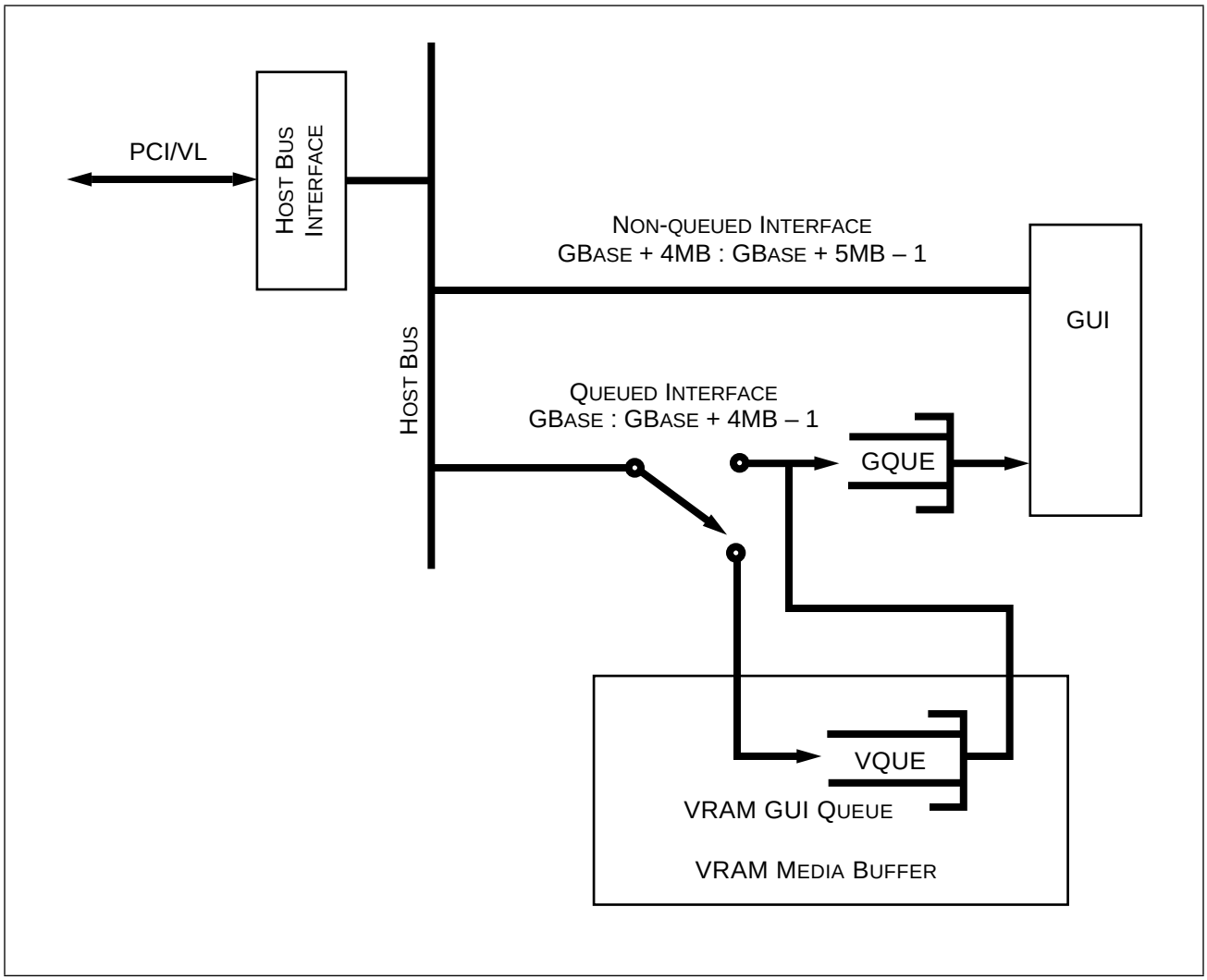
The GUI enable bit in configuration register (GRP_CFG4[0]) must be set before this register can be accessed.

Table 73. GUI FIFO Depth Register (GUIREG_DEPTH)

Bit(s)	Name	Description
31:24	unused	reserved
23	GUI_BLT_ UNDERFLOW	1=GUI screen --> host BLT underflow error
22	GQUE_ OVERFLOW	1=GUI GQUE overflow error
21	GUI_BLT_ DATA_RQD	1=GUI host --> screen write data required
20	GUI_BLT_ DATA_RDY	1=GUI screen --> host read data ready
19	GUI_BUSY	1=GUI command processor busy
18	RESULT_FIFO_ BUSY	1=GUI result FIFO non-empty
17	GUI_QUEUE_ BUSY	1=GUI queue processor busy
16	GUI_FIFO_ NON_EMPTY	1=GUI GQUE FIFO non-empty
15:0	GUI_QUEUE_ DEPTH	Unsigned up-down counter used to track GUI FIFO depth of usage. Depth counter = 0 for empty FIFO. Count increments for each word written to FIFO; decrements for each word taken from FIFO and sent to GUI. This field reflects the combined total depth in both the GQUE and VQUE.

All GUI commands, parameter values, and write data is queued either on-chip in the GQUE or off-chip in the VQUE. In addition, GUI register values which can be queued will load their specified register "IN-ORDER" with the GUI commands. As shown in Figure 11, there is an on-chip GQUE that gets consumed before the overflow is written to VRAM so that if the GUI accelerator is keeping up with its command arrival rate the VQUE will not be used and the VRAM bandwidth associated with the FIFO will not be consumed. If the CPU attempts to append to the VFIFO at the same time that the GUI command processor attempts to retrieve from the VFIFO, the CPU is granted access for the higher priority append operation.

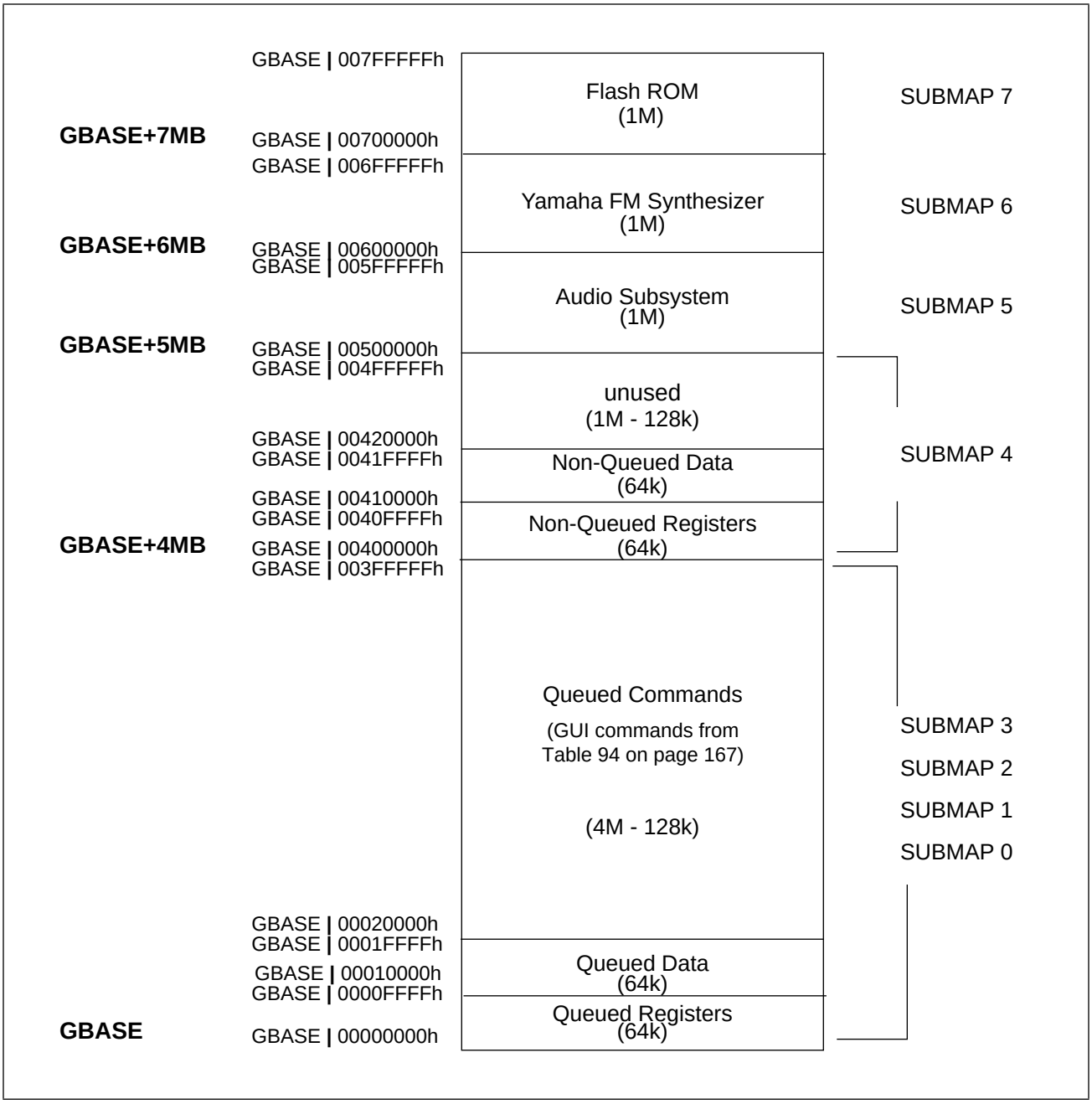
Figure 11. Non-queued, Queued And VRAM Queued GUI Interface



As shown in Figure 12, the command/register map is divided into eight submaps of one MB each, numbered 0 through 7. Each submap contains 16 blocks, each 64K long. Submap 4 is a non-queued interface to the GUI accelerator and begins at the GBASE address plus 4MB. The queued interface map (submaps 3

through 0) begins at GBASE+0MB. Any command written to the queued interface will use the GQUE on the chip and may use the VQUE in VRAM. All commands to the non-queued interface will "go-around" this FIFO. All reads of any kind go around the FIFO, thus driver software must insure that the GUI accelerator state is consistent with a read operation by reading the GUIREG_DEPTH to determine whether the GUI is idle.

Figure 12. GUI Command/Register Map



Below, Table 74 indicates the read/write characteristics of the contents of the submaps.

Table 74. Submap Read/Write Characteristics

SubMap Region	Read	Write
Non-Queued Data	Used in screen-to-host BLTs	Ignored
Non-Queued Registers	Possible	Possible
Queued Commands	Ignored	Possible
Queued Data	Ignored	Used in host-to-screen BLTs
Queued Registers	Ignored	Possible

Software can read the current depth of the GQUE/VQUE through the non-queued interface to effect efficient management of the GQUE/VQUE. The FIFO is implemented with two address registers and an up/down counter. The address registers implement a ring buffer, wrapping at a power of two boundary, as specified by driver initialization code.

Writing to GUIREG_FIFO will clear the GQUE/VQUE, resetting the counter to zero and reloading the starting address for the VQUE. Notice that each command, register, or data packet in the GQUE/VQUE will contain a 32 bit address followed by zero or more 32 bit data values.

The GQUE/VQUE implements a compression technique to reduce the unnecessary storage of addresses in either the VRAM or on-chip FIFOs. This compression stores the address field of the first word of a command, however for subsequent parameter dwords no address is stored. Similarly, for host to screen BLTs, the data length is stored in the first location in the FIFO, subsequent data words are stored without inserting address values. For queued register writes, an address is stored for every register data word in the FIFO. Thus an n parameter command would occupy n+1 locations in the FIFO. Similarly, an m dword data transfer would store m+1 locations in the FIFO. In calculating whether a command can be sent to the GUI, one must make sure that there is *at least* n+1 locations remaining in the FIFO by checking the GUIREG_DEPTH register. As reading the GUIREG_DEPTH register is a time consuming proposition, one might keep track of the available space in a variable in the X86 memory.

Flash Rom Support

Submap 7 at GBASE+7MB is used to access all of the Flash EPROM residing on the VRAM data bus. In the BtV2115, this can be up to 1MB of Flash ROM. The ROM in this submap is used to hold the Windows NT hardware abstraction layer. The VGA BIOS ROM is visible as the first 32K of this Flash ROM space.

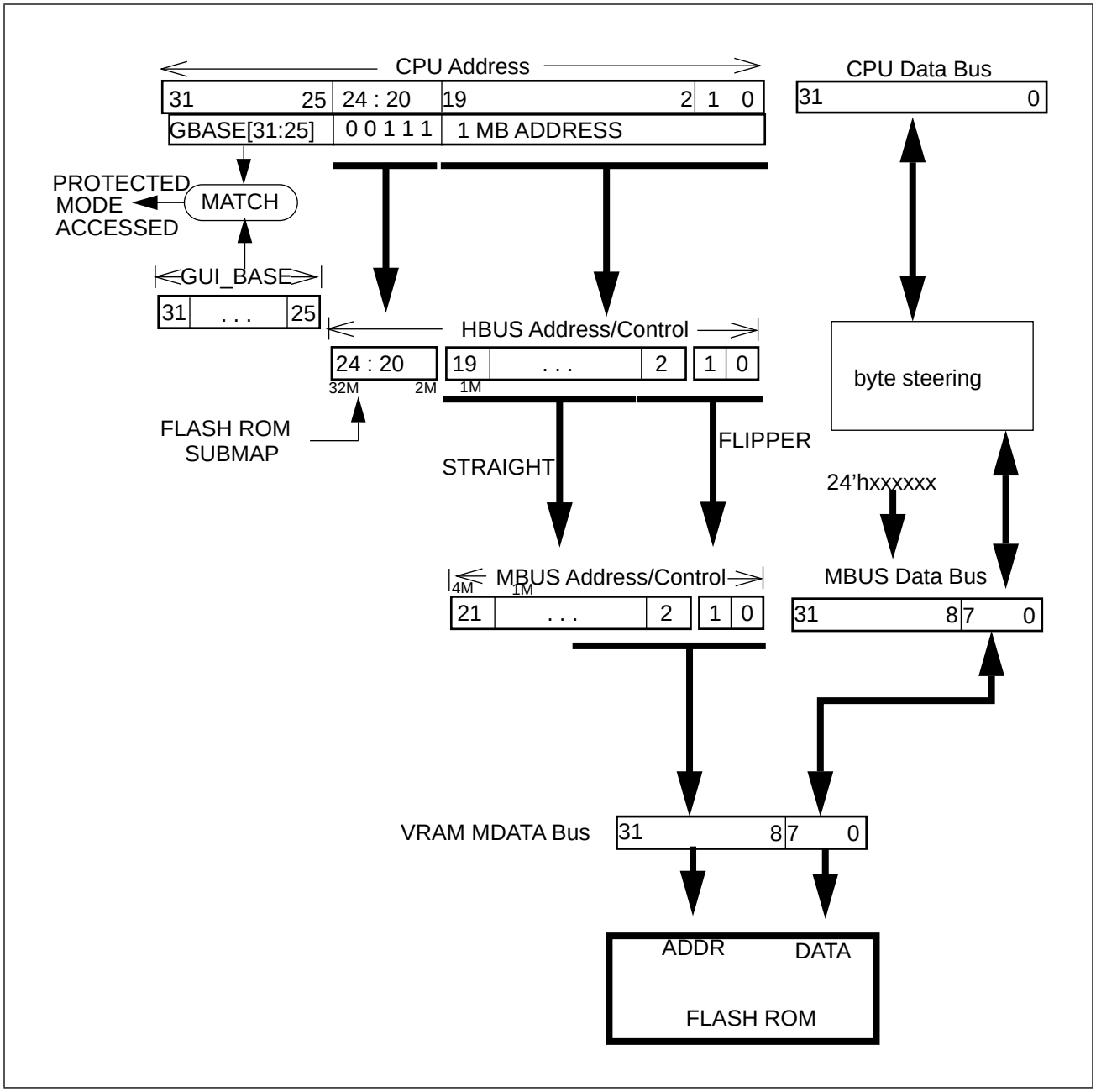
Flash ROM can be read or written through the protected mode aperture. Writing is only enabled when the appropriate configuration register bit is also set.

NOTE

Flash ROM can only be written after the unlock sequence (see “Unlock Register” on page 67) has been sent to it and must be written in “sectors,” refer to the appropriate Flash ROM vendor’s specification.
In addition, Flash ROM writes are inhibited by the default state of configuration register GRP_CFG4 bit 6 (see “Configuration Registers” on page 47).

Below, Figure 13 shows that up to one MB of Flash ROM can be accessed when the CPU address bits 31:25 match the GUI_BASE register bits 31:25 (PM_BASE) and CPU address bits 24:20 equal 5’b00111. CPU address bits 17:2 and the byte enables are passed out the VRAM data bus so that one MB is addressed for read or write.

Figure 13. Flash ROM Access Address Mapping



If we make the following C definitions:

```
#define PM_ROM 0x00700000 // PROT MODE ROM
```

Then location 300 in the ROM can be accessed through the protected mode as follows:

```
((unsigned char *) (GBASE | PM_ROM | 0x00000300L))
```

Note that Flash ROM can also be accessed through the “VGA ROM Aperture” on page 129.

Yamaha Support

In BtV2115, the four bytes of Yamaha OPL3 register (or the eight bytes of OPL4) are repeated continuously through the entire 1MB space of submap number 6. The Legacy Audio I/O space is mapped by the host bus interface into exactly the same HBUS address range as the protected mode aperture uses when addressing this submap.

NOTE

Legacy Audio supports the Yamaha 2 and 4 operator mode FM synthesizer chip family, with direct support for OPL3.

Figure 14 shows that the Yamaha FM synthesizer can be accessed when the CPU address bits 31:25 match the GUI_BASE register bits 31:25 (PM_BASE) and CPU address bits 24:20 equal 5’b00110 CPU address bits 17:2 and the byte enables are passed out the VRAM data bus so that the eight bytes of OPL4 registers are addressed for read or write.

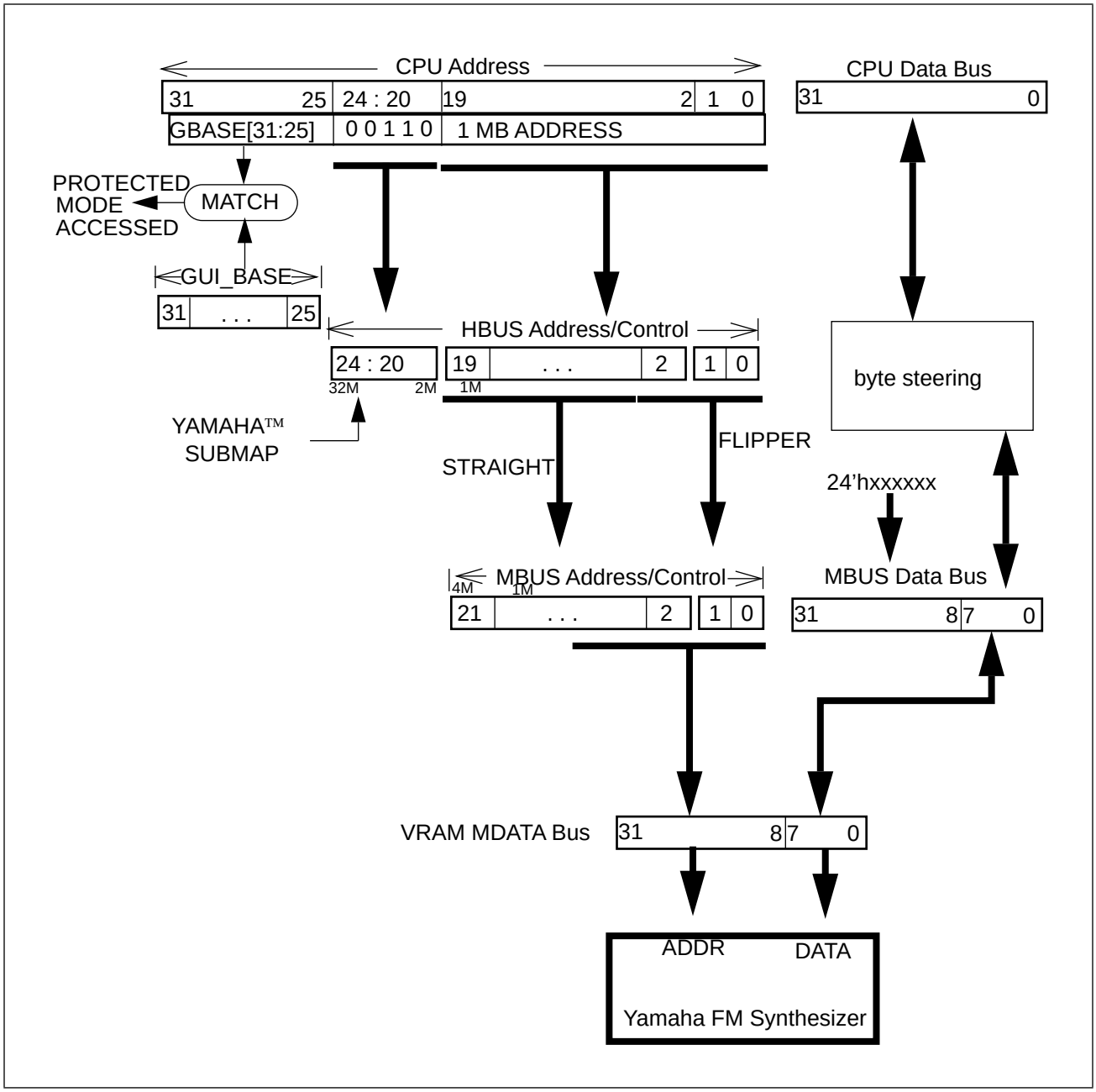
If we make the following C definitions:

```
#define PM_YAMAHA 0x00600000 // PROT MODE YAM
```

Then location 388 in the Yamaha Legacy Audio I/O space can be accessed through the protected mode as follows:

```
((unsigned char *) (GBASE | PM_YAMAHA | 0x00000388L))
```

Figure 14. Yamaha Access Address Mapping



Audio Subsystem

The registers for the audio subsystem of the BtV2115 are mapped into the 1MB of submap number 5, see Figure 12. The audio subsystem provides a Legacy Audio Pro compatible interface for use in the DOS environment and an improved windows sound device driver support mode. The host bus interface maps the corresponding Legacy Audio I/O into an HBUS address thus the actual Legacy Audio registers are accessible either as I/O registers or as memory mapped I/O in the protected mode aperture.

If we make the following C definitions:

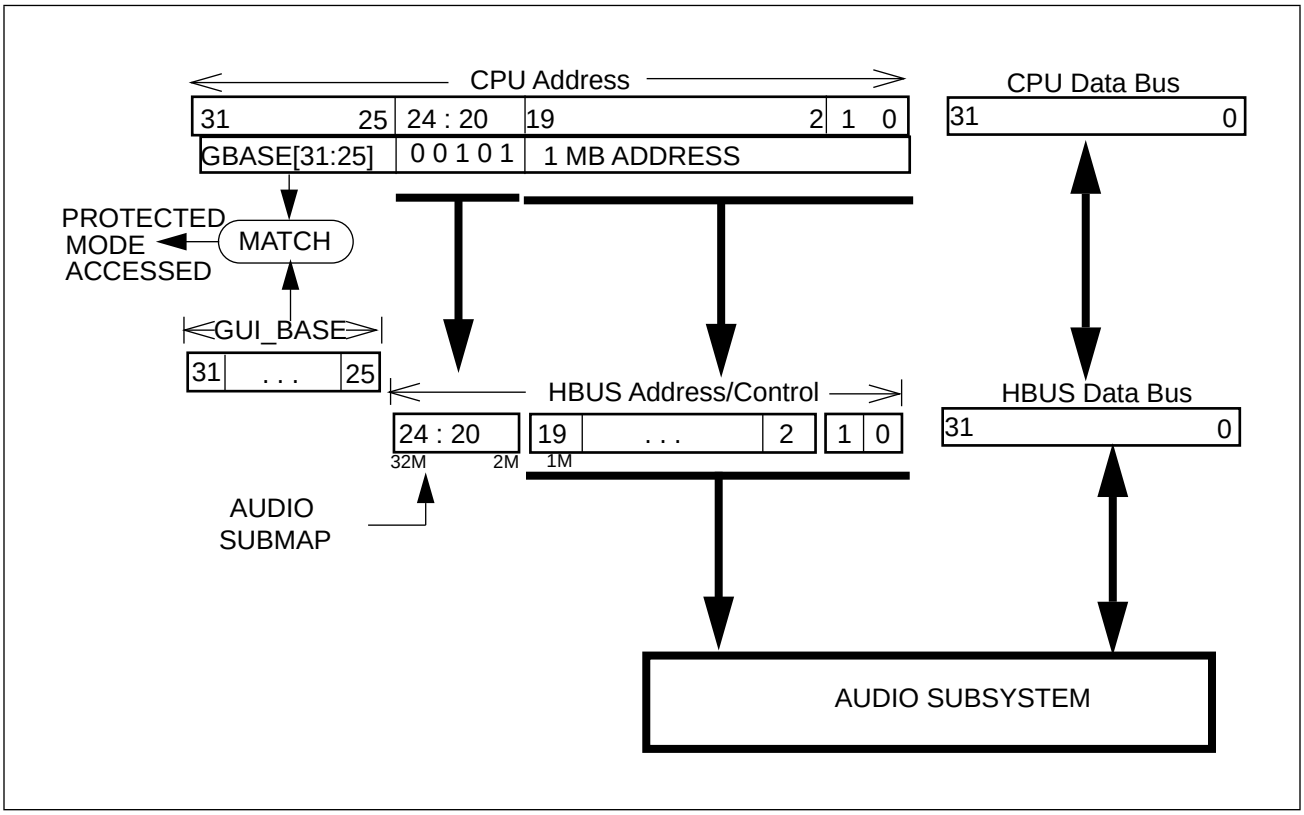
```
#define PM_AUDIO 0x00500000L // PROT MODE AUDIO
```

Then location 100 in the AUDIO subsystem can be accessed through the protected mode as follows:

```
((unsigned char *) (GBASE | PM_AUDIO | 0x00000100L))
```

For details on the audio subsystem, refer to the "Audio Interface" chapter starting on page 271.

Figure 15. Audio Access Address Mapping



GUI Base Address Register

name: GRP_GUI_BASE[3:0]
index: 23h:20h, read/write
size: 32 bit
function: The GUI base address register occupies four bytes of VGA graphics register indexed space. These bytes are in the extension space of the VGA, which means they are normally not accessible to VGA programs unless the extensions have been unlocked. Once unlocked, the GUI base address register is accessible at index 20h through 23h. The indices are loaded into the VGA graphics index register at I/O address 03CEh. The register values are accessed through the VGA graphics data register at I/O address 03CFh.

The GUI base address register is a 32 bit register with the least significant byte at index 20h and the most significant byte at 23h so that:

- index 20h has GUI base [07:00]
- index 21h has GUI base [15:08]
- index 22h has GUI base [23:16]
- index 23h has GUI base [31:24]

Table 75 defines the bit contents of the GUI base address register for both 32K and 64K apertures. Upon reset, all bits are cleared.

Table 75. GRP_GUI_BASE Address Register *(1 of 2)*

Bits in 64K apert.	Bits in 32K apert.	Field	Description
31:25	31:25	PM_BASE	These seven bits are used to match against the upper seven bits of all VL and PCI memory addresses. The protected mode aperture is accessed when such a match occurs. In addition, no match will occur unless at least one of the PM_BASE bits is set to one. This insures that the protected mode aperture can not overlay DOS memory from 00000000h to 000FFFFFh (blocking access to the lower 32MB of the CPU memory address space). Thus software cannot inadvertently cause a collision between system memory and the protected mode aperture. The protected mode aperture is disabled when this register is set to zero.
24	24	PM_ENABLE	This bit must be set to one to enable the protected mode aperture. This bit controls whether the entire 32MB aperture is accessible.
23	23	reserved	
22	22	ENABLE_A0000	1 = Ignore graphics miscellaneous register Memory Map = 01 in the A0000 to AFFFF range and force the decoder to accept CPU read/write cycles. (See Table 14 on page 42.) 0 = Graphics miscellaneous register controls A0000 to AFFFF.
21	21	†32K_APERTURES	1= 32K real mode aperture 0= 64K real mode aperture

Table 75. GRP_GUI_BASE Address Register *(2 of 2)*

Bits in 64K apert.	Bits in 32K apert.	Field	Description
__	20	†ENABLE_32K_ APERTURES	If bit 21 =1 and bit 20 = 1: enable 32K real mode apertures
20:19	___	unused	If bit 21=0 then these are unused
18	___	†ENABLE_64K_ REAL	If bit 21=0 and bit 18 = 1: enable 64K real mode apertures
17:9	19:10	†REAL_MODE_ APERTURE1	These bits are appended as the high order address bits of a 25 bit address which is applied to the HBUS as if it had come from a protected mode reference to GUI_BASE. The lower 16 bits for 64K mode or 15 bits for 32K mode come from the memory address of a read or write to a location determined as follows. 64K mode: 000B0000h - 000BFFFFh 32K mode: 000A8000h - 000AFFFFh
8:0	9:0	†REAL_MODE_ APERTURE0	These bits appended as the high order address bits of a 25 bit address which is applied to the HBUS as if it had come from a protected mode reference to GUI_BASE. The lower 16 bits for 64K mode or 15 bits for 32K mode come from the memory address of a read or write to location determined as follows. 64K mode: 000A0000h -000AFFFFh 32K mode: 000A0000h - 000A7FFFh
† To set 32K mode, bits 20 and 21 must be 1 To set 64K mode, bit 18 must be 1 and bit 21 must be 0 (Refer to “Real Mode Aperture” on page 127 for more information.)			

MBA Control Register

name: GUIREG_MBA
address: GBASE | 004000F0h, read/write
size: 32 bit
function: The Media Buffer Aperture (MBA) control register controls various functions of the MBA, like Flippin Map flip controls, Funky Map component access versus quadrature modulation modes, etc. and holds the VRAM pointer to the quadrature modulator sin/cos table. There are independent flip controls for Funky Map, Flippin Map and most everything else, but there is no way to flip the Direct Map, ROM reads/writes, and accesses to the Yamaha and audio subsections. Table 76 defines the bit contents of the GUI MBA register. Upon reset, all bits are cleared.

NOTE

The GUI enable bit in configuration register (GRP_CFG4[0]) must be set before this register can be accessed.

Table 76. GUI MBA Control Register

Bits	Field	Description
31:28	SINCOS_ADDR	SIN/COS table pointer upper four bits of an 8MB MBUS memory address. Only three bits are used in the BtV2115.
27:26	MODUL_BASE	Modulation output base address 00 = SEV frame buffer is in first 2MB of media buffer. 01 = SEV frame buffer is in second 2MB of media buffer. 10 = SEV frame buffer is in third 2MB of media buffer. 11 = SEV frame buffer is in fourth 2MB of media buffer.
25	UV_6BIT	6-bit/8-bit UV values for modulation 1 = U and V values are most significant 6 of the supplied 8 bits. 0 = All eight bits of U and V are used
24	YCRB_422MODE	4:2:2/4:1:1 mode for YCrCb 1 = 4:2:2 format 0 = 4:1:1 format
23	MODUL_EN	Enable modulation logic 1 = Funky Map configured to perform quadrature modulation for SEV. 0 = Funky Map configured to perform component access for read and write.
22	Reserved	Must be 0
21	FLIPPIN_WORD	Flippin Map word flip control, 1=flip
20	FLIPPIN_BYTE	Flippin Map byte flip control, 1=flip
19	FLIPPIN_BIT	Flippin Map bit flip control, 1=flip
18	Reserved	Must be 0
17	CMD_WORD	Command map word flip control, 1=flip
16	CMD_BYTE	Command map byte flip control, 1=flip
15	CMD_BIT	Command map bit flip control, 1=flip
14	Reserved	Must be 0
13	FUNKY_WORD	Funky Map word flip control, 1=flip
12	FUNKY_BYTE	Funky Map byte flip control, 1=flip
11	FUNKY_BIT	Funky Map bit flip control, 1=flip
10:4	YAMAHA_DELAY	Yamaha delay in mclks, default = 16 mclks
3:0	Reserved	Must be 0

Real Mode Aperture

The VGA aperture mechanism allows access to the 32MB protected mode aperture from 486 real mode. This mechanism involves changing the interpretation of the VGA memory aperture at 000A0000h-000BFFFFh so that it provides access to the various HBUS agents instead of to the VGA frame buffer. Two VGA apertures are provided to offer the most robust support of the VESA BIOS modes.

For 64K mode, two 64KB apertures are available for accessing different HBUS agents or for accessing different places within a single HBUS agent. This access of HBUS agents is enabled when GUI_BASE[18] is set to one and GUI_BASE[21] is set to zero. REAL_MODE_APERTURE0 responds to CPU reads/writes in the address range 000A0000h through 000AFFFFh. The HBUS is accessed by a 25 bit address; GUI_BASE[8:0] (REAL_MODE_APERTURE0) defines the upper 9 bits and CPU address [15:0] defines the lower 16 bits to form the HBUS address as follows:

```
HBUS_ADDR[24:0]={GUI_BASE[8:0],CPU_ADDRESS[15:0]}
```

NOTE

CPU address[1:0] is actually expressed via byte enables.

The resulting address is fed to the HBUS decoder as if it had come through the protected mode aperture. Thus when GUI_BASE[8:0] is loaded with 071h and when the CPU accesses VGA_APERTURE0, then CPU reads/writes to address range 000A0000h - 000AFFFFh will access the HBUS agent (Flash ROM) at ROM locations 10000 through 1FFFF, i. e. the second 64K byte block (refer to Figure 12 on page 116).

64K mode REAL_MODE_APERTURE1 responds to CPU reads/writes in the address range 000B0000h through 000BFFFFh. The HBUS is accessed by a 25 bit address; GUI_BASE[17:9] (REAL_MODE_APERTURE1) defines the upper 9 bits and CPU address [15:0] defines the lower 16 bits to form the HBUS address as follows:

```
HBUS_ADDR[24:0]={GUI_BASE[17:9],CPU_ADDRESS[15:0]}
```

This mechanism is useful for accessing all parts of BtV2115 when the CPU is running in real mode, as occurs at power on.

In 32K mode, access of HBUS agents is enabled when GUI_BASE[20] and GUI_BASE[21] are set to one. REAL_MODE_APERTURE0 responds to CPU reads/writes in the address range 000A0000h and 000A7FFFh. The HBUS is accessed by a 25 bit address; GUI_BASE[9:0] (REAL_MODE_APERTURE0) de-

fines the upper 10 bits and CPU address [14:0] defines the lower 15 bits to form the HBUS address as follows:

```
HBUS_ADDR[24:0]={GUI_BASE[9:0],CPU_ADDRESS[14:0]}
```

32K mode REAL_MODE_APERTURE1 responds to CPU reads/writes in the address range 00A7FFFh through 000AFFFFh. The HBUS is accessed by a 25 bit address; GUI_BASE[19:10] (REAL_MODE_APERTURE1) defines the upper 10 bits and CPU address [14:0] defines the lower 15 bits to form the HBUS address as follows:

```
{GUI_BASE[19:10],CPU_ADDRESS[14:0]}
```

The VGA aperture mechanism has many applications. For example, it is useful for BIOS routines that want to read more ROM data than is available in the VGA video BIOS space from C0000 to CFFFF, since setting the VGA_APERTURE register to 070h sets the MBA address to submap 7, i.e. the Flash PROM space. Setting the VGA_APERTURE register to 050h makes the audio registers available from real mode 80486 code. When ENABLE_VGA_CRTC is set (GRP_CFG4[2]), memory references to the VGA aperture are redirected to an HBUS agent.

VGA ROM Aperture

The VGA ROM aperture runs from 000C0000h to 000C7FFFh. It can be selectively disabled by a BIOS enable configuration bit (GRP_CFG4[5], refer to Table 22 on page 50) when BtV2115 is mounted on the motherboard and the system ROM is used to hold video ROM code. Even when the VGA ROM aperture is enabled, a section from C6000 to C67FF can be disabled. When the VGA ROM aperture is enabled, HBUS addresses are formed by concatenating the CPU address bits [14:2] with GRP_ROMPAGE[5:0] and pre-pending "00111" to the top to access the Flash ROM:

```
HBUS_ADDR[24:2]={[5'b00111,GRP_ROMPAGE[5:0],CPU_ADDR[14:2]}
```

Thus any 32K page of the possible 1MB Flash ROM can be mapped into the VGA ROM Aperture.

The VGA ROM Aperture is not available on PCI base systems. Instead, a PCI compliant Expansion ROM Address register under the control of the PCI system BIOS is used to map the 1MB ROM space. When the VGA ROM aperture is enabled, HBUS addresses are formed by concatenating the CPU address bits [19:2] with "00111" to access the Flash ROM:

```
{[5'b00111,CPU_ADDR[19:2]}
```

For further information, refer to "PCI Configuration Space" on page 219.

CPU Apertures In a PCI-Bus Environment

PM_BASE Field

The PM_BASE field of the GUI_BASE register is also read and written in a PCI-bus environment whenever PCI configuration registers are accessed for the BtV2115. Specifically, whenever PCI_BASE0_REG is accessed for PCI function 0. For further detail, refer to "PCI Base Address Registers" on page 222.

When PCI system BIOS writes all ones to PCI_BASE0_REG then only the bits in PM_BASE respond, telling the BIOS that a 32MB aperture is available. The BIOS next loads PCI_BASE0_REG with the address assignment for the BtV2115 protected mode aperture. PCI_BASE0_REG is flagged non-prefetchable so the PCI system BIOS knows to mark this aperture as non-prefetchable in the PCI bridge. This aperture is marked because the Audio/Yamaha and GUI submaps have read side effects.

Prefetch Base

Because a non-prefetchable aperture status in an intelligent PCI bridge can cost significant performance, the BtV2115 offers a second and independent 32MB aperture when in PCI mode. This aperture is the Prefetchable Aperture and when enabled is accessed via the PCI_PREFETCH_BASE PCI configuration register. This register is marked prefetchable and will be assigned to a separate and non-overlapping 32MB address range in the CPU address space. From an address decode perspective, the BtV2115 treats this 32MB aperture as an alias of the protected mode aperture found through PM_BASE.

One can safely access the Direct, Flippin and Funky Maps as well as the ROM submap through the prefetchable aperture. One should never access anything in the first 7MB of the prefetchable aperture. Because the ROM can be safely accessed through either map it serves as the neutral zone between safe and unsafe. Software should restrict ROM accesses to the protected mode aperture so that an intelligent bridge will not attempt to opportunistically read ahead into the Yamaha submap.

PCI Constraints on GBASE

Because of the plug'n' play nature of the PCI, the protected mode aperture address value can not be specified as a constant for any software that will run in a PCI environment. The definition of GBASE, above, was intentionally over simplified as a C preprocessor constant. In actuality it must be a program variable, loaded at initialization time from the PM_BASE field of the GUI_BASE register.

In addition, the possible existence of a prefetchable aperture means the two address pointer variables must be defined and utilized. Any software touching the Direct, Flippin or Funky Maps should use the second variable. At initialization, software should look at GRP_CFG[7] bit 7, which if set, indicates that the second aperture is operational and should be used if PCI system BIOS enabled it. If the prefetchable base is turned off, then both address pointers should be loaded from PM_BASE. To review, the prefetch aperture could have been turned off by a configuration strap, by the PCI BIOS being unable to assign a 32MB address range to it, or by virtue of the BtV2115 being installed in a VL environment.

CONFIGURATION REGISTERS

Introduction

The BtV2115 MediaStream Controller provides a set of registers for specifying operating parameters for interaction of the multimedia subsystem components (both internal and external to the BtV2115). The Configuration registers consist of eight bytes controllable by software within the VGA Graphics Controller Extension Registers, writing values 40h-47h to the Index/Address Register (03CEh) and reading or writing data at the Data Register location (03CFh). Address 40h corresponds to GRP_CFG0, address 41h corresponds to GRP_CFG1, etc.

For details on the bit contents of the eight configuration registers, refer to “Configuration Registers” on page 47 of the “Register Definitions” chapter.

Register Initialization

At reset all bits contained in the Configuration registers are initialized. Those bits which have reset values definable by the user are initialized based on logic levels present on MDATA pins. Not all bits of the MDATA bus are used, but those that are used can be programmed by strapping the pin to either VDD or VSS. Pin strappings should utilize a high value resistor (20K-ohms is recommended) to tie the pin to either VDD or VSS, as shown in Figure 16.

Figure 16. Configuration Resistor Straps

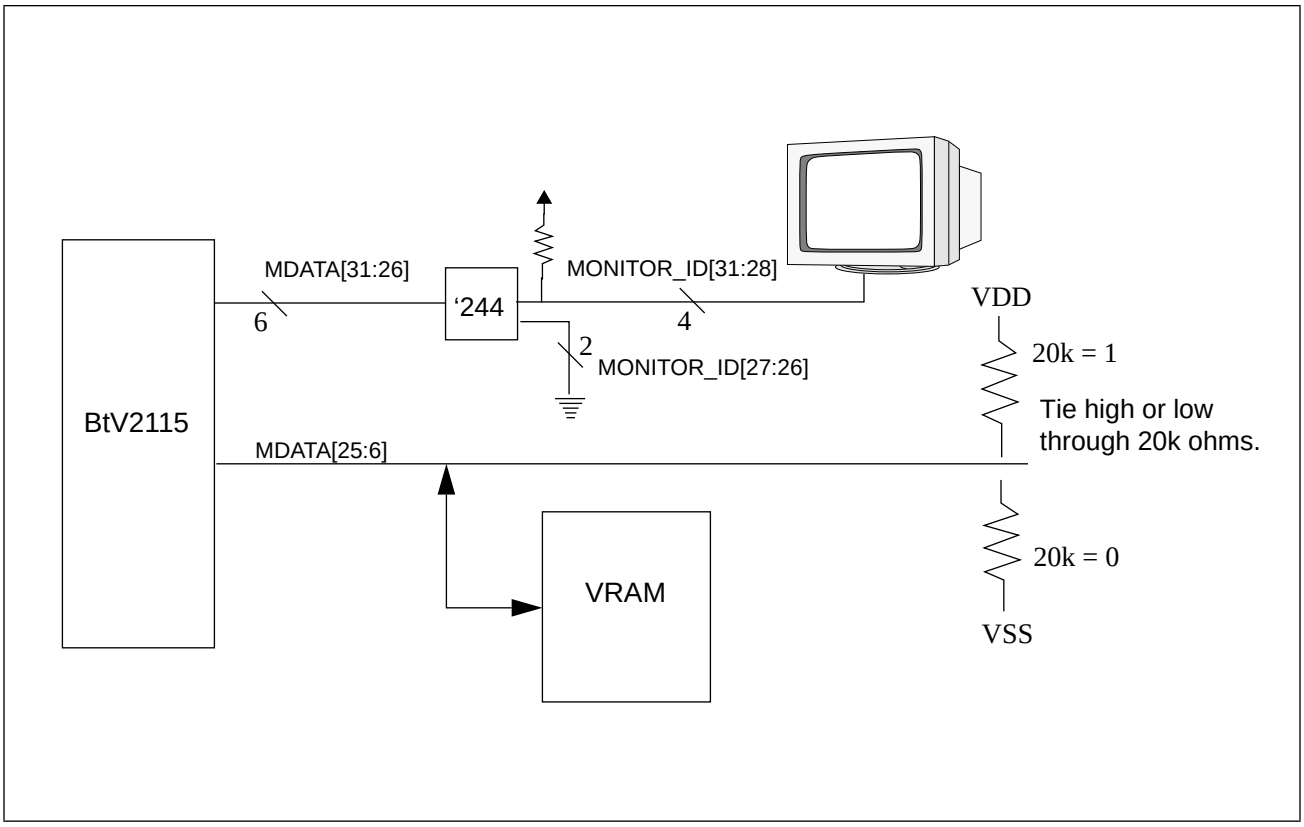


Figure 17 shows the pins of the MDATA bus which affect the initialization state of the Configuration registers. Note that pins MDATA[15:6] directly affect the initialization of VRAM_TYPE field. The value applied to MDATA[15:6] appears in GRP_CFG7 and GRP_CFG0. These Configuration registers contain VRAM-specific parameters which are initialized at reset to their expected values by loading their values from MDATA[15:6]. The bits contained in GRP_CFG0 may be customized by the user after reset by simply overwriting the initial value as needed. The value in GRP_CFG7[5:0] indicates the memory configuration of the system and therefore is read-only.

The CHIP_MODEL_CODE bits are also read-only since they reflect the functional capability built into the BtV2115. Note that CHIP_MODEL_CODE[0], which reflects the availability of a PCI bus interface, is not directly readable by the program. Furthermore, notice that the bus type is read-only. The bus type is programmed by configuration strap resistors according to which bus type the BtV2115 is attached, as follows:

Table 77. Bus Type Encode

MDATA [18:17]	Config Reg 3 bits 4:3	Bus Type
VSS: VSS	0: 0	VESA Local Bus
VSS: VDD	0: 1	PCI 32
VDD: VSS	1: 0	Reserved
VDD: VDD	1: 1	Reserved

The mapping between chip model code bits and BtV2115 part numbers is given in Table 78. Notice that the /V versus /P coding cannot be determined programmatically. If a chip's configuration registers can be successfully read via PCI cycles, then it is a /P version of the BtV2115. If a chip's configuration registers can be successfully read by a VL bus cycle, it can be either a /V or /P part number.

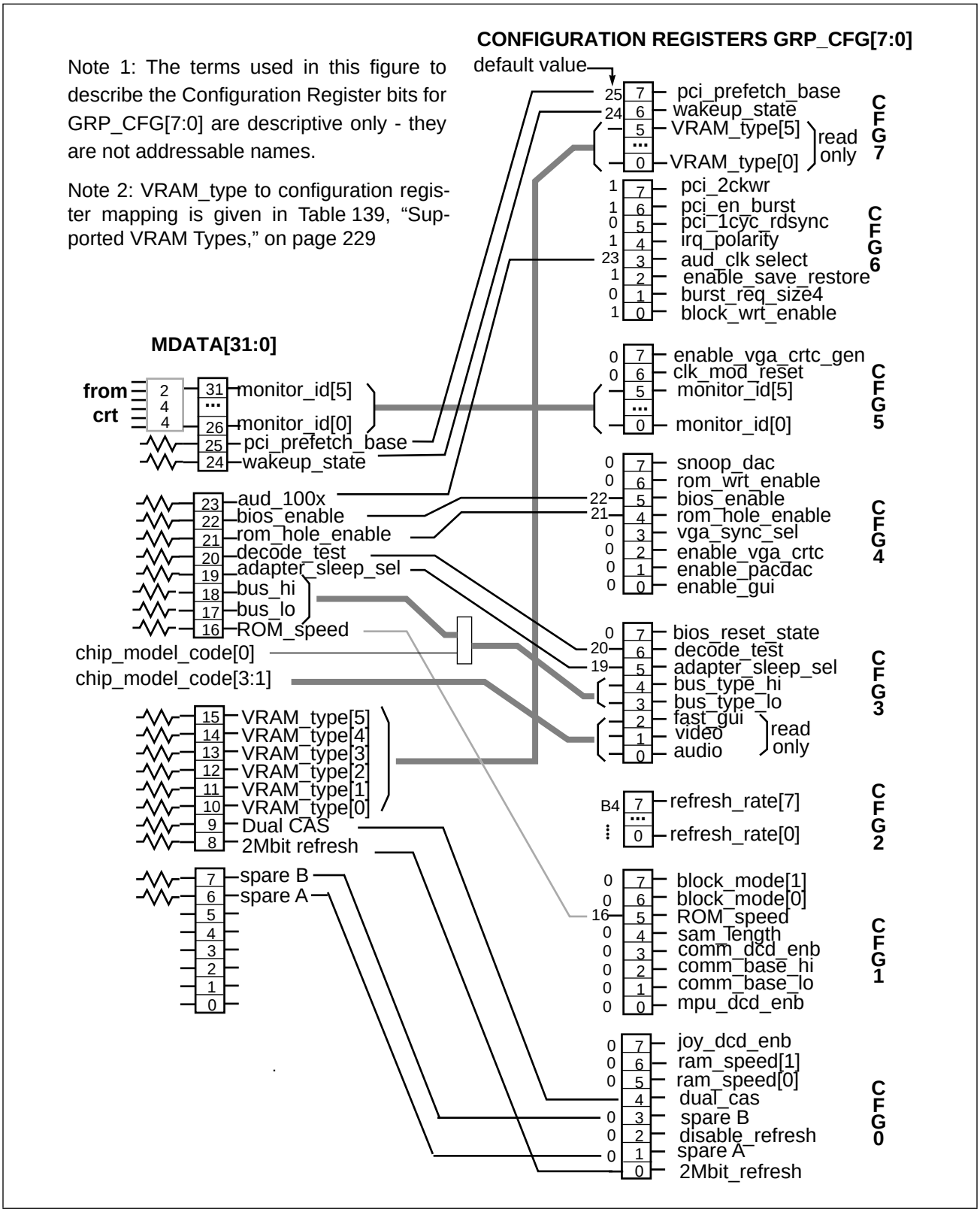
Table 78. Chip Model Coding

CHIP_MODEL_CODE [2:1]	PART # for CHIP_ MODEL_CODE [0]=0	PART # for CHIP_ MODEL_CODE [0] = 1
00	Reserved	Reserved
01	Reserved	Reserved
10	Reserved	Reserved
11	BtV2115/V	BtV2115/P

CHIP_MODEL_CODE[3] indicates whether the VRAM GUI queue is present in the BtV2115.

The MONITOR_ID bits [31:26] are also sampled at reset, indicating the type of monitor connected. MONITOR_ID bits [31:28] are pulled-up and bits [27:26] are pulled-down but not used. As shown in Figure 16, a design providing monitor ID bits would use a '244 to gate these bits onto MDATA bits [31:26] during reset.

Figure 17. Configuration Registers and Strapping Bits for BtV2115



I^2C MASTER AND SLAVE CONTROLLERS

Introduction

This chapter specifies the programming interface to both the I^2C master and I^2C slave modules of the BtV2115 Media Controller. Both of these blocks operate independently of each other so they are documented separately.

The BtV2115 supports two independent I^2C buses, referred to here as IIC and IIC_DDC. Both support ACCESS.bus protocols. IIC_DDC has additional logic to support VESA monitor DDC1, DDC2, and DDC2B signalling. Since there is only one master controller and one slave controller, only one I^2C bus can be designated as the ACCESS.bus. Typically if a DDC2B monitor is detected, it will provide fanout of the ACCESS.bus.

The I^2C block consists of the I^2C master module, the I^2C slave module and a small amount of shared logic. The I^2C master module is capable of transmitting and receiving data over the I^2C bus as an I^2C bus master. This functionality is required for the I^2C links to the BtV2487 and BtV2811A chips and is also required to support the ACCESS.bus protocol which is built on top of the I^2C protocol. The I^2C slave module is capable of receiving data over the I^2C bus as an I^2C bus slave. This functionality is not required for the I^2C links to BtV2487 and BtV2811A; it is required solely to support the ACCESS.bus protocol.

Two sets of I^2C pins allow the BtV2115 to support the IIC and IIC_DDC busses. This means that the BtV2487 and BtV2811A I^2C bus can be isolated from the ACCESS.bus bus. The I^2C master and slave modules can be independently switched from one bus to the other. The switching of the master and slaves modules to the IIC and IIC_DDC busses is controlled by the GRP_I2C_CTRL(R/W) Control register, which is defined in the next subsection.

The I^2C master module is capable of transmitting and receiving data over the I^2C bus and is controlled through two VGA extension registers. Both master transmit and master receive modes are required by the BtV2115 to BtV2487 I^2C communication link. This module fulfills all these requirements. Support for the ACCESS.bus spec requires the master transmit and slave receive modes, however this module does not provide the slave receive mode; this must be fulfilled by the I^2C slave module.

The I^2C slave module is capable of receiving data over the I^2C bus as a slave device and is controlled through two VGA extension registers.

This controller is designed to work either polled or interrupt driven in an operating environment that has many other interrupts being serviced for other devices. For example, if a clock interrupt occurs in the middle of an I²C receive using a polling design, no I²C data will be lost.

This module is designed to implement the operations of an I²C and ACCESS.bus interface *with the assistance of system software*. It is possible for the system software to make this hardware perform in violation of either the I²C or ACCESS.bus specs. This design assumes that the system software does not attempt to do any such illegal activities and the design makes no attempt to protect against this eventuality.

NOTE

The I²C interrupts are defined in Table 165, "Interrupt Status, State and Mask Register Bits," on page 291.

For electrical specification and timing for I²C I/O stages and bus lines, refer to the Philips Semiconductor publication "*The I²C-bus and How to Use It*."

I²C Control WRITE Register

name: GRP_I2C_CTRLW
index: 4Bh, write
size: 8 bit
function: This register enables the I²C serial control interface. The bit contents of the GRP_I2C_CTRLW Control register are shown in Table 79.

Table 79. GRP_I2C_CTRLW Control Register *(1 of 2)*

Bit	Function	Detail
7	I2CMA_SEL	1 = select IIC bus for master module 0 = select IIC_DDC bus for master module
6	I2CSL_SEL	1 = select IIC bus for slave module 0 = select IIC_DDC bus for slave module
5:4	reserved	set to zero.

Table 79. GRP_I2C_CTRLW Control Register *(2 of 2)*

Bit	Function	Detail
3	$\overline{\text{OVR\_IIC\_SCL}}$	Override SCL clock signal in IIC bus 1= release SCL clock 0= force SCL clock low
2	$\overline{\text{OVR\_IIC\_SDA}}$	Override SDA data pins in IIC bus 1= release SDA data 0= force SDA data low
1	$\overline{\text{OVR\_DDC\_SCL}}$	Override SCL clock signal in IIC_DDC bus 1= release SCL clock 0= force SCL clock low
0	$\overline{\text{OVR\_DDC\_SDA}}$	Override SDA data pins in IIC_DDC bus 1= release SDA data 0= force SDA data low

As shown in the above table, bits GRP_I2C_CTRLW[3:0] allow the override the output levels of the SCL clock and SDA data pins for both I²C busses. These overrides could be used to freeze one I²C bus while another is busy; for example, temporarily switch the I²C master module to the other bus for a transmit/receive. Or, these bits could be used to implement an I²C interface purely in software (a CPU-intensive task).

Loopback occurs when the master and slave are both switched to the same IIC or IIC_DDC bus. Under the control of software, the master can receive data transmitted by the slave; and the slave can receive data transmitted by the master.

I²C Control READ Register

name: GRP_I2C_CTRLR
index: 4Bh, read only
size: 8 bit
function: This register enables the I²C serial control interface. The bit contents of the GRP_I2C_CTRLR Control register are shown in Figure 79.

Table 80. GRP_I2C_CTRLR Control Register *(1 of 2)*

Bit	Function	Detail
7	I2CMA_SEL	1 = select IIC bus for master module 0 = select IIC_DDC bus for master module
6	I2CSL_SEL	1 = select IIC bus for slave module 0 = select IIC_DDC bus for slave module
5	I2CMA_DONE	I²C master module DONE. Read-only value from GRP_I2C_MCTRLR
4	I2CSL_DONE	I²C slave module DONE. Read-only value from GRP_I2C_SCTRLR

Table 80. GRP_I2C_CTRLR Control Register *(2 of 2)*

Bit	Function	Detail
3	$\overline{\text{RD\_IIC\_SCL}}$	Read SCL clock signal in IIC bus (read only)
2	$\overline{\text{RD\_IIC\_SDA}}$	Read SDA data pins in IIC bus (read only)
1	$\overline{\text{RD\_DDC\_SCL}}$	Read SCL clock signal in IIC_DDC bus (read only)
0	$\overline{\text{RD\_DDC\_SDA}}$	Read SDA data pins in IIC_DDC bus (read only)

As shown in the above table, bits GRP_I2C_CTRLR[3:0] provide for reading the pin values of the four I^2C pins. These bits could be used to implement an I^2C interface purely in software (a CPU-intensive task). These bits also provide support for VESA monitor DDC1 signalling.

I²C Master Module Software Interface

The software interface to the I²C master block is via two VGA Graphics Extension registers: GRP_I2C_MCTRL (index 4Eh) and GRP_I2C_MDATA (index 4Fh). Both of these registers behave differently when read and written so the interface actually consists of two write-only registers and two read-only registers.

In general it is necessary to access both the GRP_I2C_MCTRL and GRP_I2C_MDATA registers at least once per byte of data transmitted / received. The order of reading the GRP_I2C_MCTRL and GRP_I2C_MDATA registers is important. The data received in the GRP_I2C_MDATA register should not be considered valid until after the DONE bit has been read as "1" in the GRP_I2C_MCTRLR register. Similarly the TRAN or RECV bits in the GRP_I2C_MCTRLW register (start to transmit / receive data) should not be written "1" until after the relevant data has been read from or written to the GRP_I2C_MDATA register.

Note that the I²C master module detects but does not recover from arbitration loss, non-acknowledgment of data, etc. It is the responsibility of the system software to handle any problems like this that occur and resubmit any failed action to the I²C master module.

The software interface to the I²C master module is via the following registers.

I²C Master Data Register

name: GRP_I2C_MDATA
index: 4Fh, read/write
size: 8 bit
function: This register supplies the contents of the input data buffer containing data received over the I²C bus. The bit contents of the GRP_I2C_MDATA register are shown in Table 81.

Table 81. I²C Master Receive Data GRP_I2C_MDATA

Bit	Function	Detail
7:0	DATA	Received data byte/ data byte to transmit

If this register is read while data reception is still in progress the data read will be incorrect (not all bits will be shifted in yet). It is the responsibility of the system software to wait until the data reception on the I²C bus is complete before it considers the data read from the GRP_I2C_MDATA register to be correct. This can be done by examining the DONE bit in the GRP_I2C_MCTRLR register and waiting until it is "1."

When written, the GRP_I2C_MDATA register supplies the data to be transmitted over the I²C bus. If this register is written while data transmission is still in progress the data transmitted will be incorrect. It is the responsibility of the system

software to wait until the data transmission on the I²C bus is complete before it attempts to write to the GRP_I2C_MDATA register. This can be done by examining the DONE bit in the GRP_I2C_MCTRLR register and waiting until it is "1" before writing to the GRP_I2C_MDATA register.

Please note that the registers used for data reads and writes are the same, writing data to the GRP_I2C_MDATA register will overwrite any unread data from the previous I²C master module reception. Similarly, any data written to the GRP_I2C_MDATA register could be corrupted by an I²C receive operation. Also, if two consecutive I²C transmit operations are performed with the same data, you still need to rewrite the required data to the GRP_I2C_MDATA register between the two I²C transmissions in order to remain compatible with future hardware revisions.

Finally, note that upon reset the initialization of the contents of this register is not defined. Assume that after a reset this register contains junk data until data has been written to it by the system software or until data has been loaded into it by an I²C master module receive operation.

I²C Master Control Read Register

name: GRP_I2C_MCTRLR
index: 4Eh, read only
size: 8 bit
function: This register supplies status information about the progress / completion of any I²C bus transaction. The bit contents of the GRP_I2C_MCTRLR registers are shown in Table 82.

Table 82. I²C Master Read Control Register GRP_I2C_MCTRLR

Bit	Function	Detail
7	DONE	0 = I²C master module busy 1 = I²C master module finished current operation
6	LOST	0 = Arbitration not lost 1 = Arbitration lost
5	RACK	Value of last I²C acknowledge bit received
4	MAST	0 = I²C master module not bus master 1 = I²C master module is bus master
3:0	BITS	Number of bits processed. See Table 83

GRP_I2C_MCTRLR[7]

The DONE bit is set to "1" whenever the I²C master module has finished its current operation and is ready for another request from the system software. It is set to "0" whenever the I²C master module is busy servicing a request. This bit can be used by the system software to determine whether the I²C master module was the source of an interrupt (interrupts enabled) or to check whether the I²C master finished the current request (software polling, interrupts disabled). This bit is used in the generation of the I²C interrupt (it is ANDed with the INTE interrupt enable bit

together with any outstanding requests in the GRP_I2C_MCTRLW register) but it is set independently of the value of INTE (interrupt enable). It is set only when all of the request bits (TSTA, TRAN, RECV and TSTO) in the GRP_I2C_MCTRLW register are cleared.

The DONE bit is set under the following conditions.

- TSTA = 1: (Start request) The DONE bit is set after the start token is transmitted.
- TRAN = 1: (Transmission request) The DONE bit is set after the data is transmitted and the acknowledge bit is received.
- RECV = 1: (Receive request) The DONE bit is set after the data is received and the acknowledge bit is transmitted. Note that it is impossible to choose the value of the acknowledge bit dynamically based on the data received.
- TSTO = 1: (Stop request) The DONE bit is set after the stop token is transmitted onto the I^2C bus.
- LOST = 1: (Arbitration lost) The DONE bit is also set whenever the I^2C bus arbitration is lost. This will clear any outstanding requests in the GRP_I2C_MCTRLW register, i.e. losing bus arbitration clears the TSTA, TRAN, RECV and TSTO bits.

GRP_I2C_MCTRLR[6] (LOST)

Arbitration lost. If the previous action was an I^2C transmit, the LOST bit is set when the I^2C master module loses the arbitration process. The I^2C arbitration process can also be lost during a receive action if the master module tries to send a “1” acknowledge and another contending master tries to send a “0” acknowledge. It is also possible to lose arbitration during the transmission of a stop token onto the I^2C bus.

GRP_I2C_MCTRLR[5] (RACK)

Receive acknowledge. The RACK bit contains the value of the acknowledge bit received or transmitted during the last transmit / receive.

GRP_I2C_MCTRLR[4] (MAST)

Master bus. The MAST bit indicates whether the I^2C master module is actually the current I^2C bus master (1=master). Once this module becomes bus master it will remain so until either the system software tells it to release ownership by writing a “1” to the TSTO field of the GRP_I2C_MCTRLW register or until it releases ownership due to an arbitration loss.

GRP_I2C_MCTRLR[3:0] (BITS)

Number of bits processed. The BITS value indicates the number of the bit currently being processed, as shown in Table 83. Do not assume that an I^2C transaction is finished until this field has the value of 9 (“1001”). However, the correct way to detect the completion of an I^2C transaction is when the DONE status bit = 1. This field is actually the contents of an internal counter within the I^2C master module that counts the number of bits transmitted / received. Thus, in the case of arbitration loss it will indicate at which bit the arbitration was lost.

Table 83. GRP_I2C_MCTRLR Current Bit

Bits[3:0]	Value	Meaning
0000	0	1st bit in progress
0001	1	2nd bit in progress
...	...	.....
0110	6	7th bit in progress
0111	7	8th bit in progress
1000	8	Ack bit in progress
1001	9	All bits done

Below, Table 84 shows an example of possible settings of the GRP_I2C_MCTRLR bits.

Table 84. Example of Reading GRP_I2C_MCTRLR Register

DONE	LOST	RACK	MAST	BITS				Description
1	0	X	0	X	X	X	X	Idle, not bus master
0	0	X	1	0	1	0	0	Currently bus master Transmit/receive in progress 3 bits done.
0	0	1	1	1	0	0	1	Currently bus master Transmit/receive done If receive, ACK was 1.
0	0	X	0	0	0	0	0	Transmit/receive in progress. No bits done, not bus master, but waiting for bus mastership.
1	1	X	0	0	0	0	1	Not bus master Lost arbitration on bit 2 of last transmit.

I²C Master Control Write Register

name: GRP_I2C_MCTRLW
index: 4Eh, write only
size: 8 bit
function: The GRP_I2C_MCTRLW register controls the operation of the I²C master module. Writing to this register initializes the DONE and BITS fields in the GRP_I2C_MCTRLR register. Initializing the BITS field in the GRP_I2C_MCTRLR register would destroy any I²C bus transaction in progress, so do not write to the GRP_I2C_MCTRLW register while the I²C master module is busy transmitting or receiving on the I²C bus.

It is the responsibility of the system software to ensure that this register is never written to except when the I²C master module is idle (in-

dicated by MAST = 0 or BITS = 1001 in the GRP_I2C_MCTRLR register).

All the bits in this register except for TACK are all set to “0” upon reset. This reset initialization places the I^2C master module in an idle state immediately after reset. A TACK value of “0” indicates a not-acknowledged state.

The bit contents of the GRP_I2C_MCTRLW register are shown in Table 85.

Table 85. I2C Master Write Control Register GRP_I2C_MCTRLW

Bit	Function	Detail
7	unused	must write 0 here
6	400K	0 = 100kHz mode 1 = 400kHz mode
†5	TSTO	0 = not transmit a stop 1 = transmit a stop
†4	TSTA	0 = not transmit a start 1 = transmit a start
3	INTE	0 = disable interrupts (software polling) 1 = interrupt enable
2	TACK	Value of last I^2C acknowledge bit received
†1	RECV	0 = no receive data 1 = receive data
†0	TRAN	0 = no transmit data 1 = transmit data
† Valid until the action requested is finished, then automatically reset		

GRP_I2C_MCTRLW[7] This bit is currently unused in the I^2C master module, however the system software must always write “0” here to allow for possible future expansions.

RP_I2C_MCTRLW[6] (400K) 400kHz mode. When set to “1,” the 400K control bit causes the I^2C master module to transmit at the full speed 400kHz clock rate. When set to “0,” it uses a more conservative 100kHz clock rate (actually the rate is only 80kHz but the timings are set up to satisfy the I^2C 100kHz mode spec).

GRP_I2C_MCTRLW[5] (TSTO) Transmit a stop. When set to “1,” the TSTO control bit causes the I^2C master module to transmit a stop token onto the I^2C bus. If this I^2C master module is not bus master, then this control bit will be ignored. Note that unlike TSTA and TRAN, it is not possible to combine TSTO with TRAN or RECV. A separate command access by the system software is required to transmit a stop to the I^2C bus. Thus if TRAN, RECV, or TSTA are set to “1” then TSTO must be “0.”

GRP_I2C_MCTRLW[4] (TSTA) Transmit a start. When set to "1" the TSTA control bit causes the I²C master module to transmit a start token onto the I²C bus. If the bus is busy, it will wait for the bus to become free before transmitting the start token. If this I²C master module is already bus master, it will transmit the start token immediately. If the TRAN control bit is also set to "1," the I²C master will transmit the contents of the GRP_I2C_MDATA register immediately after the start token. The combination where both the TSTA and TSTO bits are written as "1" is illegal and the operation of the I²C master module is not defined.

GRP_I2C_MCTRLW[3] (INTE) Interrupt enable. When set to "1," the INTE control bit causes the interrupt output from the I²C master module to go to "1" whenever the contents of the DONE field goes to "1" (i.e. at the end of a transmit or receive or arbitration loss). If this bit is set to "0," the system software must service the I²C bus by means of software polling. This action will not result in the loss of any data but may cause the I²C bus stand idle until the system software next polls this module and sets up the next request. The actual interrupt is generated by ANDing this bit with the DONE bit of the GRP_I2C_MCTRLR Read register.

GRP_I2C_MCTRLW[2] (TACK) Acknowledge to transmit. The TACK bit contains the value that the I²C master module will transmit whenever it needs to transmit an acknowledge bit. (i.e. whenever it needs to acknowledge a I²C reception.)

GRP_I2C_MCTRLW[1] (RECV) Receive data. When set to "1," the RECV bit will start the process of receiving the contents of the GRP_I2C_MDATA register from the I2C bus. Note that the I2C master module must already be bus master (MAST = 1 in the GRP_I2C_MCTRLR register). Note also that since the I2C spec does not allow any circumstance where the I2C master follows a start with anything other than a transmit, this module does not support combining the action of the TSTA bit with the RECV bit as it does with TSTA and TRAN to enable both a start and transmit. The combination where both the RECV and TSTA bits are written as "1" is illegal and the operation of the I2C master module is not defined if this happens.

GRP_I2C_MCTRLW[0] (TRAN) Transmit data. When set to "1," the TRAN bit will start the process of transmitting the contents of the GRP_I2C_MDATA register onto the I2C bus. Note that the I2C master module must already be bus master (MAST = 1 in the GRP_I2C_MCTRLR register) or the system software must write a "1" to the TSTA bit at the same time to initiate bus mastership. Note also that the situation where both this bit and the RECV bit are written as "1" is illegal and will produce some undefined action in the I2C master module.

Below, Table 86 shows an example of possible settings of the GRP_I2C_MCTRLW bits.

Table 86. Example of writing to GRP_I2C_MCTRLW

unused	400K	TSTO	TSTA	INTE	TACK	RECV	TRAN	Description
0	1	0	0	0	0	0	0	Do nothing.
0	0	0	0	0	0	0	0	Do nothing, in 100kHz mode.
0	1	0	1	1	0	0	1	Transmit data preceded with start token. Generate interrupt when done.
0	1	0	0	0	1	1	0	Receive data into Acknowledge. No interrupt when finished.
0	1	1	0	0	0	0	0	Transmit a stop token to release I²C bus
1	1	0	0	0	0	0	0	Illegal: Unused bit must not be 1
0	1	0	0	0	0	1	1	Illegal: RECV and TRAN = 1
0	1	1	1	0	0	0	0	Illegal: TSTA and TSTO = 1
0	1	1	0	0	0	1	0	Illegal: TSTA and RECV = 1

Example Bit Setting Sequence

The following is a sample interaction between system software and the I²C master module that results in the I²C master module writing the value 3Fh to the 7-bit address 1Ah on the I²C bus.

1 First the system software checks for an idle condition on the I²C bus by reading the GRP_I2C_MCTRLR register, as illustrated below.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	1	X	X	0	1	0	0	1

Note, the I²C master module is idle when MAST = 0. However, this does not mean that the I²C bus is idle; the master module simply does not own the I²C bus and is not transmitting / receiving on it at the moment.

If MAST = 1, the GRP_I2C_MCTRLR register might contain the settings shown below.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	1	0	X	1	1	0	0	1

Here, the I²C master module currently owns the I²C bus (MAST = 1) but is idle at the moment (DONE = 1). Any value other than 1 in DONE would mean that the I²C master module is currently processing a transmit or receive at the moment. The BITS field will indicate which bit it is currently working on: 0000 = MSB, 0111 = LSB, 1000 = ACK bit, etc.

2 The system software can set up the first data to be transmitted by writing 34h to the GRP_I2C_MDATA register. The value 34h is calculated by combining the 7 bit address 1Ah in the 7 MSBs with a 0 (write) in the R/W (LSB) field, as shown below.

	MSB							LSB
GRP_I2C_MDATA	0	0	1	1	0	1	0	0

3 Next the system software can start the transmit process by writing the following data to the GRP_I2C_MCTRLW register, as shown below.

	unused	400K	TSTO	TSTA	INTE	TACK	RECV	TRAN
GRP_I2C_MCTRLW	0	1	0	1	1	0	0	1

This configuration requests the I²C master module to transmit a start token followed by the data in the GRP_I2C_MDATA register, generating an interrupt when the acknowledge has been received for this transmission. Note that the system software must write "0" to the unused field to allow future extensions.

4 The I²C transmit will now proceed without interaction with the system software. If the system software had not enabled the interrupt, it would need to monitor the progress of the transmit by reading the GRP_I2C_MCTRLR register and taking further action when the transmit finishes. If software polling is used instead of interrupts and if the I²C master module is not serviced for a long period of time after the current transmit / receive is complete, the I²C bus will be idled thereby locking out any other potential I²C bus masters until the system software gives the I²C master module a new command. No data would be lost; just I²C bus bandwidth would be wasted.

Example Bit Setting Sequence

The following is the sequence of events the system software would see if it was monitoring the status of the I²C master module by reading the GRP_I2C_MCTRLR register.

1 Initially the I²C master module would be waiting for an opportunity to transmit the start token (another I²C master module might currently own the bus). While this is the case, the GRP_I2C_MCTRLR register contains the following.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	0	0	X	0	0	0	0	0

2 The “0” in MAST indicates that the I^2C master module has not yet gained ownership of the I^2C bus. Once the I^2C master has ownership (MAST = 1) of the I^2C bus and transmitted the start token, the GRP_I2C_MCTRLR register contains the following.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	0	0	X	1	0	0	0	0

3 The “0000” value in BITS indicates that the MSB of the transmit is currently in progress. As more bits are transmitted the value in BITS will increase, as shown below.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	0	0	X	1	0	0	1	1

4 The “0011” value in BITS indicates that 2 bits have been transmitted and that the third bit is in progress. If the I^2C master module loses the bus arbitration process to any other master at any time during transmission, the GRP_I2C_MCTRLR register might read as follows.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	1	1	X	0	0	1	0	0

5 The I^2C master module lost the bus arbitration process (LOST = 1) while transmitting the 4th bit (BITS = 0100) in the current word. Note that once the I^2C master module lost the arbitration process, bus ownership is lost and the MAST bit is reset to “0.” Also when the arbitration process ends, the current transmission ends and the DONE bit is set, thereby generating an interrupt if the INTE bit was set. Note that is possible to lose an arbitration while transmitting an ACK bit for a reception so that the following bit pattern is possible.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	1	1	X	0	1	0	0	0

6 Assuming that no arbitration loss occurs, the I^2C master module would finish the LSB of the transmission and start receiving the acknowledge bit, as shown below.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	0	0	X	0	1	0	0	0

7 When the acknowledge bit is received, an interrupt will be generated and the transmit is complete.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	1	0	1	1	1	0	0	1

8 The I²C master module is now ready for the next action. The system software would now write the value 3Fh into the GRP_I2C_MDATAW register for the next I²C transmit.

	MSB							LSB
GRP_I2C_MDATA	0	0	1	1	0	1	0	0

9 The system software would again initiate the transmit by writing the following to the GRP_I2C_MCTRLW register.

	unused	400K	TSTO	TSTA	INTE	TACK	RECV	TRAN
GRP_I2C_MCTRLW	0	1	0	0	1	0	0	1

This configuration is the same as before except the TSTA bit is now “0,” indicating no start token is sent.

10 The system software is interrupted at the end of the transmit; assuming no arbitration loss, the GRP_I2C_MCTRLR register again reads as follows.

	DONE	LOST	RACK	MAST	BITS			
GRP_I2C_MCTRLR	1	0	1	1	1	0	0	1

11 The system software requests the sending of a stop token by writing the following to the GRP_I2C_MCTRLW register.

	unused	400K	TSTO	TSTA	INTE	TACK	RECV	TRAN
GRP_I2C_MCTRLW	0	1	0	1	0	0	0	0

Note that this time the INTE bit is “0,” disabling interrupts. The I²C master module would proceed to transmit a stop token but would not inform the system software with an interrupt when complete.

I²C Slave Module Software Interface

The software interface to the I²C master block is via 2 VGA graphics extension registers: the GRP_I2C_SCTRL register (index 4Ch) and the GRP_I2C_SDATA register (index 4Dh). The GRP_I2C_SDATA register is read-only (since this module is a slave receiver only and cannot transmit). The GRP_I2C_SCTRL register is both read and write and behaves differently in each case. Consequently, the software interface actually consists of one write-only register and two read-only registers.

In general, it is necessary to access both the GRP_I2C_SCTRL and GRP_I2C_SDATA registers at least once per byte of data received. The order of reading registers is important; the data received in the GRP_I2C_SDATA register should not be considered valid until after the DONE bit has been read as "1" in the GRP_I2C_SCTRLR register. Similarly the GRP_I2C_SCTRLW register should not be written until after the relevant data has been read from the GRP_I2C_SDATA register because writing to this register could initiate another data reception, thereby overwriting the data in the GRP_I2C_SDATA register.

Note that this I²C slave module is capable of losing the arbitration process just like the I²C master module. However this can only happen during transmission of the acknowledge bit. It is the responsibility of the system software to handle any problems like this that occur.

The software interface to the I²C slave module is via the following registers.

I²C Slave Data Register

name: GRP_I2C_SDATA
index: 4Dh, read only
size: 8 bit
function: The GRP_I2C_SDATA register supplies the contents of the input data buffer containing data received over the I²C bus. If this register is read while data reception is still in progress, the data read will be incorrect (not all bits will be shifted-in yet). It is the responsibility of the system software to wait until the data reception on the I²C bus is complete before considering the data read from the GRP_I2C_SDATA register to be correct. Data reception is complete when the DONE bit in the GRP_I2C_SCTRLR register contains a "1."

The bit contents for the GRP_I2C_SDATA register are shown in Table 87.

Table 87. I²C Slave Receive DATA GRP_I2C_SDATA

Bit	Function	Detail
7:0	DATA	received data byte

I²C Slave Control Read Register

name: GRP_I2C_SCTRLR
index: 4Ch, read only
size: 8 bit
function: The GRP_I2C_SCTRLR register supplies status information about the progress / completion of any I²C bus transaction. A break must be enabled before the I²C slave can supply status information; see Table 91 on page 155.

The bit contents for the GRP_I2C_SCTRLR register are shown in Table 88.

Table 88. I²C Slave Read Control Register GRP_I2C_SCTRLR

Bit	Function	Detail
7	DONE	0 = module is still busy with last request 1 = finished last request
6	LOST	0 = arbitration not lost 1 = arbitration lost
5	STOP	0 = no stop token received since last read 1 = stop token received since last read
4	MAST	0 = bus idle 1 = some module is I²C master
3:0	BITS	Current bit. See Table 89.

GRP_I2C_SCTRLR[7] (DONE)

The DONE bit is set to "1" whenever the I²C slave module has finished its current operation and is ready for another request from the system software. It is "0" whenever the I²C slave module is busy servicing a request. This bit can be used by the system software to determine whether the I²C slave module was the source of an interrupt (interrupts enabled) or to check whether the I²C slave has finished its current request (software polling, interrupts disabled). This bit is used in the generation of the I²C interrupt (it is ANDed with the INTE interrupt enable bit) but it is set independently of the value of INTE (interrupt enable). If more than one break request is included in the write to the GRP_I2C_SCTRLW register, the DONE bit will be set when a break conditions occurs. For each of the break conditions, the DONE bit will be set under the following conditions.

- BSTA = 1: (Break on start) The DONE bit will be set after the start token is received.
- BRCV = 1: (Break on receive) The DONE bit will be set after 8 data bits are received. This request facilitates the system software dynamically, determining the value of the ACK bit to be transmitted based on the data received.

- BACK = 1: (Break on acknowledge) The DONE bit will be set after the acknowledge bit has been transmitted. This request facilitates the system software predetermining the value of the ACK bit before any data has been received.
- BSTO = 1: (Break on stop) The DONE bit will be set after the stop token is received.

Finally, the DONE bit will also be set if the I²C bus arbitration is lost.

GRP_I2C_SCTRLR[6] (LOST) Arbitration lost. The LOST bit is set to 1 whenever the I²C slave module loses the I²C arbitration process. Note, since this module is only capable of I²C receptions, it can only lose the arbitration process during the transmission of the acknowledge bit

GRP_I2C_SCTRLR[5] (STOP) Stop token received. The STOP bit indicates that a stop transaction was received on the I²C bus. This bit is cleared whenever the GRP_I2C_SCTRLW register is written. Therefore if this bit is set, a stop transaction was received since the last GRP_I2C_SCTRLR write.

GRP_I2C_SCTRLR[4] (MAST) Some module is master. A MAST bit value of "1" indicates that some I²C master module is the current bus master. In order to detect a reception or stop transaction on the I²C bus, this bit must be set. If this module is reset (MAST = 0) while some other I²C master module is driving the I²C bus, this module will not be able to receive data or detect a stop transaction until after the current I²C transaction is completed.

GRP_I2C_SCTRLR[3:0] (BITS) Current bit being processed. The BITS value indicates the number of the bit currently being processed during a receive, as shown in Table 89. Do not assume that an I²C reception is finished until this field contains the value of 9 ("1001"). However, the correct way to detect the completion of an I²C transaction when the DONE status bit = 1. This field is actually the contents of an internal counter within the I²C master module that counts the number of bits received.

Table 89. GRP_I2C_SCTRLR Current Bit

Bits[3:0]	Value	Meaning
0000	0	1st bit in progress
0001	1	2nd bit in progress
...	...	.....
0110	6	7th bit in progress
0111	7	8th bit in progress
1000	8	Ack bit in progress
1001	9	All bits done

Below, Table 90 shows an example of possible settings of the GRP_I2C_MCTRLR bits.

Table 90. Examples of Reading GRP_I2C_SCTRLR

DONE	LOST	STOP	MAST	BITS				Description
1	X	0	0	X	X	X	X	Idle, not bus master. This module idle.
0	X	0	0	X	X	X	X	I²C bus idle. This module busy (waiting for start, etc.).
0	0	0	1	0	1	0	0	I²C bus not idle. This module busy (receive in progress, 3 bits done).
1	0	0	1	1	0	0	1	I²C bus not idle. This module done.
1	1	0	1	1	0	0	0	I²C bus not idle. Lost arbitration on ACK bit of last receive
1	0	1	1	X	X	X	X	I²C bus not idle. Stop received since last read.

I²C Slave Control Write Register

name: GRP_I2C_SCTRLW
index: 4Ch, write only
size: 8 bit
function: The GRP_I2C_SCTRLR register controls the operation of the I²C slave module. Writing to this register initializes the DONE and BITS fields in the GRP_I2C_SCTRLR register. Initializing the BITS field in the GRP_I2C_SCTRLR register would destroy any I²C bus transaction in progress, so this register must not be written to while the I²C slave module is busy. It is the responsibility of the system software to ensure that this register is never written to except when the I²C master module is idle (indicated by DONE = 1 in the GRP_I2C_SCTRLR register).

All the bits in this register except for TACK are set to zero upon reset. This reset initialization places the I²C slave module in an idle state immediately after reset. The TACK value of "1" is transmits an ACK bit of "1," which indicates a "not acknowledge" state. With these reset values in the GRP_I2C_SCTRLW register, the I²C slave module will receive data from the I²C bus as it appears but it will never transmit any I²C acknowledge bits and it will never generate an interrupt.

A break (BSTO, BSTA, BRCV, or BACK) must be enabled so that the slave state machine operates properly. A break acts as an interrupt causing the slave to hold SCLK low, stretching the cycles until released. The next start or stop will reset the slave.

The bit contents of the GRP_I2C_SCTRLW register are shown in Table 91.

Table 91. I^2C Slave Write Control Register GRP_I2C_SCTRLW

Bit	Function	Detail
7:6	reserved	must write 0's here
5	BSTO	0 = no break after receive stop 1 = break after receive stop
4	BSTA	0 = no break after receive start 1 = break after receive start
3	INTE	0 = disable interrupt (software polling) 1 = interrupt enable
2	TACK	0 = value to transmit for ACK bit 1 = no ACK value to transmit
1	BRCV	0 = no break after receive data 1 = break after receive data
0	BACK	0 = no break after transmit ACK 1 = break after transmit ACK

GRP_I2C_SCTRLW[7:6] These bits are currently unused in the I^2C slave module, however the system software must always write "00" here to allow for possible future expansions.

GRP_I2C_SCTRLW[5] (BSTO) Break on stop. When set to "1," this control bit causes this module to generate a DONE bit/interrupt after the next stop token reception on the I^2C bus.

GRP_I2C_SCTRLW[4] (BSTA) Break on start. When set to "1," this control bit causes this module to generate a DONE bit/interrupt after the next start token reception on the I^2C bus.

GRP_I2C_SCTRLW[3] (INTE) Interrupt enable. When set to "1," this control bit causes the interrupt output from the I^2C slave module to go to "1" whenever the contents of the DONE field goes to "1." If this bit is set to "0," the system software must service this module by means of software polling. This process will not result in the loss of any data but may cause the I^2C bus idle until the system software next polls this module and sets up the next request. Note that the actual interrupt is generated by ANDing this bit with the DONE bit of the CTRLR read register.

GRP_I2C_SCTRLW[2] (TACK) ACK bit to transmit. This is the value that the I^2C slave module will transmit whenever it needs to transmit an acknowledge bit. (i.e. whenever it needs to acknowledge a I^2C reception.)

GRP_I2C_SCTRLW[1] (BRCV) Break on receive. When written with "1," this bit will cause this module to generate a DONE bit/interrupt after the next end of receiving 8 bits of data.

GRP_I2C_SCTRLW[0] (BACK) Break on acknowledge. When set to "1," the bit will cause this module to generate a DONE bit/interrupt after the next transmission of the acknowledge bit.

Determining Break Condition

You can combine several of the possible break requests into a single request to the I²C slave module; the interrupt will be generated whenever the first break condition occurs. If more than one break condition is specified, it is usually possible upon receipt of the interrupt to determine which break condition caused the interrupt by reading the GRP_I2C_SCTRLR register.

The BSTA (break on start) request sets MAST=1 and BITS=0000 in the GRP_I2C_SCTRLR register.

The BRCV (break on receive) request sets MAST=1 and BITS=1001 in the GRP_I2C_SCTRLR register. The data received can be read from GRP_I2C_SDATA, and the acknowledge bit to be transmitted can be controlled by writing to the TACK bit of the GRP_I2C_SCTRLW register.

The BACK (break on acknowledge) request sets MAST=1 and BITS=1001 in the GRP_I2C_SCTRLR register. The data received can be read from GRP_I2C_SDATA.

The BSTO (break on stop) request sets STOP=0 in the GRP_I2C_SCTRLR register, indicating that the interrupt was generated by a stop transaction on the I²C BUS. Using this break condition to generate interrupts which will be used by software is an inherently dangerous procedure. Unlike all the other "break on something" conditions, BSTO *does not* hold the I²C bus to wait for service from the software interrupt service routing (the I²C spec won't allow this). This is especially difficult in a polled (i.e. interrupts disabled) environment because a significant period of time might pass before the I²C module is polled. If this happens and if the interrupt is not serviced in time, the I²C bus could continue off into another transaction, the start of which would be missed by the BtV2115 I²C slave module.

Minimum Stop Transaction

The I²C spec guarantees a minimum of 1.3µs between a stop transaction and a following start transaction on the I²C bus, so if the software interrupt service routing can guarantee to service the interrupt in less than 1.3µs there won't be a problem. If there is any doubt that the software interrupt service routine can service a break on stop interrupt quickly enough then the break on stop condition *must* be combined with another break request (break on start, break on receive or break on acknowledge) to allow the I²C slave module to continue with the next reception without loss of data. When this is done the software interrupt service routine *must* check for the possibility that both break conditions (break on stop and the other break condition) have occurred when servicing the interrupt and take appropriate action if the second break condition has also occurred to avoid missing the second interrupt.

Note also that any of the above break conditions except for BSTO (Break on stop) will cause the I²C slave module to hold the I²C bus idle until the GRP_I2C_SCTRLW register is written again with a new request. It is the responsibility of the system software to ensure that the GRP_I2C_SCTRLW register is written to un-idle the I²C bus within a reasonable period of time. The system software is also responsible for ensuring that the I²C slave module is given a new re-

quest sufficiently fast after a break on stop condition. This is necessary because the break on stop does not idle the I²C bus, which could therefore begin another transaction while the I²C slave module is still waiting for system software service, causing the loss of data. Fortunately it is never necessary to detect a break on stop condition in the normal operation of the I²C; the system software could easily ignore the stop token and instead break on start to detect the start of the next I²C bus transaction. Also, the I²C spec requires that the I²C bus remain idle for 1.3µs after the transmission of a stop token, guaranteeing the system software at least 1.3µs to respond to the break on stop request.

Passive I²C Bus Snooper

It is possible to use the I²C slave module as a completely passive I²C bus snooper. Set the GRP_I2C_SCTRLW register to break at the various points of interest in the I²C transactions and set the TACK bit to "1." Data is received as in normal I²C slave module operations, but you can also see what value ACK bit was transmitted by looking at the value of the LOST bit in the GRP_I2C_SCTRLR register. If the ACK bit on the I²C bus was "0," the LOST bit will be set to "1" (since the I²C slave module was trying to send an ACK of 1 and saw an ACK of "0," it sets the LOST bit). Similarly an ACK bit of "1" on the I²C bus will cause the LOST bit in the GRP_I2C_SCTRLR register to remain at zero. To ensure that the I²C slave module is completely passive, all the "break on something" conditions must be serviced immediately. Polling or interrupting is not sufficient because the I²C slave module might halt the I²C bus if it is left waiting for software service for too long.

Below, Table 92 shows an example of possible settings of the GRP_I2C_SCTRLW bits.

Table 92. Example of writing to GRP_I2C_SCTRLW

unused		BSTO	BSTA	INTE	TACK	BRCV	BACK	Description
0	0	0	0	0	0	0	0	Do nothing
0	0	0	0	1	0	0	0	Pointless: no break requested so no interrupt will be generated
0	0	0	1	1	0	1	0	Interrupt upon reception of a start token or 8-bits of data
0	0	0	0	1	0	0	1	Receive 8 bits of data Acknowledge with ACK=0, then interrupt
0	0	1	1	1	0	0	0	Interrupt upon reception of start or stop token
1	0	0	0	0	0	0	0	Illegal: Unused bit not "0"

Example Bit Settings Sequence

The following is a sample interaction between system software and the I²C slave module during which the I²C slave module receives a series of bytes of data. After the reception of the first two bytes the system software is able to determine that this I²C module is not being addressed by the current I²C transaction and ignore the rest of the transaction.

1 First the system software checks whether the I²C bus is idle by reading the GRP_I2C_SCTRLR register.

	DONE	LOST	STOP	MAST	BITS			
GRP_I2C_SCTRLR	1	X	0	0	1	0	0	1

Here it can be seen that the I²C bus is idle (MAST = 0). Note that unlike the MAST bit in the GRP_I2C_MCTRLR register of the I²C master module, this bit indicates whether the I²C bus is idle. In the I²C master module, the MAST bit indicates whether the I²C master module is currently bus master.

If the I²C bus is currently busy (MAST = 1), the GRP_I2C_SCTRLR register might contain the following settings.

	DONE	LOST	STOP	MAST	BITS			
GRP_I2C_SCTRLR	1	0	0	1	1	0	0	1

Any value other than "1" in DONE would mean that the I²C slave module is currently processing a request. During data reception, the BITS field will indicate which bit is currently being received: 0000 = MSB, 0111 = LSB, 1000 = ACK bit being transmitted, etc.

2 The system software can then set up the first request to be processed by writing 0Ah to the GRP_I2C_SCTRLW register. This value will generate an interrupt once 8 bits of data has been received.

	unused		BSTO	BSTA	INTE	TACK	BRCV	BACK
GRP_I2C_SCTRLW	0	0	0	0	1	0	1	0

3 The I²C slave module can now proceed without interaction with the system software. If the system software had not enabled the interrupt, it would need to monitor the progress of the transmit by reading the GRP_I2C_SCTRLR register and taking further action when the reception finishes. If software polling is used instead of interrupts and if the I²C slave module is not serviced for a long period of time after the current receive is complete, the I²C bus will be idled thereby locking out any other I²C traffic until the I²C slave module interrupt is serviced. Except for the case of detecting stop tokens mentioned above (which can be avoided by correct design of the system software), no data would be lost because of delayed servicing of I²C slave module interrupts / DONE bits - just I²C bus bandwidth would be wasted.

The following is the sequence of events the system software would see if it was monitoring the status of the I²C slave module by reading the GRP_I2C_SCTRLR register.

Initially the I²C slave module would be waiting for an I²C master module to become bus master and transmit a start token. While this was the case, the GRP_I2C_SCTRLR register would contain the following.

	DONE	LOST	STOP	MAST	BITS			
GRP_I2C_SCTRLR	0	0	X	0	X	X	X	X

The "0" in MAST indicates the I²C bus is still idle. Once some I²C master gets ownership of the I²C bus and transmits the start token the GRP_I2C_SCTRLR register would read as follows.

	DONE	LOST	STOP	MAST	BITS			
GRP_I2C_SCTRLR	0	0	X	1	0	0	0	0

The "1" in MAST indicates that some I²C master module now owns the I²C bus. The "0000" value in BITS indicates that the MSB of the transmit is currently in progress. As more bits are received the value in BITS will increase, as shown below.

	DONE	LOST	STOP	MAST	BITS			
GRP_I2C_SCTRLR	0	0	X	1	0	0	1	1

The "0011" value in BITS indicates that 2 bits have been received and that the third bit is in progress. Once all 8 bits have been received, an interrupt will be generated. The I²C slave module will idle the I²C bus while waiting for service by the system software.

4 When responding to the interrupt, the system software can check the progress of the I²C reception by reading the GRP_I2C_SCTRLR register, as shown below.

	DONE	LOST	STOP	MAST	BITS			
GRP_I2C_SCTRLR	1	0	X	1	1	0	0	0

The "1" in the DONE bit indicates the I²C slave module has finished the last request. MAST = 1 indicates the I²C bus is busy. And BITS = 1000 indicates that all 8 bits of data have been received.

The system software can see the data received by reading the GRP_I2C_SDATA register. Then it can setup the I²C slave module to receive the next byte of data by again writing 0Ah to the GRP_I2C_SCTRLW register. This value will transmit an acknowledge bit of "0," then generate an interrupt once another 8 bits of data has been received, as shown below.

	unused		BSTO	BSTA	INTE	TACK	BRCV	BACK
GRP_I2C_SCTRLW	0	0	0	0	1	0	1	0

Once all 8 bits have been received, an interrupt will be generated. The I²C slave module will again idle the I²C bus while waiting for service by the system software.

5 When responding to this interrupt, the system software can again check the progress of the I²C reception by reading the GRP_I2C_SCTRLR register, as shown below.

	DONE	LOST	STOP	MAST	BITS			
GRP_I2C_SCTRLR	1	0	X	1	1	0	0	0

This indicates that the I²C slave module has done the last request, that the I²C bus is busy and that 8 bits of data has been received.

The system software can see the data received by reading the GRP_I2C_SDATA register. At this stage the system software can determine that the current I²C bus transaction is addressed to some other I²C slave module. It can instruct the I²C slave module to not acknowledge this byte by transmitting an acknowledge value of "1" and to ignore all further I²C activity until the next start token by writing 1Ch to the GRP_I2C_ SCTRLW register, as shown below.

	unused		BSTO	BSTA	INTE	TACK	BRCV	BACK
GRP_I2C_SCTRLW	0	0	0	1	1	1	0	0

The I²C slave module will ignore all transactions on the I²C bus until it next sees a start token. It will then generate another interrupt and will idle the I²C bus while it waits for service by the system software.

6 The system software can begin again with the new I²C transaction by again writing 0Ah to the GRP_I2C_ SCTRLW register, requesting an interrupt at the end of reception of the 8 bits of data.

GUI ACCELERATOR

GUI Theory Of Operation

To better understand this chapter, first read “Introduction” on page 97 and “Protected Mode Aperture” on page 101 in the “CPU Address Space Apertures” chapter.

The BtV2115 MediaStream controller consists of a programming interface section and an autonomous drawing engine capable of performing BITBLT, character rasterization, line draw, and STRETCHBLT. The command interface to the accelerator is completely queued through a buffer management and command pacing mechanism that requires essentially no reads from the GUI unless expressly transferring a pixmap from the frame buffer to the CPU. The GUI accelerator registers, commands, and parameters are transferred through the lower six MB of the protected mode aperture. See “GUI Command/Register Map” on page 112 for a discussion of the queueing mechanism.

Retained Bitmap Context

One of the acceleration features provided is the caching of bitmap context for the screen as well as for up to three off screen bitmaps. In addition, type information is retained for up to four bitmaps residing in host memory. Source and Destination pixmaps are efficiently identified with a command as selectors of these eight retained pixmap register contexts, see “Bitmap Context Registers” on page 173 for additional discussion on the 8 sets of bitmap context caching registers.

Raster Operation

The BtV2115 GUI has a two-operand raster-op engine which logically combines a source and a destination operand. There are sixteen possible combinations; the selection of the desired operation is controlled by a five bit field in the GUI configuration register, as shown in the ROP CODE field of Table 107. Each command has a COPY and a ROP version. When the COPY version is used, no logical combination is performed; the source is always copied to destination. When the ROP version is used, the logical operation selected by the control value is performed. One of the sixteen operations is copy, so this operation can be performed either way. In addition, the source can be any valid source bitmap: solid colors, patterns, host (mono and color), offscreen, and screen.

Communication of GUI commands, parameters, and register values are paced to the accelerator such that all register modifications which follow a command will be held until that command is completed. For example, the procedure to draw five line segments in five different colors is as follows (assuming the appropriate set-up). Five pairs of foreground register values and their line draw commands are sent. If the accelerator is idle when the first foreground register value is supplied, it will immediately load the foreground color register. When the accelerator receives the first line draw command and its associated parameters, it will go busy and remain that way until completion of the line draw. If the second foreground register value is received while the accelerator is busy, the value is held in the queue until the accelerator goes idle. At this time the new color value is loaded into the foreground register. When the accelerator receives the second line draw command and parameters (either from the CPU or the queue), it will go busy and commence drawing. This sequence is repeated until all five lines are drawn in five different colors. Since each command contains a selector for the bitmap cached context, the lines could also be directed to different destinations with the extents, pitches, and pixel depths automatically managed by the hardware. Notice that any drawing operation can reference host bitmaps as a destination however, only the BLT drawing operator referencing host bitmaps as a destination makes any useful sense.

CPU Addressing of GUI

The 32 bit CPU address controls the GUI queued interface, shown in Figure 18. When the following signal conditions are met, CPU address bits 21:2 are used as a selector to determine an action for the memory read or write command.

- CPU address bits 31:25 match the value of GUI_BASE[31:25] (PM_BASE field)
- GUI_BASE[24] (PM_ENABLE field) is set to 1
- CPU address bits 24:22 are set to zero

The meaning of the selector address bits 21:2 is found in Table 93, which shows that CPU address bits 21:16 define the GUI command to be performed. Table 93 further shows that there is a different definition for the lower 16 address bits depending on whether the command is a drawing operation, a register access, or a data transfer.

Table 94 maps the selector bits 21:16 to a specific GUI command and shows the types of commands supported by the accelerator.

Figure 18. Queued GUI Address Mapping

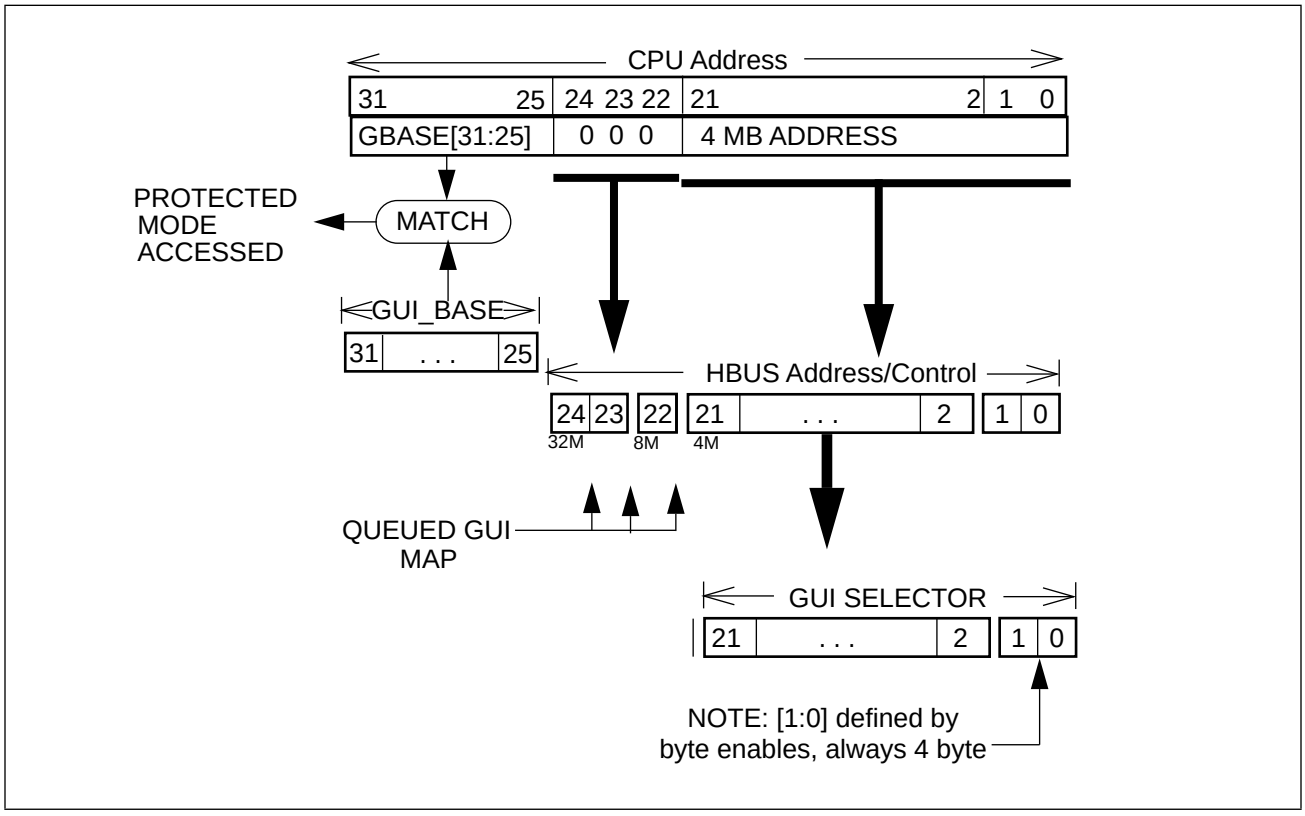


Table 93. Address Fields For GUI Accesses

CPU Address Bits	Field Description		
GUI Selector #	**Register Access**	**CPU-GUI Data Transfers**	**Drawing Operations**
21:16	GUI Command Number = 6'b000000	GUI Command Number = 6'b000001	GUI Command Number > 6'b000001
15:14	Unused	1-16K Data dword address roll field	Reserved
13:11			†Source Bitmap Selector
10:8			†Destination Bitmap Selector
7:5	Register Number		Parameter Count
4:2			Parameter Address Roll Field
1:0	Always set to zero, i.e. a 32 bit transfer		
† See Table 102 on page 174			

Notice in Table 93 that register accesses occupy the first 64KB of the GUI access space. This is the queued method of register access described above in which register loads are guaranteed to behave in strict sequential order with respect to the drawing operations. The register access commands are repeated at GBASE+4MB where they are not queued (a register write will occur immediately regardless of whether the accelerator is busy or idle). It is generally too dangerous to access GUI Accelerator registers using the non-queued address space while the accelerator is busy.

The non-queued method is used for various non-accelerator registers, which are defined in "GUI Command/Register Map" on page 112 of the "CPU Address Space Apertures" chapter. The non-queued GUIREG_MBA register controls the type of bit, byte, word, and dword flipping occurs throughout the Flippin map of the protected mode aperture and is relatively unrelated to GUI accelerator command queueing. However during driver initialization, this register is loaded with a value that controls whether GUI commands/register/data access are *big endian* or *little endian*.

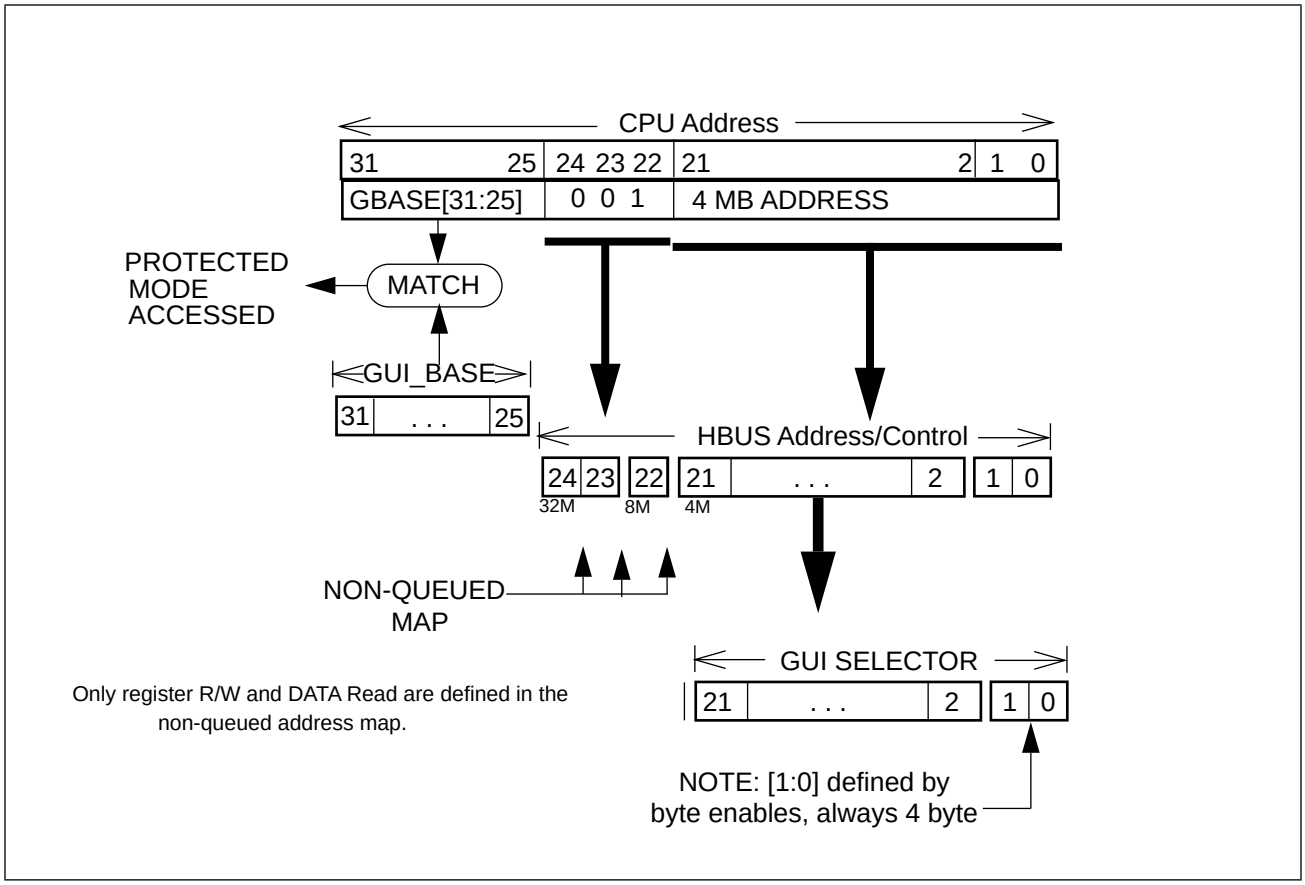
Shown in Table 93, a bit field is reserved within both drawing operation addresses and data word transfer addresses so that addresses from a REP MOVSD instruction can "roll" through the parameters. Thus you can keep the basic command address in the program, load it into EDI (a 32-bit offset register), then pass out 8K dwords of host-to-screen BLT data using the **"**ADDRESS ROLL**"** field. Thus a drawing command could be started at any one of eight possible addresses corresponding to one of the values of the PARAMETER ADDRESS ROLL FIELD.

Whether a dword is a command or a parameter is *not* encoded in this field. The two are distinguished by their placement in the transfer sequence relative to the previously issued command. Every dword comes over the bus in one cycle and contains an address and a data field. The first parameter (if any) accompanies the

first address field. For example, if a drawing command was issued with parameter count 2, the next command of whatever type will appear as the second dword following the drawing command.

Reads from the queued map are unsupported because they have the potential to hang the CPU for very long periods of time. Consequently, to read the final state of the destination XY increment register, the GUI must be completely idle before issuing a non-queued read using the access shown in Figure 19. Because of this hang potential and because a read operation to most of the first 4MB has no meaningful definition, you should always read through the non-queued interface. If reading GUI accelerator registers or screen-CPU data, first use the status bits in "GUI FIFO Depth Register" on page 114 to insure that the accelerator is completely idle or actually has read data ready before issuing the first read. Once the accelerator has the first read data word available, driver programs can use REP MOVS instructions to transfer data from screen to host CPU. The accelerator will pace the CPU via the ready line and will, in general, keep up with the CPU demand.

Figure 19. Non-queued GUI Address Mapping



For drawing operations, variable numbers of parameters are permitted (encouraged) using the count mechanism in the command (VL/PCI address bus bits). A zero or one parameter command takes one bus cycle to transfer the command and optional parameter. A two parameter command takes exactly two bus cycles, etc. In addition, a one parameter command takes two word locations in the VRAM FIFO, etc.

When commands are inserted into the VRAM queue software must know how many dwords will be occupied so that the queue depth can be accurately modeled to prevent overflow. Table 94, "GUI Commands," on page 167 shows how to compute the number of VRAM GUI queue dwords will be occupied by each command for a given number of parameters or data dwords in the GUI QUESIZE expression.

GUI Commands

The GUI commands are listed in Table 94 and defined in the following pages. The GUI Command Number is determined by CPU address bits [21:16] (see Table 93).

Table 94. GUI Commands *(1 of 2)*

GUI Command Number	Command	Raster Ops	No. of Parms.	Queued access	NON-queued access	Description
000000	RW REGISTER	n/a	n/a	WRITE	READ WRITE	Read or write one of 64 GUI registers. This command is also available through the non-queued map described above. Notice that all writes are queued in sequence with their corresponding GUI commands, e.g. BLT. NOTICE also that no reads are queued. (GUI QUESIZE = 2). See “RWREGISTER Command” on page 168.
000001	RWGUI DATA	n/a	n/a	WRITE	READ	This command allows data to pass to or from the host CPU. On the first cycle of a write the data bus contains a count of the number of dwords to be sent with subsequent RW GUI DATA commands. (GUI QUESIZE = #DWORDS+1). See “RWGUIDATA Command” on page 170.
000010-101111	RESERVED	n/a	n/a	NONE	NONE	
110000	STRETCH BLT	COPY	0-4	WRITE	NONE	(GUI QUESIZE = #parameters +1)
110001	TRAP (reserved)	COPY				
110010	LINE	COPY	0-2	WRITE	NONE	(GUI QUESIZE = #parameters +1)
110011	BITBLT	COPY	0-3	WRITE	NONE	(GUI QUESIZE = #parameters+1)
110100	STRETCH BLT	COPY TRANS-PARENT	0-4	WRITE	NONE	(GUI QUESIZE = #parameters +1)
110101	TRAP (reserved)	COPY TRANS-PARENT				

Table 94. GUI Commands *(2 of 2)*

GUI Command Number	Command	Raster Ops	No. of Parms.	Queued access	NON-queued access	Description
110110	LINE	COPY TRANS-PARENT	0-2	WRITE	NONE	(GUI QUESIZE = #parameters +1)
110111	BITBLT	COPY TRANS-PARENT	0-3	WRITE	NONE	(GUI QUESIZE = #parameters+1)
111000	STRETCH BLT	ROP	0-4	WRITE	NONE	(GUI QUESIZE = #parameters +1)
111001	TRAP (reserved)	ROP				
111010	LINE	ROP	0-2	WRITE	NONE	(GUI QUESIZE = #parameters +1)
111011	BITBLT	ROP	0-3	WRITE	NONE	(GUI QUESIZE = #parameters+1)
111100	STRETCH BLT	ROP TRANS-PARENT	0-4	WRITE	NONE	(GUI QUESIZE = #parameters +1)
111101	TRAP (reserved)	ROP TRANS-PARENT				
111110	LINE	ROP TRANS-PARENT	0-2	WRITE	NONE	(GUI QUESIZE = #parameters +1)
111111	BITBLT	ROP TRANS-PARENT	0-3	WRITE	NONE	(GUI QUESIZE = #parameters+1)

RWREGISTER Command

If we define a C constant to reference the queued register map, it could be coded:

```
#define QUEUED_REG 0x00000000
```

To access the GUIREG_MBA register through the queued register interface we could form an address such as:

```
((unsigned long *)(GBASE| QUEUED_REG | GUIREG_MBA))
```

We can also define a C constant to reference the non-queued register map:

```
#define NON_QUEUED_REG 0x00400000
```

To access the GUIREG_MBA register through the non-queued register interface we could form an address such as:

```
(unsigned long *)(GBASE| NON_QUEUED_REG | GUIREG_MBA)
```

Four drawing operators are defined for the GUI accelerator, namely BITBLT, RWGUIDATA, LINE, STRETCHBLT, discussed below.

BITBLT Command

Many different BLT operations can be very simply encoded with the BtV2115 BLT mechanism. Because the source and destination bitmap context selector within the command can point to either host or media buffer bitmaps, all operations can be specified quite succinctly. A BLT command can be accompanied by up to three parameters so that a maximum BLT command is defined in Table 95.

Table 95. BLT Command

Address Bus	Data Bus	GUI ACCELERATOR VALUE
BLT	DEST XY ADDRESS	COMMAND + P0
Parameter count and address roll field are don't cares	XY EXTENTS	P1
Parameter count and address roll field are don't cares	SOURCE XY ADDRESS	P2

The destination XY address specifies the XY offset into the bitmap of the first pixel to be written. The XY extents specify the width and height of the rectangle to be written.

The source XY address specifies the XY offset into the source bitmap. In the case of a bitmap in CPU memory, the X value is used to determine the alignment relative to the destination array. The Y value is not used. In the case of a pattern as the source, the X and Y values provide the starting offset into the pattern.

Patterns are stored in packed dword format, e.g. a monochrome 8x8 pattern is stored as two consecutive dwords in the frame buffer. The lowest-addressed element of each line of the pattern is the first pixel displayed for each line of the pattern. In other words, the right-most or least significant element becomes the left-most pixel.

Patterns are aligned relative to the offset frame buffer origin rather than to the starting destination address. The effect is that no matter where a pattern fill is started, the same pixels are drawn in the destination area as would have been if the pattern drawn had started at pixel (0,0) and continued to the destination area. The source (X,Y) address for a pattern fill is always taken to mean the (X,Y) address of the first pixel of the first line of the complete pattern.

When the source of the BLT is a host bit map, an RWGUIDATA command (GUI command number 000001) must immediately follow the BLT command which transfers up to 16K dwords to the GUI accelerator.

It should be noted that whenever an XY address is specified in a parameter or register it has the format shown in Table 96.

Table 96. XY Address Format

Bit(s)	Field Name	Description
31:28	Reserved	Must be set to zero
27:16	Y ADDRESS	Vertical index or row number
15:12	Reserved	Must be set to zero
11:0	X ADDRESS	Horizontal index or column number

RWGUIDATA Command

This command is used for host-to-screen or screen-to-host transfers only. Table 97 shows the form that the RWGUIDATA command takes for an N dword transfer.

Table 97. RWGUIDATA Command

Queue Dword No.	Bitmap Dword No.	Dword Use
1	N/A	RWGUIDATA_LENGTH (host-to-screen only)
2	1	Bitmap dword 1
3	2	Bitmap dword 2
...	...	
N	N-1	Bitmap dword N-1
N+1	N	Bitmap dword N

Table 98 defines the RWGUIDATA_LENGTH variable. It is a programming error to supply a RWGUIDATA_LENGTH word for screen-to-host transfers. After sending the command, the host simply reads the required number of dwords from anywhere within the 16K dword RWGUIDATA address roll space.

Table 98. RWGUIDATA_LENGTH

Bits	Description
31:27	Unused
26:16	# of Y scan lines in source (HOST) bitmap (n-1)
15:12	Unused
11:0	# of dwords per scan line in source (HOST) bitmap (n-1)

LINE Draw Command

The LINE command (defined in Table 99) supports both single and polyline capabilities.

Table 99. LINE Command

Address Bus	Data Bus	GUI ACCELERATOR VALUE
LINE	Ending XY point	COMMAND +P0
Parameter count and address roll field are don't cares	Starting XY point	P1

The end points of the line are both specified for single line commands. Only the end point is specified for polylines in which case the starting point is the previous endpoint.

Notice that the "LINE Control Register" on page 180 contains control bits that determine whether the first or last pixels of a line segment will be skipped or drawn. Also, this register determines several "styles" of Bresenham line algorithms, including whether the line can be drawn reversibly.

If no parameters are sent, i.e. parameter count field = 0 in the command, then the line drawing registers are assumed to have already been set up (end points, error terms, etc.). Notice that the Ending XY (P0) unconditionally gets copied to STARTINGXY (P1) after every line draw.

STRETCHBLT Command

This command is used to implement both STRETCH_BLT and STRETCH_DIB windows driver routines. Table 100 defines the STRETCH_BLT.

Table 100. STRETCHBLT Command

Address Buss	Data Bus	GUI ACCELERATOR VALUE
BLT	DEST XY ADDRESS	COMMAND + P0
Parameter count and address roll field are don't cares	DEST XY EXTENTS	P1
Parameter count and address roll field are don't cares	SOURCE XY ADDRESS	P2
Parameter count and address roll field are don't cares	SOURCE X EXTENT	P3

Notice that for the STRETCH_BLT command, parameter 3 can only specify an x extent by which to stretch since this command only handles horizontal stretching.

GUI Registers

Table 101 lists the GUI commands, their addresses, and descriptions. An address in the GUI Register is generated by ORing the GBASE value with the register address to create the offset address.

NOTE

The GUI enable bit in configuration register (GRP_CFG4[0]) must be set before these registers can be accessed.

Table 101. GUI Registers *(1 of 2)*

Queued Address (hex)	Non-queued Address (hex)	Name	Description
00000000	00400000	PARAMETER0	First parm of a command. This is a register RW view of the parameter which is normally sent as part of a command not as a register RW.
00000004	00400004	PARAMETER1	Second parm of a command.
00000008	00400008	PARAMETER2	Third parm of a command.
0000000C	0040000C	PARAMETER3	Fourth parm of a command
0000001C	0040001C	COMMAND	Last command value sent to GUI. This is primarily for diagnostic use
00000020	00400020	GUI_FG_COLOR	“Foreground Color Register” on page 178
00000024	00400024	GUI_BG_COLOR	“Background Color Register” on page 179
00000028	00400028	LINE_PATT	“Line Pattern Register” on page 181
0000002C	0040002C	GUI_DST_XY_INC	“Destination XY Increment Register” on page 179
00000030	00400030	GUI_CFG	“GUI Configuration Register” on page 176
00000034	00400034	BLT_CONTROL	“BLT Control Register (Direction)” on page 179
00000038	00400038	LINE_CONTROL	“LINE Control Register” on page 180
00000040	00400040	BMAP0_TYPE	“Bitmap Context Registers” on page 173
00000044	00400044	BMAP0_PITCH	“Bitmap Context Registers” on page 173
00000048	00400048	BMAP1_TYPE	“Bitmap Context Registers” on page 173
0000004C	0040004C	BMAP1_PITCH	“Bitmap Context Registers” on page 173
00000050	00400050	BMAP2_TYPE	“Bitmap Context Registers” on page 173

Table 101. GUI Registers *(2 of 2)*

Queued Address (hex)	Non-queued Address (hex)	Name	Description
00000054	00400054	BMAP2_PITCH	“Bitmap Context Registers” on page 173
00000058	00400058	BMAP3_TYPE	“Bitmap Context Registers” on page 173
0000005C	0040005c	BMAP3_PITCH	“Bitmap Context Registers” on page 173
00000060	00400060	BMAP4_TYPE	Only bits 29:24 are present
00000064	00400064	BMAP4_PITCH	Reserved (no pitch value)
00000068	00400068	BMAP5_TYPE	Only bits 29:24 are present
0000006C	0040006C	BMAP5_PITCH	Reserved (no pitch value)
00000070	00400070	BMAP6_TYPE	Only bits 29:24 are present
00000074	00400074	BMAP6_PITCH	Reserved (no pitch value)
00000078	00400078	BMAP7_TYPE	Only bits 29:24 are present
0000007C	0040007C	BMAP7_PITCH	Reserved (no pitch value)
00000080	00400080	BRES0_ADDR	“Bresenham 0, Address Register” on page 182
00000084	00400084	BRES0_ERR	“Bresenham 0, Error Register” on page 182
00000088	00400088	BRES0_K1	“Bresenham 0, K1 Register” on page 182
0000008C	0040008C	BRES0_K2	“Bresenham 0, K2 Register” on page 183
00000090	00400090	BRES0_INC1	“Bresenham 0, Increment 1 Registers” on page 183
00000094	00400094	BRES0_INC2	“Bresenham 0, Increment 1 Registers” on page 183
00000098	00400098	BRES0_LENGTH	“Bresenham 0, Length Register” on page 184
0000009C : 000000EC	0040009C : 004000EC		Reserved
don’t use	004000F0	GUIREG_MBA	“MBA Control Register” on page 125
don’t use	004000F4	GUIREG_DEPTH	See “GUI FIFO Depth Register” on page 114.
don’t use	004000F8	GUIREG_FIFO	See “GUI FIFO Register” on page 112.
don’t use	004000FC	GUIREG_DATA	Diagnostic use only. Within the chip, this code is used for the read/write GUI data command. Register transfers to/from this location are undefined in normal operation.

Bitmap Context Registers

BMAP[7:0]_TYPE, BMAP[7:0]_PITCH — There are eight pairs of 32-bit bitmap registers (BMAP0_TYPE and BMAP0_PITCH, BMAP1_TYPE and BMAP1_PITCH, etc.). Registers BMAP[3:0]_TYPE contain both type and offset information. Registers BMAP[7:4]_TYPE contain only type information. Registers BMAP[3:0]_PITCH contain pitch information. Registers BMAP[7:4]_PITCH are used for host bitmaps and solid fill functions and are not available for use (as indicated in Table 101).

The contents of the bitmap context registers are shown in Table 102. Table 103 defines the TYPE field. Table 104 defines the OFFSET field. Table 105 defines the PITCH field.

The GUI selector number is determined by the CPU address. Bits [13:11] and [10:8] specify either a source or destination bitmap selector, respectively (see Table 93 on page 164).

Table 102. Bitmap Context Registers

GUI Selector #	32 Bit Register	Upper 8 bits [31:24]	Lower 24 bits [23:0]
000	BMAP0_TYPE	TYPE	OFFSET
	BMAP0_PITCH	reserved	PITCH
001	BMAP1_TYPE	TYPE	OFFSET
	BMAP1_PITCH	reserved	PITCH
010	BMAP2_TYPE	TYPE	OFFSET
	BMAP2_PITCH	reserved	PITCH
011	BMAP3_TYPE	TYPE	OFFSET
	BMAP3_PITCH	reserved	PITCH
100	BMAP4_TYPE	TYPE	reserved
	BMAP4_PITCH	reserved	reserved
101	BMAP5_TYPE	TYPE	reserved
	BMAP5_PITCH	reserved	reserved
110	BMAP6_TYPE	TYPE	reserved
	BMAP6_PITCH	reserved	reserved
111	BMAP7_TYPE	TYPE	reserved
	BMAP7_PITCH	reserved	reserved

As shown in Table 103, the bitmap register type uses the saved context to tell the GUI accelerator how to treat the specified source or destination bitmap.

Table 103. Bitmap Register TYPE Field

Bit(s)	Description
7:6	Reserved
5:4	Pattern Size 00 = arbitrary (reserved in BtV2115) 01 = 8x8 10 =16x16 11 = 32x32
3	1 = Solid fill with background color
2	1 = Pattern
1	1 = Host
0	1 = Mono

Table 104. Bitmap Register OFFSET Field

Bits	Description
24:20	Reserved
19:0	Offset value. The bitmap's starting location in VRAM

Table 105. Bitmap Register PITCH Field

Bits	Description
31:12	Reserved
11:4	†Pitch pixel value: 0010 0000 = 512 0100 0000 = 1024 0010 1000 = 640 0101 0000 = 1280 0011 0010 = 800 0110 0100 = 1600 0011 1001 = 912 1000 0000 = 2048
3:0	Always 0
†Except for the 912, each line of bitmap/pattern must start on a dword boundary.	

Table 106 shows an example of how the bitmap context registers might be allocated for one instant in the life of a Windows driver. Notice contexts 7:4 are either allocated to host bitmaps or constant source (SRC) data patterns. To change the drawing context, all the driver has to do is send bitmap context register changes down the VRAM queued register write path within the sequence of drawing operations. Suppose the sequence looks like: BLT#1, queued write to context 1 offset, queued write to context 1 pitch, BLT#2. Suppose that both BLT#1 and BLT#2 use bitmap context 1 as their destinations (DST). BLT#1 could be directed to a screen

window while BLT#2 could be directed to an offscreen bitmap. Since drawing context is retained and since context changing register writes can be enqueued in sequence there is virtually no need for the Windows driver to synchronize to the actual current drawing state of the GUI accelerator. Therefore the driver can proceed far ahead of the hardware and allow full overlap of application execution with drawing setup and rasterization.

Table 106. Example Bitmap Context Register Allocation

Bit Map #	Selector #	Typical Allocation
0	000	Mono Pattern
1	001	Color Pattern
2	010	Screen Copy
3	011	Spare
4	100	Mono Host†
5	101	Color Host†
6	110	Solid Fill†
7	111	Free†
† Reserved for host and solid fill operations		

If the source (SRC) bitmap has the host bit set but also has the solid fill bit set or if a ROP is specified that doesn't use SRC data, then it is a programming error to supply host data. Unused SRC data will halt all data and command flow through the GUI FIFO, requiring a GUI_RESET to recover.

GUI Configuration Register

name: GUIREG_CFG
address: GBASE | QUEUED_REG | 00000030h, read/write
address: GBASE | NON_QUEUED_REG | 00000030h, read/write
size: 32 bit
function: The GUI configuration register controls the overall context of the GUI accelerator determining such functions as pixel color depth for line or blt, raster operation to be used, etc. The contents of the GUI configuration register are shown in Table 107. At reset all bits are cleared except [18:16], as indicated.

Table 107. GUI Configuration Register (GUIREG_CFG) *(1 of 2)*

Bit(s)	Description
31:23	reserved
22	Read only status bit for GUI RAM BIST (see GRP_CFG6[0] Table 20 on page 48) 1 = BIST complete on GUI result FIFO RAMs
21	Read only status bit for GUI RAM BIST (see GRP_CFG6[0] Table 20 on page 48) 1 = Control RAM BIST pass 0 = Control RAM BIST fail
20	Read only status bit for GUI RAM BIST (see GRP_CFG6[0] Table 20 on page 48) 1 = Data RAM BIST pass 0 = Data RAM BIST fail
19	Byte 3 write control 1 = preserve alpha channel 0 = modify alpha channel The Byte 3 write control bit should only be set in 32bpp mode to preserve the “alpha channel” data in byte 3 of 32-bit pixels. When set to 1, the bit disables byte enable 3 on all dwords the GUI writes to VRAM. To ensure predictable results, be sure that the GUI is idle and the result fifo is empty (GUIREG_DEPTH[19:0] =0) before the byte 3 write control bit is toggled.
18:16	Color depth 000= 1bpp (mono) reserved 001= 4bpp 010= 8bpp (reset value) 011= reserved 100= 16 bpp 101= reserved 110= 24 bpp 111= 32 bpp
15:14	reserved
13:12	Transparency control 00= no transparency 01= source transparency 10= reserved 11= reserved

Table 107. GUI Configuration Register (GUIREG_CFG) *(2 of 2)*

Bit(s)	Description		
11:9	reserved		
8	MONO_FLIP_ENABLE 0= pass monochrome bitmap bytes through unscathed. 1= reverse the bits within each byte in monochrome bitmaps before alignment.		
7:5	reserved		
4:0	ROP CODE		
	00h=	S-->D	source replaces dest
	01h=	S AND D --> D	AND source with dest
	02h=	S AND $\overline{D}$ --> D	AND source with not dest
	03h=	0 --> D	0s replace dest
	04h=	S OR $\overline{D}$ --> D	OR source with NOT dest
	05h=	S XNOR D --> D	XNOR source with dest
	06h=	$\overline{D}$ --> D	invert dest
	07h=	S NOR D --> D	NOR source with dest
	08h=	S OR D --> D	OR source with dest
	09h=	D --> D	no change
	0Ah=	S XOR D --> D	XOR source with dest
	0Bh=	$\overline{S}$ AND D --> D	AND NOT source with dest
	0Ch=	1 --> D	1s replace dest
	0Dh=	$\overline{S}$ OR D --> D	OR NOT source with dest
	0Eh=	S NAND D ->D	NAND source with dest
	0Fh=	$\overline{S}$ --> D	NOT source replaces dest
	10h:1Fh	RESERVED	

Foreground Color Register

name: GUI_FG_COLOR
address: GBASE | QUEUED_REG | 00000020h, read/write
address: GBASE | NON_QUEUED_REG | 00000020h, read/write
size: 32 bit
function: In all modes where the color depth is less than 32 bits per pixel, color values must be replicated to fill the foreground color register. For example, in 8bpp mode, the 8 bit color index to be used for foreground write operations must be replicated across all four bytes of the foreground register. In 24 bpp mode, byte3 must be set equal to byte0. The bit contents of the foreground color register are shown in Table 108.

Table 108. Foreground Color Register

Bit(s)	Description
31:0	foreground color

Background Color Register

name: GUI_BG_COLOR
address: GBASE | QUEUED_REG | 00000024h, read/write
address: GBASE | NON_QUEUED_REG | 00000024h, read/write
size: 32 bit
function: This value is used for writing pixels for solid fills or when in one of the patterned modes, and the pattern bit is a zero. If color depth is less than 32 bpp, then color values must replicated in the same manner as for the foreground register. The bit contents of the background color register are shown in Table 109.

Table 109. Background Color Register

Bit(s)	Description
31:0	Background color value

Destination XY Increment Register

name: GUI_DEST_XY_INC
address: GBASE | QUEUED_REG | 0000002Ch, read/write
address: GBASE | NON_QUEUED_REG | 0000002Ch, read/write
size: 32 bit
function: This value is used to increment the internal destination XY register after the completion of a BLT command, allowing a blt offset to be specified.

The bit contents of the destination XY increment register are shown in Table 110.

Table 110. Destination XY Increment

Bit(s)	Description
31:28	y sign extended
27:16	Signed y increment value
15:12	x sign extended
11:0	Signed x increment value

BLT Control Register (Direction)

name: GUI_BLT_CONTROL
address: GBASE | QUEUED_REG | 00000034h, read/write
address: GBASE | NON_QUEUED_REG | 00000034h, read/write
size: 32 bit
function: If BLT backwards is selected, the DST and SRC XY both point to the last pixel of the respective fields. Also both X and Y are decremented throughout the operation. The bit contents of the background color register are shown in Table 111.

Table 111. BLT Control Register

Bit(s)	Description
31:1	Reserved
0	Direction (1=backward), cleared after each operation.

LINE Control Register

name: GUI_LINE_CONTROL
address: GBASE | QUEUED_REG | 00000038h, read/write
address: GBASE | NON_QUEUED_REG | 00000038h, read/write
size: 32 bit
function: The line control register defines how several line drawing situations are handled. A single pixel line [(X0,Y0)=(X1,Y1)] with either SKIP_FIRST or SKIP_LAST is effectively a NO-OP (Bresenham registers are modified but not frame buffer, nor is the line pattern stepped). The bit contents of the line control register are shown in Table 112.

Table 112. Line Control Register *(1 of 2)*

Bits	Field	Description
31:11	unused	
10	[a.] XMAJOR	1 = X axis is the major axis 0 = Y axis is the major axis
9	[a.] SIGN_DX	1 = DX is negative 0 = DX is positive
8	[a.] SIGN_DY	1 = DY is negative 0 = DY is positive
7:6	unused	
5	[b.] X_REVERSIBLE	1 = Draw line reversibly, X-Windows Style. When NT_REVERSIBLE is set, it overrides X_REVERSIBLE.
4	CALC_ONLY	1 = Calculate all constants and addresses but don't draw
3	SKIP_FIRST	1 = Do not draw the first pixel on a line segment

Table 112. Line Control Register *(2 of 2)*

Bits	Field	Description
2	SKIP_LAST	1 = Do not draw the last pixel on a line segment.
1	[b.] INVERT_ZERO_TEST	1 = For zero-error-term cases, inverts the decision to step in the major and minor or just major axis for NT_REVERSIBLE, X_REVERSIBLE, and non-reversible line draws 0 = Not enable invert zero test
0	[b.] NT_REVERSIBLE	1 = Draw line reversibly, NT style. For NT lines always choose lower X or Y value. When set, overrides X_REVERSIBLE. 0 = Disable NT style. If both NT_REVERSIBLE and X_REVERSIBLE are set to 0, non-reversible lines are drawn
a. Bits 10:8 control the effective drawing octant for zero-parameter lines. These bits are ignored and overwritten when the line draw engine calculates the Bresenham parameters. b. If bits 0, 1, and 5 are set to "0," the line engine will step both axes when the error term is zero.		

Zero-parameter line draws are not affected by CALC_ONLY, allowing a sequence of (1) DRAW_LINE (CALC_ONLY) (2) modify Bresenham registers (3) DRAW_LINE (no parameters).

Line Pattern Register

name: GUI_LINE_PATTERN
address: GBASE | QUEUED_REG | 00000028h, read/write
address: GBASE | NON_QUEUED_REG | 00000028h, read/write
size: 32 bit
function: This register is used to control the style of lines drawn. When bit 0 is set to "1," the current pixel is drawn in foreground color. When bit 0 is set to "0," the current pixel is drawn in background color. The contents of the line pattern register are rotated one bit position to the right for every line pixel drawn; i.e., bits 31:1 become bits 30:0 and bit 0 becomes bit 31. The line pattern register is not reset between line draw operations, so it will track through polyline operations. Once the GUI is enabled (GRP_CFG4[0] = 1), the pattern is reinitialized only by writing to this pattern register. The contents of this register are ignored unless a source bitmap with the pattern bit is selected (see Table 103, "Bitmap Register TYPE Field," on page 175).

Bresenham 0, Address Register

name: GUI_BRES0_ADDR
address: GBASE | QUEUED_REG | 00000080h, read/write
address: GBASE | NON_QUEUED_REG | 00000080h, read/write
size: 32 bit
function: The value in this register is normally computed from the XY to linear address logic which converts incoming XY addresses to linear frame buffer addresses. The bit contents are shown in Table 113.

Table 113. Bresenham 0, Address Register

Bit(s)	Description
31:25	reserved, set to zero
24:5	dword address
4:2	nibble address for 4 bpp modes
1:0	reserved, set to zero
Note: 8bit address requires byte, 16bit address requires a word, 24bit address requires byte, 32bit address requires dword	

Bresenham 0, Error Register

name: GUI_BRES0_ERR
address: GBASE | QUEUED_REG | 00000084h, read/write
address: GBASE | NON_QUEUED_REG | 00000084h, read/write
size: 32 bit
function: The value in this register is normally computed from the line command setup logic, unless a zero parameter line command was issued. The value is also computed in the CALC_ONLY mode of line draw. The value is set as required for the Bresenham line draw algorithm. The bit contents are shown in Table 114.

Table 114. Bresenham 0, Error Register

Bit(s)	Description
31:0	Error value

Bresenham 0, K1 Register

name: GUI_BRES0_K1
address: GBASE | QUEUED_REG | 00000088h, read/write
address: GBASE | NON_QUEUED_REG | 00000088h, read/write
size: 32 bit
function: The value in this register is normally computed from the line command setup logic, unless a zero parameter line command was issued. The value is also computed in the CALC_ONLY mode of line draw. The value is set as required for the Bresenham line draw algorithm. The bit contents are shown in Table 115.

Table 115. Bresenham 0, Constant K1

Bit(s)	Description
31:0	K1 value

Bresenham 0, K2 Register

name: GUI_BRES0_K2
address: GBASE | QUEUED_REG | 0000008Ch, read/write
address: GBASE | NON_QUEUED_REG | 0000008Ch, read/write
size: 32 bit
function: The value in this register is normally computed from the line command setup logic, unless a zero parameter line command was issued. The value is also computed in the CALC_ONLY mode of line draw. The value is set as required for the Bresenham line draw algorithm. The bit contents are shown in Table 116.

Table 116. Bresenham 0, Constant K2

Bit(s)	Description
31:0	K2 value

Bresenham 0, Increment 1 Registers

name: GUI_BRES0_INC1
address: GBASE | QUEUED_REG | 00000090h, read/write
address: GBASE | NON_QUEUED_REG | 00000090h, read/write
size: 32 bit
function: The value in this register is normally computed from the line command setup logic, unless a zero parameter line command was issued. The value is also computed in the CALC_ONLY mode of line draw. The value is set as required for the Bresenham line draw algorithm. The bit contents are shown in Table 117.

Bresenham 0, Increment 2 Registers

name: GUI_BRES0_INC2
address: GBASE | QUEUED_REG | 00000094h, read/write
address: GBASE | NON_QUEUED_REG | 00000094h, read/write
size: 32 bit
function: The value in this register is normally computed from the line command setup logic, unless a zero parameter line command was issued. The value is also computed in the CALC_ONLY mode of line draw. The value is set as required for the Bresenham line draw algorithm. The bit contents are shown in Table 117.

Table 117. Bresenham 0, Increment 1 and 2 Registers

Bit(s)	Description
31:18	Sign extend bit[17] (writes bit 17 into this field)
17:2	Increment value
1:0	Set to zero

Bresenham 0, Length Register

name: GUI_BRES0_LENGTH
address: GBASE | QUEUED_REG | 00000098h, read/write
address: GBASE | NON_QUEUED_REG | 00000098h, read/write
size: 32 bit
function: The value in this register is normally computed from the line command setup logic, unless a zero parameter line command was issued. The value is also computed in the CALC_ONLY mode of line draw. The value is set as required for the Bresenham line draw algorithm. The bit contents are shown in Table 118.

Table 118. Bresenham 0 Length Register

Bit(s)	Description
31:12	Reserved, set to zero
11:0	Line pixel count - 1, e.g. (1,1)->(3,3) yields a length register value of 2.

SOFTWARE ENCODED VIDEO (SEV) PLAYER

Introduction

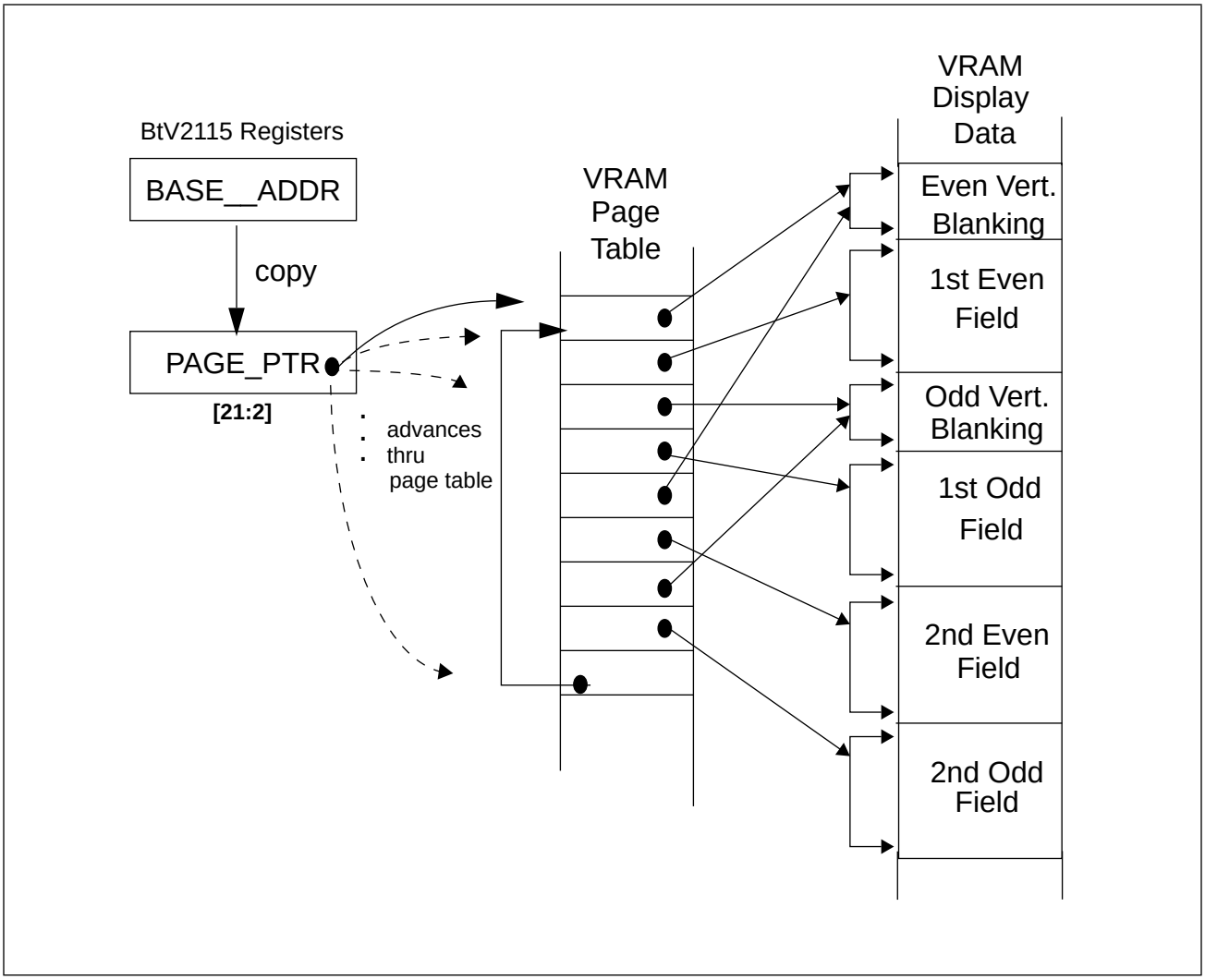
The BtV2115 SEV module is capable of supplying data to the BtV2487 PACDAC SEV module from the VRAM serial port. This module is capable of operating independently of system software for one or more frames but it has complicated requirements for the organization of its data in VRAM memory.

This module has non-trivial requirements for VRAM to BtV2487 bandwidth that vary based on the compression achieved in the run length encoding and several other factors. If the necessary bandwidth is not available the SEV FIFO in the BtV2487 will underflow.

Organization of Display Data In Memory

The display data is organized in VRAM memory in a linked list data structure, as shown in Figure 20.

Figure 20. Organization of Display Data



A page table consisting of 32 bit dwords contains a pointer to the start of a data block and a count specifying the length of the display block. Table 119 shows the bit definitions of the page table entry.

Table 119. Page Table Entry

Bits	Field Name	Description
31:18	DATA_CNT	Number of dwords to transfer for this page table entry
17:0	DATA_PTR	Pointer to starting location in VRAM [21:4]

The DATA_CNT field of the page table entry specifies the length of the block of display data in memory. The length is specified in dwords; the length in bytes is 4 times this value.

The DATA_PTR field of the page table entry points to the start of a block of display data in memory. The DATA_PTR field is 18 bits wide; these bits go to form A22-A4 which together with assumed values of 0 for A3-A0 allow the data block to begin at any 4 dword / 16 byte address in VRAM. Note that is not possible to point to any data block in memory at anything other than a 4 dword / 16 byte address (i.e. anything with address bits A3-A0 non-zero).

Note that these restrictions on the starting address and length of the display data block in memory ensure that the display data block ends on a dword boundary. The restrictions on starting and stopping addresses for the display data in VRAM are essential because the BtV2115 is unable to transfer anything other than dwords over the VRAM serial port. Without these restrictions this module would be impossible to implement - however these restrictions impose some requirements on the programmer when generating the display data. The simplest way that these requirements could be satisfied is if the programmer adjusts some repeat blocks within the run length data to ensure that each block of display data is a multiple of 4 bytes in length - this will ensure that no SEV repeat data structure spans over more than one display data block in memory. If this is not possible then the programmer must ensure that the data in the second and subsequent display data block is set up to remain in sync with the run-length encoded data coming from the previous block.

A value of 0 in the page table entry DATA_CNT field signifies the end of the page table. When the SEV module gets to this entry in the page table it ignores the pointer part of the entry, goes back to the start of the page table and begins processing from there again.

There is no limitation on the smallest repeat count allowable in the DATA_CNT field of a page table entry, however because each page table entry initiates at least one VRAM serial transfer to the BtV2487 a long sequence of very small display data blocks will likely lead to the BtV2487 SEV block running out of data. The actual minimum value will depend on various other issues like how much bandwidth is made available on the VRAM serial port for the transmission of SEV data, how compressed the SEV data is, etc. It is the responsibility of the programmer to ensure that sufficient VRAM serial port bandwidth is available and that it is used efficiently enough to ensure that the BtV2487 SEV module does not run out of data.

Limitations on the Display Data

The limitations on the Display table are listed below.

- All page table and display data must be in the VRAM.
- The start of the page table can be anywhere in VRAM - i.e. it must be at an address with an A1-A0 of zero.
- The page table can be any length and is limited only by the size of VRAM. The end of the page table is indicated by a page table entry with a count value of 0.
- Each block of display data must begin on a 16 byte or 4 dword boundary. (i.e. A VRAM address where A3-A0 are zero.)

- Each block of the display data must end on a dword boundary. (i.e. A VRAM address where A1-A0 are zero.)
- The maximum length of a block of display data is 2^{14} dwords, or 64KB. it is possible to make up a longer block of display data by stringing together two or more page table entries.

Programming the SEV Block

The vast majority of the programming involved with the BtV2115 SEV block is in setting up the display data and page tables in VRAM memory. Once all the data is set up the programmer can start the SEV module by writing the SEV GRP_SV_CTRLW register and observing the operation of the SEV module by reading the SEV GRP_SV_CTRLR register.

SEV Controller Write Register

name: GRP_SV_CTRLW[3:0]
index: B4h:B1h, write only
size: 32 bit
function: This is a graphics extension register used to control the operation of the soft video module. The contents of the bit registers are shown in Table 120. The bits are further explained in the paragraphs that follow.

Table 120. SEV GRP_SV_CTRLW Register

Bits	Field Name	Description
31	ENABLE	1 = Enable SEV operation 0 = Reset SEV module
30	PTOP	1 = Pause at top of list 0 = Normal operations
29	PIMM	1 = Pause immediately upon setting this bit 0 = Normal operation
28:24	unused	
23:22	PKT_SIZE	00 = 32 dword packets 01 = 64 dword packets 10 = 128 dword packets 11 = 256 dword packets
21:2	BASE_ADDR	Pointer to first page table entry in VRAM.
1:0	reserved	Set to zero

GRP_SV_CTRLW[31] (ENABLE)

The ENABLE bit enables/disables the operation of the BtV2115 SEV module. Currently the ENABLE bit prevents the SEV module from operating - if the ENABLE bit is de-asserted while the SEV module is already operating it will be interpreted as a reset and the SEV module will drop what it is doing and stop generating SEV data for the BtV2487. The reset condition of the ENABLE bit is de-asserted.

GRP_SV_CTRLW[30] (PTOP)

The PTOP (pause at top) bit pauses the SEV module when it gets to the top of the page table. This is useful for when the programmer is writing to the GRP_SV_CTRLW register. If this bit is asserted, the SEV module pauses just before copying the BASE_ADDR field to the PAGE_PTR register. This is useful because the access to the BASE_ADDR field is split over 3 8-bit writes and it could be hazardous to allow the possibility of the BASE_ADDR field being transferred to the PAGE_PTR register while it is partially updated.

GRP_SV_CTRLW[29] (PIMM)

The PIMM (pause immediately) bit is similar to the PTOP bit in that it stops the SEV module just after it's next transfer of video data to the BtV2487 but before it updates the PAGE_PTR, DATA_PRT or DATA_CNT registers. With the current implementation of the SEV module the programmer must wait for at least one cycle after the SEV module sees the PIMM bit asserted before reading the PAGE_PTR or DATA_CNT fields of the GRP_SV_CTRLR register. The SEV module is guaranteed to have gotten at least as far through the video data as indicated by the values read from the GRP_SV_CTRLR register. The SEV module may have proceeded further by as much as a VRAM packet - however due to the way this module is designed there is no way for the programmer to determine if this is so.

Setting either the PTOP or PIMM bit for any appreciable period of time could cause erroneous behavior in the SEV module and waste VRAM to BtV2487 bandwidth causing the BtV2487 SEV module to run out of video data to display. The programmer should only assert either of these bits for a short time. Assert the PTOP bit just before writing to the GRP_SV_CTRLW register BASE_ADDR field, and assert the PIMM bit just before reading the GRP_SV_CTRLR register DATA_CNT or PAGE_PTR fields. Immediately deassert the relevant bit when finished.

The ENABLE, PTOP, and PIMM bits are in a separate byte all by themselves so that it is easy to write to them without upsetting any of the other fields.

GRP_SV_CTRLW[23:22] (PKT_SIZE)

The PKT_SIZE field is a 2 bit field that controls the maximum size of serial packet to use in transferring data. The available packet sizes are shown in Table 121. The reset value of the PKT_SIZE field is not defined.

Table 121. Packet Size Used

PKT_SIZE	Bytes/packet	Dword Cycles/packet
2'B00	128	32
2'B01	256	64
2'B10	512	128
2'B11	1024	256

GRP_SV_CTRLW[21:2] (BASE_ADDR)

The BASE_ADDR field is 20 bits wide and indicates the start address for the page table in VRAM memory. This field is copied to the PAGE_PTR register when ENABLE is first asserted and again whenever the SEV module reaches the end of the page table in memory. This is a 20 bit dword address into the 4M address space of the VRAM. The reset value of the BASE_ADDR field is not defined.

SEV Controller Read Register

name: GRP_SV_CTRLR[3:0]
index: B4h:B1h, read only
size: 32 bit
function: This is a graphics extension register used to control the read operation of the soft video. The GRP_SV_CTRLR register allows you to check the status of the BtV2115 SEV module. Both the DATA_CNT and PAGE_PTR fields are the contents of internal registers within the SEV module.

The contents of the bit registers are shown in Table 122. The bits are further explained in the paragraphs that follow.

Table 122. SEV GRP_SV_CTRLR Register

Bits	Field Name	Description
31:18	DATA_CNT	Number of dwords yet to be transferred in the currently active PAGE_TABLE entry
17:0	PAGE_PTR	Address of the PAGE_TABLE entry that is currently being processed.

GRP_SV_CTRLR[31:18] (DATA_CNT)

The DATA_CNT register contains the number of dwords of data remaining to be transferred in the current display data block. Whenever the SEV module moves to a new page table entry, this register is loaded from the DATA_CNT field of the page table entry. As packets of data are transferred over the VRAM serial bus to the BtV2487, this register is decremented to keep track of the number of dwords remaining to be transferred. The DATA_CNT register is decremented by the appropriate packet size (as defined by the PKT_SIZE field, see Table 120 on page 189) whenever video data is sent to the BtV2487 from the VRAM serial port. If the value is not a integral number of packets, then everything proceeds as before except that the fractional part of a packet is transferred as a smaller last packet.

Thus, the location of the SEV module within the display frame is determined. The PAGE_PTR field contains the currently processed display data block. The DATA_CNT specifies where in that data block the processing currently is.

GRP_SV_CTRLR[15:0] (PAGE_PTR)

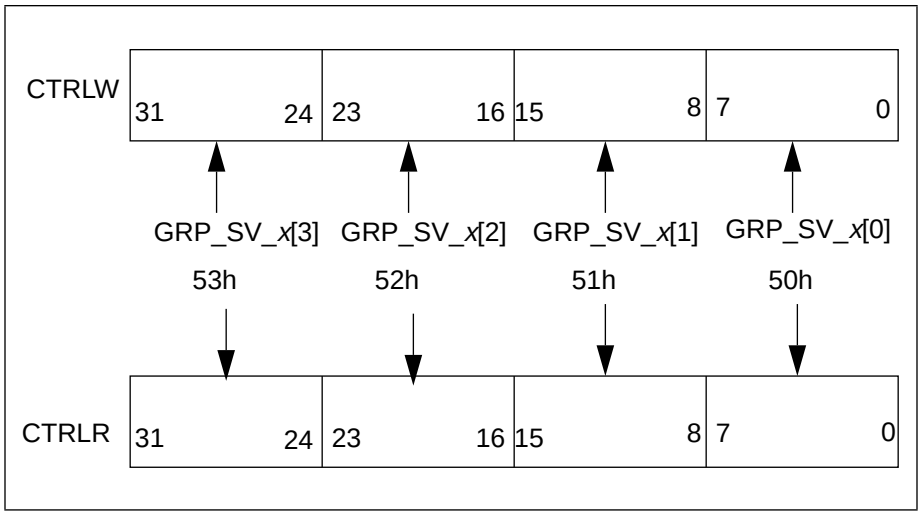
The PAGE_PTR register points to the current page table entry (on a dword address). Initially after the ENABLE bit in the GRP_SV_CTRLW register is first asserted, this register is copied from the BASE_ADDR field in the GRP_SV_CTRLW register to point at the first entry in the page table in memory. As the SEV module moves through the page table, this register is also incremented and can be read to see how far through the page table the BtV2115 SEV module has advanced. Note that the BASE_ADDR field in the GRP_SV_CTRLW register is 22 bits wide and the PAGE_PTR field in the GRP_SV_CTRLR register is only 18 bits. The missing bits are two MSBs and two LSBs. These bits are ignored in the BASE_ADDR field because only dword accesses are available from the VRAM.

Accessing SEV Registers

The access mechanism to the GRP_SV_CTRLR and GRP_SV_CTRLW registers is via the VGA graphics extension bus. The GRP_SV_CTRLW and GRP_SV_CTRLR registers are mapped to the same addresses on the VGA graphics extension bus. Since GRP_SV_CTRLW is write only and GRP_SV_CTRLR is read only, they can share addresses with no problems.

The soft video registers are 32 bits wide and the VGA graphics extension bus is 8 bits wide. This difference in register length could lead to problems because one read/write could be broken into separate 8-bit read/writes. To prevent the accidental read/write of incorrect data, access to each of these registers is divided into four separate 8-bit reads/writes, as shown in Figure 21.

Figure 21. Accessing SEV Registers Through VGA Extension Bus



When reading the GRP_SV_CTRLR register, both the DATA_CNT (GRP_SV_CTRLR[31:18]) and PAGE_PTR (GRP_SV_CTRLR[17:0]) fields could be changing while the soft video is running. These fields require two or more 8-bit reads, which could result in an incorrect result if the value of these fields change between the 8-bit reads. If the software driver does not need to access the full contents of these fields (for example, one page table is smaller than 256 entries), then the software could read only, for example, the least significant byte of GRP_SV_CTRLR to determine where in the page table the soft video module is currently processing. If full accuracy is required, a possible solution is to pause the soft video module while the full contents of the GRP_SV_CTRLR register is read. To do this, assert the PIMM bit of the GRP_SV_CTRLW register, then write the PIMM bit again to allow the soft video module to continue. This procedure will

cause the soft video module to pause immediately; in an extreme case this could cause the video FIFO in the PACDAC to underflow.

When writing to the GRP_SV_CTRLW register, the BASE_ADDR field (GRP_SV_CTRLW[21:2]) must be broken into three separate 8-bit writes. It is possible that the soft video module will reach the end of its VRAM PAGE_TABLE and attempt to reload the PAGE_PTR register from a BASE_ADDR field which is only partially updated. To avoid this problem, write the most-significant byte of this register to set the PTOP bit. This action will pause the soft video module if it reaches the top of the VRAM PAGE_TABLE, thus preventing the use of an incomplete BASE_ADDR. When the three low order bytes are written to the GRP_SV_CTRLW register, the most significant byte can be written again to clear the PTOP bit. This procedure pauses the soft video module only when it reaches the end of its VRAM PAGE_TABLE and is ready to return to the top.

PACDAC CONTROLLER

Introduction

The PACDAC Controller of the BtV2115 is essentially a state machine capable of interpreting a VRAM resident data structure. This data structure is a compacted representation of the display environment which determines system parameters such as pixel width, screen resolution, size and position of video windows, cursor location, serial clock and pixel clock rates, etc. In short the data structure stores all information necessary to completely define the entire display state. The PACDAC Controller is responsible for processing this data structure and clocking the pertinent elements from VRAM to the BtV2487. This chapter describes BtV2115 functional support of these features, the format of the data structure and its usage.

The BtV2115 registers used to control the PACDAC are defined in the "Register Definitions" starting on page 55.

For more information on the PACDAC, refer to the *BtV2487 PACDAC Specification*.

PACDAC Data Types

Data is transferred by the PACDAC Controller from VRAM to the BtV2487 PACDAC via the VRAM serial port. Blocks of data are tagged using the PT[3:0] pins of the BtV2115 to indicate to the BtV2487 PACDAC the type of data currently at the inputs of the PACDAC. Table 123 shows the data type assignments for the PT[3:0] pins.

Table 123. PT[3:0] Data Type Assignments

Address	Data Type	Description
0h		Reserved
1h	CD	Cursor pixels
2h	GD	Graphics pixels
3h	V1	Video FIFO 1 pixels
4h	V2	Video FIFO 2 pixels
5h	DR	BtV2487 (DAC) register data
6h	TS	Timing structure data
7h - Dh		Reserved
Eh	NOOP	No operation
Fh	RST	Reset

The PT and serial data from the VRAM are synchronous to LCLK[1:0]. Each LCLK rising edge clocks a full 32 bit dword into PACDAC. PT types are never mixed within a dword.

The RST PT data type is used to reset the BtV2487 PACDAC without performing a system reset. Setting GRP_CFG5[6] (see Table 21 on page 49) will cause the PACDAC Controller to warm reset the BtV2487 PACDAC by issuing RST PT data types until bit 6 is cleared.

VRAM serial data coincident with the DR PT data type contains either a register address or register data to be written into the register file of the BtV2487 PACDAC.

The TS PT data type indicates information which is used by the BtV2487 PACDAC to determine when to begin loading or unloading its internal FIFO set, the correct assertion of CRT control signals such as HSYNC and BLANK, and the timing for the LSYNC handshake signal.

The NOOP PT data type is used to initialize the VRAM serial register between packet types.

Master Structure

The core of the PACDAC Controller data structure is the Master Structure. The elements within the Master Structure's sequentially accessed list contain either direct information or pointers to substructures that define aspects of the display state. The Master Structure is processed at the top of each frame allowing screen format changes at the frame rate.

The BtV2115 supports two Master Structures to allows alterations to the inactive Master Structure and substructures without effecting the current display state. The dual structures and the associated Master Structure switching mechanism permit graceful mode switching between screen formats.

The dual Master Structure pointers, labeled A and B, are accessed via the I/O registers PDC_MSPTRA and PDC_MSPTRB (defined in "Master Structure A Address Register" on page 55 and "Master Structure B Address Register" on page 56). The switching mechanism is accessed via the PDC_CNTL register (see page 57).

PDC_MSPTRA and PDC_MSPTRB contain byte addresses which point to the first dword of the respective Master Structure. Bits [1:0] of PDC_MSPTRA and PDC_MSPTRB always contain zeroes.

The PACDAC Controller also uses the DAC Register Structure, which contains the desired register state for the BtV2487 PACDAC during the ensuing frame and the Timing Structure, which contains information describing the screen formatting used during the ensuing frame.

Table 124 defines the contents of the Master Structure. Details on the different data types are provided following the table. An example of a Master Structure setup is provided at the end of this chapter.

Table 124. Master Structure Entry Format *(1 of 3)*

	Entry Name	Field	Bits	Description
Graphics Data	PKT_ORDR		31:00	Packet ordering control
	GD_PNTR		31:20	Reserved
		PNTR	19:00	Dword pointer to start of graphics data
	GD_Y		31:28	Reserved
		Y_END	27:16	Screen row ending position + 1
			15:12	Reserved
		Y_STRT	11:00	Screen row starting position
	GD_X	PTCH	31:16	Pitch of graphics data
			15:12	Reserved
		X_CNT	11:00	Number of dwords to send to PACDAC per screen row
Cursor Data	CD_PNTR		31:20	Reserved
		PNTR	19:00	Pointer to start of cursor data
	CD_Y		31:28	Reserved
		Y_END	27:16	Screen row ending position + 1
			15:12	Reserved
		Y_STRT	11:00	Screen row starting position
	CD_X	PTCH	31:16	Pitch of cursor data
			15:12	Reserved
		X_CNT	11:00	Number of dwords to send to PACDAC per screen row

Table 124. Master Structure Entry Format *(2 of 3)*

	Entry Name	Field	Bits	Description
Video 1	V1_PNTRA		31:20	Reserved
		PNTR	19:00	Pointer to video 1 data, 1 of 2
	V1_PNTRB		31:20	Reserved
		PNTR	19:00	Pointer to video 1 data, 2 of 2
	V1_Y		31:28	Reserved
		Y_END	27:16	Ending row position + 1
			15:12	Reserved
		Y_STRT	11:00	Starting row position
	V1_X	PTCH	31:16	Pitch of video 1 data
			15:12	Reserved
		X_CNT	11:00	Number of dwords to send to PACDAC per screen row
	V1_YSCL		31:15	Reserved
		RSVD	14:13	Reserved (write zeroes)
		Y_SINC	12:00	Y scale increment, video 1 (I.FFFFFFFFFFFF)
Video 2	V2_PNTRA		31:20	Reserved
		PNTR	19:00	Pointer to video 2 data, 1 of 2
	V2_PNTRB		31:20	Reserved
		PNTR	19:00	Pointer to video 2 data, 2 of 2
	V2_Y		31:28	Reserved
		Y_END	27:16	Ending row position + 1
			15:12	Reserved
		Y_STRT	11:00	Starting row position
	V2_X	PTCH	31:16	Pitch of video 2 data
			15:12	Reserved
		X_CNT	11:00	Number of dwords to send to PACDAC per screen row
	V2_YSCL		31:15	Reserved
		UP_MODE	14:13	See Table 125 on page 202
		Y_SINC	12:00	Y scale increment, video 2 (I.FFFFFFFFFFFF)

Table 124. Master Structure Entry Format *(3 of 3)*

	Entry Name	Field	Bits	Description
Timing Structure Data	TS_PNTR		31:20	Reserved
		PNTR	19:00	Pointer to timing structure
	TS_LENG		31:20	Reserved
		LENG	19:00	Sum of all NATS fields (refer to Table 127) in the timing structure
DAC Register Data	DR_PNTR		31:20	Reserved
		PNTR	19:00	Pointer to DAC registers
	DR_END		31:28	Reserved
		BLK	27:20	Register transfer block size
		END	19:00	End of DAC register
Clock Data	MISC		31:28	Reserved
		NXT_LINE	27:16	Initial next line value
		CLK_MOD	15:00	[15:4] Graphics clock modulation [3:00] IO clock modulation

Packet Ordering

The order by which PT data types (also called packet types) are sent to the BtV2487 PACDAC is controllable via the PKT_ORDR Master Structure Entry. Each nibble of PKT_ORDR defines a PT data type. The sequence is processed from LS nibble to MS nibble; all six nibbles are processed during each scan line with the same packet order used for each.

All PT data types shown in Table 123 are valid, except the RST PT data type which is generated in hardware within the PACDAC Controller. Each nibble entry in PKT_ORDR must be unique except the NOOP which may be added as necessary to pad out the PKT_ORDR entry.

For example, a typical PKT_ORDR entry could be 124356h, which would cause the PACDAC Controller to send PT data types in this sequence: TS, DR, V1, V2, GD, CD. A system which did not require video or cursor functions might use the following PKT_ORDR entry, EE2E56h.

PKT_ORDR determines only the ordering of PT data type. The PACDAC controller determines whether a given PT data type is needed for a given scan line and sends only those which are required to format the scan line.

Graphics Data

The Master Structure field GD_PNTR.PNTR points to the location of the first dword of graphics data within the frame buffer. This entry is a twenty bit dword address.

GD_Y contains values which define the vertical boundaries for displayed graphics pixels. Screen position values are based on an arbitrary reference point usually line 0. Typically line 0 will be the first scan line of the VSYNC period. The

starting screen row, line 0 in this example, can be programmed using the NXT_LINE field of the MISC Master Structure entry. GD_Y.Y_END contains the position of the last screen row of displayed graphics pixels+ 1. GD_Y.Y_STRT contains the screen row of the first line of displayed graphics pixels.

GD_X contains values which constrain the horizontal boundaries for graphics data in units measured in dwords. GD_X.PTCH is the graphics data pitch. Pitch is defined as the difference between the addresses of two vertically adjacent pixels, in dwords. GD_X.X_CNT specifies the number of dwords contained in a single line of graphics data. Commonly PTCH and X_CNT will contain the same value if graphics data is packed without gaps between the last pixel of line N and the first pixel of line N+1. To determine the correct value for X_CNT multiply the number of bits per pixel times the number of horizontal pixels and divide the quantity by 32.

The majority of the remaining data types shares a common format with that explained above.

Cursor Data

The Master Structure field CD_PNTR.PNTR points to the starting location of the cursor data register. This entry is a twenty bit dword address.

CD_Y.Y_STRT contains the value of the first line of the cursor. CD_Y.Y_END contains the last line of the cursor + 1.

CD_X.PTCH contains the pitch of vertically adjacent cursor pixels as stored in the frame buffer. CD_X.X_CNT contains the number of dwords in a single row of cursor data.

Video Data

The PACDAC Controller supports two double buffered video windows. The windows are labeled 1 and 2 and the dual buffers for each are labeled A and B. The Master Structure fields V1_PNTRA.PNTR, V1_PNTRB.PNTR, V2_PNTRA.PNTR, and V2_PNTRB.PNTR point to the starting location of the data for video window 1A and 1B and video window 2A and 2B, respectively. The formatting for buffers A and B are exactly the same, they share PTCH, Y_SINC, etc. The dual buffering mechanism allows removal of artifacts caused by the asynchronous video capture and display scan rates common in other systems.

The V1_PNTRA.PNTR, V1_PNTRB.PNTR, V2_PNTRA.PNTR, and V2_PNTRB.PNTR Master Structure entries are twenty bit dword addresses.

Upscaling in X and Y is supported in the BtV system; the PACDAC performs virtually all upscaling along the X axis. PACDAC Controller support is required for upscaling in the Y direction. The heart of this mechanism is a 12+1 bit DDA that determines which lines of screen data to send to the PACDAC.

Two types of upscaling are available in the PACDAC. Interpolated upscaling analyzes two pixel values to determine which value to display. Replicated upscaling displays a single pixel value multiple times. Interpolation results in a higher quality image, however only a single interpolated window is supported within the BtV architecture. In replicated mode, two independent upscaled windows are possible.

The Master Structure supports up to four different video buffers: V1_PNTRA, V1_PNTRB, V2_PNTRA, and V2_PNTRB. In a configuration which does not require upscaling or uses replicated upscaling the use of the four pointers is straight forward, V1_PNTRA and V1_PNTRB control the display and capture buffers for video window 1. Likewise, V2_PNTRA and V2_PNTRB control the display and capture buffers for video window 2. In interpolated configurations these pointers have slightly different usages.

The upscaling mode is defined in the field UP_MODE within the V2_YSCL Master Structure entry, bits V2_YSCL[14:13]. (V2 is considered the primary window.) V1_YSCL[14:13] is reserved for future architectures and should be written with 0x0.)

The UP_MODE values specify how to increment the video pointers. Table 125 shows the UP_MODE bit definitions and the number of windows supported; additional information follows the table.

Table 125. UP_MODE Bit Definitions

Bit Defs.	Function	Windows	Description
00	No upscaling	1 or 2	Increment both A and B pointers for a given window after sending a line of video. Incrementing is accomplished by adding the pitch value for the window to both pointers. Pointers A and B for each window move in lock step.
01	Upscale with replication	1 or 2	Increment both A and B pointers for a given window after sending a line of video. Incrementing is accomplished by adding the pitch value for the window to both pointers only if a DDA overflow occurs when Y_SINC is added.
10	Upscale with interpolation	1	Pointers V1_PNTRA, V1_PNTRB, V2_PNTRA, and V2_PNTRB are altered. Two pointers are required for each buffer. See explanation below.
11	Reserved		

The replication upscaling is performed based on Y_SINC. The DDA adds the Y_SINC value to the accumulator and checks for overflow. If overflow occurs the video pitch is added to both A and B pointers. If overflow does not occur the pointers are unchanged. The Y_SINC field contains a 13 bit field defined as:
I.FFFFFFFFFFFF

The decimal point is located to the right of the MSB. Setting the MS bit to a 1 forces an overflow on the 12 bit accumulator value, essentially disabling the DDA

and causing the PACDAC Controller to increment the pointers by the pitch value each time.

Interpolation upscaling alters the pointers V1_PNTRA, V1_PNTRB, V2_PNTRA, and V2_PNTRB. Two pointers are required for each buffer. A pair of pointers are used to indicate line N and line N+1 within each video buffer. During one line of displayed video, both line N and line N+1 are transmitted to the PACDAC from the active buffer. The PACDAC uses its internal interpolator to process pixels from M and M+1 and determine the resulting pixel for display. Interpolation upscaling supports beam crossing, but only one window is displayable.

For interpolated upscaling the convention is that V1_PNTRA points to line N of the A buffer and V2_PNTRA points to line N+1 of the A buffer, similarly for the B buffer. Once both lines are sent for the active buffer, the PACDAC Controller uses the DDA to determine whether to increment the pointer values. The increment value V2 Y_SINC is used for calculations in this mode, V1 Y_SINC is extraneous. Y_SINC is added to the accumulator value. If an overflow occurs, V2_PNTRA is copied to V1_PNTRA and V2_PNTRB is copied to V1_PNTRB. V2_PNTRA and V2_PNTRB are then incremented by V1_X PTCH. V1_X PTCH is not used in this configuration. The buffer pointer pairs increment in lock step. In this configuration, video FIFO unload commands for FIFO 1 and 2 must be simultaneous in the timing structure.

Beam crossing (or frame rate conversion) deals with removing visible artifacts from live video, especially scenes which contain motion. These artifacts are a result of the differing frame rates of the sampled captured video and the display system. Video sources tend to have a lower frame rate than high resolution graphics display systems. The result is that at some point a single frame of displayed video will have data from the previous frame and data from the current frame on the display at the same time. For still images this is not as visible. However, since the eye is very sensitive to motion, live images which contain a high degree of motion will show undesirable artifacts. The solution to this problem is the use of two buffers for each video window: one for display and one for capture. At appropriate times, the pointers to these buffers are switched, removing motion artifacts. This switch control is accomplished outside of the PACDAC Controller in the video data structures.

V1_PNTR.PNTR is a twenty bit dword address pointing to the first valid data in the Video Window 1 A Buffer. V1_PNTRB.PNTR is a pointer to the first valid data in the Video Window 1 B Buffer.

V1_Y.Y_END indicates the ending row position + 1 of Video Window 1. V1_Y.Y_STRT indicates the starting screen row position for Window 1. Buffers A and B use the same Y_END and Y_STRT values.

V1_X.PTCH signifies the Video Window 1 pitch for vertically adjacent pixels of buffers A and B, as stored in the frame buffer.

V1_X.X_CNT contains the number of dwords which must be sent to PACDAC during active video lines for each row of Video Window 1.

V1_YSCL.Y_SINC is a 12+1 bit quantity which is used in upscaling. Y_SINC is the value which is added to the accumulator after each video line is sent, if the accumulator overflows then V1_X.PTCH is added to V1_PNTR.PNTR and stored

in V1_PNTR.PNTR. If the accumulator does not overflow the accumulator value is retained for the next line. The format of Y_SINC is I.F. F is the 12 bit fractional value, I is the integer value. If I is set to a one the accumulator overflows on each calculation, PTCH is added to PNTR, essentially disabling the DDA.

V2_PNTR.PNTR is a twenty bit dword address pointing to the first valid data in the Video Window 2 A Buffer. V2_PNTRB.PNTR is a pointer to the first valid data in the Video Window 2 B Buffer.

V2_Y.Y_END indicates the ending row position + 1 of Video Window 2. V2_Y.Y_STRT indicates the starting screen row position for Window 2. Buffers A and B use the same Y_END and Y_STRT values.

V2_X.PTCH signifies the Video Window 2 pitch for vertically adjacent pixels of buffers A and B, as stored in the frame buffer.

V2_X.X_CNT contains the number of dwords which must be sent to PACDAC during active video lines for each row of Video Window 2.

V2_YSCL.Y_SINC is a 12+1 bit quantity which is used in upscaling. Its usage is similar to that described for V1_YSCL.Y_SINC.

V2_YSCL.UP_MODE is used by both Video Window 1 and Video Window 2 to indicate the upscale mode required. Table 125 summarizes the usage of this field.

Timing Structure Data

The Timing Structure is a sub-element which is linked to the Master Structure via the TS_PNTR.PNTR entry. Its purpose is to format the display and provide control signals which inform the PACDAC when to begin unloading its internal FIFOs or when to assert HSYNC, etc. The mechanism for this is described below.

TS_LENG.SUM_NATS indicates the total number of timing atoms in the Timing Structure. The correct value for SUM_NATS is obtained by adding the NATS fields of every TS_CNT entry in the timing structure. The discussion below describes timing atoms and their format.

Timing Atom

The smallest element of the Timing Structure is called a Timing Atom. The Timing Atom contains a 12 bit run length and a 16 bit control state. The control state indicates the values which are to be assigned to various signals within PACDAC. The 12 bit run length (Horizontal Repeat Count, X_REP) indicates the number of pixel clocks to hold this control state. A separate atom is required for each change in the control state. Therefore, each atom defines a specific timing region within a screen line. All PACDAC signals will have the same value within a region, therefore timing atoms run length encode the assertion of the control state in the horizontal direction. Table 126 defines the timing atom bits.

Table 126. Timing Atom Bit Definitions

BIT	Item	Description	Detail
31	LS	Linesync	1= ready for new data 0= not ready
30	HS	HSYNC out	1= drive HSYNC high 0= drive HSYNC low
29	VS	VSYNC out	1= drive VSYNC high 0= drive VSYNC low
28	U1	Unload Video 1	1 = read from Video FIFO #1 0 = ignore Video FIFO #1
27	U2	Unload Video 2	1 = read from Video FIFO #2 0 = ignore Video FIFO #2
26	UG	Unload Graphics	1 = read from Graphics FIFO
25	BL	Composite Blank	0= display blanked 1= display not blanked
24	SY	Composite Sync	1= sync not active 0= sync active (sync on green modes only)
23:18		Reserved	
17	CO	Cursor Origin	1= enable cursor x counter 0= disable cursor x counter
16	GA	Graphics Signature	1= enable signature capture 0 = disable signature capture
15:12		Reserved	
11:0	AC	Atom Count	repeat count for timing atom

Timing Structure Format

The timing structure describes the position of the timing regions within a single frame by referencing timing atoms and formatting their presentation to PACDAC. While the timing atom is responsible for horizontal run length encoding of the control state, the timing structure in essence run length encodes the control state in the vertical direction.

The timing structure is divided into sets of values, first a single entry which describes the atom set, called TS_CNT, followed by a varying number of timing atoms. This sequence is repeated as required to describe the display state for the frame.

TS_CNT contains a field labeled NATS, which determines the number of timing atoms in this set. TS_CNT also incorporates a field labeled Y_REP, which indicates the valid number of screen lines. The one bit field FR_SYNC is used to indicate the top of frame for use in other sections of the BtV2115.

Table 127 illustrates the timing structure format.

Table 127. Timing Structure Format

Name	Field	Bits	Description
TS_CNT	FR_SYNC	31	Frame start flag
	DISP_ACTIVE	30	Active video is being displayed
	NATS	29:16	Number of atoms in set
	Y_REP	15:00	Vertical repeat count
TS_ATOM†	C_ST	31:16	Control State
		15:12	Reserved
	X_REP	11:00	Horizontal repeat count
TS_ATOM†	C_ST	31:16	Control State, repeat
		15:12	Reserved
	X_REP	11:00	Horizontal repeat count, repeat

† TS_ATOM is repeated as needed to describe a horizontal line. TS_CNT and TS_ATOM(s) are repeated as needed to describe a full frame.

DAC Data

The DAC Register Structure holds the complete register state for the PACDAC. The PACDAC has an internal auto-incrementing address register, which requires only a starting address when programing a contiguous block of registers. The first entry in the DAC Register Structure will contain a register index value. The PACDAC recognizes this as an address when it detects a 1 in bit 31 of the dword. Then, when this dword is transferred to the PACDAC by the PACDAC Controller, bit 31 causes the PACDAC to load bits [9:0] into the auto-incrementing address register. Thereafter, dwords which are sent with the DR PT data type are loaded into the register array as long as bit 31 of each dword is cleared. The PACDAC address register post increments following register data writes. Addresses may be mixed with register data as required to load a subset of the total PACDAC register set. Each register data entry in the DAC Register Structure occupies a full dword one for each register.

DR_PNTR.PNTR (a twenty bit dword address) points to the first entry in the DAC Register Structure.

DR_END.BLK determines the number of DAC Register Structure entries which are sent per line. Sending the full complement of PACDAC registers in a single line would typically overflow the PACDAC internal DAC Register FIFO, therefore a mechanism is provided to break the register set into blocks of size BLK.

DR_END.END is the twenty bit dword address of the last entry in the DAC Register Structure.

The total number of entries in the DAC Register Structure (DR_END - DR_PNTR + 1) must be an integer multiple of DR_BLK. This can be accomplished by padding the DAC register structure with the DR_INDEX, followed by the appropriate number of palette entries or alternatively simply padding the end of the structure with multiple writes to the address register.

Sample Master Structure Setup

This section provides a snippet of C code for setting up a generic master structure in VRAM and a DAC register structure. As a minimum, at least these two structures need to be initialized via software before the PACDAC Controller is enabled.

For this example, the Master Structure B pointer registers are used. The same methodology can easily be used to initialize another structure in VRAM for Master Structure A. This example writes VRAM through the protected mode aperture, however the real mode aperture could have also been used.

When the VGA is enabled, it automatically generates the timing structure based on the values written to the VGA registers at the address in VRAM pointed to by the master structure table entry TS_PNTR. The length is calculated and the value is placed in the TS_LENG entry. The VGA also writes the DAC Register Structure based on the address contained in DR_PNTR and DR_END. For the graphics structure, the VGA calculates the values for GD_X and GD_Y and updates the table accordingly. If the VGA is not enabled, then software is responsible for writing and maintaining all of these structures.

```
//Begin example
//Setup structure B pointer
outp(GRP_INDEX,PDC_MSPTRB_0);   //VRAM address 0x0ffa00(1 meg)
outp(GRP_DATA,0x00);
outp(GRP_INDEX,PDC_MSPTRB_1);
outp(GRP_DATA,0xfa);
outp(GRP_INDEX,PDC_MSPTRB_2);
outp(GRP_DATA,0x0f);

//Initialize Master Structure B in VRAM/
//First, packet order-->timing,DAC regs,graphics
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x0,0x00000256l);
//Graphics data structure pointers
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x4,0x80000 >> 2);//GD_PNTR
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x8,0);           //GD_Y
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0xc,0);           //GD_X
//Cursor data structure pointers
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x10,0);          //CD_PNTR
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x14,0);          //CD_Y
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x18,0);          //CD_X
//Display video 1 structure pointers
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x1c,0);          //V1 ptr a
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x20,0);          //V1 ptr b
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x24,0);          //V1_Y
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x28,0);          //V1_X
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x2c,0);          //V1_YSCL
```

```
//Display video 2 structure pointers
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x30,0);          //V2 ptr a
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x34,0);          //V2 ptr b
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x38,0);          //V2_Y
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x3C,0);          //V2_X
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x40,0);          //V2_YSCL
//Timing structure pointers
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x44,0xff600 >> 2);//TS_PNTR
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x48,0);           //TS_LENG
//DAC register structure pointers, pointer address - 1 dword
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x4C,0xfeffc >> 2);   //DR_PNTR
//Size of dac register table =300 dwords,block size=64
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x50,0x6400000 | ((0xFEFFC >> 2) + 300 -1)); /
/DR_END
//Clock modulation for graphics and I/O packets
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ffa00l | 0x54,0xc00c);    //CLK_MOD


//Begin DAC structure setup
outp(GRP_INDEX,GRP_PDAC_0);   //Setup DAC structure pointer
outp(GRP_DATA,0xFE);          //VRAM address 0xff000, load bits 21:11, this is LSB
outp(GRP_INDEX,GRP_PDAC_1);
outp(GRP_DATA,1);             //MSB of bits 21:11
outp(GRP_INDEX,GRP_25PLL_0);  //Setup PLL register values
outp(GRP_DATA,0xa9);          //25Mhz setup
outp(GRP_INDEX,GRP_25PLL_1);  //
outp(GRP_DATA,0x0a);          //
outp(GRP_INDEX,GRP_25PLL_2);  //
outp(GRP_DATA,0x00);          //
outp(GRP_INDEX,GRP_28PLL_0);  //28Mhz setup
outp(GRP_DATA,0x7c);          //
outp(GRP_INDEX,GRP_28PLL_1);  //
outp(GRP_DATA,0x0b);          //
outp(GRP_INDEX,GRP_28PLL_2);  //
outp(GRP_DATA,0x00);          //

//First dac pallet address word, for PDAC regs 0x100-0x1ff,see BtV2487 spec for details
//DR_PNTR points here
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000feffcl,0x80000100l);

//DAC register structure, for PDAC regs 0-0xff,see BtV2487 spec for details
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff400l,0x80000000l);
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff404l,0x00000000l);//border color index
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff408l,0x000000000);//pixel clock  PLL
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff40cl,0x00000afbl);//serial pll 32.9542
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff410l,0x00000000l);//cfg reg
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff414l,0x00000000l);//graphics format(bits/pix)
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff418l,0x00000206l);//cursor X position
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff41cl,0x00000000l);//reserved
```

```
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff420l,0x00000000l);//graphics diag reg
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff424l,0x00000000l);//softVideo diag reg
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff428l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff42Cl,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff430l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff434l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff438l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff43cl,0x00000000l);//fifo error status
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff440l,0x80000010l);//change address to 0x10
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff444l,0x00000000l);//video format reg
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff448l,0x00000000l);//Vid 1 color key reg
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff44cl,0x00000000l);//Vid 1 color key mask
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff450l,0x00000000l);//Vid 1 X scale
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff454l,0x00000000l);//Vid 2 color key
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff458l,0x00000000l);//Vid 2 color key mask
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff45cl,0x00000000l);//Vid 2 X scale
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff460l,0x00000000l);//Vid Y scale increment
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff464l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff468l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff46cl,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff470l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff474l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff478l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff47cl,0x80000200l);//change address 0x200
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff480l,0x00000000l);//border color reg
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff484l,0x00000000l);//cursor color 1
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff488l,0x00000000l);//cursor color 2
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff48cl,0x00000000l);//cursor color 3
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff490l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff494l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff498l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff49cl,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff4a0l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff4a4l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff4a8l,0x00000000l);//reserved
MEMWR_DWORD(GBASE | MBA_DIRECT | 0x000ff4acl,0x00000000l);//reserved
```

CPU HOST BUS INTERFACE

VESA Local Bus Interface

The BtV2115 interfaces to the CPU either via the Video Electronics Standards Association (VESA™) local bus (VL-Bus™) or the PCI Bus interface. For a description of the PCI interface, refer to “PCI Bus Interface” on page 216.

The VESA local bus is an architectural, timing, electrical, and physical interface that allows the BtV2115 to interface to the local bus of a host CPU. The VESA interface is defined in the following paragraphs. The overbar above a signal indicates that the signal is active-LOW.

VL Address Bus VL_ADDR[31:2] — These inputs provide the IO port addresses or physical memory addresses to the BtV2115.

VL Byte Enables $\overline{\text{VL\_BE[3:0]}}$ — These inputs indicate which byte lanes of the 32-bit data bus are involved with the current VL-Bus transfer.

VL Data Bus VL_DATA[31:0] — These inputs provide the bidirectional data path between the BtV2115 and the CPU. During a read transfer operation, the BtV2115 drives data onto VL_DATA[31:0]. Byte enables $\overline{\text{VL\_BE[3:0]}}$ determine which byte lane(s) are valid.

VL Address Strobe $\overline{\text{VL\_ADS}}$ — This input indicates that a new cycle has begun.

VL Reset $\overline{\text{VL\_RESET}}$ — This input is a master reset that is asserted after system power-up and before any valid CPU cycles begin. The $\overline{\text{VL\_RESET}}$ signal resets all functions on the BtV2115 before execution. The relationship between the rising and falling edges of the $\overline{\text{VL\_RESET}}$ and the phase of VL_CLOCK need not be guaranteed.

VL Clock VL_CLOCK — This clock signal is a 1x clock that follows the same phase as a 486-type CPU. The BtV2115 supports the VESA local bus at up to 40 MHz.

VL Memory or IO Status

VL_M_$\overline{IO}$ — This CPU output signal indicates the type of access currently executing on the VL-Bus. A high VL_M_$\overline{IO}$ indicates a memory cycle; a low VL_M_$\overline{IO}$ indicates an IO cycle.

VL Data or Code Status

VL_D_$\overline{C}$ — This status signal indicates whether the current transfer contains data or code.

VL Read or Write Status

VL_W_$\overline{R}$ — This CPU output signal indicates the type of access currently executing on the VL-Bus. A high VL_W_$\overline{R}$ indicates a write; a low VL_W_$\overline{R}$ indicates a read.

VL Identifier Pin

VL_ID2 — The VL-Bus controller is capable of handling a high speed zero wait state write transfer. The BtV2115 inserts a wait state when VL_ID2 is low.

VL Ready Return

$\overline{VL\_RDYRTN}$ — This signal tells the BtV2115 when the cycle has ended.

VL Local Device Select

$\overline{VL\_LDEV}$ — This output signal informs the VL-Bus controller that the BtV2115 recognizes the current cycle as an access to it.

VL Local Ready

$\overline{VL\_LRDY}$ — This input terminates the current active bus cycle and tells the VL-Bus controller that the bus is done. If VL_ID2 is set, the BtV2115 starts driving $\overline{VL\_LRDY}$ during the first T2 state for writes. Most write cycles are completed in two clock cycles and $\overline{VL\_LRDY}$ is driven low during the first T2 state. If VL_ID2 is cleared, the BtV2115 starts driving $\overline{VL\_LRDY}$ in the second T2 state.

Interrupt Request Line 9

IRQ9 — The IRQ9 line is a level-triggered interrupt that is electrically connected to IRQ9 on the ISA bus.

VL Bus Timing

This section presents timing diagrams for key VL waveforms. Below, Table 128 defines the minimum and maximum values for the specified parameters.

Table 128. VL-Bus Timing

	Symbol	Parameter	Max	Min	Units
VL_CLOCK (see Figure 22)	t_1	VL_CLOCK rise time	3	—	ns
	t_2	VL_CLOCK fall time	3	—	ns
	t_3	VL_CLOCK high period	60% of t_5	40% of t_5	ns
	t_4	VL_CLOCK low period	60% of t_5	40% of t_5	ns
	t_5	VL_CLOCK period	—	25	ns
$\overline{\text{VL\_ADS}}$ and $\overline{\text{VL\_LDEV}}$ (see Figure 23)	t_6	Address, status, $\overline{\text{VL\_ADS}}$ setup to VL_CLOCK	—	5	ns
	t_7	$\overline{\text{VL\_LDEV}}$ active delay from address, status (20pF loading)	20	—	ns
	t_8	$\overline{\text{VL\_LDEV}}$ inactive delay from address, status	20	—	ns
$\overline{\text{VL\_LRDY}}$ (see Figure 24)	t_9	$\overline{\text{VL\_LRDY}}$ active delay from VL_CLOCK	14	3	ns
	t_{10}	$\overline{\text{VL\_LRDY}}$ inactive delay from VL_CLOCK	14	3	ns
	t_{11}	$\overline{\text{VL\_LRDY}}$ high before high-impedance (Z)	60% of t_5	40% of t_5	ns
Data (see Figure 25)	t_{12}	Read data setup to VL_CLOCK	—	7	ns
	t_{13}	Read data hold from VL_CLOCK	—	2	ns

The following timing diagrams depict the associated VL time intervals. The variable Z represents high-impedance.

Figure 22. VL_CLOCK Timing Diagram

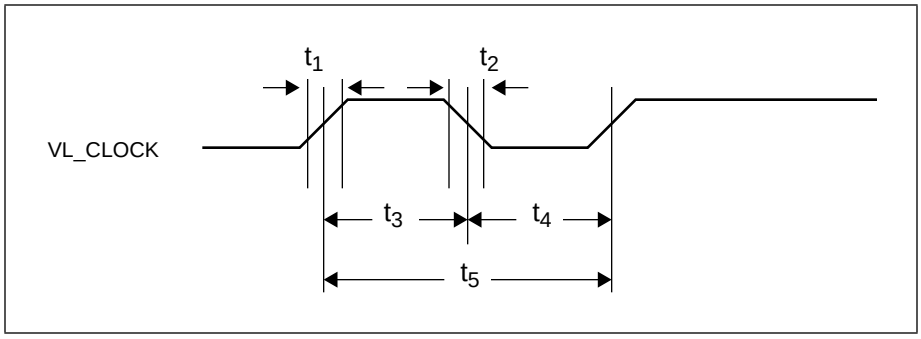


Figure 23. VL_ADS and VL_LDEV Timing Diagram

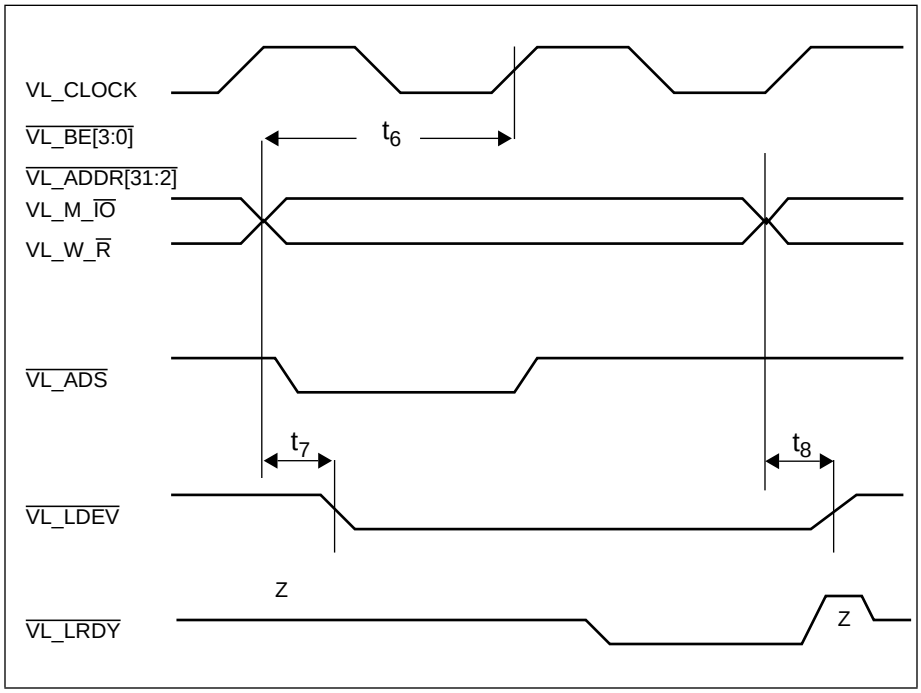


Figure 24. VL_LRDY Delay Timing Diagram

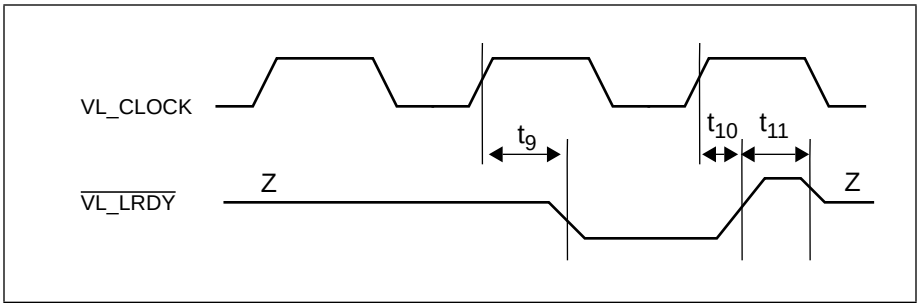
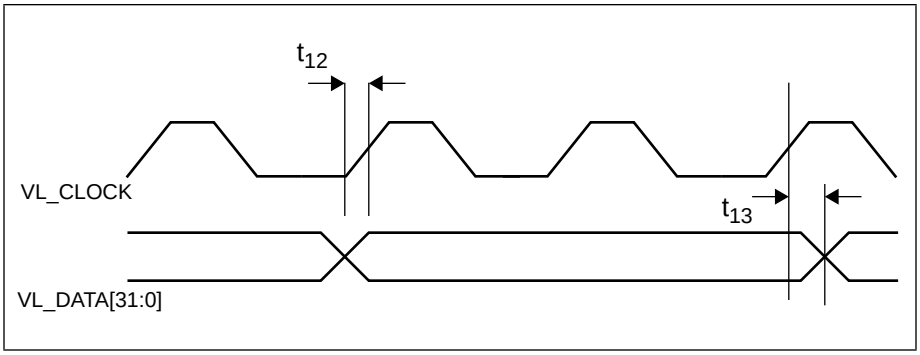


Figure 25. Data Timing Diagram



PCI Bus Interface

The PCI local bus is an architectural, timing, electrical, and physical interface that allows the BtV2115 to interface to the local bus of a host CPU.

The supported PCI bus cycles are as follows:

- I/O Read
- I/O Write
- Memory Read
- Memory Write
- Configuration Read
- Configuration Write
- Memory Read Multiple
- Memory Read Line
- Memory Write and Invalidate

Memory Write and Memory Write and Invalidate are treated in the same manner. Memory Read and Memory Read Multiple are treated in the same manner.

The unsupported PCI bus cycles are as follows:

- Interrupt Acknowledge
- Special Cycle
- Dual Address Cycle

All I/O, memory, and configuration writes are three clock cycles unless held off by a full host FIFO. The only two cycle writes are defined by GRP_CFG6[7], see Table 20 on page 48. Subsequent writes in the same transaction are zero wait states as long as the internal FIFO is not full.

The PCI Bus interface pins are defined in the following paragraphs. The overbar above a signal indicates active-LOW.

The term "sustained tri-state™" is used in this section. It is defined as an active-LOW, tri-state signal owned and driven by a single agent at a time. The agent that drives this pin low must drive it high for at least one clock before allowing it to float. A new agent cannot start driving a sustained tri-state signal sooner than one clock after the previous owner tri-states it. To sustain the inactive status until a new agent drives it, a pullup must be provided by the central resource.

PCI Address and Data Bus

PCI_AD[31:0] — This tri-state, bi-directional, IO pin handles both address and data information. A bus transaction consists of an address phase followed by one or more data phases for either read or write operations.

The address phase is the clock cycle in which $\overline{\text{PCI\_FRAME}}$ is first asserted. During the address phase, PCI_AD[31:0] contains a byte address for IO operations and a dword address for configuration and memory operations. During data phas-

es, PCI_AD[7:0] contains the least significant byte and PCI_AD[31:24] contains the most significant byte.

Read data is stable and valid when $\overline{\text{PCI\_TRDY}}$ is asserted and write data is stable and valid when $\overline{\text{PCI\_IRDY}}$ is asserted. Data is transferred during the clocks when both $\overline{\text{PCI\_TRDY}}$ and $\overline{\text{PCI\_IRDY}}$ are asserted.

PCI Bus Command and Byte Enables

$\overline{\text{C\_BE}}$[3:0] — These tri-state, bi-directional, IO pins handle both bus command and byte enable information. During the address phase of a transaction, $\overline{\text{C\_BE}}$[3:0] contain the bus command. During the data phase, $\overline{\text{C\_BE}}$[3:0] are used as byte enables. The byte enables are valid for the entire data phase and determine which byte lanes carry meaningful data. $\overline{\text{C\_BE}}$[3] refers to the most significant byte and $\overline{\text{C\_BE}}$[0] refers to the least significant byte.

PCI Parity

$\overline{\text{PCI\_PAR}}$ — This tri-state, bi-directional, IO pin provides even parity across PCI_AD[31:0] and $\overline{\text{C\_BE}}$[3:0]. This means that the number of 1's on $\overline{\text{PCI\_PAR}}$, PCI_AD[31:0], and $\overline{\text{C\_BE}}$[3:0] equals an even number.

$\overline{\text{PCI\_PAR}}$ is stable and valid one clock after the address phase. For data phases, $\overline{\text{PCI\_PAR}}$ is stable and valid one clock after either $\overline{\text{PCI\_TRDY}}$ is asserted on a read or $\overline{\text{PCI\_IRDY}}$ is asserted on a write. Once valid, $\overline{\text{PCI\_PAR}}$ remains valid until one clock after the completion of the current data phase. $\overline{\text{PCI\_PAR}}$ and PCI_AD[31:0] have the same timing, but $\overline{\text{PCI\_PAR}}$ is delayed by one clock. The BtV2115 drives $\overline{\text{PCI\_PAR}}$ for read data phases; the master drives $\overline{\text{PCI\_PAR}}$ for address and write data phases.

PCI Clock

PCI_CLK — This input provides timing for all PCI transactions. All PCI signals except $\overline{\text{PCI\_RESET}}$ and the optional interrupt are sampled on the rising edge of PCI_CLK, and all other timing parameters are defined with respect to this edge. The BtV2115 supports a PCI clock of up to 33 MHz.

PCI Reset

$\overline{\text{PCI\_RESET}}$ — This input resets all functions on the BtV2115 before execution. $\overline{\text{PCI\_RESET}}$ may be asynchronous to PCI_CLK when asserted or deasserted.

PCI Cycle Frame

$\overline{\text{PCI\_FRAME}}$ — This sustained tri-state signal is driven by the current master to indicate the beginning and duration of an access. $\overline{\text{PCI\_FRAME}}$ is asserted to signal the beginning of a bus transaction. Data transfer continues throughout assertion. At deassertion, the transaction is in the final data phase.

PCI Initiator Ready

$\overline{\text{PCI\_IRDY}}$ — This sustained tri-state signal indicates the bus master's readiness to complete the current data phase.

$\overline{\text{PCI\_IRDY}}$ is used in conjunction with $\overline{\text{PCI\_TRDY}}$. When both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted, a data phase is completed on that clock. During a read, $\overline{\text{PCI\_IRDY}}$ indicates when the initiator is ready to accept data. During a write, $\overline{\text{PCI\_IRDY}}$ indicates when the initiator has placed valid data on PCI_AD[31:0]. Wait cycles are inserted until both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted together.

PCI Target Ready

$\overline{\text{PCI\_TRDY}}$ — This sustained tri-state signal indicates the BtV2115's readiness to complete the current data phase.

$\overline{\text{PCI\_IRDY}}$ is used in conjunction with $\overline{\text{PCI\_TRDY}}$. When both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted, a data phase is completed on that clock. During a read, $\overline{\text{PCI\_TRDY}}$ indicates when the target is presenting data. During a write, $\overline{\text{PCI\_TRDY}}$ indicates when the target is ready to accept the data. Wait cycles are inserted until both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted together.

PCI Stop

$\overline{\text{PCI\_STOP}}$ — This sustained tri-state signal indicates the BtV2115 is requesting the master to stop the current transaction.

PCI Initialization Device Select

PCI_IDSEL — This input is used to select the BtV2115during configuration read and write transactions.

PCI Device Select

$\overline{\text{PCI\_DEVSEL}}$ — This sustained tri-state signal indicates device selection. The BtV2115 always asserts $\overline{\text{PCI\_DEVSEL}}$ in the third clock cycle, except in cases of two-clock writes.

When actively driven, $\overline{\text{PCI\_DEVSEL}}$ indicates the driving device has decoded its address as the target of the current access.

PCI Interrupt A

$\overline{\text{PCI\_INTA}}$ — This signal is an Open Drain interrupt output.

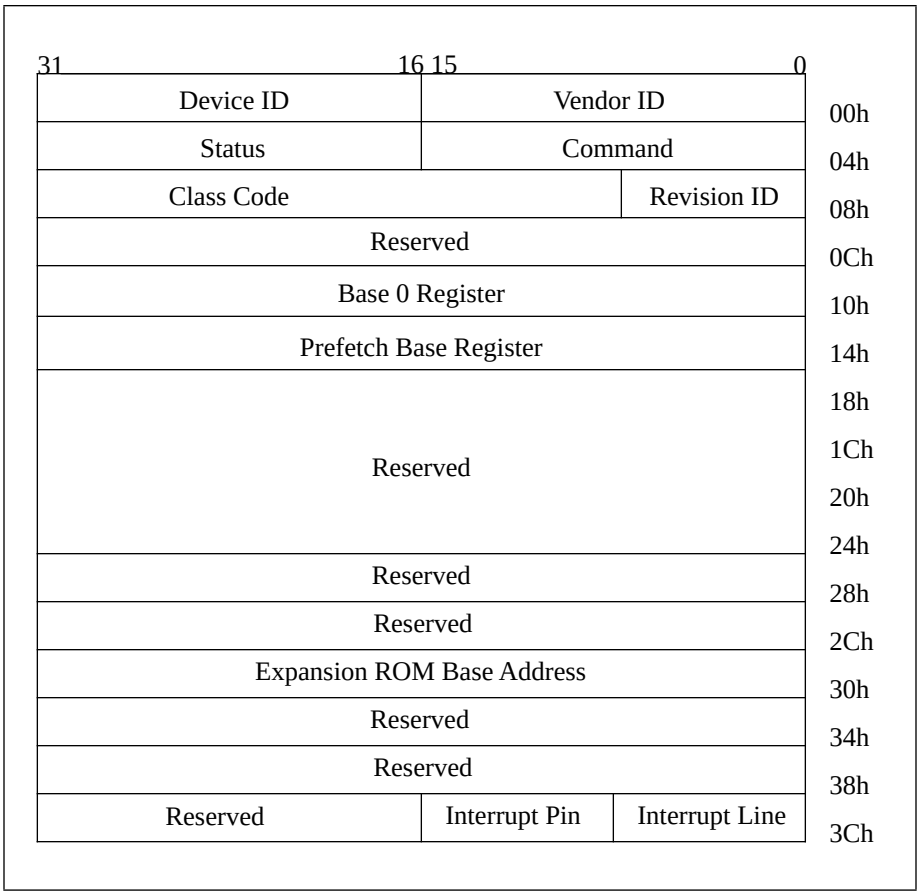
PCI High Speed Read

HSREAD — When the host is the destination, an input of 1 allows high speed read of GUI blit data. A 0 input disables the high speed read. This signal should be set to 1 only if the PCI bus clock is at or above 16 MHz, and should be stable during BtV2115 operation.

PCI Configuration Space

The PCI configuration space defines the registers used to interface between the host and the PCI local bus. This section defines the organization of the registers within the 64 byte predefined header portion of the configuration space. Figure 26 shows the configuration space header. For details on the PCI bus, refer to the *PCI Local Bus Specification, Revision 2*.

Figure 26. PCI Configuration Space Header



The BtV2115 is a multifunction device. Function 0 responds as a VGA Graphics Controller. Function 1 responds as an audio device. In the paragraphs below, assume that the two functions have the same definitions, unless specified.

The configuration space registers are stored in dwords and defined by byte addresses. Therefore a register one byte in length can have a bit definition other than 7:0 (for example 31:24), depending on its location in the configuration space. For a discussion on configuration cycle addressing, refer to Section 3.6.4.1 of the *PCI Local Bus Specification, Revision 2*.

All writable bits are reset to 0 by the system reset. After reset, the BtV2115 is disabled and will only respond to CFGWR and CFGRD cycles. All reserved and unimplemented registers are read as zeroes.

PCI Vendor ID Registers

name: PCI_VENDOR_ID
address: 01h:00h, read only
size: 16 bit
function: This register contains the unique vendor ID assigned to Brooktree Corporation. This field will always return the value of 109Eh.

PCI Device ID Registers

name: PCI_DEVICE_ID
address: 03h:02h, read only
size: 16 bit
function: This register contains the unique device ID assigned by Brooktree Corporation. This field will return the BtV MediaStream Controller model number 2115 in hexadecimal.

PCI Command Registers

name: PCI_COMMAND0, PCI_COMMAND1
address: 05h:04h, read/write as shown below
size: 16 bit
function: This register controls the ability to respond to PCI cycles. The bit contents for function 0 (PCI_COMMAND0) are shown in Table 129. The bit contents for function 1 (PCI_COMMAND1) are shown in Table 130.

Table 129. PCI Command Register - Function 0

Bit	Field	Detail
15:6		Read only. Contains all 0's.
5	PCI_SNOOP_DAC	1 = Enable snooping
4:2		Read only. Contains all 0's
1	PCI_MEM_EN	1 = Enable memory cycles 0 = Disable memory and ROM cycles
0	PCI_IO0_EN	1 = Enable IO cycles in the VGA address space or alternate address space

Table 130. PCI Command Register - Function 1

Bit	Field	Detail
15:1		Read only. Contains all 0's.
0	PCI_IO1_EN	1 = Enable IO cycles in the audio range

PCI Status Registers

name: PCI_STATUS
address: 07h:06h, read only
size: 16 bit
function: This register records status information for PCI bus related events. The bit contents are shown in Table 131.

Table 131. PCI Status Register

Bit	Field	Detail
31:27		Read only. Contains all 0's.
26:25	PCI_DEVSEL_ TIMING	Read only, set to 01. Medium, DEVSEL always in third clock or faster.
24		Returns 0
23	PCI_BK_TO_BK	Read only, set to 1, supports back-to-back cycles.
22:16		Read only. Contains all 0's.

PCI Revision ID Register

name: PCI_REVISION_ID
address: 08h, read only
size: 8 bit
function: This register contains the revision ID for the BtV2115. The value is 03h.

PCI Class Code Registers

name: PCI_CLASS0_ID, PCI_CLASS1_ID
address: 0B:09h, read only
size: 32 bit
function: This register identifies the function of the BtV2115. The bit contents for function 0 (PCI_CLASS0_ID) are shown in Table 132. The bit contents for function 1 (PCI_CLASS1_ID) are shown in Table 133.

Table 132. PCI Class Code Register - Function 0

Bit	Field	Detail
31:24		Display controller = 03h alternate decode = FFh
23:16		VGA - 00h
15:8		Contains all 0's.

Table 133. PCI Class Code Register - Function 1

Bit	Field	Detail
31:24		Multimedia device - 04h
23:16		Audio - 01h
15:8		Contains all 0's.

PCI Base Address Registers

name: PCI_BASE0_REG, PCI_BASE1_REG
address: 10h, read/write
size: 8 bit
function: The bit contents for PCI_BASE0_REG, function 0, is shown in Table 134. For function 1, PCI_BASE1_REG is not implemented and is read as 32'h00000000.

When the following conditions are met during a memory cycle, the BtV2115 is accessed: bits 31:25 are non-zero and match bits 31:25 of the GUI base address, and memory cycles are enabled in the PCI Command Register 0 (see Table 129 on page 220). Non-zero writes to this register will cause the assertion of the protected mode enable bit (GRP_GUI_BASE[24]). 32MB of CPU space is reserved for BtV2115 operations.

For more information on the address space, refer to "CPU Address Space Apertures" starting on page 97.

Table 134. PCI Base Register - Function 0

Bit	Detail
31:25	If all zeroes, BtV2115 not accessible via memory cycles in a non-prefetchable range. If non-zero, specifies the address of the 32MB GUI base aperture space (PM_BASE). Refer to "GUI Base Address Register" on page 123.
24:4	Read only, all 0's
3	Read only. 0 indicates not prefetchable.
2:1	Read only. 00 indicates locatable anywhere in 32-bit address space.
0	Read only. 0 specifies memory space

PCI Prefetchable Base Address Registers

name: PCI_PREF_BASE0, PCI_PREF_BASE1
address: 14h, read/write
size: 8 bit
function: The bit contents for PCI_PREF_BASE0, function 0, is shown in Table 135. For function 1, PCI_REF_BASE1 is not implemented and reads back 32'h00000000.

When the following conditions are met during a memory cycle, the BtV2115 is accessed: bits 31:25 are non-zero and match bits 31:25 of the GUI base address, and memory cycles are enabled in the PCI Command Register 0 (see Table 129 on page 220). Non-zero writes to this register will cause the assertion of the protected mode enable bit (GRP_GUI_BASE[24]). 32MB of CPU space is reserved for BtV2115 operations.

For more information on this address space, refer to "CPU Address Space Apertures" starting on page 97. GRP_PBASE is the VGA register that reads the address specified by bits [31:25]; refer to "Read PCI Prefetchable Base Address Register" on page 59.

Table 135. PCI Prefetch Base Register - Function 0

Bit	Detail
31:25	If all zeroes, BtV2115 not accessible via memory cycles in a non-prefetchable range. If non-zero, specifies the address of the 32MB GUI base aperture space (PM_BASE). Refer to "GUI Base Address Register" on page 123.
24:4	Read only, all 0's
3	Read only. 1 indicates prefetchable.
2:1	Read only. 00 indicates locatable anywhere in 32-bit address space.
0	Read only. 0 specifies a memory space indicator

PCI ROM Base Address Register

name: PCI_ROM_BASE0, PCI_ROM_BASE1
address: 33h:30h, read/write
size: 32 bit
function: These registers specify a 16MB space for the BtV2115 to respond to ROM cycle requests. These registers are valid for function 0 only. For function 1, these registers are not implemented and read back as 32'h00000000. The bit contents for PCI_ROM_BASE0, function 0, is shown in Table 136. GRP_ROMBASE is the VGA register that reads the address specified by bits [31:24]; refer to "Read PCI ROM Base Address Register" on page 59.

Table 136. PCI ROM Base Register - Function 0

Bit	Detail
31:24	Specifies a 16MB space for ROM cycles.
23:1	Reserved
0	PCI_ROM_EN, Read/write 1 = ROM cycles enabled 0 = ROM cycles disabled

PCI Interrupt Line Register

name: PCI_INT_LINE[7:0]
address: 3Ch, read/write
size: 8 bit
function: This register specifies which system interrupt controller input is connected to the BtV2115's interrupt pin. This register is valid for function 0 only. For function 1, this register is not implemented and is read as 8'h00.

PCI Interrupt Pin Register

name: PCI_INT0_PIN, PCI_INT1_PIN
address: 3Dh, read only
size: 8 bit
function: For function 0, this register returns the value 8'h01, which indicates a response on $\overline{\text{PCI\_INTA}}$ only. For function 1, this register is not implemented and returns 8'h00.

VIDEO RAM INTERFACE

Introduction

The BtV2115 provides data steering, addressing, timing, and control for an external memory bus to which 1 - 4 MB of dual ported video RAM, ROM, and an optional Yamaha™ FM Synthesizer are attached. The memory bus provides access to the multimedia data and control information stored in VRAM, as well as ROM and Yamaha data.

The BtV2115 attaches to the parallel port of the VRAM array via the memory bus. The BtV2115 does *not* attach to the VRAM serial data port, even though it controls all the accesses to VRAM, both parallel and serial (between the VRAM serial port and the BtV2487 PACDAC). In addition, the BtV2115 provides the ability to write Flash ROM to simplify BIOS upgrades.

An internal BtV2115 bus is used to steer data between the external memory bus, the multi-media bus (BtV2811A and BtV2300), and internal BtV2115 sub-modules. These internal sub-modules provide the functionality of BtV2115, including buffering and steering of data between the CPU bus and the other components.

Several types of VRAM are supported. The type of VRAM used in the subsystem design is configurable at the BtV2115 pins via strapable bits. The VRAM type determines the VRAM parameter bits (such as speed, size, and Write/CAS configuration) in the configuration registers. These initial values can be overridden, if necessary, by software.

VRAM Signal Description

The external memory bus includes 32 bidirectional (tristate) data lines, 9 address lines, 4 RAS (Row Address Strobe) lines, 6 control lines to generate CAS (Column Address Strobe) and WE (Write Enable), 1 VRAM output enable, and 1 VRAM special function select signal. The VRAM data lines double as configuration strapping bits, with MDATA optionally tied through 20K ohm external resistors to V_{DD} or GND and sampled at reset to provide board-dependent initial values for many of the Configuration Register bits. (Note: PCI-bus attachment uses a different method of setting initial values.)

The CAS[3:0] and WE[1:0] pins on the BtV2115 are bimodal. Based on the VRAM type configured, the BtV2115 maps the column address strobe and write enable functions to the appropriate BtV2115 pins to allow the same board wiring for either dual CAS or dual WE memories in the same board design without jumpers or straps in the memory subsystem.

The naming convention for the BtV2115 assumes the “dual CAS” mode, however the CAS[3:0] and WE[1:0] pins have different functions in dual CAS or dual we mode. Below, Table 137 summarizes the VRAM dual WE/dual CAS modes. Table 138 provides the VRAM bus signals.

Table 137. VRAM Dual WE/Dual CAS Modes

BtV2115 Pin Name	Function: Dual CAS Mode	Function: Dual WE Mode
$\overline{CAS0}$	Column address strobe byte 0	Write enable byte 0
$\overline{CAS1}$	Column address strobe byte 1	Column address strobe byte 0 & 1
$\overline{CAS2}$	Column address strobe byte 2	Write enable byte 2
$\overline{CAS3}$	Column address strobe byte 3	Column address strobe byte 2 & 3
$\overline{WE0}$	Write enable byte 0 & 1	Write enable byte 1
$\overline{WE1}$	Write enable byte 2 & 3	Write enable byte 3

Table 138. VRAM Bus Signals

Name	#	Description	I/O
MDATA[31:8]	24	Bidirectional tri-state VRAM memory data, address lines for Flash ROM and register selects for Yamaha FM synthesizer are multiplexed on MDATA[27:8].	I/O
MDATA[7:0]	8	Bidirectional tri-state VRAM , ROM, and Yamaha data.	I/O
MADDR[8:0]	9	Multiplexed row and column address.	O
$\overline{\text{RAS}}$[3:0]	4	VRAM Row Address Strobe (RAS).	O
$\overline{\text{CAS0}}$	1	Dual CAS: VRAM Column Address Strobe (CAS) for the least significant byte of the 32 bit data bus.	O
		Dual WE: Write Enable (WE) for byte0.	
$\overline{\text{CAS1}}$	1	Dual CAS: VRAM Column Address Strobe (CAS) for byte1.	O
		Dual WE: $\overline{\text{CAS}}$ for least significant 16 data bits (byte1 and byte0).	
$\overline{\text{CAS2}}$	1	Dual CAS: VRAM Column Address Strobe (CAS) for byte2.	O
		Dual WE: Write Enable (WE) for byte2.	
$\overline{\text{CAS3}}$	1	Dual CAS: VRAM Column Address Strobe (CAS) for byte3.	O
		Dual WE: CAS for most significant 16 data bits (byte3 and byte2).	
$\overline{\text{WE0}}$	1	Dual CAS: WE for least significant 16 data bits (byte1 and byte0). Also used as write control for Flash ROM and OPLx.	O
		Dual WE: Write Enable (WE) for byte1. Also used as write control for Flash ROM and OPLx.	
$\overline{\text{WE1}}$	1	Dual CAS: WE for most significant 16 data bits (byte3 and byte2).	O
		Dual WE: Write Enable (WE) for byte3.	
$\overline{\text{TRG}}$	1	VRAM Transfer Enable. $\overline{\text{TRG}}$ is also used as read control for Flash ROM and OPLx.	O
DSF	1	VRAM Special Function Select.	O

VRAM Configurability

In order to simplify board design with various VRAM configurations, the BtV2115 is implemented with internal configuration registers containing memory parameters, including different memory speeds (80, 70 or 60 nanosecond parts), 8 or 16 millisecond refresh period, etc. These configuration bits are initialized based on the VRAM type which is set at reset by strapping MDATA[15:10] using 20k ohm resistors. The configuration registers which are initialized using this strapping mechanism are also accessible by the host CPU. It is possible to alter the powerup configuration after reset via the BIOS initialization.

VRAM Types Supported

Table 139 describes the types of VRAM supported. The Dual CAS and 2Mbit CBR values are set by strap resistors. The other values are set by BIOS. Refer to “Configuration Registers” on page 133.

Table 139. Supported VRAM Types *(1 of 2)*

Part#	Vendor	Dual CAS	RAM Speed	2 Mbit CBR	SAM Length	Block Mode
TMS55160-80	Texas Instruments	1	00	0	0	1
TMS55160-70	Texas Instruments	1	01	0	0	1
TMS55160-60	Texas Instruments	1	10	0	0	1
TMS55165-80	Texas Instruments	0	00	0	0	1
TMS55165-70	Texas Instruments	0	01	0	0	1
TMS55165-60	Texas Instruments	0	10	0	0	1
IBM025160L-70	IBM	1	01	0	0	0
IBM025160L-80	IBM	1	00	0	0	0
IBM025161L-70	IBM	1	01	0	0	0
IBM025161L-60	IBM	1	10	0	0	0
UPD482445-80 UPD482444-80	NEC	0	00	0	1	0
UPD482445-70 UPD482444-70	NEC	0	01	0	1	0
UPD482445-60 UPD482444-60	NEC	0	10	0	1	0
HM538253-80	Hitachi	0	00	0	1	1
HM538253-70	Hitachi	0	01	0	1	1
KM428C256-80	Samsung	0	00	0	1	1
KM428C256-70	Samsung	0	01	0	1	1
KM428C256-60	Samsung	0	10	0	1	1
KM428C257-80	Samsung	0	00	0	1	1
KM428C257-70	Samsung	0	01	0	1	1
KM428C257-60	Samsung	0	10	0	1	1
TC528257-80	Toshiba	0	00	0	1	1
TC528257-70	Toshiba	0	01	0	1	1

Table 139. Supported VRAM Types *(2 of 2)*

Part#	Vendor	Dual CAS	RAM Speed	2 Mbit CBR	SAM Length	Block Mode
UPD482234-80	NEC	0	00	0	1	1
UPD482234-70	NEC	0	01	0	1	1
UPD482235-80	NEC	0	00	0	1	1
UPD482235-70	NEC	0	01	0	1	1

VRAM Parallel Data Cycle Types Supported

The BtV2115 is designed to control a BtV MediaStream Family of chips. In the design, the timing specifications of the supported VRAM types are satisfied, within the load limitations for the subsystem implementation (see "Operating Specifications" on page 317 for the design specifications). The types of VRAM cycles which are generated by the BtV2115 are shown in the following diagrams. The diagrams show relative timings; the internal 50 MHz clock is shown for reference.

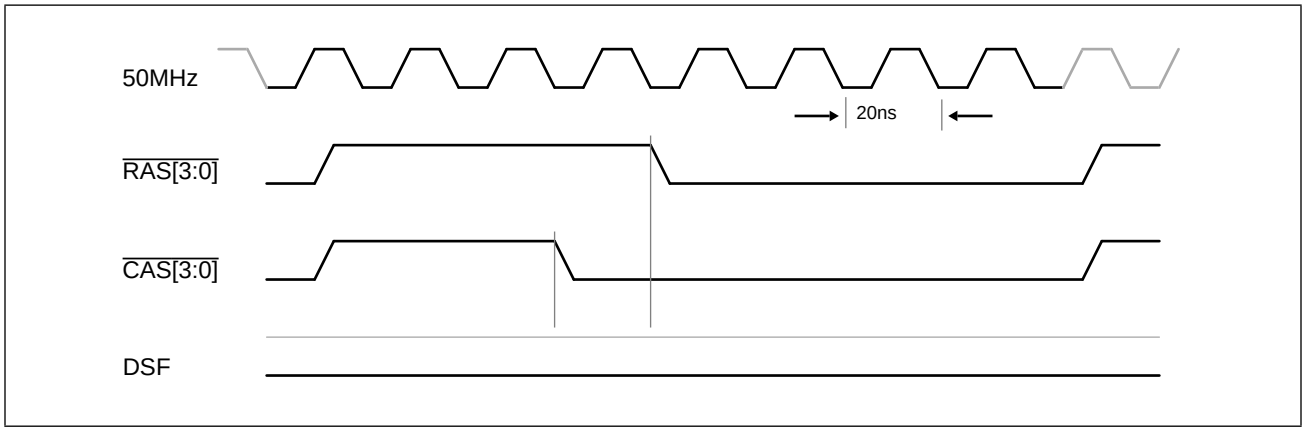
The diagrams which follow use a signal naming convention that assumes the BtV2115 is configured for "dual CAS" mode. The relative timings for the Column Address Strobe and Write Enable functions are accurate for both the dual CAS and dual write enable modes.

In the following timing diagrams, the term "DRAM" refers to the DRAM portion of the VRAM.

REFRESH Cycle

The BtV2115 issues CAS-Before-RAS Refresh (CBRR) Cycles with Option Reset.

Figure 27. VRAM Refresh Cycle



DRAM Read 1 to 4 bytes transferred from memory to the MDATA bus.

Figure 28. VRAM Read Cycle (80ns cycle timing)

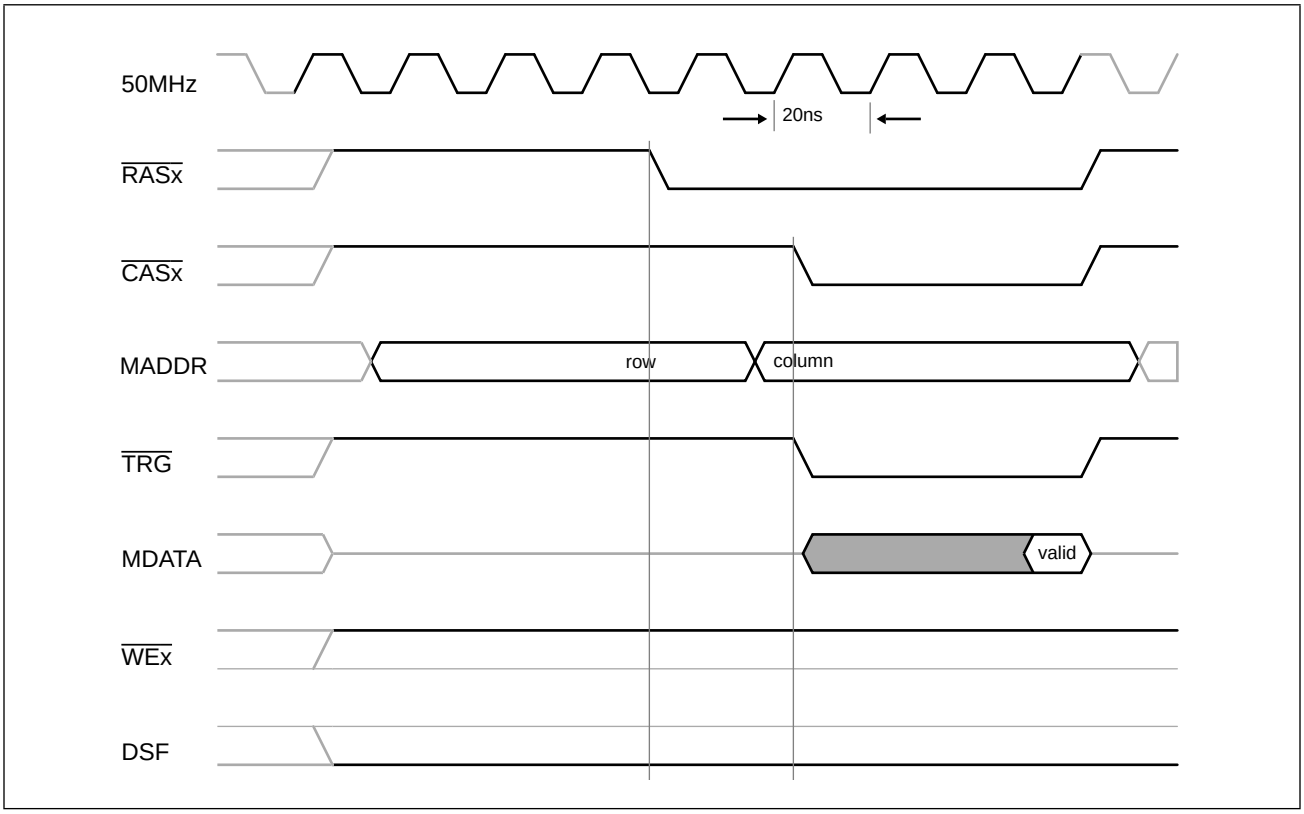


Figure 29. VRAM Read Cycle (70ns cycle timing)

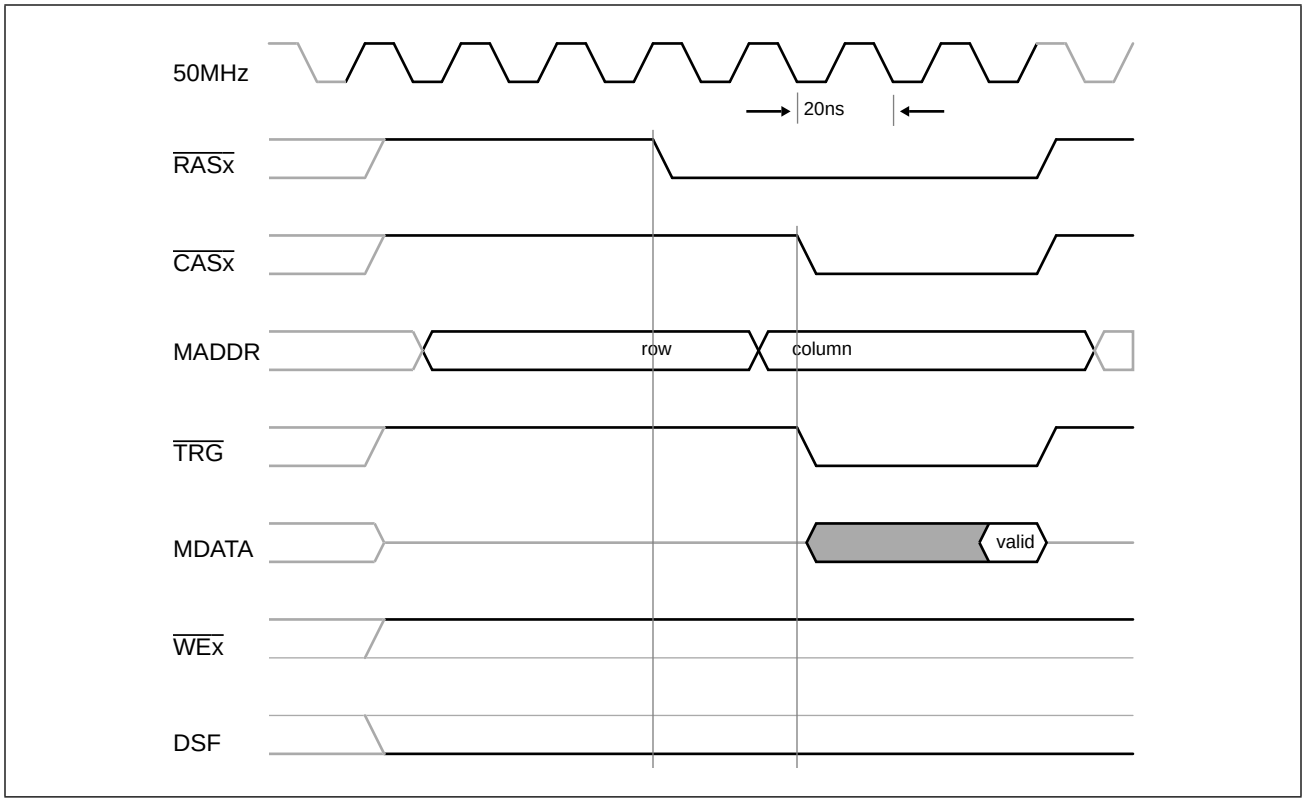
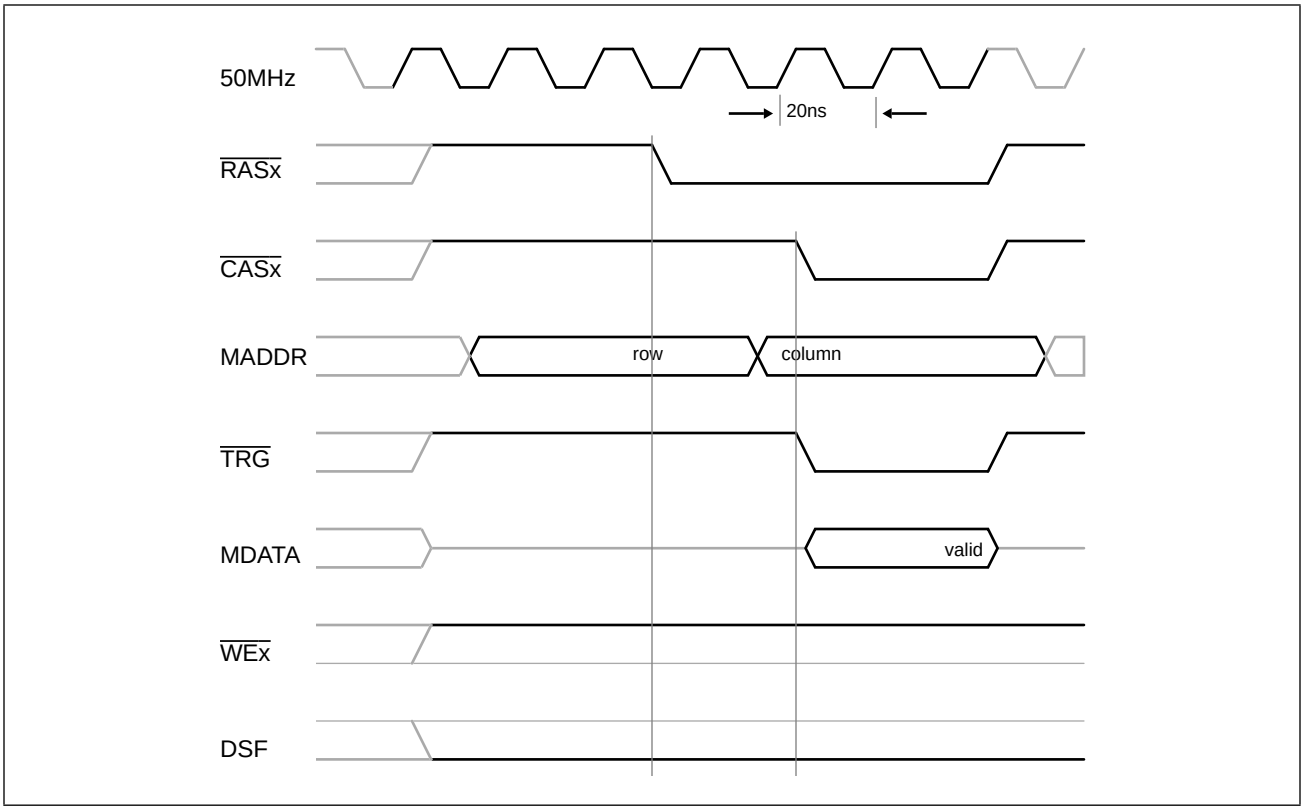


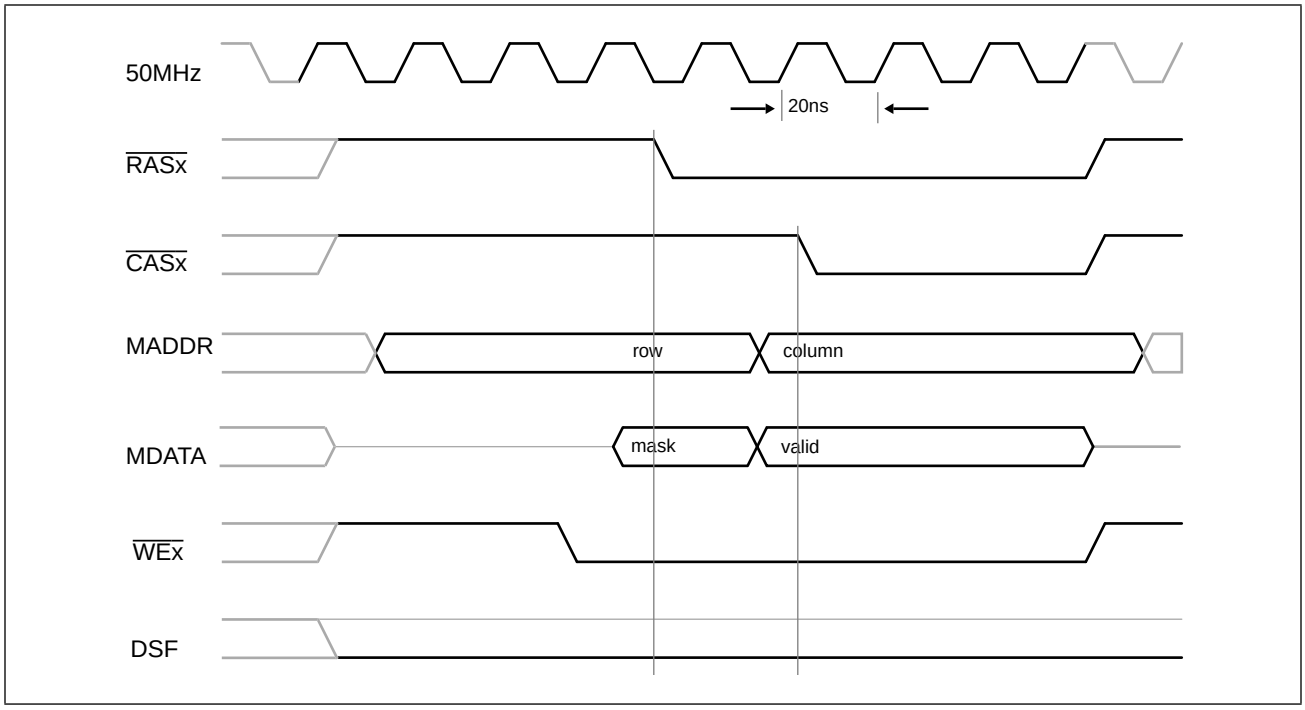
Figure 30. VRAM Read Cycle (60ns cycle timing)



DRAM Write

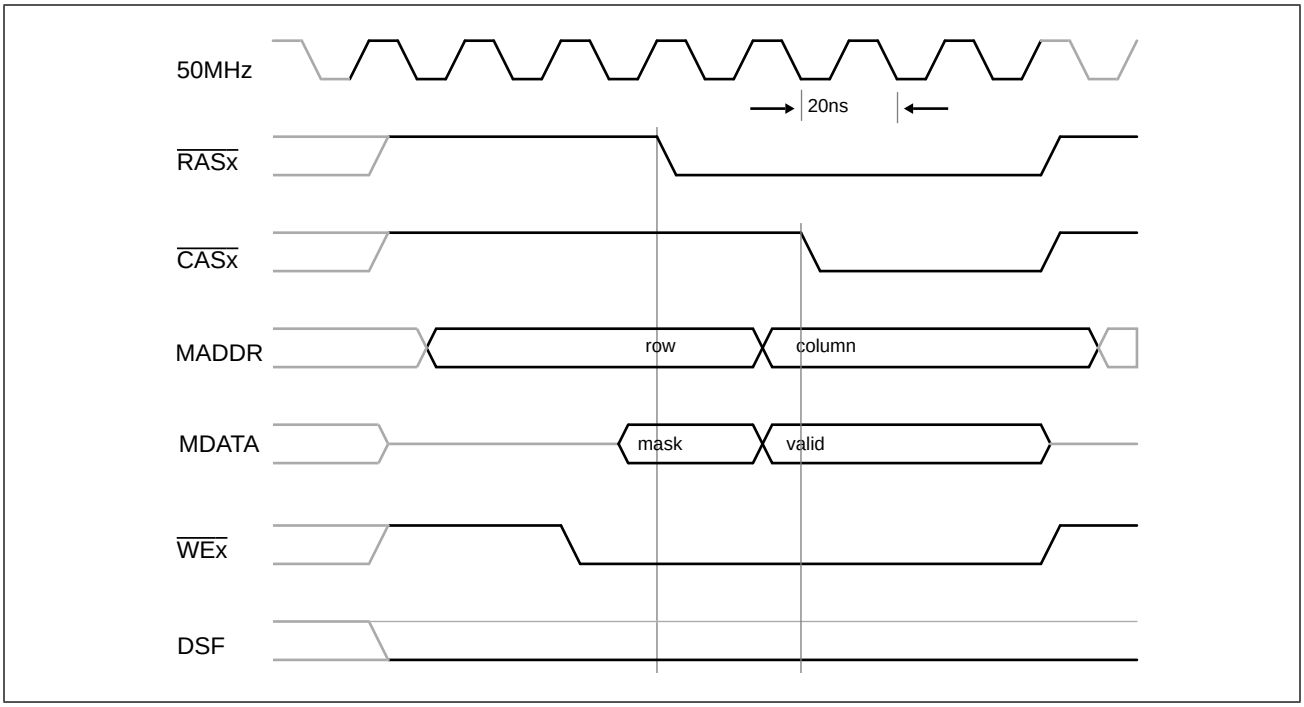
1 to 4 bytes transferred from the MDATA bus to memory. Non-block BtV2115 writes use Masked (Write-per-bit) mode exclusively (selected by $\overline{WE}$=0 at $\overline{RAS}$ fall). In Write-per-bit mode, the VRAM ANDs a mask with the data bits, allowing selective writing of bits. This mask can be provided through an internal mask register in VRAM, or by sampling the data lines at $\overline{RAS}$ fall. The BtV2115 does not support use of the internal VRAM mask register. The BtV2115 has its own mask register addressable in I/O space. This mask is used whenever bit level writes are needed, and is set to all ones for other writes.

Figure 31. DRAM Masked Write Cycle (80ns cycle timing)



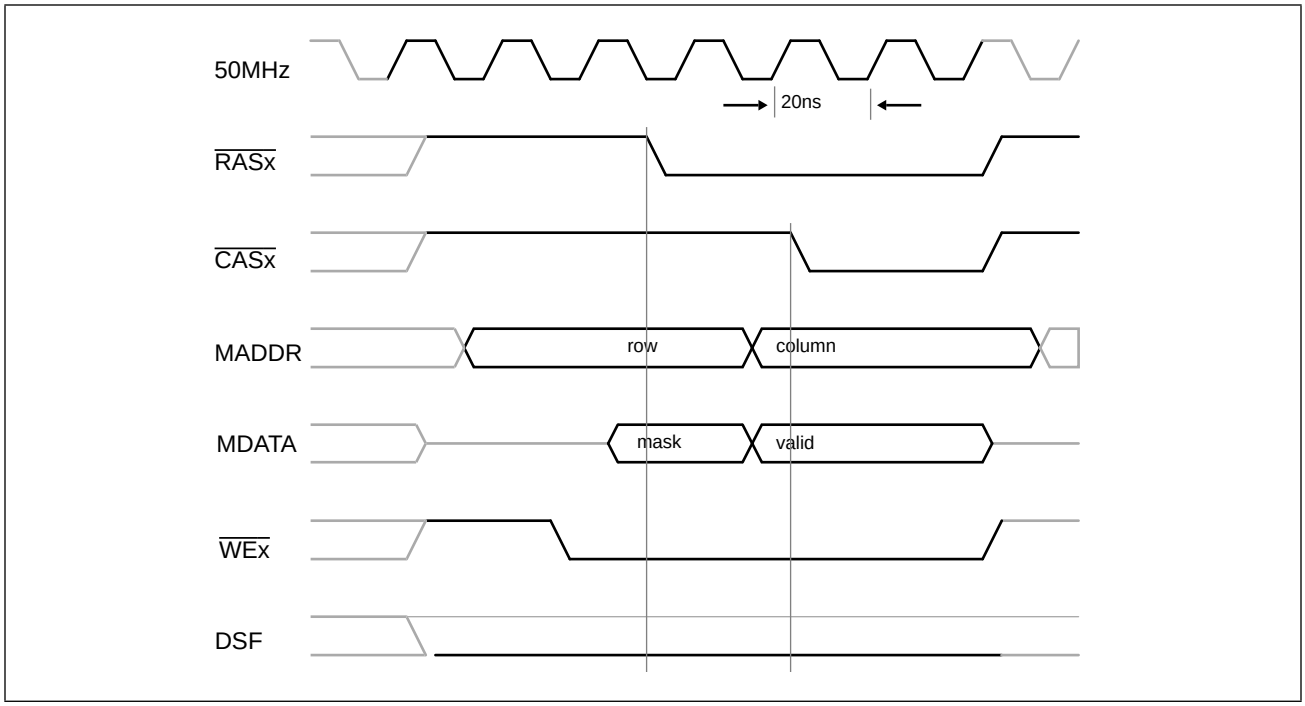
The BtV2115 uses only non-persistent write-per-bit mode ($\overline{WE}$=0 at RAS fall) for writes. The VRAM persistent mask register is not written; a mask value is driven by the BtV2115 (and forced to all ones during “normal” cycles when write-per-bit is not needed).

Figure 32. DRAM Masked Write Cycle (70ns cycle timing)



The BtV2115 uses only non-persistent write-per-bit mode ($\overline{WE}$=0 at RAS fall) for writes. The VRAM persistent mask register is not written; a mask value is driven by the BtV2115 (and forced to all ones during “normal” cycles when write-per-bit is not needed).

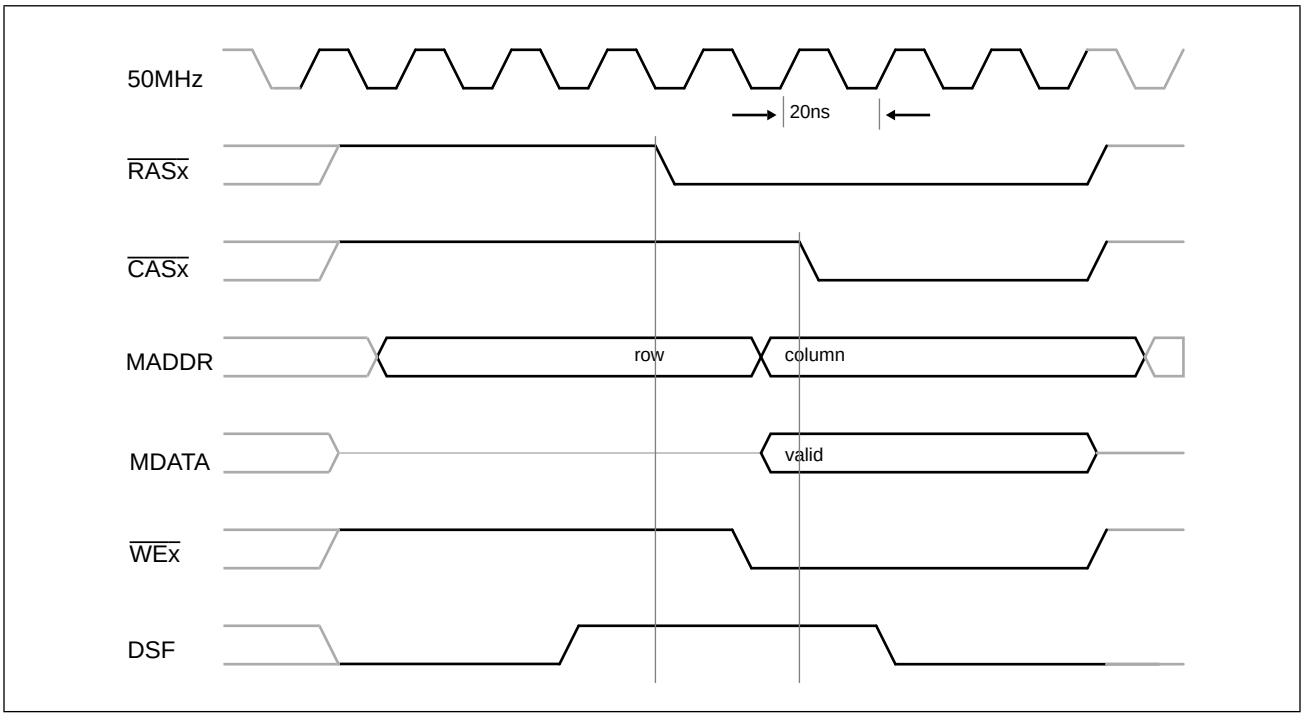
Figure 33. DRAM Write Cycle (60ns cycle timing)



The BtV2115 uses only non-persistent write-per-bit mode ($\overline{WE}$=0 at RAS fall) for writes. The VRAM persistent mask register is not written; a mask value is driven by the BtV2115 (and forced to all ones during “normal” cycles when write-per-bit is not needed).

Load Color Register The Color Register contents are used during Block Writes.

Figure 34. Load Color Register Cycle (all speeds)



$\overline{WE}$=1 and $\overline{DSF}$=1 at $\overline{RAS}$ fall causes a VRAM register load, and DSF=1 at $\overline{CAS}$ fall selects the Color Register. The MDATA value present at CAS fall is then loaded into the VRAM Color Register for later use in Block Write Cycles. The BtV2115 does not use the internal VRAM mask register selectable when DSF=0 at $\overline{CAS}$ fall.

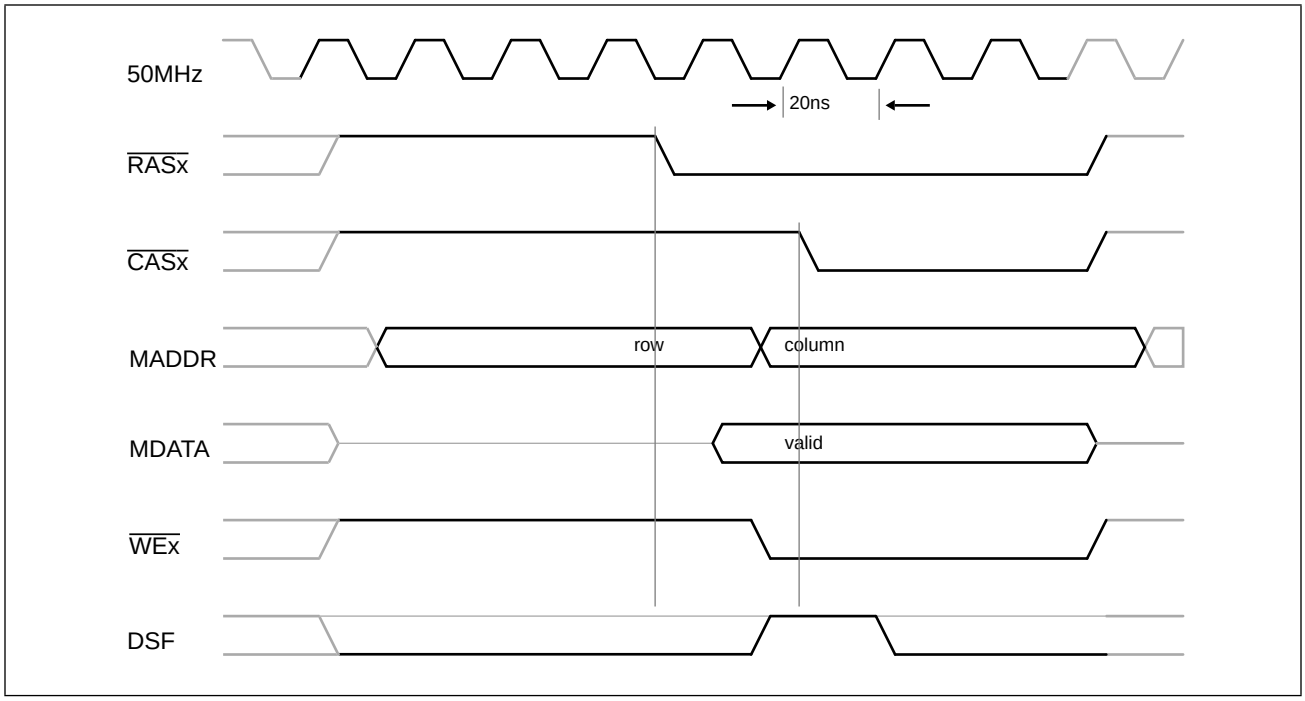
DRAM Non-masked Block Write Used to write the previously loaded Color Register value into DRAM.

Micron Block Mode increases eightfold the number of locations concurrently written to VRAM by affecting a block of 8 columns (instead of a single column). This is made possible by using each enabled byte transferred from the MDATA bus (at normal write data sample time) as a column mask, controlling 8 consecutive columns (ignoring the low 3 column address bits sampled at $\overline{CAS}$ fall, instead of selecting 1 of 8 columns). Each byte of column mask affects the block of 8 columns for a group of 8 data bits.

TI Block Mode quadruples the number of locations concurrently written to VRAM by affecting a block of 4 columns (instead of a single column). This is made possible by using the halves of each enabled byte transferred from the MDATA bus (at normal write data sample time) as a column mask, controlling 4 consecutive columns (ignoring the low 2 column address bits sampled at $\overline{CAS}$ fall, instead of selecting 1 of 4 column). Each half byte of column mask affects the block of 4 columns for a group of 4 data bits (1 quadrant).

Non-masked Mode is selected by $\overline{TRG}$=1, $\overline{WE}$=1, and DSF=0 at $\overline{RAS}$ fall. Block Mode is then selected by DSF=0 at $\overline{CAS}$ fall.

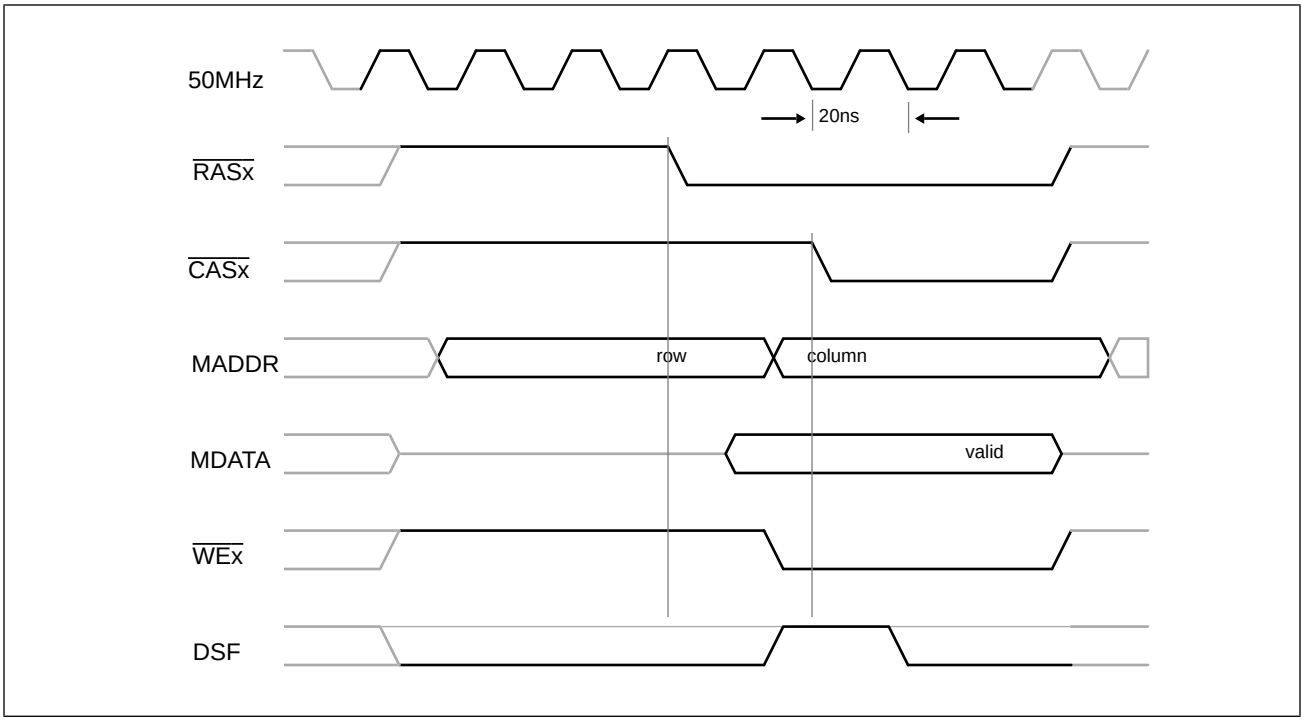
Figure 35. DRAM Non-Masked Block Write Cycle (80ns cycle timing)



Masked Write (Write-per-bit) Mode is selected when $\overline{WE}$=1 and DSF=0 at $\overline{RAS}$ fall of a non-refresh DRAM cycle.

Block Mode is selected when DSF=1 at $\overline{CAS}$ fall during a non-refresh DRAM cycle.

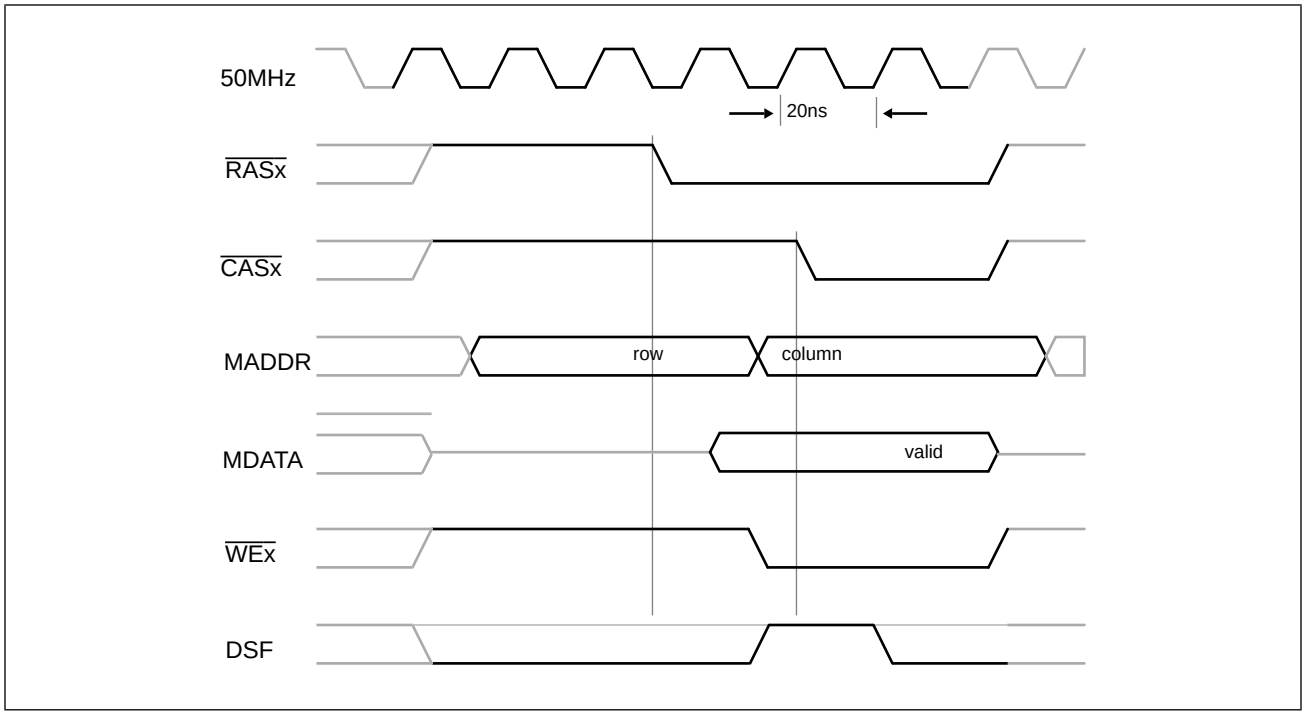
Figure 36. DRAM Non-Masked Block Write Cycle (70ns cycle timing)



Masked Write (Write-per-bit) Mode is selected when $\overline{WE}$=1 and DSF=0 at $\overline{RAS}$ fall of a non-refresh DRAM cycle.

Block Mode is selected when DSF=1 at $\overline{CAS}$ fall during a non-refresh DRAM cycle.

Figure 37. DRAM Non-Masked Block Write Cycle (60ns cycle timing)



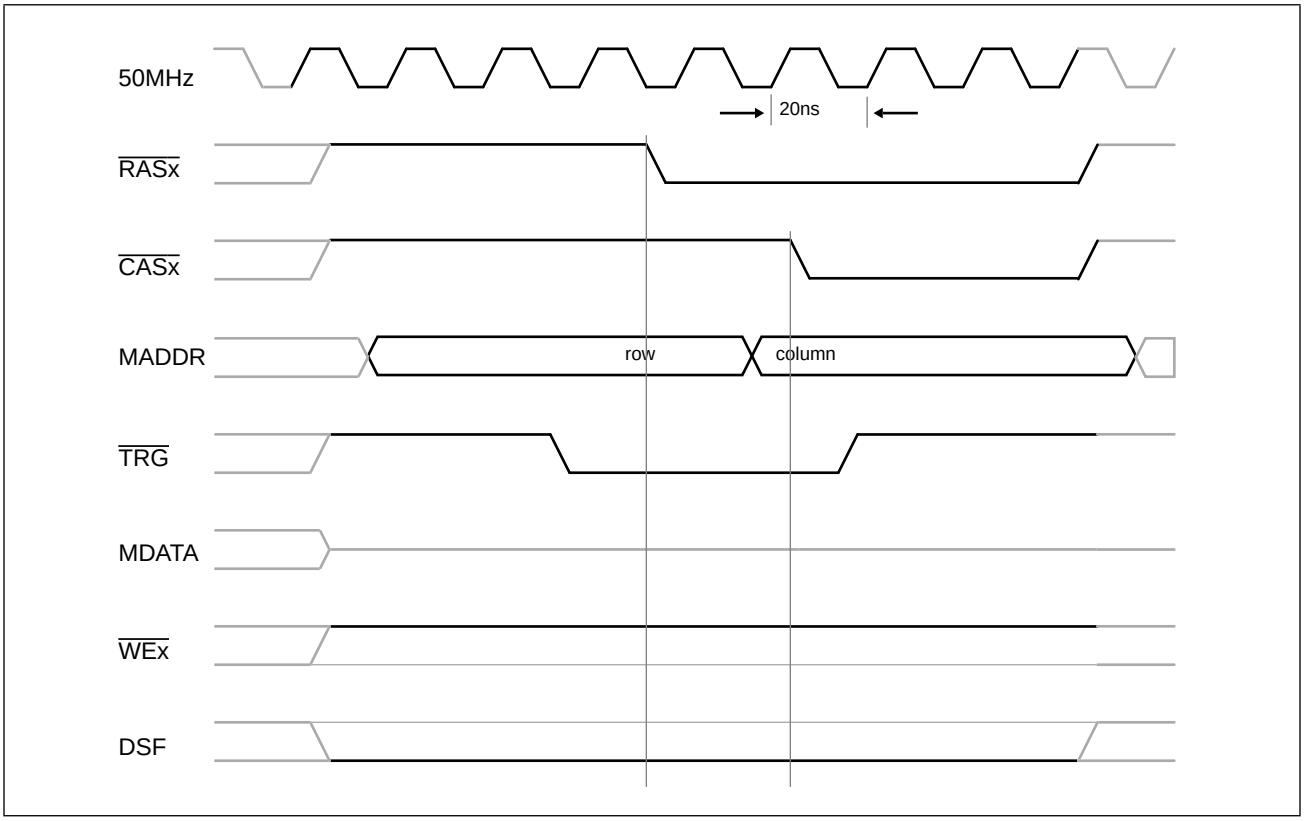
Masked Write (Write-per-bit) Mode is selected when $\overline{WE}$=1 and DSF=0 at $\overline{RAS}$ fall of a non-refresh DRAM cycle.

Block Mode is selected when DSF=1 at $\overline{CAS}$ fall during a non-refresh DRAM cycle.

Register Transfer (DRAM to SAM)

Data transferred (internal to the VRAM) from DRAM to the SAM Serial Data Register. Selected by $\overline{TRG}$=0 (and $\overline{CAS}$=1) and DSF=0 at $\overline{RAS}$ fall. The SAM data can then be output through the VRAM serial data bus to BtV2487.

Figure 38. DRAM Memory-to-Register Transfer Cycle (80ns cycle timing)



$\overline{TRG}$=0 and $\overline{WE}$=1 at RAS fall during a non-refresh cycle selects the operation which transfers a data row from DRAM into the VRAM internal serial data register.

Figure 39. VRAM Memory-to-Register Transfer Cycle (70ns cycle timing)

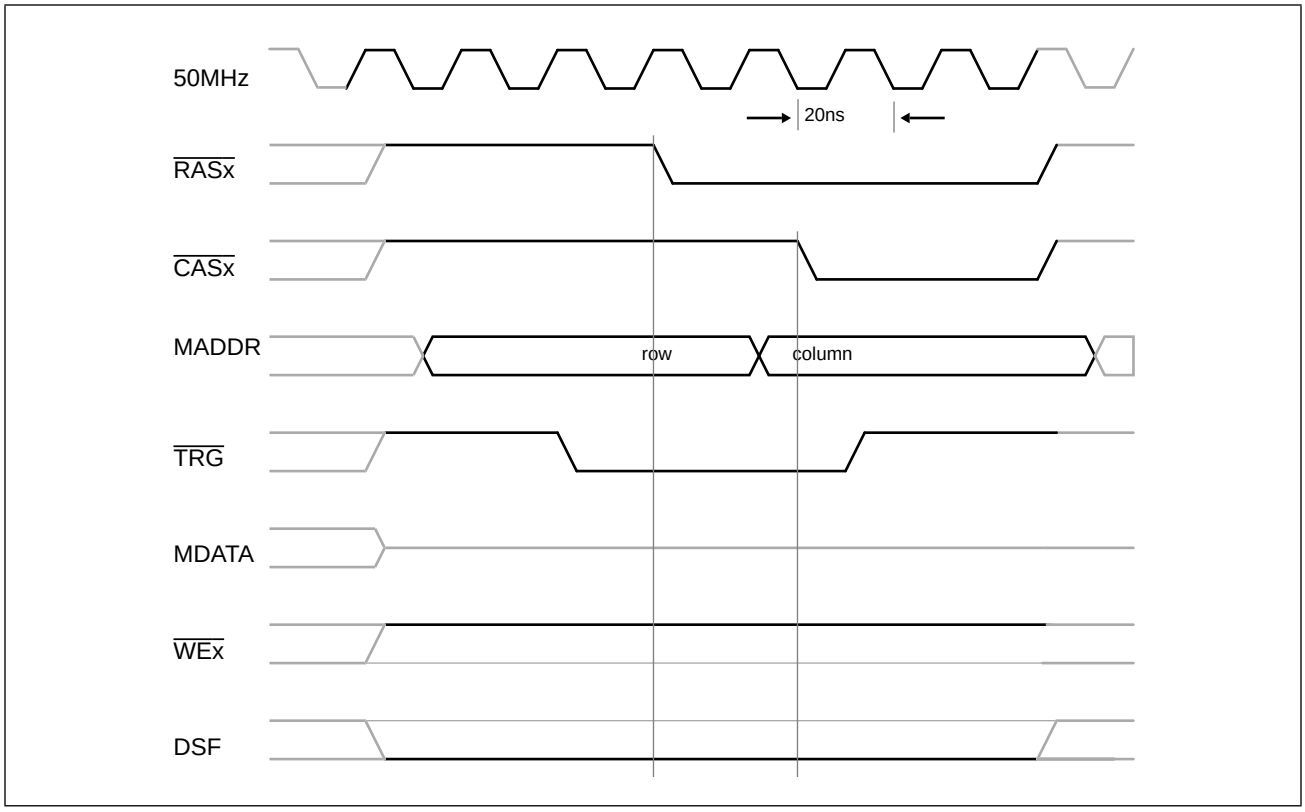
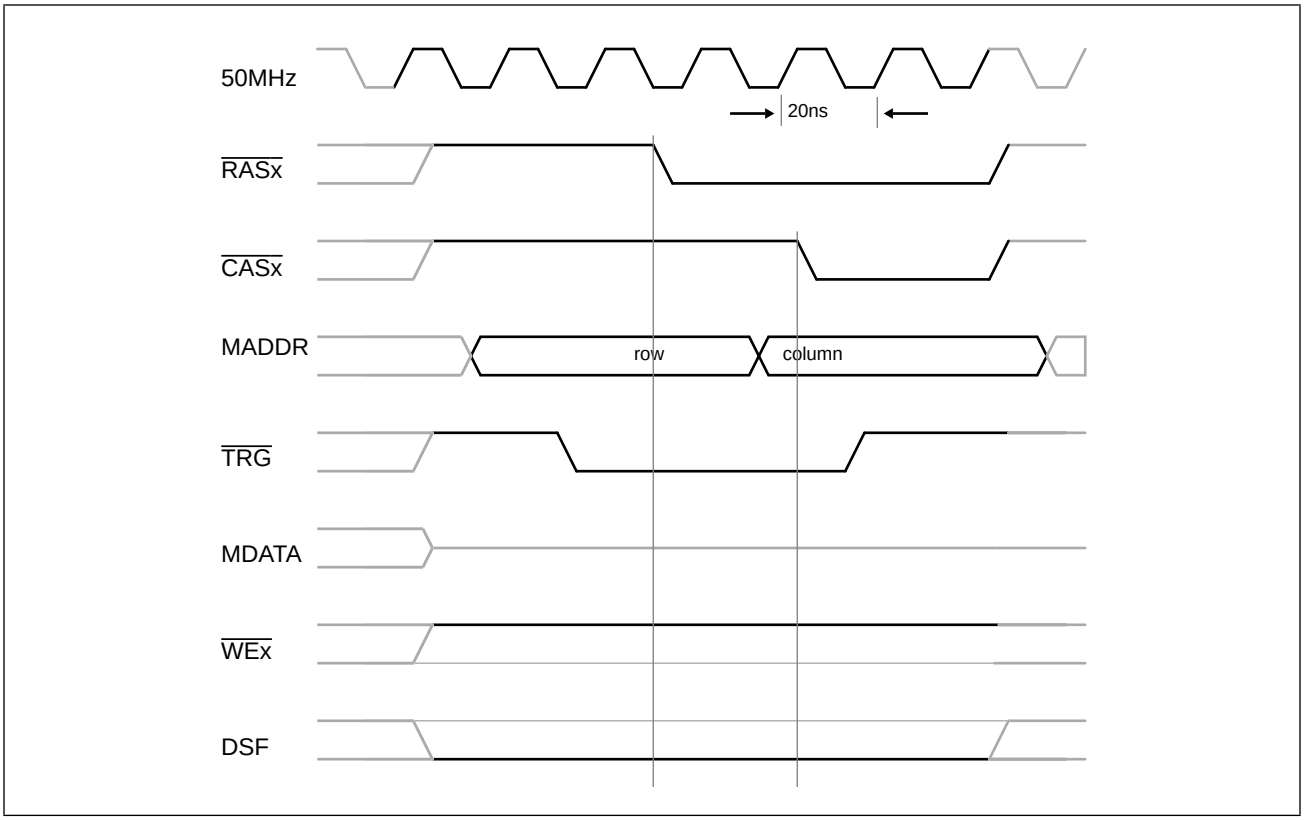


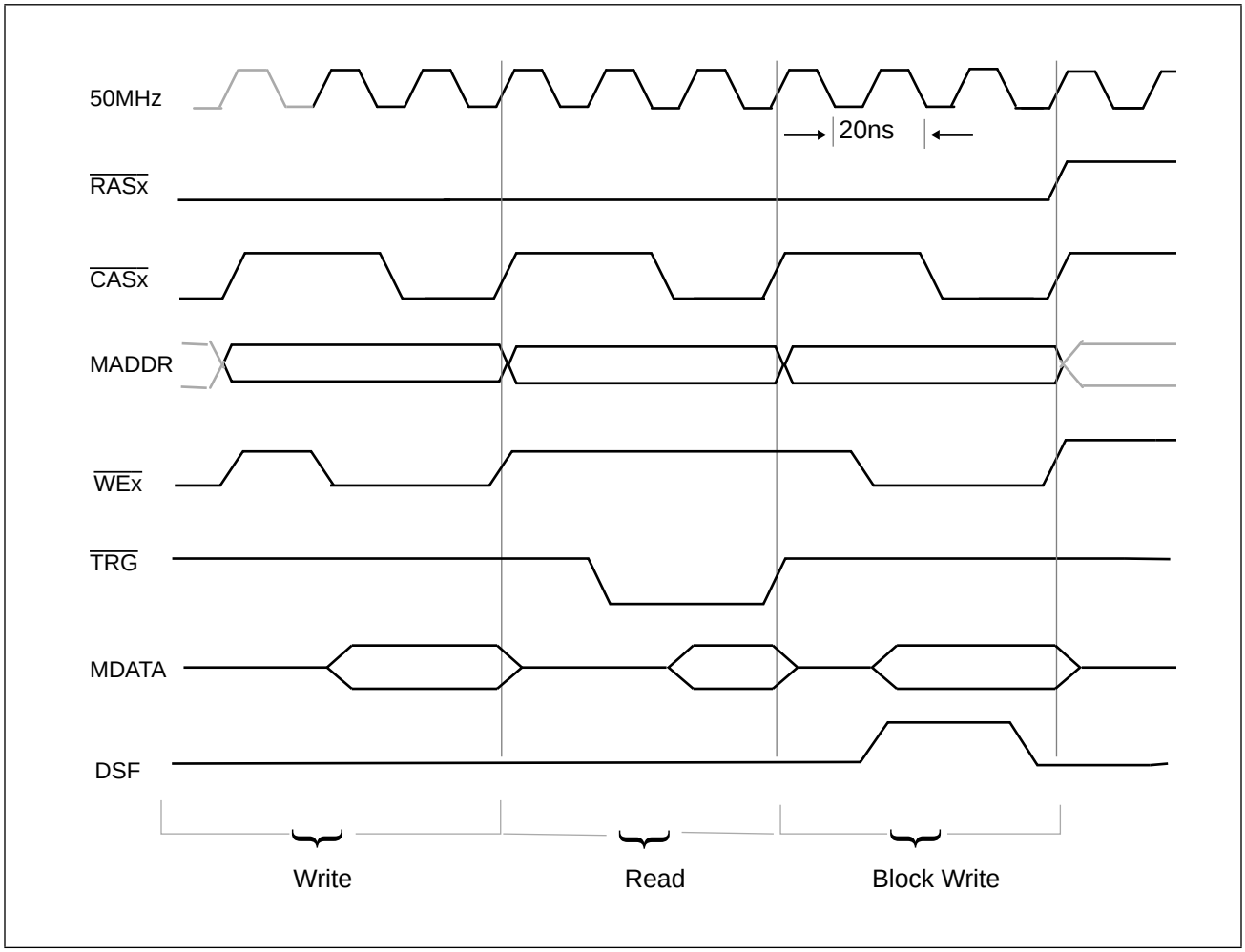
Figure 40. VRAM Memory-to-Register Transfer Cycle (60ns cycle timing)



Page Mode Cycles

The BtV2115 supports page mode cycles for read, write, and block write modes, as shown in Figure 41.

Figure 41. Page Mode Cycles



VRAM Timing Considerations

The BtV2115 is designed to work properly on a board meeting the minimum and maximum load limits for the BtV subsystem design (see "VRAM Timing Parameters" on page 317). The BtV2115 uses an on-chip phase-locked loop to quadruple the input 25MHz clock, producing internal 100MHz and 50MHz clocks. The VRAM control signals (as well as input address and data lines) are then generated off the rising edge of the internal 100MHz clock, producing a high degree of accuracy. The timing diagrams for the BtV2115 memory cycle types supported show the nominal placement of signals referenced to the internal 50 MHz clock.

The BtV2115 meets the timing specifications for the VRAM types supported. Timing values for signals generated by the BtV2115 are contained in. For comparison and detailed explanation of the corresponding VRAM parameters, see the specifications for the supported VRAM part number.

NON-VRAM Attachment

The VRAM bus also supports single byte interface connection of ROM and the Yamaha FM Synthesizer. The VRAM bus does not provide a general purpose attachment. It is for special purpose attachment, designed with careful loading considerations. Because of the load on the VRAM address lines at the full 8 module limit, addressing for ROM and Yamaha is provided by connecting to the higher order data lines. The ROM (A19 to A0) address lines are connected to the MDATA bus (bits 27:8). The Yamaha (A2 to A0) address lines are connected to the MDATA bus (bits 10:8). The address values are multiplexed by the BtV2115 during Read and Write cycles.

NOTE

OPL2 has only 2 registers and OPL3 has 4 registers; neither has an A2 pin, thus, no connection to MDATA[10].

Both ROM and Yamaha occupy a 1MB address space with a single chip select generated for any address within that space. The BtV2115 supports several types of ROM, but only a single ROM module can be driven directly (due to a single 1M chip select). External logic would be required to generate additional distinct chip selects.

ROM Configurability

The BtV2115 supports attachment of several different ROM Types, generating control signals at two different speeds. Configuration Register GRP_CFG1[5] (see Table 25 on page 53) sets the ROM speed, which is strapable at reset time on MDATA[16]. The BtV2115 will set the ROM speed bit at FAST or SLOW, based on the ROM type. Parts with speeds above the FAST threshold will use the 45ns cycle, while parts below the threshold will be configured as SLOW and will run with the 70ns timing. The BtV2115 supports the 2700 Family of non-Flash ROMs (PROMs) and the Flash ROMs listed in Table 140.

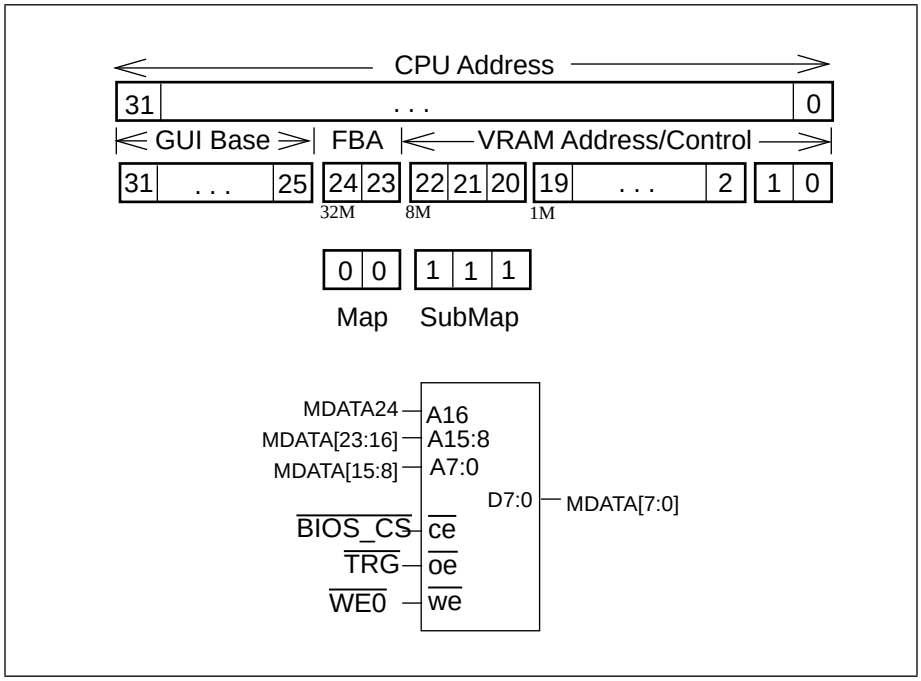
Table 140. Flash ROM Types Supported

Part #	Description
AM29F010-70	AMD Flash ROM 128Kx8 70ns
AM29F010-120	AMD Flash ROM 128Kx8 120ns
i28F010-70	Intel Flash ROM 128Kx8 70ns
i28F020-120	Intel Flash ROM 256Kx8 120ns

ROM Connection

In addition to normal ROM, the BtV2115 can utilize Flash Memory for BIOS code, allowing software upgrade. The ROM is addressable at an offset of +7MB above the GUI Base Address 32M address boundary, that is, a ROM Cycle is selected with FBA Map = 00 and SubMap = 111. It is also addressable at the VGA ROM address. Figure 42 shows the signal connections for a typical Flash ROM.

Figure 42. ROM Connection

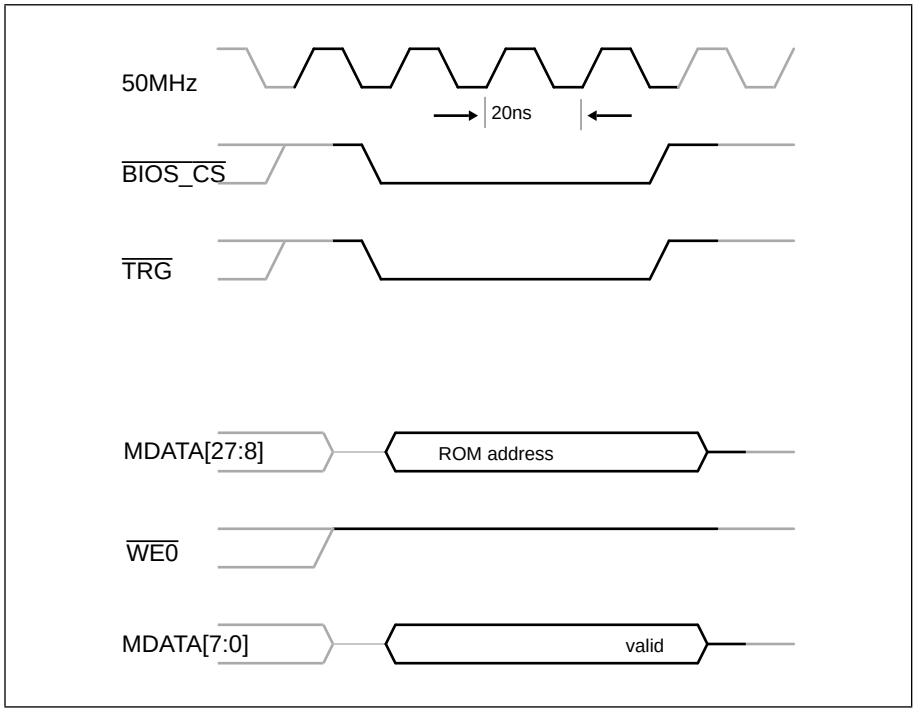


$\overline{WE0}$ is used only for Flash ROM Writes.

ROM READ Cycle

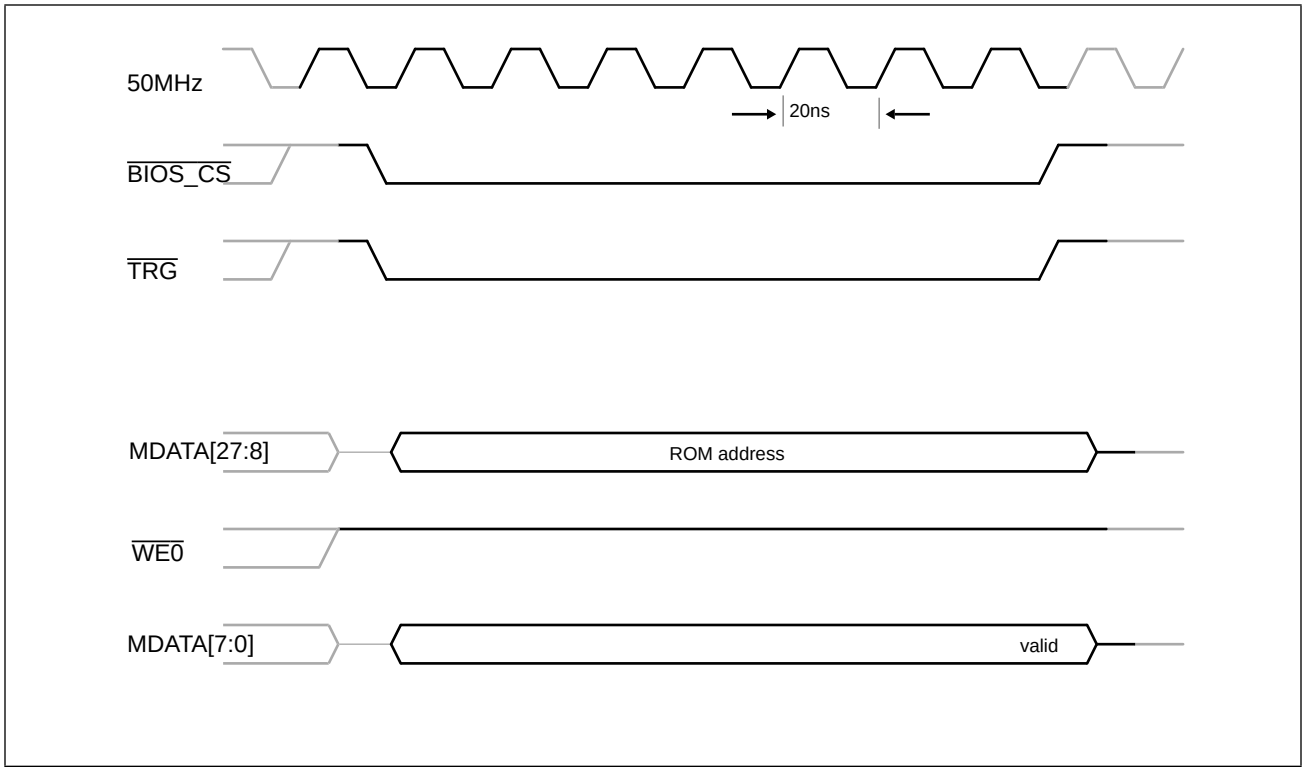
The BtV2115 reads ROM data at two different speeds corresponding to 70ns and 120ns access times.

Figure 43. ROM Read Cycle (70ns part)



ROM address bits A19-A0 are connected to MDATA[27:8] due to load limitations on VRAM bus addresses. Depending on ROM size, some of the bits are not be connected.

Figure 44. ROM Read Cycle (120ns part)

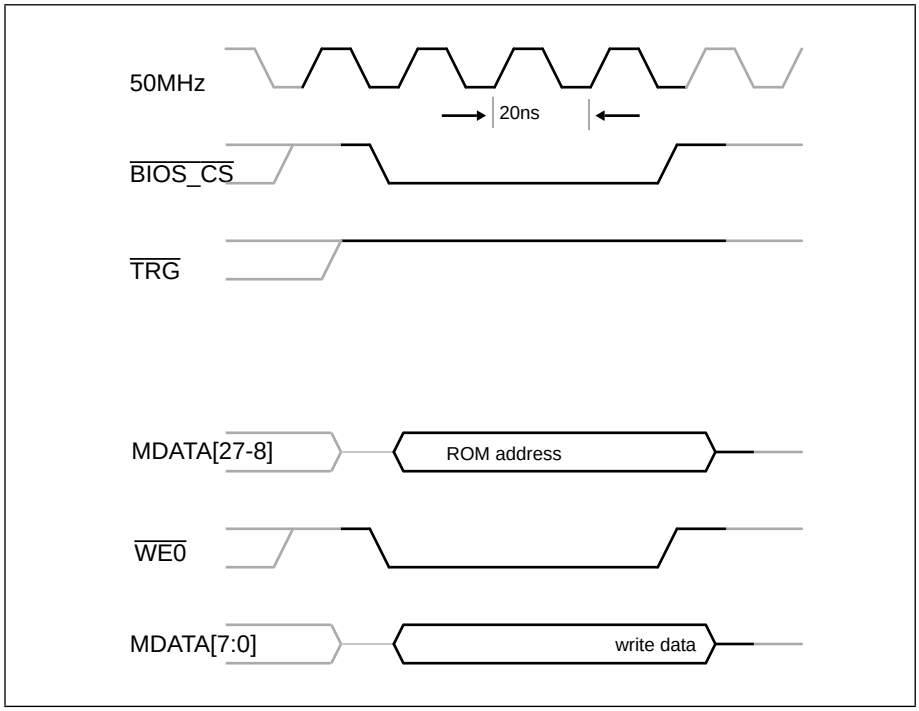


ROM address bits A19-A0 are connected to MDATA[27:8] due to load limitations on VRAM bus addresses. Depending on ROM size, some of the bits are not be connected.

Flash ROM Write Cycle

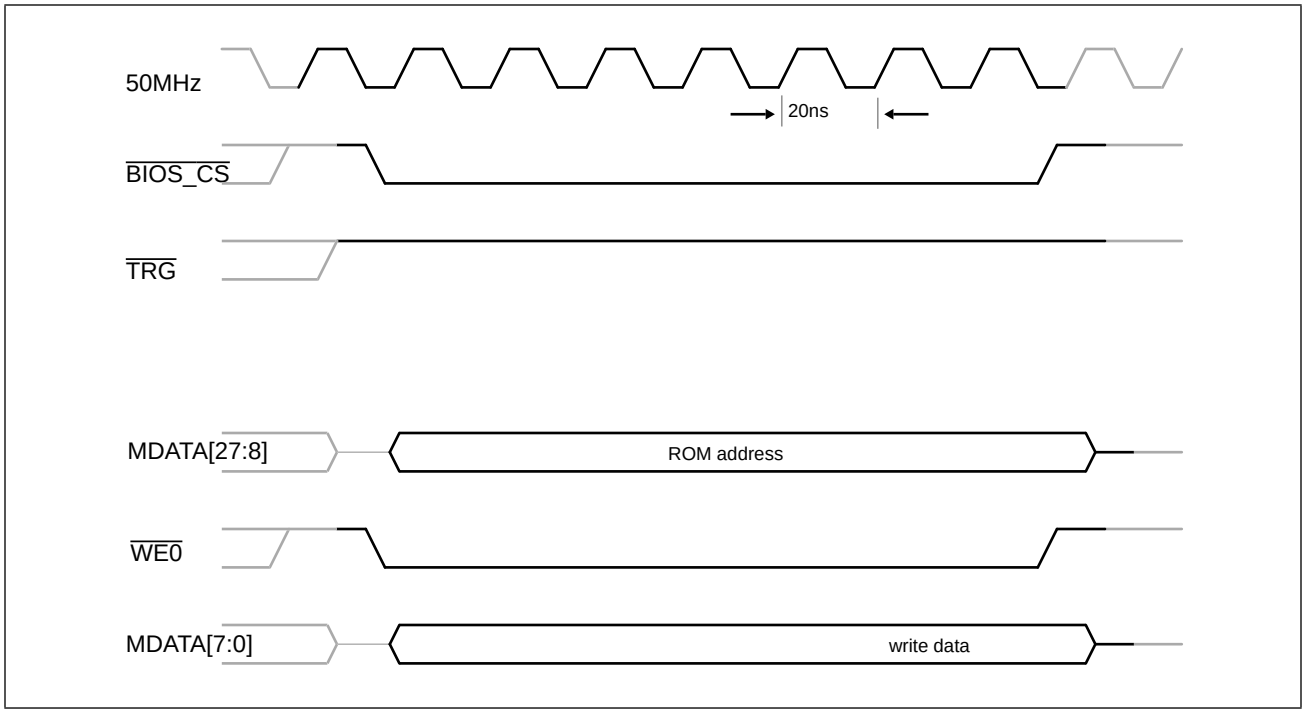
The BtV2115 supports writing of Flash ROM data at two different speeds. The special sequence necessary to Write Flash ROM is supported by the BtV2115 BIOS.

Figure 45. Flash ROM Write Cycle (70ns part)



ROM address bits A19-A0 are connected to MDATA[27:8] due to load limitations on VRAM bus addresses. Depending on ROM size, some of the bits are not be connected.

Figure 46. Flash ROM Write Cycle (120ns part)

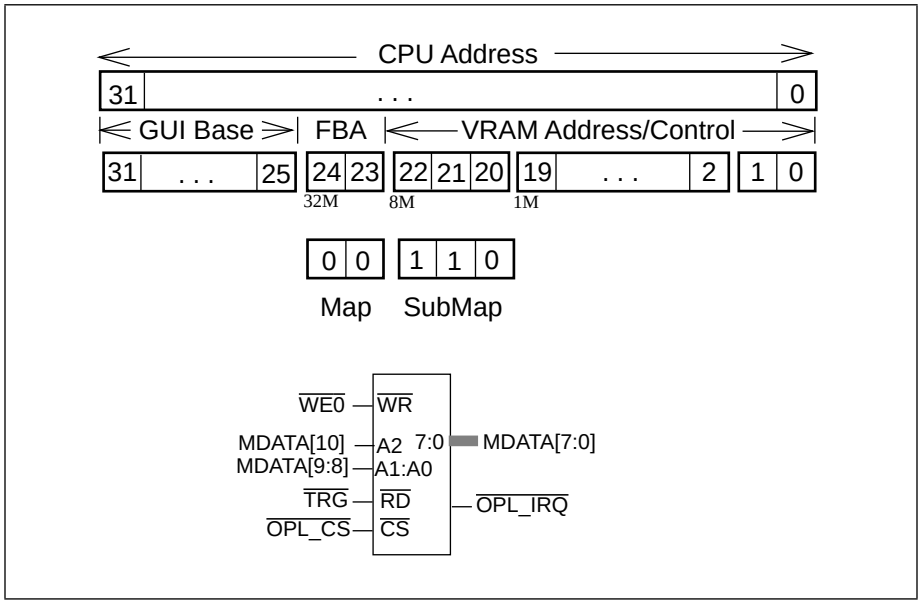


ROM address bits A19-A0 are connected to MDATA[27:8] due to load limitations on VRAM bus addresses. Depending on ROM size, some of the bits may not be connected.

Yamaha Configurability

The Yamaha YMF278 (OPL4), YMF262 (OPL3), and YM3812 (OPL2) FM Synthesizers can easily be connected in a BtV Subsystem (as shown in Figure 47). The OPL4 has an additional Register Select (address) bit to support new registers.

Figure 47. BtV2115 Connections to Yamaha FM Synthesizer



NOTE

Yamaha must also be connected to other chips in the BtV subsystem. Those connections are not shown here.

Legacy Audio supports the Yamaha 2 and 4 operator mode FM synthesizer chip family, with direct support for OPL3.

Yamaha FM Synthesizer Connection

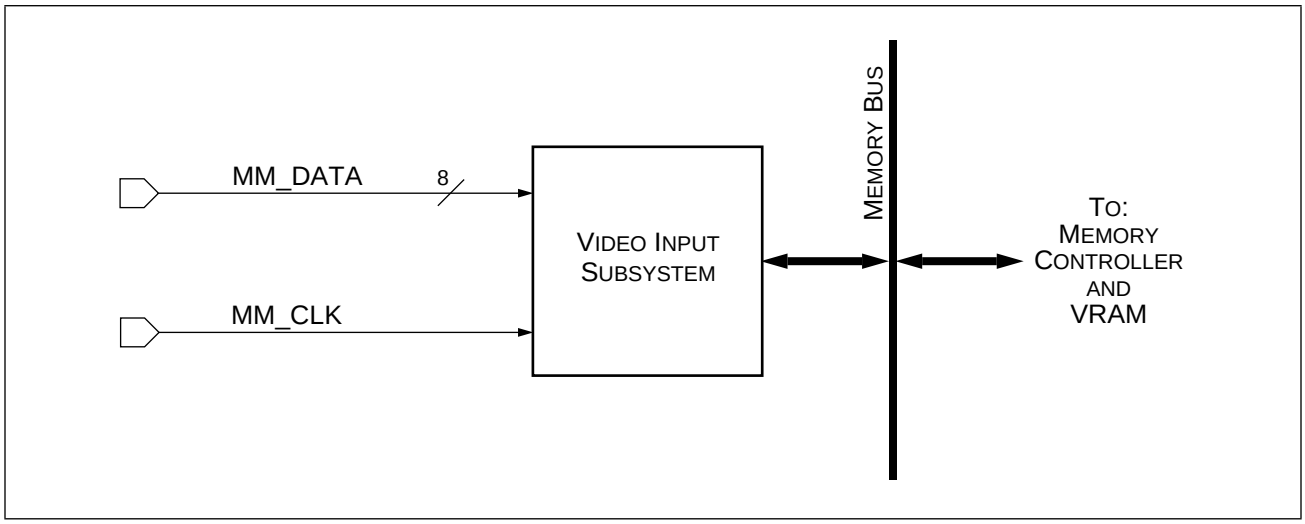
The Yamaha data lines are connected to the low byte of the VRAM data (MDATA[7:0]). Data is steered by the internal BtV2115 memory controller to/from the proper system byte during Yamaha register accesses. The Register Selects do not connect to VRAM address bits (due to address loading), but connect to VRAM data bits (MDATA[10:8]), which are multiplexed to contain the proper addressing information for the register selects.

VIDEO INPUT SUBSYSTEM

Introduction

The BtV2115's Video Input Subsystem interfaces between the BtV2811A Video Decoder and the Memory Controller and VRAM via the Memory Bus. Within the Video Input Subsystem, the video input logic takes a continuous byte stream of video (MM_DATA) from the BtV2811A, extracts the encoded control information, formats the pixel streams, and packs it into the specified VRAM buffer. Figure 48 shows an overview of the Video Input Subsystem.

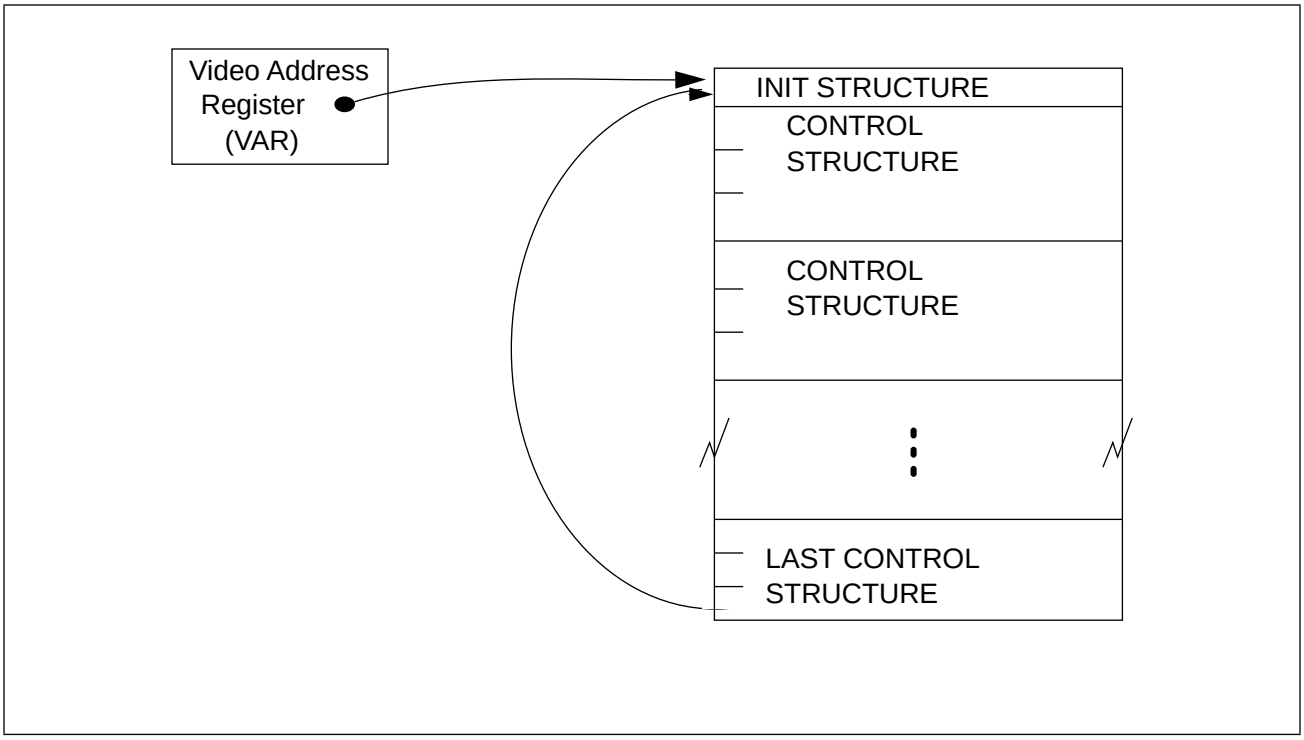
Figure 48. Video Input Subsystem



Within the VRAM, a control structure list controls the video input logic. This DMA channel command-like architecture provides a great deal of flexibility to capture and display programs.

The Video Input Subsystem accesses the control structure list via the Video Address Register (VAR), which points to the top of the list. The dword at the top of the list (the INIT STRUCTURE) contains initial control values. Each control structure in the list consists of three sequentially located dwords of VRAM. Thus, Figure 49 represents a typical control structure list.

Figure 49. Video Input Control Structures



The end_of_list bit of the video input control structure indicates the end of the video control structure list. When a control structure is completed with this bit set, the VAR starts the process over again. Any control structure can be marked to execute an interrupt upon completion. In addition, control structures can specify coverage of either vertical active lines or vertical blank lines, such as the vertical interval-time code marker or closed captioning information.

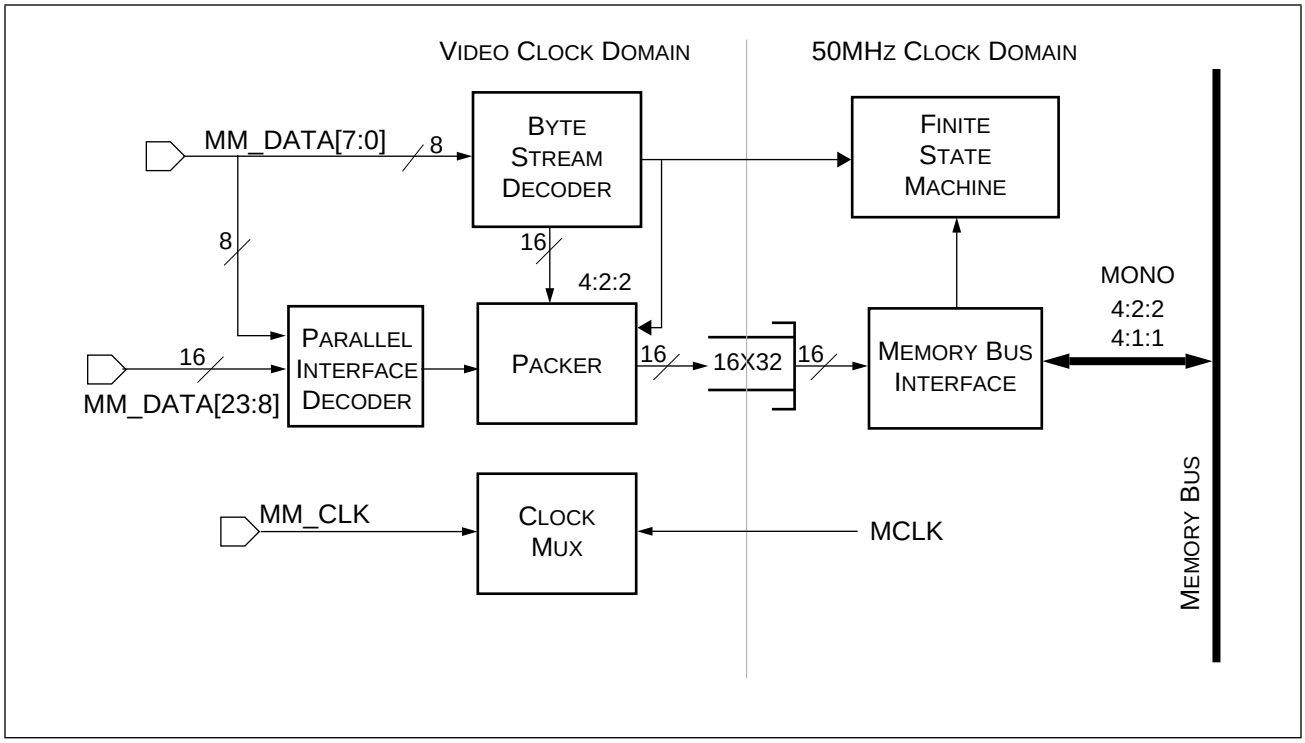
Each control structure can have unique a capture format. For example, closed captioning data can be captured in mono format (Y only) while the active vertical lines are captured in 4:2:2 YCrCb format. In addition, a structure can be marked to *drop* specified lines of a specified field by marking off the required time without actually storing data into the VRAM, thus, easily skipping over either odd or even fields.

The BtV2811A provides the byte stream protocol that describes the video image delivered to the BtV2115 controller. The BtV2811A encoded information includes factors such as vertical reset, horizontal scaling, and vertical decimation.

A byte stream decoder within the BtV2115's video input logic reconstructs the two dimensional video image and any associated data, e.g. closed captioning, from this incoming byte stream. The desired portions of the data can be converted from 4:2:2 YCrCb into 4:1:1 or mono only and then packed into VRAM to minimize size/bandwidth. In addition, the pixels can remain in 4:2:2 format or can be decimated vertically and horizontally by a factor of 4. This further decimation is useful for obtaining icon-sized live video.

The video input control structure list can be used to control double buffering for both the video capture and video display processes of the PACDAC controller. For video display, control of the A/B buffer selection of the PACDAC controller video 1 FIFO or video 2 FIFO can be controlled by completion of a capture structure.

Figure 50. Video Input Subsystem Block Diagram



As shown in Figure 50, the BtV2811A data is presented to the Byte Stream Decoder block which interprets the packed format into a valid pixel stream with surrounding video frame control signals extracted. The pixel stream is formatted and decimated into dwords destined for the VRAM media buffer. As each dword is formed, it is inserted into the 16x32 FIFO and passed from the video clock domain to the BtV2115 internal MCLK domain.

On the MCLK side, a finite state machine is in overall control of all aspects of the video input subsystem. It traverses the structure list, fetches each control or init structure in turn and presents the various fields, and handles the writing of the video data from the 16x32 FIFO to VRAM.

Buffer Management Hardware

The video input registers share the audio subsystem's submap 5 (see Figure 12 on page 116).

Video Control Register

name: AUD_VCR
address: GBASE | 005003C0h, read/write
size: 8 bit
function: This register contains various control bits for the video input function as shown in Table 141.

Table 141. Video Control Register

Bit	Field Name	Description
7	Enable Video	0= Reset video subsystem 1= Enable video input processing. This bit should be held at zero until after clock selection is made by AUD_VCR[6] This bit is reset by the pause function in the control structures.
6	Clock Select	0= Use 50MHz MCLK 1= Use video input clock, i.e. the MM_CLK. Set this bit to one if a BtV2811A is known to be present from I^2C inquiries. Once the clock bit is set then the enable bit, AUD_VCR[7] can be set on a subsequent I/O operation.
5:3	Reserved	
2	Enable Parallel Decoder	PCI-Bus only 0 = BtV2811A interface 1= Parallel decoder interface (see "Parallel Decoder Interface" on page 266)
1:0	Reserved	

Video One Register

name: AUD_V1R
address: GBASE | 005003C1h, read/write
size: 8 bit
function: This single bit in this register drives the PACDAC controller's video1 buffer address logic to select either video FIFO 1 A pointer or B pointer. This allows direct software control of double buffering for the video

window associated with video FIFO one in the BtV2487. Notice that the state of this bit can also be automatically controlled by bits in the video input control structure, below.

Table 142. Video One Register

Bit	Description
7:1	Reserved
0	Select PACDAC Controller Video FIFO 1 B. This bit can be set or reset by the video structures. 0 = Use video buffer A on video FIFO 1 at the top of next frame 1 = Use video buffer B on video FIFO 1 at the top of next frame

Video Two Register

name: AUD_V2R
address: GBASE | 005003C2h, read/write
size: 8 bit
function: This single bit in this register drives the PACDAC controller's video2 buffer address logic to select either video FIFO 2 A pointer or B pointer. This allows direct software control of double buffering for the video window associated with video FIFO two in the BtV2487. Notice that the state of this bit can also be automatically controlled by bits in the video input control structure, below.

Table 143. Video Two Register

Bit	Description
7:1	Reserved
0	Select PACDAC Controller Video FIFO 2 B. This bit can be set or reset by the video structures. 0 = Use video buffer A on video FIFO 2 at the top of next frame 1 = Use video buffer B on video FIFO 2 at the top of next frame

Video Address Register

name: VAR
address: GBASE | 005003C4h, read/write
size: 32 bit
function: The structure address field in this register is used at the end of every video capture list to know where to begin the next capture list. The bit contents of this registers are shown in Table 144.

Table 144. Video Address Register

Bit(s)	Field Name	Description
31:22	Current Tag	Read Only view of tag field of control structure in process. Reset to zero by chip reset. Not defined when enable video (bit 7, video control) is not set.
21:2	Structure Address	VRAM dword address of first structure in the control structure list. Note: this always points to an INIT STRUCTURE followed by a variable number of control structures.
1:0	Reserved	

Init Structure Definition

The BtV2811A's vertical timing loop allows the number of lines in the vertical active region to vary if the BtV2811A is decoding a video tape recorder in any mode other than normal play. As a result, the BtV2811A provides an accurate leading edge detection of vertical active. However the number of lines expected may differ, and the actual leading edge of vertical active could be on an unexpected line.

To make the control structure selection of starting line values work correctly, the vertical line counter in the byte stream decoder is reloaded from the value in the INIT STRUCTURE whenever the first vertical active horizontal reset is detected. Normally, active video for an NTSC signal starts on line 22. When a video tape recorder is in a mode other than normal play, the active video may not start until much later (for example, line 28). If a capture control structure is coded to start capture on line 22 but active video does not actually start until line 28, six lines of inactive video would be captured.

To circumvent this, the VERT_RELOAD value should be programmed to a larger value (for example, line 32) and the capture control structure set to start at that same value. When the BtV2115 detects the start of vertical active data, it will set its internal line counter to 32 (in this example) and start capturing data for the capture control structure.

Table 145. Init Structure

Bit(s)	Field Name	Description
31:30	Reserved	
29:20	BLANK_SRC_PIXELS	Max number of video source pixels that can be captured on a vertical blanked line.
19:10	ACTIVE_SRC_PIXELS	Max number of video source pixels that can be captured on a vertical active line.
9:0	VERT_RELOAD	Vertical line counter reload value.

In a similar fashion, the BtV2811A's horizontal timing loop can shift the horizontal active to acquire good pixels in fast forward/reverse. The exact number of pixels to transfer per line is set by the other fields in the INIT STRUCTURE. These fields allow for detection of loss of video lock horizontally and guarantee that only the anticipated amount of memory will be loaded by a control structure, i.e. a badly formatted video source cannot unintentionally overwrite VRAM registers.

Capture Control Structure Definition

The capture structure consists of three dwords of VRAM (DWORD0, DWORD1, DWORD2) as defined in Table 146, Table 147, and Table 148.

Table 146. Capture Control Structure DWORD0 *(1 of 2)*

Bit(s)	Field Name	Description
31	END_OF_LIST	0 = Process the next sequential control structure. 1= This is the last control structure in the list. Find the next one to process by following the Video Address Register pointer.
30	SET VIDEO INTERRUPT	1=Set the video interrupt flag at the end of processing this control structure.
29	PAUSE	0= Continue video list processing at end of processing this structure. 1= Reset the enable_video bit, VCR[7] at the end of processing this structure and after all data has been stored in VRAM.
28:27	FORMAT	00= 4:2:2 01= 4:1:1 10= mono (Y only) 11= 4 to 1 decimation of 4:2:2
26	SCAN	0= Even field only 1= Odd field only
25	DROP	0= Normal operation, write to VRAM 1= Drop this field
24	CHROMA_MODE	PCI-Bus only 1 = Enable chroma keying on luminance LSB (see "Parallel Decoder Interface" on page 266) 0= Pass LSB Y through to BtV2487
23	V2 SELECT ENABLE	0= No change in buffer selection 1= DWORD0[21] selects video 2 A/B buffer

Table 146. Capture Control Structure DWORD0 *(2 of 2)*

Bit(s)	Field Name	Description
22	V1 SELECT ENABLE	0= No change in buffer selection 1= DWORD0[20] selects video 1 A/B buffer
21	V2SELECT	0= Video 2 use buffer A for next frame 1= Video 2 use buffer B for next frame
20	V1SELECT	0= Video1 use buffer A for next frame 1= Video 1 use buffer B for next frame
19:10		Reserved
9:0	STARTING VERTICAL LINE	Matched against byte stream decoder vertical line counter to determine when to start data transfer for this control structure.

The STARTING VERTICAL LINE field is compared to the current vertical line in the current input video field. If the field is the correct type (odd or even), all lines after the specified vertical line are captured until the number of lines specified in the VERTICAL LINE EXTENT field is captured or until vertical reset. Consequently, care must be taken when starting capture after reset. The first structure in the capture list may cause incorrect data to be captured if the BtV2115 starts capturing data at a location other than the beginning of the incoming video field. To avoid this problem, set the first capture structure to drop an entire field. Valid capture will then start in the second capture control structure.

Table 147. Capture Control Structure DWORD1

Bit(s)	Field Name	Description
31:29	Reserved	
28:11	DWORD LINE PITCH	Number of dwords to add to address of start of line capture. The 2 LSBs of this address must be zero, i.e., this pointer must point to a 4 dword (16 byte) aligned location in VRAM.
10	Reserved	
9:0	VERTICAL LINE EXTENT	Number of vertical lines to be transferred by the control structure

Table 148. Capture Control Structure DWORD2

Bit(s)	Field Name	Description
31:22	TAG	This tag field is visible in the video address register when this control structure is being processed.
21:2	MBUS ADDRESS	VRAM dword address pointer to location in VRAM to store data captured by this control structure. The 2 LSBs of this address must be zero, i.e., this pointer must point to a 4 dword (16 byte) aligned location in VRAM.
1:0	Reserved	

Line Numbering System

The BtV2811A has a horizontal and vertical scaling mechanism. Lines that are decimated must still be processed by the BtV2811A so lines coming to the BtV2115 are marked as either present or not. The marking is done by withholding the start of active pixels from lines that are discarded. The BtV2115 video input subsystem counts only active lines. Thus the line numbering system used in the control structures is based on a post scale count not the actual NTSC or PAL line number.

The BtV2811A only scales vertically inside the active vertical region of the video frame. Consequently, all lines in the inactive region are presented (and counted) by the BtV2115. This ensures that the closed caption data will always be available, independent of the scaling set in the BtV2811A.

The BtV2811A scales horizontally on all lines. Therefore be sure to account for horizontal scaling in closed caption decoding.

Horizontal pixel counts within the BtV2115 are post scaled pixel counts.

Live Video Input

Before performing video capture, the following tasks must be performed:

- The BtV2811A must be queried and initialized via the I^2C bus.
- The control structure list is set up in VRAM
- The video address register is initialized to point to the control list structure.
- The clock select bit in VCR[6] is set so that the BtV2811A clock is used for capture.
- The video enable bit in VCR[7] is set so that the video capture logic will come out of reset and begin processing the structure list.

For normal live video operation, the pause bit should not be set in any control structure.

In order to display a CIF resolution live video window, set up a control structure list with an INIT STRUCTURE and two CONTROL STRUCTUREs. The first control structure (DWORD0) is set to wait for the even field and is set to *drop* the video. The second control structure (DWORD1) is setup to wait for the odd field vertical line 32 with a mode 4:1:1 format. The MBUS address in this structure points to the first dword of the target buffer for the video image. The vertical line extent is set for 240 lines. The word line pitch field is set to add the number of dwords per line to the starting address of each line. In this simple case, no double buffering is used so the VCR register should be set so that it controls the VIDEO A/B selection for the PACDAC Controller.

To capture two fields, the first control structure (DWORD0) is set to capture the odd field and points to the first scan line of the frame buffer. The second control structure (DWORD1) is set to capture even fields and points to the second line of the buffer. The word line pitch of both control structure is set to twice the number of dwords per scan line.

Closed Captioning Capture With A Video Icon

A large video control structure list and a very small amount of buffer can be used to decode closed captioning while the video window is iconified. To implement this case, set up the list to process 30 separate NTSC frames and mark the last control structure to interrupt the CPU and continue. Each frame is processed by a set of two control structures (3 dwords in each structure). The first structure grabs the closed captioning line by being setup for the odd field line 11 with a format of "MONO" so that only the Y values are grabbed into a buffer. The second structure is setup to grab an iconified version of the video into a single small buffer. This structure's format is set to a 4 to 1 decimation of 4:2:2.

Each closed captioning structure points to a different line buffer while each icon structure points to the same video buffer. Thus the CPU will be interrupted once per second to process an entire second's worth of closed captioning information. This requires approximately 181 dwords to hold the structure list, 4K dwords to hold the close captioning and 256 dwords to hold the video icon.

To allow for interrupt latency in the CPU, set the actual structures to capture in a double-buffered fashion with two one second sequences of video captioning so that software can ping pong back and forth (i.e., capture 60 frames worth). Set the tag values of the first 30 frames in the control structures to 01h; and set the tags of the second 30 frames to 02h. Thus, allowing software to quickly and easily tell which second's worth of close capturing is currently being processed.

Single Frame Record Example

To simplify single frame capture, set the PAUSE and INTERRUPT bits in the last control structure element. Upon completion of the list, the video control structure list processor will stop and an interrupt will be generated to the CPU.

A more complicated example includes both a live video window loop and a capture list. The live video loop occurs first in the control structure list with the END_OF_LIST bit set in the last structure. As video is captured into the live window video buffer, it encounters this EOL bit and restarts at the top.

If the application builds two control structure extensions immediately after the live window portion, the capture part is set to capture full resolution for both fields with its last control structure marked as INTERRUPT and PAUSE. Thus, when the application is instructed to grab a single frame, it turns off the END_OF_LIST bit in the last control structure of the live video window and waits for the interrupt to indicate completion of the capture. With this strategy, the live video that is captured is a fresh, full time coherent frame.

Parallel Decoder Interface

For PCI-Bus configurations, the BtV2115 can optionally use an alternate decode interface instead of the BtV2811A VideoStream Decoder. This option is enabled with bit 2 of the AUD_VCR register; see Table 141 on page 256.

Below, Table 149 shows the corresponding signals between the BtV2115 and the BtV2811A and a parallel decoder interface. Refer to Figure 51 for a visual representation of the reference signals used for a scaling window.

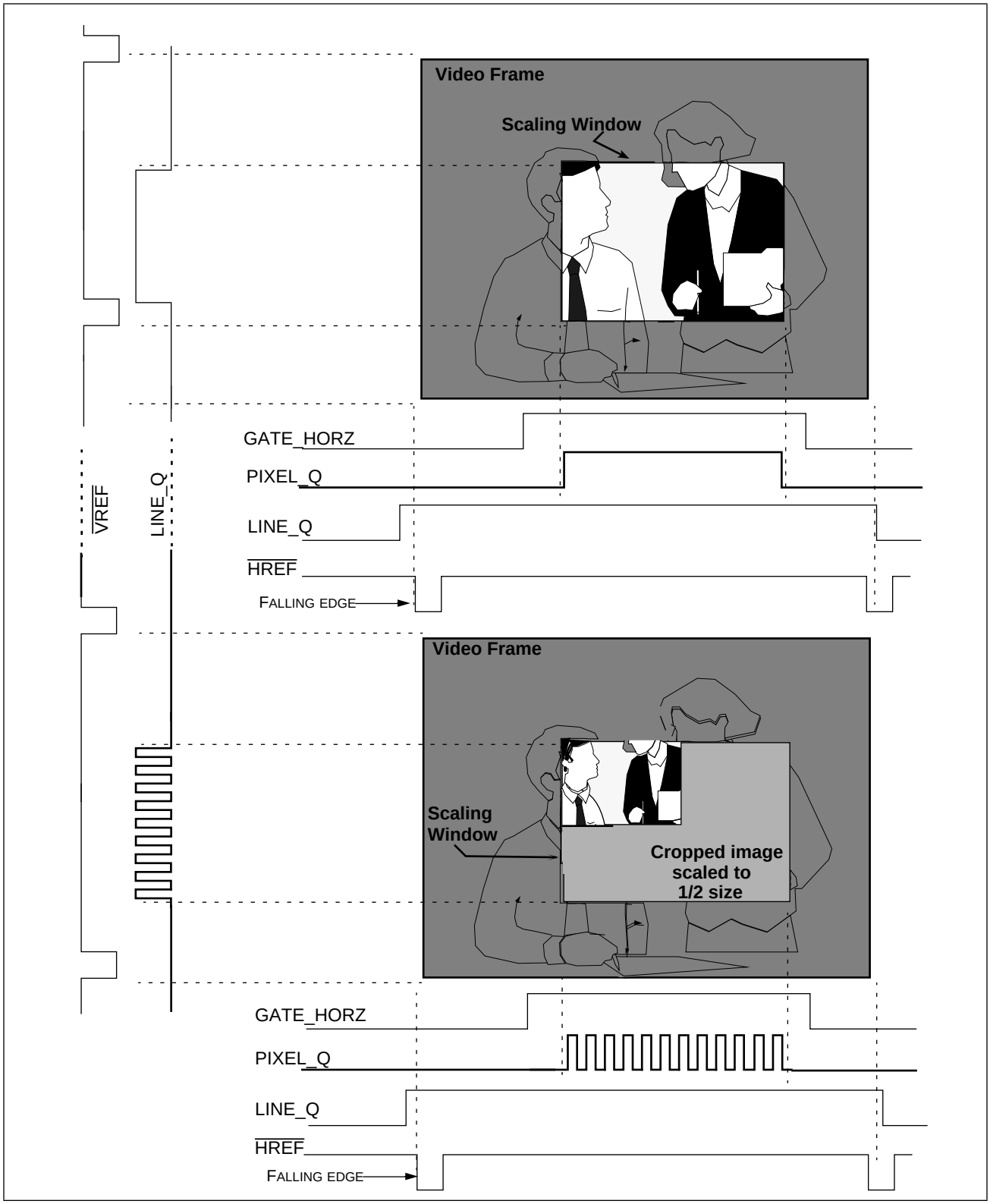
Table 149. Parallel Decoder Interface

BtV2115 Pin Signals	Description	BtV2811A 8-Bit Interface (PCI- or VL-Bus)	Parallel Interface (PCI-Bus only)
MM_DATA[7:0]	Luminance input signals	VD[15:8]	LUMA_DATA
MM_DATA[15:8]	Chrominance input signals[a.]	N/A	CHROMA_DATA
MM_DATA[16]	Pixel qualifier input signal to indicate active pixels of a qualified line	N/A	PIXEL_Q
MM_DATA[17]	Line qualifier input signal to indicate active video phase	N/A	LINE_Q
MM_DATA[18]	Delay compensated horizontal reference signal	N/A	HREF
MM_DATA[19]	Vertical sync signal related to scalar input	N/A	VREF
MM_DATA[20]	Horizontal gate signal, a 1 indicates horizontal direction	N/A	GATE_HORZ
MM_DATA[21]	Reserved		
MM_DATA[22]	VRAM input odd/even bit according to the internal field processing	N/A	OE_FIELD
MM_DATA[23]	Keying signal of the chroma keyer	N/A	ALPHA
a. Chroma keying is enabled with bit 24 of VRAM DWORD0; see Table 146 on page 259.			

When using the video capture parallel interface, the video module can't determine the difference between vertically blanked and non-blanked signals, except when using the ByteStream interface. To work around this limitation (say, to capture teletext or close-caption data), program the device generating the parallel video data to send the vertically blanked lines at the start of the frame. Then use an appropriate video structure to capture/step-over this information.

While in this mode, the video module refers to only the BLANK_SRC_PIXELS field of the INIT Structure (see Table 145 on page 258). The ACTIVE_SRC_PIXELS field has no effect.

Figure 51. Reference Signals for Scaling Window



Signal Definitions

The video parallel interface signals are defined below.

MM_DATA[7:0] — LUMA_DATA
MM_DATA[15:8] — CHROMA_DATA

The first pixel on a line is a Cb pixel. Every second pixel is a Cr or Cb pixel. The video module keeps track of Cr and Cb pixels by counting pixels from the first pixel on a line.

MM_DATA[16] — PIXEL_Q
MM_DATA[17] — LINE_Q

Video data is captured from the LUMA_DATA and CHROMA_DATA inputs on the rising edge of MM_CLK during cycles when both qualifier inputs (PIXEL_Q and LINE_Q) are asserted (active high). If only one qualifier input is needed, then the LINE_Q should be tied off to a logic 1.

MM_DATA[18] — HREF

HREF rising edge is used to detect the start of a new line. This rising edge of this signal is only valid when LINE_Q is asserted. This rising edge must also occur at least two clocks before the first pixel on the video line. HREF falling edge is used to reset the Cr/Cb pixel counter to expect a Cb pixel at the start of the next line.

MM_DATA[19] — VREF

VREF is used to reset the line count to zero at the start of a field. This signal is active high and should be asserted for a short while at the start of every field.

MM_DATA[20] — GATE_HORZ

GATE_HORZ falling edge is used to detect the end of the video line which causes the video FIFO to be flushed and the address pointers to increment to the next line address in preparation for the next video line to be captured. This signal falling edge must occur as quickly as possible after the end of the video data on any video line to allow enough time for the video FIFO to be flushed into VRAM and possibly a new video control structure read from VRAM before the start of active video on the next line. Standard NTSC/PAL/SECAM video timing allows sufficient completion time; for non-standard methods, contact Brooktree for support.

HREF and GATE_HORZ are two separate signals with different but similar meanings. HREF is equivalent to a HRESET signal that is used for counting video lines; it should see a rising edge as early as possible in the video line. GATE_HORZ is equivalent to an ACTIVE signal that should be asserted and deasserted as quickly as possible before and after the active pixels on a line. If there is

only one HREF/GATE_HORZ type signal and it satisfies the timing requirements of both of these signals (rising edge as early as possible in the video line but at least two clocks before the first pixel and falling edge as soon as possible after the last active pixel), then both inputs can be tied together.

MM_DATA[22] — OE_FIELD

The OE_FIELD bit is used to indicate the odd/even field of the video being captured: low for even field, high for odd field.

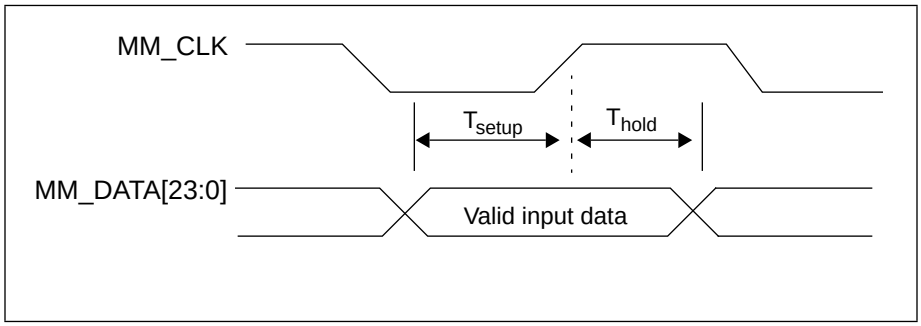
MM_DATA[23] — ALPHA

The ALPHA bit is muxed into the least significant bit of the luma data when the video capture logic is programmed to the alpha keying capture mode. Note that this bit is subsampled by two; it is only sampled for Cb (even) pixels but is stored in the luma LSB for both the Cb and corresponding Cr pixel.

Video Interface Timing

Figure 52 provides the video interface timing diagram.

Figure 52. Video Interface Timing Diagram



Bytestream and parallel video interface timing:
T_{setup} = 5.0ns before MM_CLK rising edge
T_{hold} = 4.0ns after MM_CLK rising edge

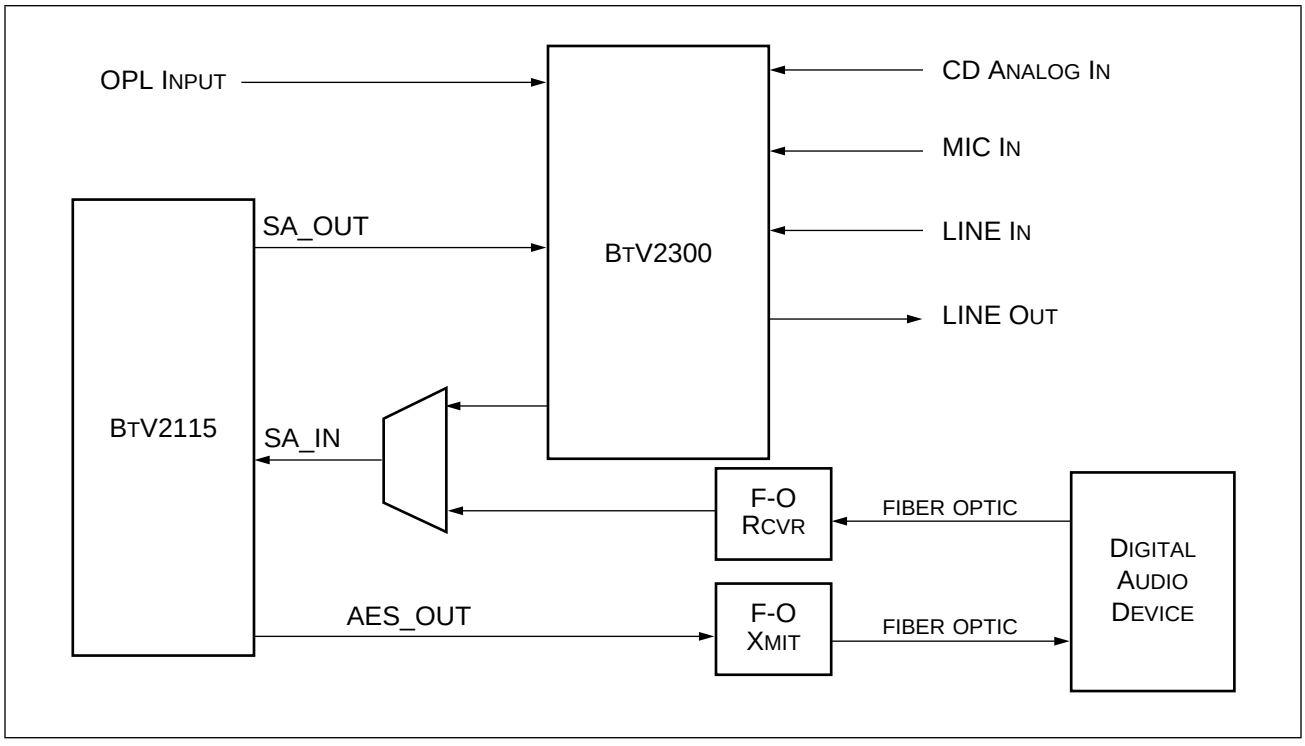
AUDIO INTERFACE

Introduction

The BtV chip set provides a completely self-contained audio subsystem for Microsoft Windows 3.1, Windows 95, and NT, as well as supports compatibility with the many DOS-based games. In addition, the BtV2115 provides a gateway to and from the consumer electronics world of digital audio via the AES/EBU (Audio Engineering Society/European Broadcasting Union) serial audio input and output pins.

Figure 53 shows a typical audio system configuration.

Figure 53. Typical Audio System Configuration



AES/EBU Format

Understanding the AES protocol is fundamental to understanding the BtV audio components, therefore a brief description is in order.

The block-oriented protocol is transmitted at 128 times the sample clock (i.e. 6.144 MHz for 48 KHz samples) in a self-clocking (Manchester/bi-phase mark), block-oriented protocol with synchronization. Since the BtV2115 is over-sampling this signal for decode, it will accommodate the full gamut of audio sample rates. Support of parity, validity, user or channel status is not needed for implementation within the BtV2300's the serial encoder (but, is required for external communications).

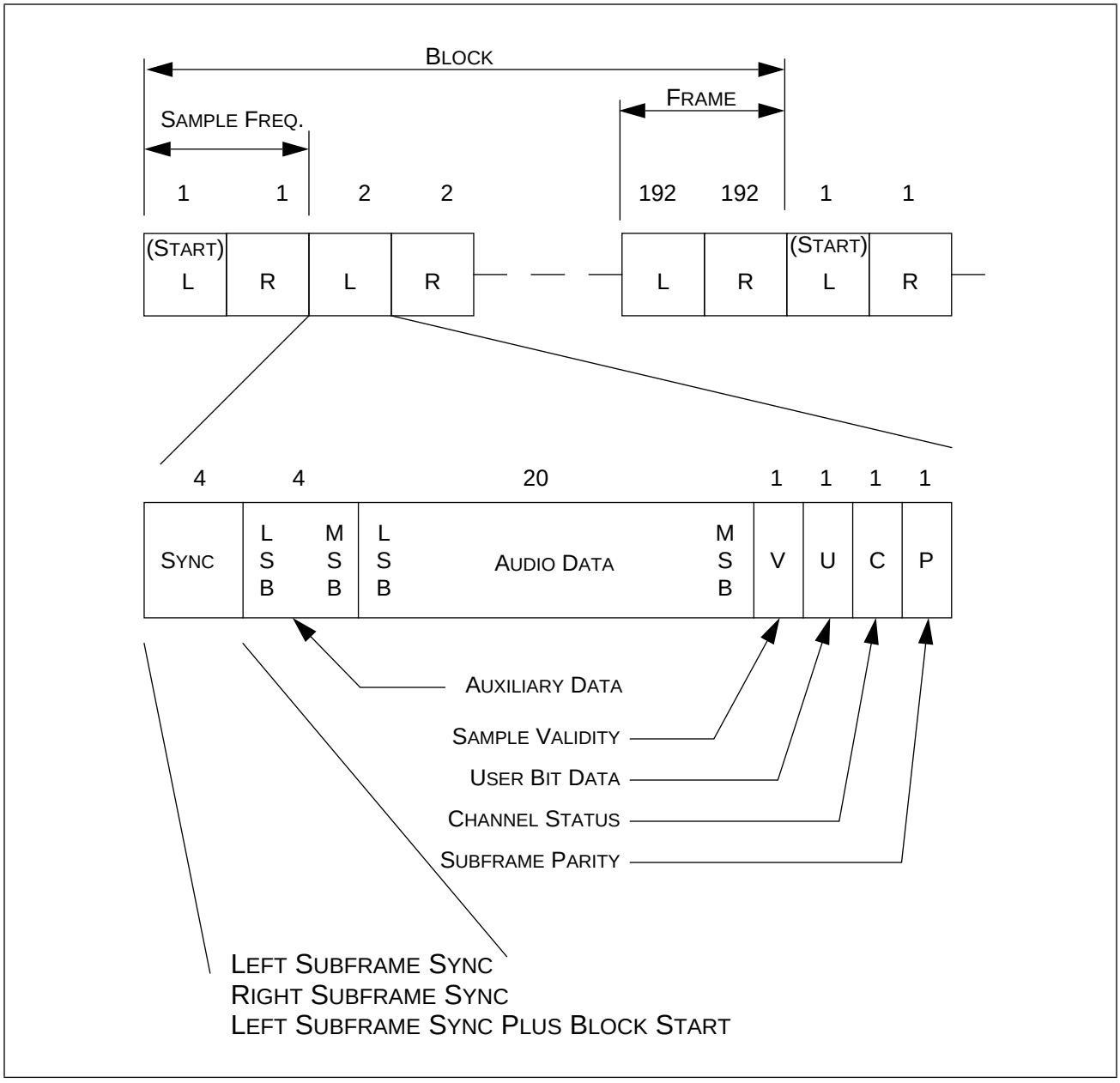
The block-oriented protocol consists of a series of frames. Each frame consists of two channel subframes, 32-bits each, as shown in Figure 54. The rate of transmission of each frame corresponds to the source sampling frequency. That is, the clock corresponding to each bit time within a frame is equal to 64 times the sample clock. The manchester encoding brings the basic encoder clock rate to 128 times the sample rate. Figure 54 shows that each subframe contains one 20 bit audio sample for either the left or right channel, a 4 bit sync or preamble field, a 4 bit auxiliary audio field and 4 one bit fields (defined below).

The BtV2115 always transmits 2-channel 16 bit stereo data, even when the source material is mono (in which case it is duplicated on both left and right channels). Both the channel status block and the user block are identical for each channel. The 16 bit sample is MSB justified within the 20 bit audio field of a frame. The 20 bit samples are sent LSB first in twos complement notation.

Frames are sent in blocks of from 5 to 256 frames. The block length is programmed via the AUD_EBL Register. For communicating solely with the BtV2300 audio chip, the block length is arbitrary (but the shorter the block length, the lower the BtV2300 register update latency). When communicating with an AES/SPDIF device (e.g. a minidisc), the block length should be set for exactly 192 frames by loading 191 (BFh) into the block length register.

Notice that each frame in Figure 54 contains 8 bits of "AUX" data, four from the left subframe and four from the right subframe. When communicating with the BtV2300, each frame contains one byte of data going to or from the BtV2300. The byte associated with the first frame of a block to the BtV2300 contains the register address. The first frame of a block is marked with a unique sync or preamble field that indicates both start of block and left frame, see Figure 54. The usage of the AUX bits will be defined in more detail below.

Figure 54. AES Framing



The four one-bit fields are used in two ways. Two of the fields, V and P, have meaning on a subframe basis only. The validity bit, V, is used to mark each audio sample as valid or invalid (present or not present) with zero indicating valid. Parity bit, P, is used to send even parity for the subframe.

The U and C bits are used in a very different way. Each represents a 1-bit serial channel of data running at a rate of 1 bit per audio sample. Thus, for 48KHz CD audio, there are 48KBits of Left User data, 48KBits of right User data, 48KBits of left Channel Data and 48KBits of right Channel data per second.

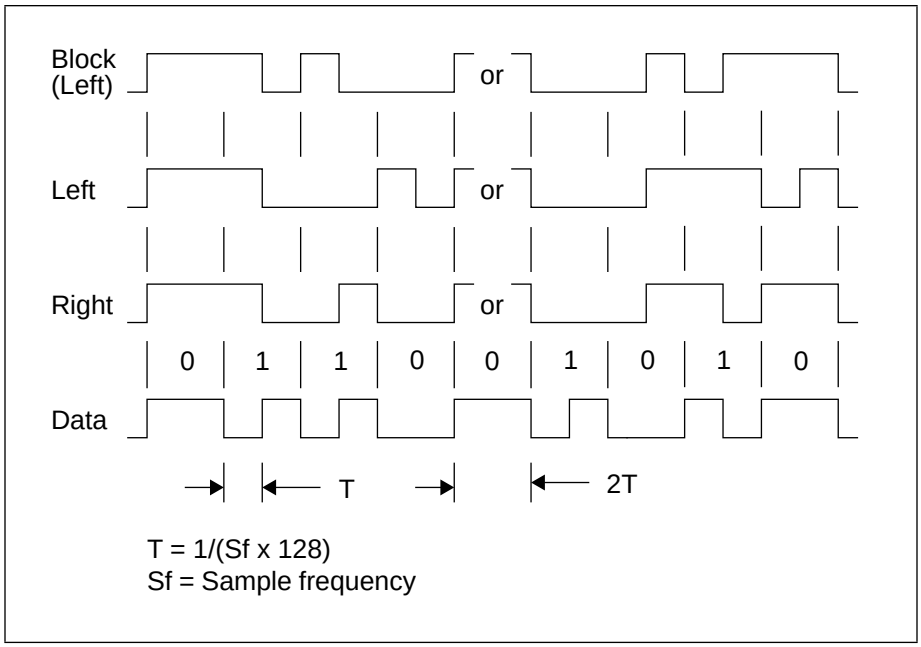
V = Validity (0 = valid)
U = User data, 1 bit
C = Channel data, 1 bit
P = Parity (generates even), 1 bit

Notice that the block start preamble is used to mark the beginning of User and Channel bit streams so that in 192 frame blocks, there are 24 bytes of each bit stream per block. The BtV chip set does not use the User and Channel bit streams for communication. However, the BtV2115 can generate or store User and Channel data when communicating with an AES/SPDIF consumer/professional audio device.

See “Channel Status” on page 313 for a discussion of the BtV2300 Channel stream versus that generated for external audio components.

Figure 55 defines the AES timing.

Figure 55. AES Timing Diagram



AES References

AES Recommended Practice for Digital Audio Engineering -- Serial Transmission Format for Linearly Represented Digital Audio Data, AES3-1985 (ANSI S4.40-1985); Audio Engineering Society, Inc., 60 East 42nd Street, New York, New York 10165, USA

John Watkinson, *The Art of Digital Audio*, 1991, Focal Press.

Clif Sanchez and Roger Taylor, *Overview of Digital Audio Interface Data Structures*; Crystal Semiconductor Corporation Application Note, from Volume 1 Data Book, April 1992

Digital Audio Interface, EIAJ CP-340, September 1987.

Audio Subsystem

Figure 56 shows a block diagram of the BtV2115 Audio Subsystem.

The BtV2115 provides two AES outputs. The serial audio out port (SA_OUT) is connected directly to the BtV2300, and AES_OUT is routed through a digital audio device (as shown in Figure 53). The two streams must be separated since BtV2300 control data is transmitted in the AES AUX field and arbitrary user data is transmitted in the AUX field for consumer audio data.

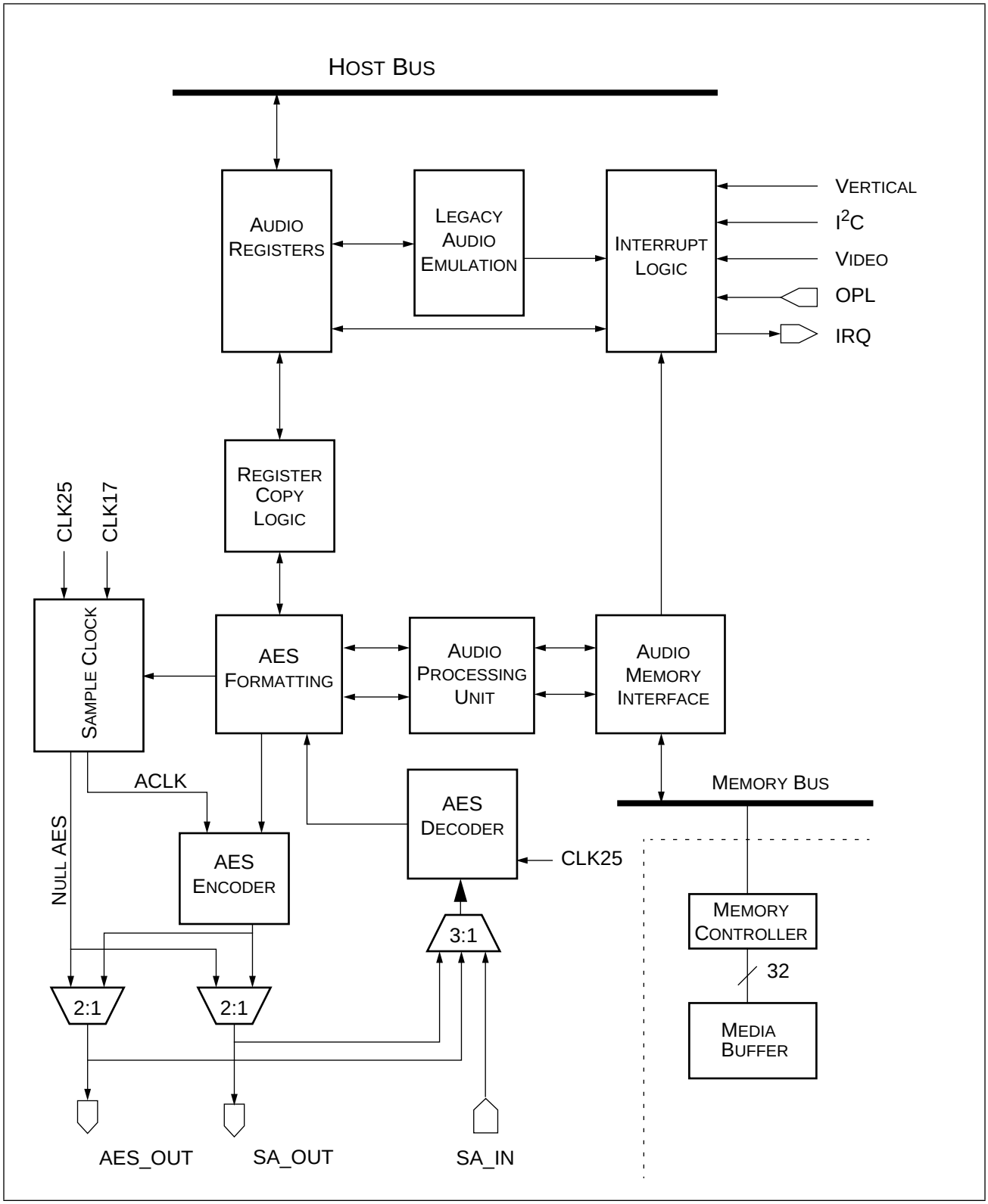
The audio subsystem contains one AES encoder (recall that the AES stream is manchester encoded). Consequently, the BtV2115 contains a "NULL AES encoder" which transmits a continuous stream of ones to quiesce either the consumer device or the BtV2300 when it is not being driven by the AES encoder.

Typically the serial audio input (SA_IN) receives an AES stream from the BtV2300. Note that the BtV2300 has an integrated mux which can pass a consumer device's AES stream through the BtV2300 if desired. The SA_IN line goes through a diagnostic wrap mux into the AES decoder.

The AES decoder always runs on CLK25 which provides sufficient over-sampling to reliably decode the AES bit stream at up to a 48 KHz stereo audio sample rate. The AES encoder receives the serial audio stream and offers a parallel interface to the rest of the BtV2115 audio subsystem.

The AES encoder receives a sample rate times 128 clock, derived either from CLK17 or CLK25. The AES encoder offers a parallel interface to the rest of the BtV2115 audio subsystem and generates the serial audio stream. ACLK is output to the AudioStream Interface via the AUDCLK_OUT signal pin.

Figure 56. BtV2115 Audio Subsystem Block Diagram



All of the BtV2300 registers are read/written via the AUX field of the AES streams. BtV2300 registers are written by sending a start address in the first byte of the AUX block as marked by the start block preamble of the AES frame. Subsequent bytes within the AUX stream are treated as data and are written to BtV2300 registers for each successive AES frame. The register address in both the BtV2300 and the BtV2115 auto-increments for each data byte sent. The address register does not wrap. Instead, it stops at the highest address and subsequent data bytes are ignored.

The registers within the BtV2300 are write-only. There is no mechanism for the BtV2115 to read back BtV2300 register data. However, the BtV2300 registers are "shadowed" within the BtV2115 so that software can read the last value written to the BtV2300.

The BtV2115 reads two registers from the BtV2300 via the SA_IN stream: the BtV2300 Chip ID register (CID) which contains a version/revision code and the BtV2300 application interface. The first byte of a block from the BtV2300 contains the CID while the second and subsequent blocks of AUX data from the BtV2300 contain samples of the application interface.

The application interface is bidirectional. The BtV2300 application direction register contains bit by bit controls of the direction of the application interface; one means output, zero means input. Data written to the BtV2115 application register (APP_WR) is transmitted to the BtV2300 for possible output on the applications interface, subject to the direction register mask.

The Register Copy Logic block in Figure 56 is responsible for watching the data written to the bank of BtV2300 shadow registers so that an appropriate address can be generated at the start of the next block of AES output data. The register copy block keeps track of the lowest register written but not yet transmitted. On each block, the register transfer will begin with the lowest modified BtV2300 shadow register.

BtV2300 register bytes are assembled from left and right subframes, as shown in Table 150.

Table 150. BtV2300 Register to AUX Mapping

AES AUX	BtV2300 Data Bit
RIGHT_AUX[3]	BIT[7]
RIGHT_AUX[2]	BIT[6]
RIGHT_AUX[1]	BIT[5]
RIGHT_AUX[0]	BIT[4]
LEFT_AUX[3]	BIT[3]
LEFT_AUX[2]	BIT[2]
LEFT_AUX[1]	BIT[1]
LEFT_AUX[0]	BIT[0]

The AES formatter manages the construction of blocks and supplies frame level parallel data to the encoder and receives it from the decoder. With the exception of passing BtV2300 shadowed register values into the AES stream and copying the BtV2300 CID and Application Read data into the BtV2115 audio registers, all data in the AES streams is zero or is copied directly to or from media buffer. When not using the media buffer for the AES "VUCB" components (see "Audio Data Formats In the Media Buffer" on page 283), a consumer format digital audio stream is generated. However, when the media buffer is used as a source or destination for channel status and user data, a professional format digital audio stream may be utilized.

Digital signal processing is performed within the Audio Processing Unit (APU); refer to Figure 57. This consists of format conversion into stereo 16-bit 2's complement audio samples, floating point multiplies to perform gain or attenuation, clipping detection, maximum value holds, and sample summation.

The BtV2115 audio system supports two simultaneous audio streams. Each stereo stream permits either playback or record, thus allowing two stereo playback streams to the BtV2300. The APU contains a digital mixer that can be used to blend both left streams together and both right streams together. The output of the AES decoder is always available for mixing with the playback data stream, even if neither of the audio streams is set up for recording.

The APU can provide decompression of various PCM and ADPCM formats as summarized in Table 153 through Table 163.

The APU provides other services such as peak value detection and an average value accumulation.

Figure 57. Audio Processing Unit

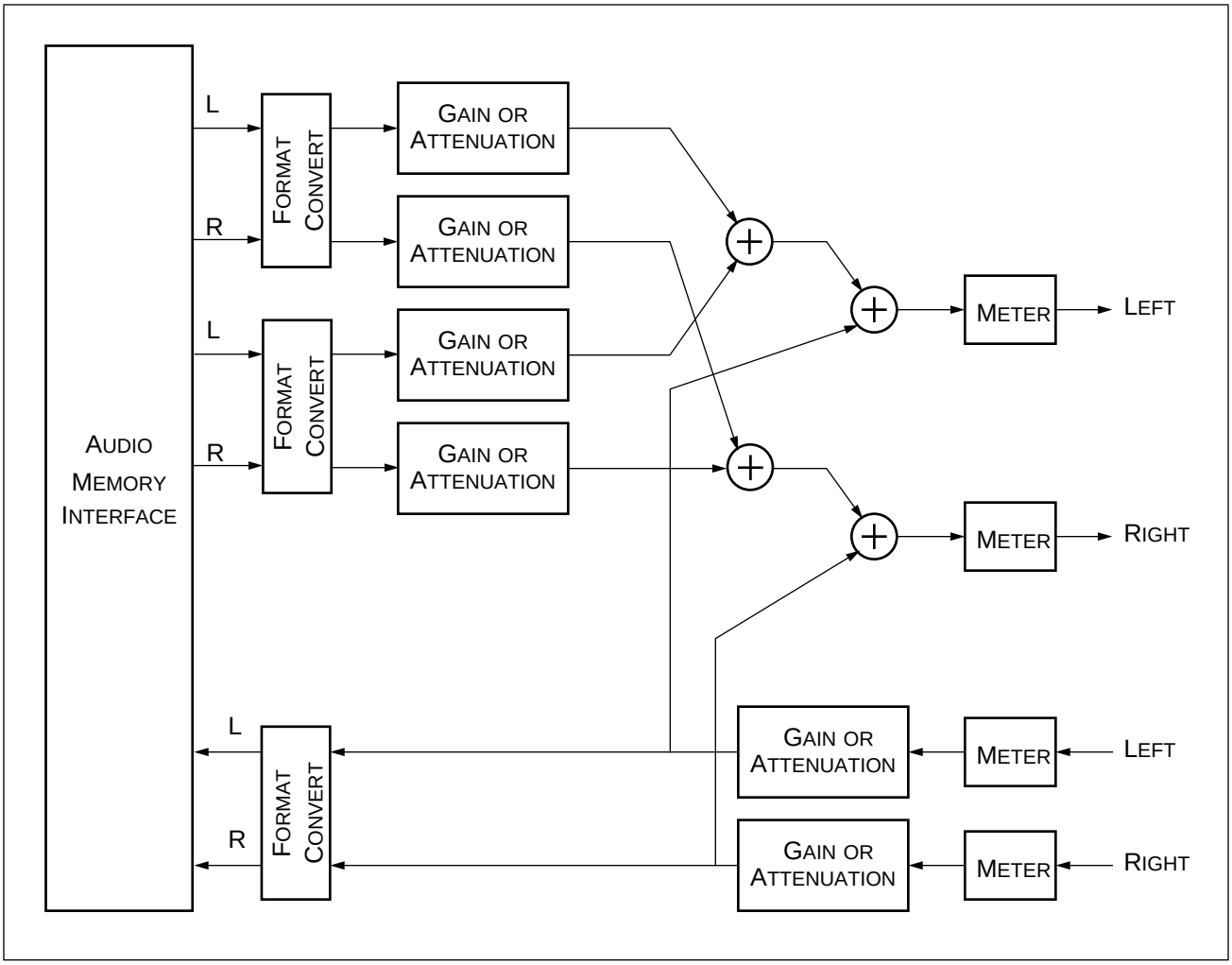
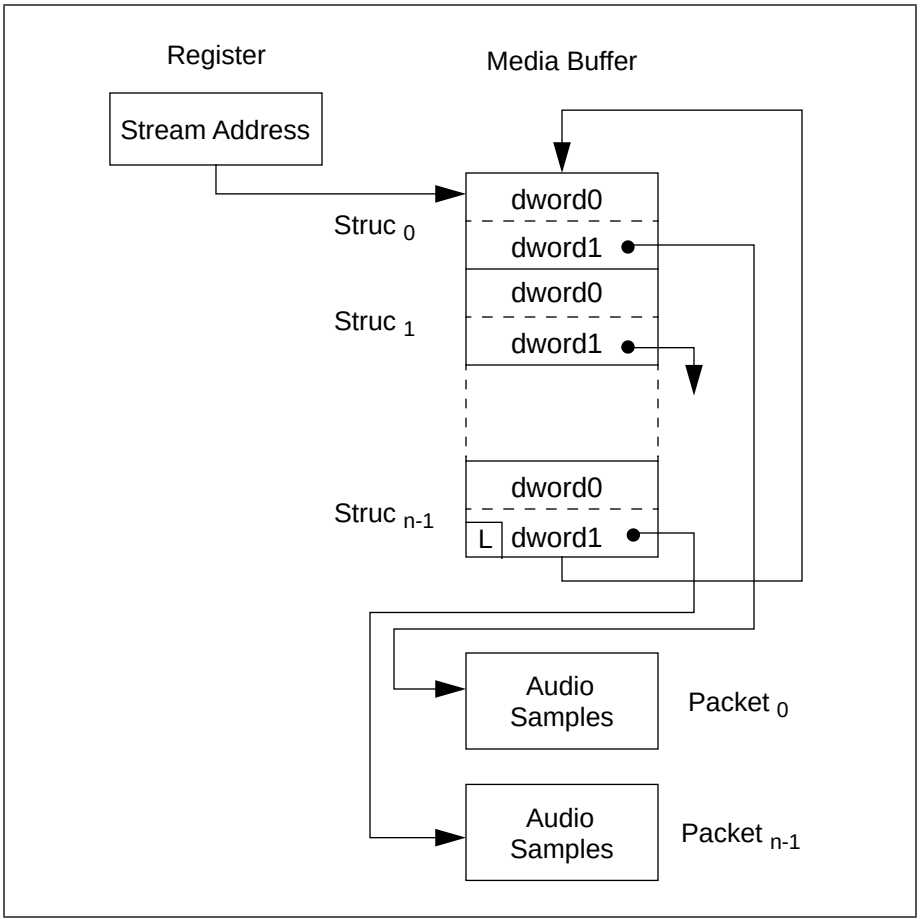


Figure 58. Stream and Data Structure



In addition to handling fetching or storing of the two stereo streams, the Audio Memory Interface (AMI) is responsible for fetching, following and executing the audio structure list (as shown in Figure 58). Each structure element in the media buffer belongs either to the primary list or the secondary list. The first structure in the primary list is pointed to by the primary structure address register. Similarly the first structure in the secondary list is pointed to by the secondary structure address register. All structure elements consist of two dwords in the format shown in Table 151 and Table 152.

Table 151. Audio Structure Dword 0

Bit	Field	Description
31	LAST	1= Go to top of structure list as designated by address in register 0 = Use next 2 dwords to continue
30	INTERRUPT	1= Generate interrupt when number of samples goes to zero 0= No interrupt
29:26	0	Reserved (set to zero).
25	SILENCE	1 = Set samples to zero, ignoring data 0 = Normal, use data
24:23	MODE	00= Stereo 01=Mono,only left channel 10=Mono, only right channel 11=Mono, both channels Mono puts mono source data on both channels for playback. For record, converts stereo source data to mono
22:19	STRUCTURE_ID	Structure ID for application use
18:16	0	Reserved (set to zero)
15:0	NUMBER_SAMPLES	Number of samples to playback or record in the specified format and mode

Table 152. Audio Structure Dword 1

Bit	Field	Description
31	0	Reserved (set to zero)
30:29	FB_FORMAT	00= data only 01= data+aux+vucb 10=vucb only 11= aux+vucb
28	ADPCM_REF	1 = If the data format is ADPCM then this packet contains a reference byte as first sample 0 = No reference byte
27:25	DATA_FORMAT	000= 16-bit PCM 001= 8-bit PCM 010 = Reserved 011 = Reserved 100 = Reserved 101 = 4-bit ADPCM† 110 = 2.6-bit ADPCM† 111 = 2-bit ADPCM†
24	PLAYBACK	1= playback 0 = record
23:20	0	Reserved (set to zero)
19:0	DATA_ADDRESS	Dword media buffer address of data for this structure
† Only for playback (not record)		

Audio Data Formats In the Media Buffer

In all stereo modes, the even samples contain left data and the odd samples contain right data. In all 16-bit formats, the data is represented as a 2's complement integer, and the sign bit is the most significant bit of the most significant byte, in little endian style. In all 8-bit PCM formats, the data is represented as an unsigned integer.

The VUCB format contains the same Valid, User, and Channel status bits as the VUCP format described in "AES/EBU Format" on page 272. The B bit indicates block boundaries (instead of Parity, which is generated within the encoder). For record operations, the B bit is set on both left and right VUCB samples at the block boundary. For playback, the B bit is used to mark the beginning of a new serial audio block. Note that if the playback format incorporates AUX, and the serial data stream is routed to the BtV2300 (as opposed to a null stream), all BtV2300 register programming is assumed by that media buffer packet and care should be taken to avoid unexpected results.

The primary and secondary streams are completely independent. Both may be used to play into a single stream. For example, primary may be used for data-only (in one of various formats) while secondary may be used in VUCB format, in which case both streams are merged in the APU. Also, data-only formats from both streams of different types are also merged within the APU.

Table 153 through Table 163 show the decompression of various PCM and AD-PCM formats.

Table 153. 16-bit Format

Bit	Field	Description
31:16	Sample 1	Sample 1
15:0	Sample 0	Sample 0

Table 154. 8-bit Format

Bit	Field	Description
31:24	Sample 3	Sample 3
23:16	Sample 2	Sample 2
15:8	Sample 1	Sample 1
7:0	Sample 0	Sample 0

Table 155. 4-bit ADPCM Format with Reference

Bit	Field
ONLY MONO SUPPORTED	
31:28	Sample 5
27:24	Sample 6
23:20	Sample 3
19:16	Sample 4
15:12	Sample 1
11:8	Sample 2
7:0	Sample 0 is a reference PCM byte.

Table 156. 4-bit ADPCM Format

Bit	Field
ONLY MONO SUPPORTED	
31:28	Sample 6
27:24	Sample 7
23:20	Sample 4
19:16	Sample 5
15:12	Sample 2
11:8	Sample 3
7:4	Sample 0
3:0	Sample 1

Table 157. 2.6-bit ADPCM Format with Reference

Bit	Field
ONLY MONO SUPPORTED	
31:29	Sample 7
28:26	Sample 8
25:24	Sample 9
23:21	Sample 4
20:18	Sample 5
17:16	Sample 6
15:13	Sample 1
12:10	Sample 2
9:8	Sample 3
7:0	Sample 0 is a reference PCM byte.

Table 158. 2.6-bit ADPCM

Bit	Field
ONLY MONO SUPPORTED	
31:29	Sample 9
28:26	Sample 10
25:24	Sample 11
23:21	Sample 6
20:18	Sample 7
17:16	Sample 8
15:13	Sample 3
12:10	Sample 4
9:8	Sample 5
7:5	Sample 0
4:2	Sample 1
1:0	Sample 2

Table 159. 2-bit ADPCM Format With Reference

Bit	Field
ONLY MONO SUPPORTED	
31:30	Sample 9
29:28	Sample 10
27:26	Sample 11
25:24	Sample 12
23:22	Sample 5
21:20	Sample 6
19:18	Sample 7
17:16	Sample 8
15:14	Sample 1
13:12	Sample 2
11;10	Sample 3
9:8	Sample 4
7:0	Sample 0 is a reference PCM byte.

Table 160. 2-bit ADPCM Format

Bit	Field
ONLY MONO SUPPORTED	
31:30	Sample 12
29:28	Sample 13
27:26	Sample 14
25:24	Sample 15
23:22	Sample 8
21:20	Sample 9
19:18	Sample 10
17:16	Sample 11
15:14	Sample 4
13:12	Sample 5
11;10	Sample 6
9:8	Sample 7
7:6	Sample 0
5:4	Sample 1
3:2	Sample 2
1:0	Sample 3

Table 161. VUCB Only Format

Bit	Field
31:28	VUCB 7
27:24	VUCB 6
23:20	VUCB 5
19:16	VUCB 4
15:12	VUCB 3
11:8	VUCB 2
7:4	VUCB 1
3:0	VUCB 0

Table 162. AUX+VUCB Format

Bit	Field
31:28	VUCB 3
27:24	AUX 3
23:20	VUCB 2
19:16	AUX 2
15:12	VUCB 1
11:8	AUX 1
7:4	VUCB 0
3:0	AUX 0

Table 163. DATA+AUX+VUCB Format

Bit	Field
31:28	VUCB 0
27:24	AUX 0
23:16	SET TO ZERO
15:0	DATA Sample 0

Audio Registers

The audio registers are memory mapped and are accessible from either the protected mode aperture or the real mode aperture, see “Protected Mode Aperture” on page 101 and “Real Mode Aperture” on page 127.

Although 1MB of audio space is reserved in the protected mode aperture only 1024 bytes are used. Two LSBs of the address are encoded in the byte enables. Only 8 of the remaining bits are actually decoded to yield 256 dwords of audio register space. These 256 dwords alias repeatedly through the 1MB of protected mode space reserved for audio.

Unless specified otherwise, all audio registers are set to zero at reset.

All reserved bits return zero when read, and should be written with zero to maintain compatibility with future versions or revisions of the device.

Table 164 summarizes the audio registers.

Table 164. Audio Address Map *(1 of 3)*

Register Address	Name	Description
00500000	AUD_ISR	Interrupt Status
00500002	AUD_IMR	Interrupt Mask
00500004	AUD_ISTR	Interrupt State
00500006	AUD_APPR	Application Port Read Data
00500007	AUD_CID	Codec ID
00500008	AUD_SDCR	Serial Data Control
00500009	AUD_EBL	Encoder Block Length
0050000A	AUD_EBP	Encoder Block Position
0050000B	AUD_DSR	Decoder Status
00500010	AUD_CH0	Channel Status Byte 0
00500011	AUD_CH1	Channel Status Byte 1
00500012	AUD_CH2	Channel Status Byte 2
00500013	AUD_CH3	Channel Status Byte 3
00500040	AUD_PSA	Primary Stream Address
00500044	AUD_PSC	Primary Stream Counter
00500048	AUD_SSA	Secondary Stream Address
0050004C	AUD_SSC	Secondary Stream Counter
00500050	AUD_PLL	Primary Left Level

Table 164. Audio Address Map *(2 of 3)*

Register Address	Name	Description
00500052	AUD_PRL	Primary Right Level
00500054	AUD_SLL	Secondary Left Level
00500056	AUD_SRL	Secondary Right Level
00500058	AUD_RMLL	Record Monitor Left Level
0050005A	AUD_RMRL	Record Monitor Right Level
0050005C	AUD_NS	Number summed register
00500060	AUD_LPSUM	Left Playback Sum
00500062	AUD_RPSUM	Right Playback Sum
00500064	AUD_LPMAX	Left Playback Max.
00500066	AUD_RPMAX	Right Playback Max.
00500068	AUD_LRSUM	Left Record Sum
0050006A	AUD_RRSUM	Right Record Sum
0050006C	AUD_LRMAX	Left Record Max.
0050006E	AUD_RRMAX	Right Record Max.
005000BC	AUD_CAR	Codec Address
005000C0	AUD_CLK	Clock
005000C2	AUD_FCLK	Filter Clock Divisor
005000C3	AUD_CMODE	Codec Mode
005000C4	AUD_APPO	Application Port Direction
005000C5	AUD_APPD	Application Port Write Data
005000C6	AUD_MIX	Mixer Control
005000C7	AUD_MUX	MUX Selector
005000C8	AUD_CDL	CD Left Attenuation
005000C9	AUD_CDR	CD Right Attenuation
005000CA	AUD_LL	Line Left Attenuation
005000CB	AUD_LR	Line Right Attenuation
005000CC	AUD_FML	FM Left Attenuation
005000CD	AUD_FMR	FM Right Attenuation

Table 164. Audio Address Map *(3 of 3)*

Register Address	Name	Description
005000CE	AUD_MICL	MIC Left Gain
005000CF	AUD_MICR	MIC Right Gain
005000D0	AUD_DACL	DAC Left Attenuation
005000D1	AUD_DACR	DAC Right Attenuation

Interrupt Status Register

name: AUD_ISR
address: GBASE | 00500000h, read/write
size: 16 bit

On read, this register indicates the active/inactive state for each interrupt. The actual interrupt to the bus is a reduction of the interrupt status register by a logical OR. The interrupt state register is bit-wise ANDed with the interrupt mask register to generate the status register.

Interrupt status bit definitions are as shown in Table 165.

A type of "event" means that the interrupt occurs on the leading edge of the event and will remain until cleared by writing to either the interrupt status or state register. A type of "level" means that the value of the interrupt is passed through (using the mask) and must be reset via some other mechanism, e.g. the VGA vertical interval interrupt mechanism. Writing a 1 to individual bits of the status register will reset that particular interrupt, if it is of type "event."

Table 165. Interrupt Status, State and Mask Register Bits *(1 of 2)*

Bit	Field	Type	Description
15	LA_COMMAND_INT	Event	Legacy Audio Command (event)
14	LA_PRO_WRITE_INT	Event	Legacy Audio Pro write command (event)
13	PRIMARY_INT	Event	Audio primary structure requested IRQ (event).
12	SECONDARY_INT	Event	Audio secondary structure requested IRQ (event)
11	IIC_MASTER_INT	Level	I^2C master controller IRQ (level)
10	IIC_SLAVE_INT	Level	I^2C slave controller IRQ (level)
9	VIDEO_CAPTURE_INT	Event	Video input subsystem, structure requesting IRQ (event)
8	VGA_RETRACE_INT	Level	VGA or PDC requesting IRQ (level)
7	EXTERNAL_INT	Level	From an on-board external device

Table 165. Interrupt Status, State and Mask Register Bits *(2 of 2)*

Bit	Field	Type	Description
6:5	Reserved	N/A	Reserved
4	AES_DECODER_INT	Event	AES Decoder went into lock, or went out of lock (event)
3	AUDIO_CLIPPED	Event	Audio signal was clipped; examine metering registers to determine which channel/direction (event)
2:0	Reserved	N/A	Reserved

Interrupt Mask Register

name: AUD_IMR
address: GBASE | 00500002h, read/write
size: 16 bit
function: This read/write register contains a bit-for-bit mask for each interrupt source. 1 = enable the corresponding interrupt. 0 = mask off the corresponding interrupt type. The contents of this register are bit-wise ANDed with the interrupt state register bits to form the interrupt status register and to generate a request to the CPU bus.

Refer to Table 165 for a mapping of interrupt types to bit positions for this register. Reserved interrupts cannot be masked.

Interrupt State Register

name: AUD_ISTR
address: GBASE | 00500004h, read/write
size: 16 bit
function: This read/write register contains the storage elements for interrupt types that are flagged as events. Interrupt types that are flagged as levels do not actually have a storage element here but are passed through as if they were in this register. The contents of this register are bit-wise ANDed with the interrupt mask register bits to present an interrupt request to the CPU bus. Writing a 1 to individual bits of the state register will reset that particular interrupt type in the interrupt status register, if it is of type “event.”

Refer to Table 165 for a mapping of interrupt types to bit positions for this register.

Application Port Read Data Register

name: AUD_APPR
address: GBASE | 00500006h, read only
size: 8 bit
function: This read only register provides an access port for software to read the state of the BtV2300 applications port. When the corresponding bit in AUD_APPO is a zero, the value applied externally to the BtV2300 applications port pin can be read here by software. The applications port write data is in AUD_APPD.

For this register to be valid, AUD_SDCR loopback (see Table 167) must be set to normal and the AES decoder must be locked (see Table 165) and have read at least two frames.

BtV2300 Chip ID Register

name: AUD_CID
address: GBASE | 00500007h, read only
size: 8 bit
function: This read only register returns the chip ID information of the BtV2300. In addition, it returns a status bit indicating when the circuitry in the BtV2300 is undergoing recalibration.

After reset, this register is not valid until the decoder is locked (see Table 165) and the first frame has been read. AUD_SDCR loopback (see Table 167) must be set to normal.

Bit definitions are shown in Table 166.

Table 166. AUD_CID BtV2300 Chip ID Register

Bit	Field	Description
7:4	MAJOR_ID	[4:0] = 0h is major ID
3:1	MINOR_ID	[4:0] = 4h or greater
0	CALIB_STATUS	1= BtV2300 being recalibrated 0= calibration complete

Serial Data Control Register

name: AUD_SDCR
address: GBASE | 00500008h, read/write
size: 8 bit
function: This register controls various global aspects of the serial data stream, including which serial output pin is driven by the AES encoder and which is driven by the NULL encoder. Refer to Table 167. This register defaults to 08h (loopback SA_OUT).

Table 167. Serial Data Control Register

Bit	Field	Description
7	Reserved	Reserved.
6	NULL_SA_OUT	Put nulls onto the SA_OUT pin which is connected to the BtV2300 so that it will ignore the AES stream directed to a consumer device.
5	NULL_AES_OUT	Put nulls onto the AES_OUT pin which may be connected to consumer device so that it will ignore the AES stream directed to the BtV2300.
4:3	LOOPBACK	Signal source for the decoder: 00= normal operation (SA_IN) 01= loopback SA_OUT 10= loopback AES_OUT 11= reserved
2	DISABLE_PARITY	1= Disable parity checking in the AES decoder.
1	ODD_PARITY_ENCODER	1= Generate odd parity in the encoder 0 = Generate even parity in the encoder
0	ODD_PARITY_DECODER	1= Use odd parity in the encoder 0= Use even parity in the encoder

Encoder Block Length Register

name: AUD_EBL
address: GBASE | 00500009h, read/write
size: 8 bit
function: This register provides the encoder block length. Set the block length to one less than the desired block length. For AES/SPDIF compatibility use 0BFh (191).

After reset, AUD_EBL is set to 4 for a block length of 5, which is the minimum block length for communicating with the BtV2300. Maximum block length is 256. (Actual block length range is 5 to 256, while valid AUD_EBL range is 4 to 255.) Due to the way the BtV2300 registers are shadowed and transmitted, the shorter the block length, the shorter the latency time on register changes.

Encoder Block Position Register

name: AUD_EBP
address: GBASE | 0050000Ah, read only
size: 8 bit
function: This diagnostic register shows the binary count of frames left within a block (i.e. it counts down from n-1 to zero, where n is the number of frames in a block).

Decoder Status Register

name: AID_DSR
address: GBASE | 0050000Bh, read only
size: 8 bit
function: This diagnostic and error recovery register provides information about the current status of the AES decoder. Refer to Table 168.

Table 168. Decoder Status Register

Bit	Field	Description
7	LOCKED	1= Decoder is locked to an incoming AES stream.
6	SIGNAL_PRESENT	1= Decoder detects signal transitions on the selected input data stream.
5:0	Reserved	

Reserved

name: Reserved
address: GBASE | 0050000Ch
size: 32 bit

Channel Status Byte 0 Register

name: AUD_CH0
address: GBASE | 00500010h, read only
size: 8 bit
function: This register contains the first byte of channel status data from the selected signal source selected (see Table 167), as defined by AES/SPDIF (see Table 198).

Channel Status Byte 1 Register

name: AUD_CH1
address: GBASE | 00500011h, read only
size: 8 bit
function: This register contains the second byte of channel status data from the selected signal source selected (see Table 167), as defined by AES/SPDIF (see Table 198).

Channel Status Byte 2 Register

name: AUD_CH2
address: GBASE | 00500012h, read only
size: 8 bit
function: This register contains the third byte of channel status data from the selected signal source selected (see Table 167), as defined by AES/SPDIF (see Table 198).

Channel Status Byte 3 Register

name: AUD_CH3
address: GBASE | 00500013h, read only
size: 8 bit
function: This register contains the fourth byte of channel status data from the selected signal source selected (see Table 167), as defined by AES/SPDIF (see "Channel Status Usage" on page 313). Bytes 5 through 24 can only be determined by recording in a VUCB mode.

Reserved

name: Reserved
address: GBASE | 00500014 through GBASE | 0050003C
size: 11 locations of 32 bits each

Primary Stream Address Register

name: AUD_PSA
address: GBASE | 00500040h, read/write
size: 32 bit
function: This register contains the media buffer address of the 1st control structure associated with the primary stream, along with some control bits. Refer to Table 169.

Setting ENABLE_PRIMARY to a zero will disable the processing of audio and control structure data. The transition of ENABLE_PRIMARY from a 0 to a 1 will begin the processing of control structure information beginning at the PRIMARY_ADDRESS.

Setting PAUSE_PRIMARY to a 1 will cause the primary audio stream to pause, holding the current value. Processing of the stream data and control structures will continue when PAUSE_PRIMARY is set to zero.

The primary buffer counter (see "AUD_PSC" on page 297) may be reset by setting ENABLE_PRIMARY_COUNTER to a 0. This bit must be set to a 1 to enable the fetching and processing of primary control structures and data.

Table 169. Primary Address Register

Bits	Field	Description
31	ENABLE_PRIMARY	1 = Enable primary stream. 0 = Disable primary stream.
30	PAUSE_PRIMARY	1 = Pause the processing of primary stream data. 0 = Continue processing.
29	ENABLE_PRIMARY_ COUNTER	1 = Count primary structures 0 = Hold counter in reset sate
28:22	Reserved	
21:2	PRIMARY_ADDRESS	Media buffer byte address of the first structure in the primary list.
1:0		Always 0 (dword aligned)

Primary Stream Counter Register

name: AUD_PSC
address: GBASE | 00500044h, read/write
size: 32 bit
function: This register provides read access to a number of status indicators for the primary stream. Its main function is to support the semaphore counter which allows software to copy audio buffers to the media buffer and easily mark the associated structure as available for processing. Every time a write occurs to bits 31:24, the primary stream counter is incremented. Every time hardware completes the processing of a structure the counter is decremented until zero. The next structure in the list will not be fetched or processed unless the PRIMARY_COUNTER is non-zero. Refer to Table 170.

Table 170. Primary Stream Counter *(1 of 2)*

Bits	Field	Description
31:24	PRIMARY_COUNTER	Primary stream counter is incremented by writes of any data value to this byte. Counter is decremented when an audio structures and associated data has been processed.
23:20	PRIMARY_ID	Read only†
19	Reserved	
18	SILENCE_PRIMARY	Read only†

Table 170. Primary Stream Counter *(2 of 2)*

Bits	Field	Description
17	PRIMARY_LAST	Read only†
16	PRIMARY_IRQ	Read only†
15:0	PRIMARY_LENGTH	Read only. Shows number of samples left to process. Range: 0 to n, where n = number of samples in the buffer.
† Value taken from media buffer structure currently being processed or the last one completed, if stream is idle.		

Secondary Stream Address Register

name: AUD_SSA
address: GBASE | 00500048h, read/write
size: 32 bit
function: This register contains the media buffer address of the 1st control structure associated with the secondary stream, along with some control bits. Refer to Table 171.

Setting ENABLE_SECONDARY to a zero will disable the processing of audio and control structure data. The transition of ENABLE_SECONDARY from a 0 to a 1 will begin the processing of control structure information beginning at the SECONDARY_ADDRESS.

Setting PAUSE_SECONDARY to a 1 will cause the secondary audio stream to pause, holding the current value. Processing of the stream data and control structures will continue when PAUSE_SECONDARY is set to zero.

The secondary buffer counter (see "AUD_SSC" on page 299) may be reset by setting ENABLE_SECONDARY_COUNTER to a 0. This bit must be set to a 1 to enable the fetching and processing of secondary control structures and data.

Table 171. Secondary Address Register *(1 of 2)*

Bits	Field	Description
31	ENABLE_SECONDARY	1 = Enable secondary stream. 0 = Disable secondary stream.
30	PAUSE_SECONDARY	1 = Pause the processing of secondary stream data. 0 = Continue processing

Table 171. Secondary Address Register *(2 of 2)*

Bits	Field	Description
29	ENABLE_SECONDARY_ COUNTER	1= Count secondary structures 0= Hold counter in reset sate
28:22	Reserved	
21:2	SECONDARY_ADDRESS	Address of the first structure in the secondary list.
1:0		Always zero (dword aligned).

Secondary Stream Counter Register

name: AUD_SSC
address: GBASE | 0050004Ch, read/write
size: 32 bit
function: This register provides read access to a number of status indicators for the secondary stream. Its main function is to support the semaphore counter which allows software to copy audio buffers to the media buffer and easily mark the associated structure as available for processing. Every time a write occurs to bits 31:24, the secondary stream counter is incremented. Every time hardware completes the processing of a structure the counter is decremented until zero. The next structure in the list will not be fetched or processed unless the SECONDARY_COUNTER is non-zero. Refer to Table 172.

Table 172. Secondary Stream Counter

Bits	Field	Description
31:24	SECONDARY_COUNTER	Secondary stream counter is incremented by writes of any data value to this byte. Counter is decremented as audio structures complete processing.
23:20	SECONDARY_ID	Read only†
19	Reserved	
18	SILENCE_SECONDARY	Read only†
17	SECONDARY_LAST	Read only†
16	Reserved	Read only†
15:0	SECONDARY_LENGTH	Read only. Shows number of samples left to process. Range: 0 to n, where n = number of samples in the buffer.
† Value taken from media buffer structure currently being processed or the last one completed, if stream is idle.		

Level Registers

All level registers use 6-bit values to digitally control gain or attenuation over the range of +18.0 dB to -75.0 dB (including full mute) in 1.5 dB steps. Refer to Table 173. Note that applying gain may cause the signal to clip, which can be detected in bit 3 of the Interrupt State (and potentially Status) Registers. Refer to Table 165 on page 291.

Table 173. Level Register Values

Bit[5:0]	Gain or Attenuation
111111	+18.0 dB
111110	+16.5 dB
.....	.....
110100	+1.5 dB
110011	0.0 dB
110010	-1.5 dB
.....	.....
000010	-73.5 dB
000001	-75.0 dB
000000	mute

Primary Level Registers

name: AUD_PLL, AUD_PRL
address: GBASE | 00500050h, read/write
size: 32 bit
function: This register controls the left and right audio signal level on the primary playback stream, after format conversion and prior to being mixed with the secondary or record streams. Refer to Table 174.

Table 174. Primary Level Registers

Bits	Field	Description
31:14	Reserved	
13:8	RIGHT_LEVEL	Values from Table 173
7:6	Reserved	
5:0	LEFT_LEVEL	Values from Table 173

Secondary Level Registers

name: AUD_SLL, AUD_SRL
address: GBASE | 00500054h, read/write
size: 32 bit
function: This register controls the left and right audio signal level on the secondary playback stream, after format conversion and prior to being mixed with the primary or record streams. Refer to and Table 175.

Table 175. Secondary Level Registers

Bits	Field	Description
31:14	Reserved	
13:8	RIGHT_LEVEL	Values from Table 173
7:6	Reserved	
5:0	LEFT_LEVEL	Values from Table 173

Record Monitoring Level Registers

name: AUD_RMLL, AUD_RMRL
address: GBASE | 00500058h, read/write
size: 32 bit
function: These left and right record monitoring levels do not affect the data going to memory (if recording) and are only used for mixing the monitored record data into the playback stream for monitoring. The exception to this rule is when one of the streams is recording in the mode “mono - both channels” (see Table 151 on page 282). In this case, AUD_RMLL will affect both the monitor and record data. Refer to Table 176.

Table 176. Record Monitoring Levels

Bits	Field	Description
31:13	Reserved	
12:8	RIGHT_MONITOR	Values from Table 173
7:6	Reserved	
5:0	LEFT_MONITOR	Values from Table 173

Number Summed Register

name: AUD_NS[1:0]
address: GBASE | 0050005Ch, read only
size: 32 bit
function: The APU accumulates a per channel average of the sample values so that a “quasi” power meter function can be implemented without bogging down the x86. It is the X86’s job to do the division to achieve a signal average value. This register provides the denominator for all channels in play and/or record mode and is reset when read. Refer to Table 177.

Table 177. Number Summed

Bit	Field	Description
31:16	Reserved	
15:0	NUMBER_SUMMED	Increments at sample rate. Cleared when read.

Right/Left Playback Sum Registers

name: AUD_LPSUM, AUD_RPSUM
address: GBASE | 00500060h, read only
size: 32 bit
function: The sum of playback samples is a 32-bit unsigned value and only the upper 16 bits are made available to software. Thus a significant number of samples must be accumulated before a meaningful average can be computed. All bits, including the least significant bits which are not viewable are zeroed when read. Refer to Table 178.

Table 178. Playback Sum Registers

Bit	Field	Description
31:16	RIGHT_SUM	Right playback sum.
15:0	LEFT_SUM	Left playback sum

Right/Left Playback Max Registers

name: AUD_LPMAX, AUD_RPMAX
address: GBASE | 00500064h, read only
size: 32 bit
function: The APU implements a maximum absolute value function with the current maximum value of playback held in this register. This value is reset when read. Refer to Table 179.

Table 179. Playback Max Registers

Bit	Field	Description
31:16	RIGHT_MAX	Right playback maximum (max value is 7FFFh)
15:0	LEFT_MAX	Left playback maximum (max value is 7FFFh)

Right/Left Record Sum Registers

name: AUD_LRSUM, AUD_RRSUM
address: GBASE | 00500068h, read only
size: 32 bit
function: The sum of record samples is a 32-bit signed value and only the upper 16 bits are made available to software. Thus a significant number of samples must be accumulated before a meaningful average can be computed. All bits, including the least significant bits which are not viewable, are zeroed when read. Refer to Table 180.

Table 180. Record Sum Registers

Bit	Field	Description
31:16	RIGHT_SUM	Right record sum.
15:0	LEFT_SUM	Left record sum

Right/Left Record Max Registers

name: AUD_LRMAX, AUD_RRMAX
address: GBASE | 0050006Ch, read only
size: 32 bit
function: The APU implements a maximum absolute value function with the current maximum value held in AUD_LRMAX and AUD_RRMAX. This value is reset when read. Refer to Table 181.

Table 181. Record Max Registers

Bit	Field	Description
31:16	RIGHT_MAX	Right record max
15:0	LEFT_MAX	Left record max

Legacy Audio Emulation Registers

name: LA_EMULATION_REGISTERS
address: GBASE | 00500080h through GBASE | 005000BBh
size: 32-bit
function: These registers are used to emulate the Legacy Audio function and are for internal purposes only. However, register 00500080h controls the Legacy Audio Address decoder base address select (022xh when bit 6 = 0, 024xh when bit 6 = 1), as well as enabling of the Legacy Audio I/O decoder (disabled when bit 7 = 0, enabled when bit 7 = 1). This register should be accessed in a read-modify-write manner.

BtV2300 Address Register

name: AUD_CAR
address: GBASE | 005000BCh, read only
size: 8 bit
function: This register reflects the current setting of the BtV2300 address register and is used to track which BtV2300 register is currently marked as the first one to be sent in the next AES block.

BtV2300 Register Shadows

The registers within the BtV2300 are write-only and cannot be read directly by the BtV2115. However, the BtV2300 registers are “shadowed” within the BtV2115 so that software can read the last value written to the BtV2300. This section defines the shadowed registers.

Clock Register

name: AUD_CLK
address: GBASE | 005000C0h, read/write
size: 16 bit
function: The Clock Register is used by both the BtV2115 and the BtV2300 and controls the sample frequency and clock division for the AES encoder and decoder. Two clocks are available: 16.9344 MHz (CLK17) and 24.576 MHz (CLK25), both specified to be +/- 0.01% in accuracy.

The SEL_CLK17 field selects which clock to use and is output to the AudioStream Interface via the AUDCLK_OUT signal pin.

CLK_DIVISOR is a 13-bit clock divider value used to load an up-counter (at the selected clock rate) to set the sample rate of audio, independent of whether the data stream is mono or stereo.

When a new clock divisor is written to this register, the CLK_DIV_CHANGE bit is set. This bit is then reset after the new clock divisor is transmitted to the BtV2300.

Refer to Table 182 for bit content of the clock register. See Table 183 for several representative examples.

This register defaults to 00050200h (48KHz, CLK25). In some cases, this register performs like a shadow register.

Table 182. Clock Registers

Bits	Field	Description
15	Reserved	
14	CLK_DIV_CHANGE	1= clock divisor changed (read only)
13	SEL_CLK17	1= use CLK17 0 = use CLK25
12:0	CLK_DIVISOR	See Table 183

Table 183. Sample Rate Examples

Sample Frequency	Clock	DIV,decimal	DIV,hex
48,000	CLK25	512	200
44,100	CLK17	384	180
32,000	CLK25	768	300
11,025	CLK17	1536	600
8000	CLK25	3072	C00

Filter Clock Divisor Register

name: AUD_FCLK
address: GBASE | 005000C2h, read/write
size: 8 bit
function: This register contains the filter clock divisor, as shown in Table 184. It is used to generate the clock within the BtV2300 to drive an external switched-cap filter. Either CLK17 or CLK25 (which ever is selected) is divided by two and then divided by this 6-bit divisor. The value of FILTER_DIVISOR should be such that the filter clock is 50 (or 100) times the cut-off frequency, depending on the external filter in use. See the *BtV2300 AudioStream Interface* specification and associated application notes for more detailed information.

Table 184. Filter Clock Divisor

Bits	Field	Description
8:6	Reserved	
5:0	FILTER_DIVISOR	Filter clock divisor (N-1)

Codec Mode Register

name: AUD_CMODE
address: GBASE | 005000C3h, read/write
size: 8 bit
function: This register contains the mode and configuration information for the BtV2300 AudioStream Interface. Refer to Table 185.

Table 185. Codec Mode Register

Bits	Field	Description
7:6	IO MODE	BtV2300 IO MODE 00 = stereo out 01 = stereo in 10 = mono in, mono out 11 = reserved
5:4	ADC_CONFIG	BtV2300 ADC configuration 00 = internal ADC 01 = reserved 10 = reserved 11 = reserved
3:2	DAC_CONFIG	BtV2300 DAC configuration 00 = internal DAC 01 = reserved 10 = reserved 11 = reserved
1	TRISTATE_BtV2300_AES	1 = tristate BtV2300 AES output 0= normal operation
0	SELECT_AES_IN	1 = route AES_IN to serial audio out to BtV2115. 0 = normal operation (ADC to SA_OUT)

Application Port Direction Register

name: AUD_APPO
address: GBASE | 005000C4h, read/write
size: 8 bit
function: This register sets the bit-by-bit direction of the BtV2300 application interface data register. 1= output, 0= input.

Application Port Write Data Register

name: AUD_APPD
address: GBASE | 005000C5h, read/write
size: 8 bit
function: This register holds the data byte that will be or is being output on the BtV2300 application interface data port. Read data from the BtV2300 applications port is available in AUD_APPR.

BtV2300 Mixer Control Register

name: AUD_MIX
address: GBASE | 005000C6h, read/write
size: 8 bit
function: This register holds the shadow copy of the BtV2300 mixer control register. Refer to Table 186 for bit content of this register.

Table 186. BtV2300 Mixer Control Register Shadow

Bit	Field	Description
7:6	LOOP_TEST	00 = normal 01 = analog loopback, left 10 = analog loopback, right 11 = digital loopback
5	POWER_DOWN	1 = powerdown 0 = normal
4:0	MIXER_ATTENUATION	00000 = mute 00001 = -51dB 11111 = -6dB

BtV2300 MUX Selector Register

name: AUD_MUX
address: GBASE | 005000C7h, read/write
size: 8 bit
function: This register holds the shadow copy of the BtV2300 mux selector register. Refer to Table 187 for bit content of this register.

Table 187. BtV2300 MUX Selector Register Shadow *(1 of 2)*

Bit	Field	Description
7	RIGHT_CHANNEL	1 = bits 5:0 update only right channel 0 = bits 5:0 update both left and right
6	RECALIBRATE	1 =force a recalibrate operation in the BtV2300 0 = normal operation

Table 187. BtV2300 MUX Selector Register Shadow *(2 of 2)*

Bit	Field	Description
5	MIC_MUTE	1 = pass the mic 0 = mute the mic
4	MUX_GAIN	1 = +12dB 0 = +6dB
3	MIX_GAIN	1 = +12dB 0 = +6dB
2:0	MIXER_SELECTOR	000 = mute the mux 001 = CD analog 010 = Line input 011 = FM Synthesizer (OPL3) 100 = Mic 101 = mixer output 110 = DAC output 111 = reserved

CD Left Attenuation Register

name: AUD_CDL
address: GBASE | 005000C8h,read/write
size: 8 bit
function: This register shadows the BtV2300 left CD attenuation register. Refer to Table 188 for bit content of this register.

Table 188. CD Left Attenuation Register

Bit	Field	Description
7:5	Reserved	
4:0	CD_LEFT_ATTEN	00000 = mute 00001 = -51dB 11111 = -6dB increment in steps of 1.5dB per bit

CD Right Attenuation Register

name: AUD_CDR
address: GBASE | 005000C9h, read/write
size: 8 bit
function: This register shadows the BtV2300 right CD attenuation register Refer to Table 189 for bit content of this register.

Table 189. CD Right Attenuation Register

Bit	Field	Description
7:5	Reserved	
4:0	CD_RIGHT_ATTEN	00000 = mute 00001 = -51dB 11111 = -6dB increment in steps of 1.5dB per bit

Line Left Attenuation Register

name: AUD_LL
address: GBASE | 005000CAh, read/write
size: 8 bit
function: This register shadows the BtV2300 left line input attenuation register. Refer to Table 190 for bit content of this register.

Table 190. Line Input Left Attenuation Register

Bit	Field	Description
7:5	Reserved	
4:0	LINE_LEFT_ATTEN	00000 = mute 00001 = -51dB 11111 = -6dB increment in steps of 1.5dB per bit

Line Right Attenuation Register

name: AUD_LR
address: GBASE | 005000CBh, read/write
size: 8 bit
function: This register shadows the BtV2300 right line input attenuation register Refer to Table 191 for bit content of this register.

Table 191. Line Input Right Attenuation

Bit	Field	Description
7:5	Reserved	
4:0	LINE_RIGHT_ATTEN	00000 = mute 00001 = -51dB 11111 = -6dB increment in steps of 1.5dB per bit

FM Left Attenuation Register

name: AUD_FML
address: GBASE | 005000CCh, read/write
size: 8 bit
function: This register shadows the BtV2300 left FM synthesis input attenuation register. Refer to Table 192 for bit content of this register.

Table 192. FM Synthesis Input Left Attenuation

Bit	Field	Description
7:5	Reserved	
4:0	FM_LEFT_ATTEN	00000 = mute 00001 = -51dB 11111 = -6dB increment in steps of 1.5dB per bit

FM Right Attenuation Register

name: AUD_FMR
address: GBASE | 005000CDh, read/write
size: 8 bit
function: This register shadows the BtV2300 right FM synthesis input attenuation register. Refer to Table 193 for bit content of this register.

Table 193. FM Synthesis Input Right Attenuation

Bit	Field	Description
7:5	Reserved	
4:0	FM_RIGHT_ATTEN	00000 = mute 00001 = -51dB 11111 = -6dB increment in steps of 1.5dB per bit

MIC Left Gain Register

name: AUD_MICL
address: GBASE | 005000CEh, read/write
size: 8 bit
function: This register shadows the BtV2300 left microphone input gain register. Default value is 1Fh (minimum gain). Refer to Table 194 for bit content of this register.

Table 194. MIC Input Left Gain

Bit	Field	Description
7:5	Reserved	
4:0	MIC_LEFT_GAIN	00000 = illegal 00001 =+51dB 11111 = +6dB increment in steps of 1.5dB per bit

MIC Right Gain Register

name: AUD_MICR
address: GBASE | 005000CFh, read/write
size: 8 bit
function: This register shadows the BtV2300 right microphone input gain register. A setting of zero implies infinite gain, therefore is illegal. Refer to Table 195 for bit content of this register.

Table 195. MIC Input Right Gain

Bit	Field	Description
7:5	Reserved	
4:0	MIC_RIGHT_GAIN	00000 = illegal 00001 = +51dB 11111 = +6dB increment in steps of 1.5dB per bit

DAC Left Attenuation Register

name: AUD_DACL
address: GBASE | 005000D0h, read/write
size: 8 bit
function: This register shadows the BtV2300 left DAC input attenuation register. Refer to Table 196 for bit content of this register.

Table 196. DAC Synthesis Input Left Attenuation

Bit	Field	Description
7:5	Reserved	
4:0	DAC_LEFT_ATTEN	00000= mute 00001 = -51dB 11111 = -6dB

DAC Right Attenuation Register

name: AUD_DACR
address: GBASE | 005000D1h, read/write
size: 8 bit
function: This register shadows the BtV2300 right DAC input attenuation register. Refer to Table 197 for bit content of this register.

Table 197. DAC Synthesis Input Right Attenuation

Bit	Field	Description
7:5	Reserved	
4:0	DAC_RIGHT _ATTEN	00000 = mute 00001 = -51dB 11111 = -6dB

Channel Status

The channel status field for the audio output from BtV2115 has two modes: either the values shown in Table 198 or values from the playback channel status region of the frame buffer, depending on which mode is enabled. When not using channel status data from the frame buffer, the channel status block closely resembles the AES consumer format.

Table 198. Channel Status Usage *(1 of 3)*

Byte	Professional (AES/EBU)	Consumer (S/PDIF)
0	**Bits 7:6 = Sample Rate** 00 = not indicated 01 = 44.1 KHz 10 = 48 KHz 11 = 32 KHz **Bit 5 = Source Locked** 0 = sample rate locked 1 = sample rate unlocked **Bits 4:2 = Emphasis** 000 = not indicated 001 = no emphasis 011 = 50/15 µs 111 = CCITT J.17 all others = reserved **Bit 1 = Mode** 0 = digital audio 1 = digital data **Bit 0 = Channel Format** 0 = consumer 1= professional	**Bits 7:6 = Mode** 00 = mode 0 all others = reserved **Bits 5:3 = Pre-emphasis (audio)** 000 = none 001 = 50/15 µs - 2 channel 010 = reserved - 2 channel 011 = reserved - 2 channel 1XX = reserved - 4 channel - or - **Bits 5:3 = Non-Audio** 000 = digital data all others = reserved **Bit 2 = Copy Protect** 0 = copy inhibited/copyright asserted 1 = copy permitted/copyright not asserted **Bit 1 = Mode** 0 = digital audio 1 = digital data **Bit 0 = Channel Format** 0 = consumer 1 = professional

Table 198. Channel Status Usage *(2 of 3)*

Byte	Professional (AES/EBU)	Consumer (S/PDIF)
1	**Bits 7:4 = User bits management** 0000 = no user data indicated 1000 = 192 bit block 1100 = user defined operation all others = reserved (undefined) **Bits 3:0 = Mode** 0000 = not indicated 0010 = stereo 0100 = single channel (mono) 1000 = 2 channel 1100 = primary/secondary 1111 = vector to byte 3 all others = reserved	**Bit 7 = Mode** 0 = consumer or professional 1 = program transfer mode if byte 0, bit 0 = 1
2	**Bits 7:6 = Reserved (undefined)** **Bits 5:3 = Source word length[a]** 000 = not indicated 001 = 20/16 bits 010 = 22/18 bits 100 = 23/19 bits 101 = 24/20 bits 110 = 21/17 bits all others = reserved **Bits 2:0 = Auxiliary bits usage** 000 = aux not defined (20-bit audio) 010 = single coordination signal (20-bit) 100 = used for sample data (24-bit) 110 = user defined operation all others = reserved	
3	Vector for Multi-channel 8'b0 (future)	**Bits 7:6 = Reserved** **Bits 5:4 = Clock Accuracy** 00 = Level II (±1000 ppm) 01 = Level I (±50 ppm) 10 = Level III (variable) 11 = Reserved **Bits 3:0 = Sample Frequency** 0000 = 44.1 KHz 0010 = 48 KHz 0011 = 32 KHz all others = reserved

Table 198. Channel Status Usage *(3 of 3)*

Byte	Professional (AES/EBU)	Consumer (S/PDIF)
4	**Bits 7:2 = Reserved** **Bits 1:0 = Reference Signal**[c] 00 = not reference signal 01 = grade 2 reference 10 = grade 1 reference 11 = reserved	
5	Reserved (8'b0)	
6:9	Channel origin data[d] First character is byte 6	
10:13	Channel destination data[d] First character is byte 10	
14:17	32-bit binary sample address code (incrementing every block). Byte 14 is LSB	
18:21	32-bit Time-code. Byte 18 is LSB[e]	
22	Validity Bit 7 — 1 = bytes 18:21 are unreliable Bit 6 — 1 = bytes 14:17 are unreliable Bit 5 — 1 = bytes 6:13 are unreliable Bit 4 — 1 = bytes 0:5 are unreliable Bits 3:0 — Reserved (zeros)	
23	CRC[f] for bytes 0:22	

a. Notation is: number of bits for 24-bit max / number of bits for 20-bit max
b. If the sample rate is exactly 48, 44.1, or 32 KHz, that will be reflected here; if some other sample rate is selected, "other" will be selected. In all cases, the sample rate information will be available via the user data block (though this is left up to software when using media buffer data for user data).
c. Digital audio reference signal per AES11-1990.
d. 7-bit ASCII data with odd parity bit.
e. Format is 00 hours, 00 minutes, 00 seconds, 00 frames, LSB first, where all zeros = midnight
f. Polynomial is $G(x) = X_p^8 + X_p^4 + X_p^3 + X_p^2 + 1$, with an initial state of all 1's.

VRAM Timing Parameters

Memory Interface operation of BtV2115 is controlled by the bit value contained in GRP_CFG0[6:5]. This determines the RAM timing characteristics of the memory interface. The columns in Table 199 below indicate the characteristics of the memory interface relative to the settings of GRP_CFG0[6:5]. The timing of BIOS cycles is determined by the bit value contained in GRP_CFG1[5]. BIOS cycles, OPL cycles, and the timing of the PACDAC interface are independent of the GRP_CFG0[6:5] value. The timings for BIOS, OPL, and PACDAC are replicated across the GRP_CFG0[6:5] columns solely for clarity.

Table 199. BtV2115 Output Specifications for DRAM Signals *(1 of 6)*

Parameter Description	Symbol	11		10		01		00		Notes
		min	max	min	max	min	max	min	max	
		ns		ns		ns		ns		
Cycle time, CAS-Before-RAS Refresh	CBRC	170	176	170	176	170	176	170	176	1
Cycle time, read	RC	120	124	120	124	139	145	160	166	
Cycle time, write	WC	119	125	119	125	139	145	160	166	
Cycle time, load color register	LCRC	170	176	170	176	170	176	170	176	1
Cycle time, block write	BWC	119	125	119	125	139	145	160	166	
Cycle time, memory to register transfer	RTC	119	125	119	125	139	145	160	166	
Cycle time, page mode write	PMWC	37	43	37	43	58	64	58	64	
Cycle time, page mode read	PMRC	37	43	58	64	58	64	58	64	
Cycle time, ROM read	ROMRC	241	247	241	247	241	247	241	247	1,6
Cycle time, FLASH ROM write	ROMWC	241	247	241	247	241	247	241	247	1,6
Cycle time, ROM read	ROMRC	78	84	78	84	78	84	78	84	1,7

Table 199. BtV2115 Output Specifications for DRAM Signals *(2 of 6)*

Parameter Description	Symbol	11		10		01		00		Notes
		min	max	min	max	min	max	min	max	
		ns		ns		ns		ns		
Cycle time, FLASH ROM write	ROMWC	68	74	68	74	68	74	68	74	1,7
Cycle time, OPL read	OPLRC	220	226	220	226	220	226	220	226	1
Cycle time, OPL write	OPLWC	200	206	200	206	200	206	200	206	1
RAS precharge	RPCBR	88	94	88	94	88	94	88	94	1
RAS pulse width	RASCBR	88	94	88	94	88	94	88	94	1
CAS precharge	CPCBR	68	74	68	74	68	74	68	74	1
CAS setup to RAS	CSR	17	23	17	23	17	23	17	23	1
DSF setup to RAS	FSRCBR	17	23	17	23	17	23	17	23	1,2
DSF hold from RAS low	RFHCBR	17	23	17	23	17	23	17	23	1,2
CAS hold from RAS low	CHR	88	94	88	94	88	94	88	94	1
RAS precharge	RP	48	52	48	52	58	62	68	72	
RAS pulse width	RAS	70	74	70	74	80	84	90	94	
CAS precharge	CP	18	22	18	22	28	32	28	32	
CAS pulse width	CAS	39	43	39	43	49	53	59	63	
RAS low to CAS low delay	RCD	27	33	27	33	27	33	27	33	1
Address setup to RAS	ASR	8	12	8	12	8	12	8	12	1
Row address hold	RAH	18	22	18	22	18	22	18	22	1
Address setup to CAS	ASC	8	12	8	12	8	12	8	12	1
Column address hold	CAH	28	32	28	32	38	42	48	52	
TRG high setup to RAS	TSR	48	54	48	54	58	64	68	74	
TRG high hold from RAS low	RTH	28	34	28	34	28	34	28	34	1
TRG pulse width	TRG	38	44	38	44	48	54	58	64	
Read data setup to end of cycle	RDS	3	-	3	-	5	-	7	-	3
Read data hold from end of cycle	RDH	0	-	0	-	0	-	0	-	3
WE high (read) hold from CAS	RCH	38	44	38	44	38	44	38	44	1
DSF setup to RAS	FSR	48	54	48	54	58	64	68	74	
DSF hold from CAS low	CFH	38	44	38	44	48	54	58	64	
Mask setup to RAS	MS	8	12	8	12	8	12	8	12	1

Table 199. BtV2115 Output Specifications for DRAM Signals *(3 of 6)*

Parameter Description	Symbol	11		10		01		00		Notes
		min	max	min	max	min	max	min	max	
		ns		ns		ns		ns		
Mask hold from RAS low	MH	18	22	18	22	18	22	18	22	1
Data setup to CAS	DS	9	13	9	13	9	13	9	13	1
Data hold from CAS low	DH	38	42	38	42	48	52	58	62	
WE low (write) setup to RAS	WSR	17	23	17	23	17	23	17	23	1
WE low (write) hold from CAS	WCH	38	44	38	44	48	54	58	64	
TRG high (write) setup to RAS	YS	48	54	48	54	58	64	68	74	
TRG high (write) hold from RAS low	YH	68	74	68	74	78	84	88	94	
WE high hold from RAS low	RWHLCR	18	22	18	22	18	22	18	22	1,4
WE low setup to CAS	WSCLCR	8	12	8	12	8	12	8	12	1
DSF hold from RAS low	FHR	48	54	48	54	58	64	68	74	1
WE hold from RAS low	RWH	18	22	18	22	18	22	18	22	1
WE low setup to CAS	WSC	8	12	8	12	8	12	8	12	1
DSF hold from RAS low	RFH	17	23	17	23	17	23	17	23	1
DSF setup to CAS	FSC	8	12	8	12	8	12	8	12	1
CAS precharge, page mode	CPPM	18	22	18	22	28	32	28	32	
CAS pulse width, page mode read	CASRPM	19	23	29	33	29	33	29	33	
CAS pulse width, page mode write	CASWPM	19	23	19	23	29	33	29	33	
Address setup to CAS, page mode	ASCPM	8	12	8	12	8	12	8	12	1
Address hold from CAS low, page mode	CAHPM	18	22	18	22	18	22	18	22	1
WE setup to CAS, page mode	WSCPM	8	12	8	12	8	12	8	12	1
WE hold from CAS low, page mode	WCHPM	8	12	8	12	8	12	8	12	1
TRG setup to CAS, page mode	TSCPM	8	12	8	12	8	12	8	12	1
TRG hold from CAS low, page mode	TCHPM	17	23	17	23	17	23	17	23	1
Data setup to CAS, page mode	DSPM	9	13	9	13	9	13	9	13	1

Table 199. BtV2115 Output Specifications for DRAM Signals *(4 of 6)*

Parameter Description	Symbol	11		10		01		00		Notes
		min	max	min	max	min	max	min	max	
		ns		ns		ns		ns		
Date hold from CAS low, page mode	DHPM	18	22	18	22	18	22	18	22	1
DSF setup to CAS, page mode	FSCPM	8	12	8	12	8	12	8	12	1
DSF hold from CAS low, page mode	CFHPM	17	23	17	23	17	23	17	23	1
Refresh cycle	REF	-	-	-	-	-	-	-	-	1,5
MDATA setup to BIOS_CS	MSCROM	38	44	38	44	38	44	38	44	1,6
BIOS_CS setup to WE	CSWROM	17	23	17	23	17	23	17	23	1,6
BIOS_CS hold from WE	CHWROM	17	23	17	23	17	23	17	23	1,6
MDATA hold from BIOS_CS	MHCROM	28	34	28	34	28	34	28	34	1,6
BIOS_CS pulse width	CSROM	180	186	180	186	180	186	180	186	1,6
WE pulse width	WEROM	139	145	139	145	139	145	139	145	1,6
BIOS_CS setup to TRG	CSTROM	18	22	18	22	18	22	18	22	1,6
BIOS_CS hold from TRG	CHTROM	18	22	18	22	18	22	18	22	1,6
TRG pulse width	TRGROM	139	145	139	145	139	145	139	145	1,6
BIOS_CS precharge	CSPROM	109	115	109	115	109	115	109	115	1,6
BIOS_CS setup to WE	CSWROM	7	13	7	13	7	13	7	13	1,7
BIOS_CS hold from WE	CHWROM	17	23	17	23	17	23	17	23	1,7
BIOS_CS pulse width	CSROM	68	74	68	74	68	74	68	74	1,7
WE pulse width	WEROM	38	44	38	44	38	44	38	44	1,7
BIOS_CS setup to TRG	CSTROM	7	13	7	13	7	13	7	13	1,7
BIOS_CS hold from TRG	CHTROM	17	23	17	23	17	23	17	23	1,7
MDATA setup to TRG	DSROM	20	-	20	-	20	-	20	-	1,7
MDATA hold from TRG	DHROM	0	-	0	-	0	-	0	-	1,7
TRG pulse width	TRGROM	68	74	68	74	68	74	68	74	1,7
BIOS_CS precharge	CSPROM	58	64	58	64	58	64	58	64	1,7
MDATA setup to TRG	MSTOPL	7	13	7	13	7	13	7	13	1

Table 199. BtV2115 Output Specifications for DRAM Signals *(5 of 6)*

Parameter Description	Symbol	11		10		01		00		Notes
		min	max	min	max	min	max	min	max	
		ns		ns		ns		ns		
TRG low setup to OPL_CS	TSCOPL	-3	3	-3	3	-3	3	-3	3	1
TRG hold from OPL_CS	THCOPL	-3	3	-3	3	-3	3	-3	3	1
MDATA hold from TRG	MHTOPL	17	23	17	23	17	23	17	23	1
OPL_CS pulse width	CSOPL	160	166	160	166	160	166	160	166	1
TRG pulse width	TRGOPL	160	166	160	166	160	166	160	166	1
Data setup to OPL_CS	DSOPL	10	-	10	-	10	-	10	-	
Data hold from OPL_CS	DHOPL	2	-	2	-	2	-	2	-	
OPL_CS precharge	CSPOPL	282	288	282	288	282	288	282	288	1
MDATA setup to WE	MSWOPL	7	13	7	13	7	13	7	13	1
WE low setup to OPL_CS	WSCOPL	7	13	7	13	7	13	7	13	1
WE hold from OPL_CS	WHCOPL	-3	3	-3	3	-3	3	-3	3	1
MDATA hold from WE	MHWOPL	7	13	7	13	7	13	7	13	1
WE pulse width	WEOPL	170	176	170	176	170	176	170	176	1
SCLK cycle time	SC	13.3	-	13.3	-	13.3	-	13.3	-	1
SCLK precharge	SCH	6	-	6	-	6	-	6	-	1
SCLK pulse width	SCL	6	-	6	-	6	-	6	-	1
LCLK cycle time	LC	13.3	-	13.3	-	13.3	-	13.3	-	1
LCLK precharge	LCH	6	-	6	-	6	-	6	-	1
LCLK pulse width	LCL	6	-	6	-	6	-	6	-	1

Table 199. BtV2115 Output Specifications for DRAM Signals *(6 of 6)*

Parameter Description	Symbol	11		10		01		00		Notes
		min	max	min	max	min	max	min	max	
		ns		ns		ns		ns		
LCLK0 to LCLK1 skew	LLSK	1	8	1	8	1	8	1	8	1
PT setup to LCLK0	PSL	1	-	1	-	1	-	1	-	1
PT hold from LCLK0 high	PLH	3	-	3	-	3	-	3	-	1
SOE setup to first LCLK0 rising edge	SOSL	40	-	40	-	40	-	40	-	1,8
SOE hold from last LCLK0 rising edge	SOLH	40	-	40	-	40	-	40	-	1,8

NOTES:
1. Values are independent of GRP_CFG0[6:5] setting.
2. The state of DSF during CAS-Before-RAS cycles is programmable via GRP_CFG0[0], see Table 26 on page 54
3. End of cycle is determined by the rising edge of either CAS or RAS, which ever occurs first.
4. Applies only to LCR (Load Color Register) cycles.
5. The refresh rate is programmable by writing GRP_CFG2 (Configuration Register 2, Refresh Rate Counter), see Table 24 on page 53.
6. These timing are valid for GRP_CFG1[5] = 0, 120 nsec ROM cycles.
7. These timing are valid for GRP_CFG1[5] = 1, 70nsec ROM cycles.
8. SOE always switches before the 1st dword of the next packet group is sent. “First LCLKx rising edge” refers to this 1st dword. “Last LCLKx rising edge” refers to the last packet sent prior to subsequent SOE change for the next packet group.

BtV2115 Subsystem VRAM Load Limits

BtV2115 is designed to drive up to 8 VRAM modules, with considerations for additional ROM and a Yamaha FM Synthesizer. Table 200 shows the load limits for the modules.

Table 200. BtV2115 Load Limits

Signal	Capacitive, pf	
	MIN	MAX
MDATA[7:0]		85
MDATA[10:8]		85
MDATA[27:11]		85
MDATA[31:28]		85
MADDR[9:0]		60
$\overline{\text{RAS}}$[3:0]		40
$\overline{\text{CAS}}$[3:0]		40
$\overline{\text{WE}}$0		50
$\overline{\text{WE}}$1		50
TRG		85
DSF		70
$\overline{\text{SOE}}$[3:0]		40
LDCLK0		50
LDCLK1		50
PT[3:0]		25

Absolute Maximum Ratings

Table 201. Absolute Maximum Ratings

Parameter	Symbol	Min	Typ	Max	Units
VAA, VDD (measured to GND)				7.0	V
Voltage on Any Signal Pin		GND – 0.5		VDD + 0.5	V
Ambient Operating Temperature	T_A	–55		+125	˚C
Storage Temperature	T_S	–65		+150	˚C
Junction Temperature	T_J			+150	˚C
Vapor Phase Soldering (1 minute)	T_{VSOL}			220	˚C
Stresses above those listed in this table may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those listed in the operational sections of this specification are not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. This device employs high-impedance CMOS devices on all signal pins. It should be handled as an ESD-sensitive device. Voltage values on any signal pin that extend beyond the power supply rails by more than the amount(s) specified above can cause destructive latchup.					

Recommended Operating Conditions

Table 202. Recommended Operating Conditions

Parameter	Min	Typ	Max	Units
Load Capacitance on digital outputs:				pF
$\overline{\text{TRG}}$		80	85	pF
DSF		60	70	pF
MADDR		45	60	pF
MDATA		40	85	pF
$\overline{\text{CAS}}$[3:0]		35	40	pF
$\overline{\text{WE}}$[1:0]		35	50	pF
$\overline{\text{RAS}}$[3:0]		35	40	pF
$\overline{\text{SOE}}$[3:0]		35	40	pF
$\overline{\text{BIOS\_CS}}$		20		pF
$\overline{\text{OPL\_CS}}$		20		pF
PT[3:0]		20	25	pF
$\overline{\text{RESET\_OUT}}$		80		pF
Load Capacitance on IO pins VL_DATA[31:0], list of IO pins			tbd	pF
Load Capacitance on clock outputs: LCKL[1:0], AUDCLK_OUT		60	tbd	pF
Supply Voltage	4.75	5.00	5.25	V
Junction Temperature T_j	0		125	˚C

Target DC Characteristics

Target DC characteristics are shown in Table 203, II2 characteristics in Table 204, and DDC characteristics in Table 205. Power dissipation figures are shown in Figure 59 and Figure 60.

Table 203. Target DC Characteristics

Parameter	Symbol	Min	Typ	Max	Units
Digital Inputs					
Input High Voltage	V_{IH}	2.0		V_{DD} + 0.5	V
Input Low Voltage	V_{IL}	GND–0.5		0.8	V
Input High Current (V_{in} = 2.4 V)	I_{IH}			10	µA
Input Low Current (V_{in} = 0.4 V)	I_{IL}	–10			µA
Input Capacitance (v = 1 MHz, Vin = 2.4 V)	C_{IN}			7	pF
Hysteresis if applicable			0.3		V
Digital Outputs					
Output High Voltage (I_{OH} = –400 µA)	V_{OH}	2.4			V
Output Low Voltage (I_{OL} = 3.2 mA)	V_{OL}			0.4	V
Three-State Current (0–2.4V)	I_{OZ}			50	µA

Table 204. DC Characteristics for IIC_SDA and IIC_SCL I/O (Fast Mode)

Parameter	Symbol	Min.	Max.	Units
Low level input voltage	V_{IL}	–0.5	1.5	V
High level input voltage	V_{IH}	3.0	V_{DD} + 0.5	V
Hysteresis	V_{hys}	0.2		V
Spike suppression	t_{SP}	0	50	ns
Low level output voltage (open drain) at 3 mA sink current at 6 mA sink current	 V_{OL1} V_{OL2}		 0.4 0.6	 V V
Output fall time (10–400pF bus capacitance)	t_{OF}	$20 + 0.1C_b$[1]	250[2]	ns
Input current (Vi = 0.4–0.9$V_{DD\ max.}$)	I_i	–10	10	µA
Capacitance	C_i		10	pF

1. C_b = capacitance of one bus line in pF.
2. SDA and SCL lines will not be obstructed if V_{DD} is switched off.

Table 205. DC Characteristics for DDC_SDA and DDC_SCL I/O (Standard Mode)

Parameter	Symbol	Min.	Max.	Units
Low level input voltage	V_{IL}	–0.5	1.5	V
High level input voltage	V_{IH}	3.0	V_{DD} + 0.5	V
Hysteresis	V_{hys}	0.2		V
Spike suppression	t_{SP}	0	50	ns
Low level output voltage (open drain) at 3 mA sink current at 6 mA sink current	 V_{OL1} V_{OL2}		 0.4 0.6	 V V
Output fall time (10–400pF bus capacitance)	t_{OF}	$20 + 0.1C_b$[1]	250[2]	ns
Input current (Vi = 0.4–0.9$V_{DD\ max.}$)	I_i	–10	10	µA
Capacitance	C_i		10	pF

1. C_b = capacitance of one bus line in pF.
2. SDA and SCL lines will not be obstructed if V_{DD} is switched off.

Target AC Characteristics

Table 206 and Table 207 provide the target AC characteristics for the I2C bus.

Table 206. AC Characteristics for IIC_SDA and IIC_SCL Bus Lines (Fast Mode)

Parameter	Symbol	Min.	Max	Units
SCL clock frequency		0	400	kHz
Bus free time between a STOP and START condition		1.3		µs
Hold time (repeated) START condition. After this period, the first clock pulse is generated		0.6		µs
Low period of the SCL clock		1.3		µs
High period of the SCL clock		0.6		µs
Set-up time for a repeated START condition		0.6		µs
Data hold time		0	0.9	µs
Data set-up time		100		µs
Rise time of both SDA and SCL signals		$20 + 0.1C_b$[1]	300	ns
Fall time of both SDA and SCL signals		$20 + 0.1C_b$[1]	300	ns
Set-up time for STOP condition		0.6		µs
Capacitive load for each bus line			400	pF
1. SDA and SCL lines will not be obstructed if V_{DD} is switched off.				

Table 207. AC Characteristics for DDC_SDA and DDC_SCL Bus Lines (Standard Mode) *(1 of 2)*

Parameter	Symbol	Min.	Max	Units
SCL clock frequency		0	400	kHz
Bus free time between a STOP and START condition		1.3		µs
Hold time (repeated) START condition. After this period, the first clock pulse is generated		0.6		µs
Low period of the SCL clock		1.3		µs
High period of the SCL clock		0.6		µs
Set-up time for a repeated START condition		0.6		µs
Data hold time		0	0.9	µs

Table 207. AC Characteristics for DDC_SDA and DDC_SCL Bus Lines (Standard Mode) *(2 of 2)*

Parameter	Symbol	Min.	Max	Units
Data set-up time		100		µs
Rise time of both SDA and SCL signals		$20 + 0.1C_b$[1]	300	ns
Fall time of both SDA and SCL signals		$20 + 0.1C_b$[1]	300	ns
Set-up time for STOP condition		0.6		µs
Capacitive load for each bus line			400	pF
1. SDA and SCL lines will not be obstructed if V_{DD} is switched off.				

Power Dissipation

Introduction

A system designer should consider the cost/reliability trade-offs when determining package selection and cooling requirements. As a rule, the device junction temperature should not exceed 125˚C to prevent compromising device reliability. The information in this section provides the data needed to determine the best trade-offs.

Power Dissipation

The following factors affect device thermal performance: air flow, ambient temperature, board thermal characteristics (including copper area and thickness), package thermal characteristics, device power dissipation, power dissipation of surrounding devices, and board power dissipation. These trade-off considerations assume that the surrounding devices do not contribute nor block heat dissipation and do not block airflow. It is also assumed that the circuit board contains a set amount of copper (combined weight, all layers).

Thermal Resistance

The ability of a package to dissipate power is characterized by the overall thermal resistance of the package. The thermal resistance of the available packages is determined based on experimental data. Device junction to ambient thermal resistance is commonly denoted by the symbol θ_{ja}. Packages having decreased thermal resistance are generally more expensive due to the inclusion of heat slugs, heat spreaders, more expensive lead frames, etc. The thermal resistance of a given package can also be reduced by providing airflow over the package.

Table 208 shows the thermal resistance for the die junction packages shown in Figure 59 and Figure 60.

Table 208. Package Thermal Resistance

Package	Airflow (Linear Feet per Minute)					Units
	0	50	100	200	400	
208-pin Power QUAD	17	15	13	12	10	˚C/W
208-pin MQFP	21	19	17	16	14	˚C/W

Package Power Dissipation

The thermal resistance of a package as a function of power dissipation and junction and ambient temperatures is computed using the following expression:

$$\theta_{ja} = \frac{T_j - T_a}{P}$$

where: T_j = junction temperature
T_a = ambient temperature
θ_{ja} = package thermal resistance
P = resulting package power dissipation

The BtV2115's worst case power dissipation (ICC, max, @ VCC, max) is 3.3 W.

Figure 59 charts the worst case die junction temperature versus air flow for the power quad. Figure 60 charts the worst case die junction temperature versus air flow for the heat spreader 208MQFP.

Figure 59. Worst Case Power Dissipation for Power Quad

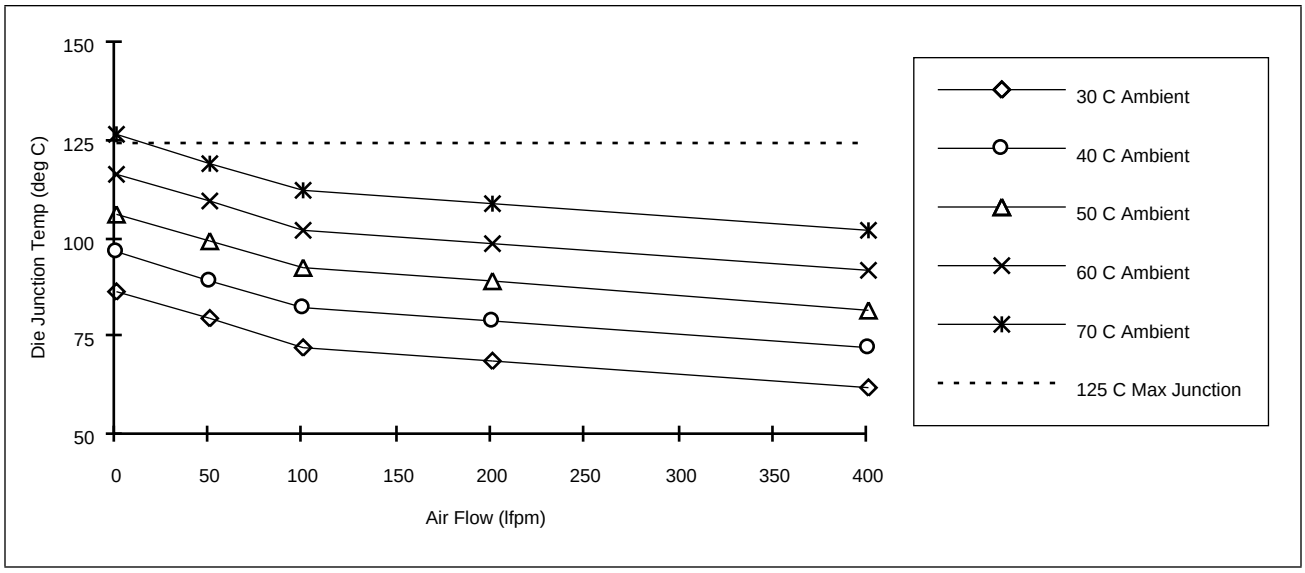
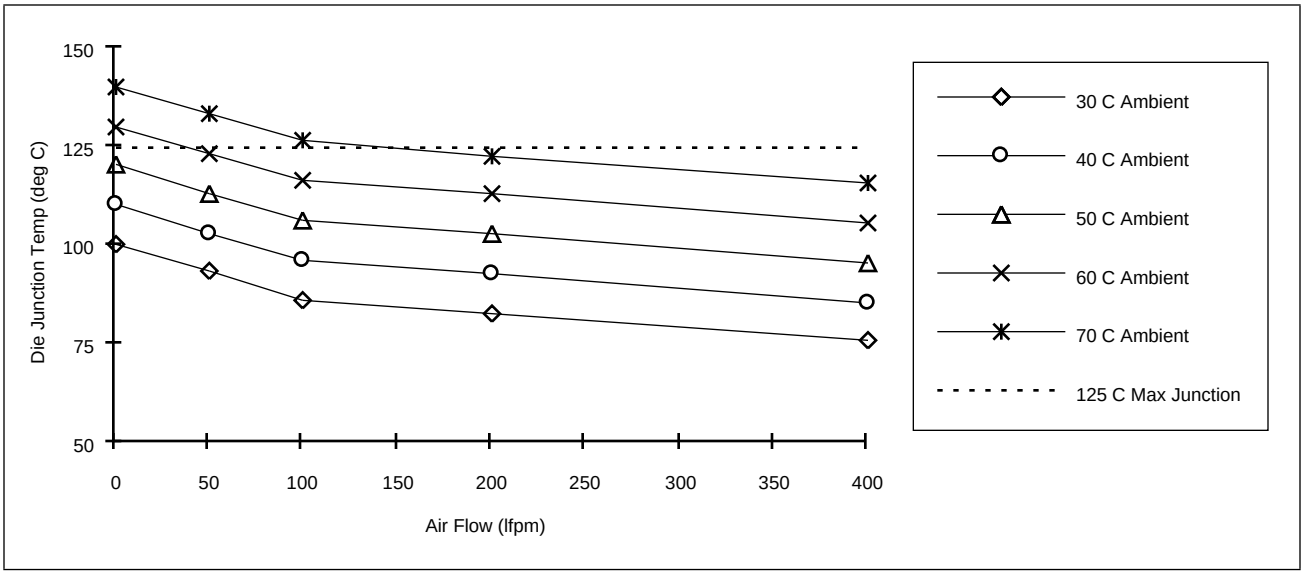


Figure 60. Worst Case Power Dissipation for Heat Spreader MQFP



Recommendations

By using the worst case power dissipation figures for a desired mode of operation and environment, one can deduce the required package thermal resistance. The system designer may then choose the appropriate package.

Suppose that the system designer wants to use the heat spreader package in an environment where no air flow can be guaranteed. With a power dissipation of 3.3 W and a maximum recommended operating junction temperature of 125°C, a maximum ambient temperature can be calculated. Rearranging the thermal resistance equation yields:

$$T_a = T_j - \theta_{ja}P$$

So:

$$T_a = 125 - 21(3.3) \text{ or } 55.7^{o}C$$

Timing Diagrams

This section contains VRAM timing diagrams. For related timing information, refer to “VL Bus Timing” on page 213.

Figure 61. VRAM Read Cycle

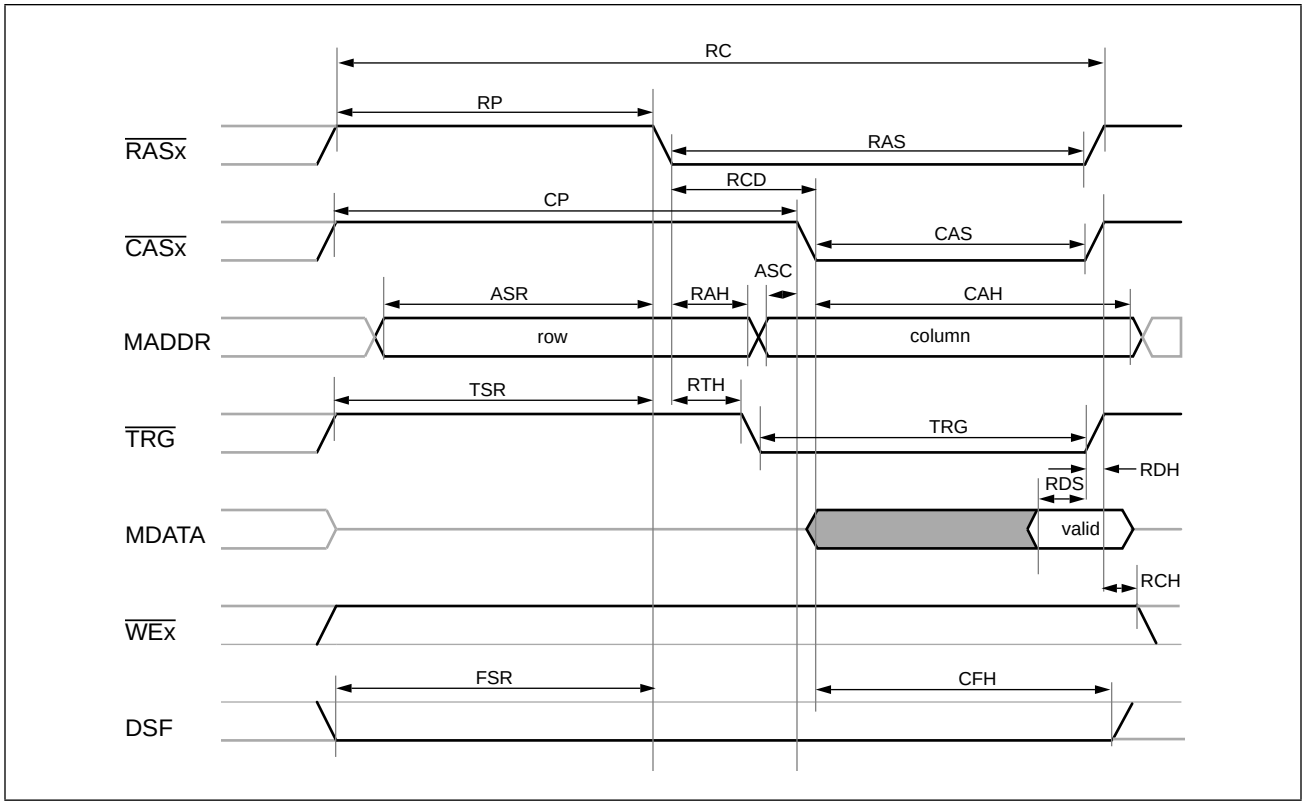


Figure 62. VRAM Masked Write Cycle

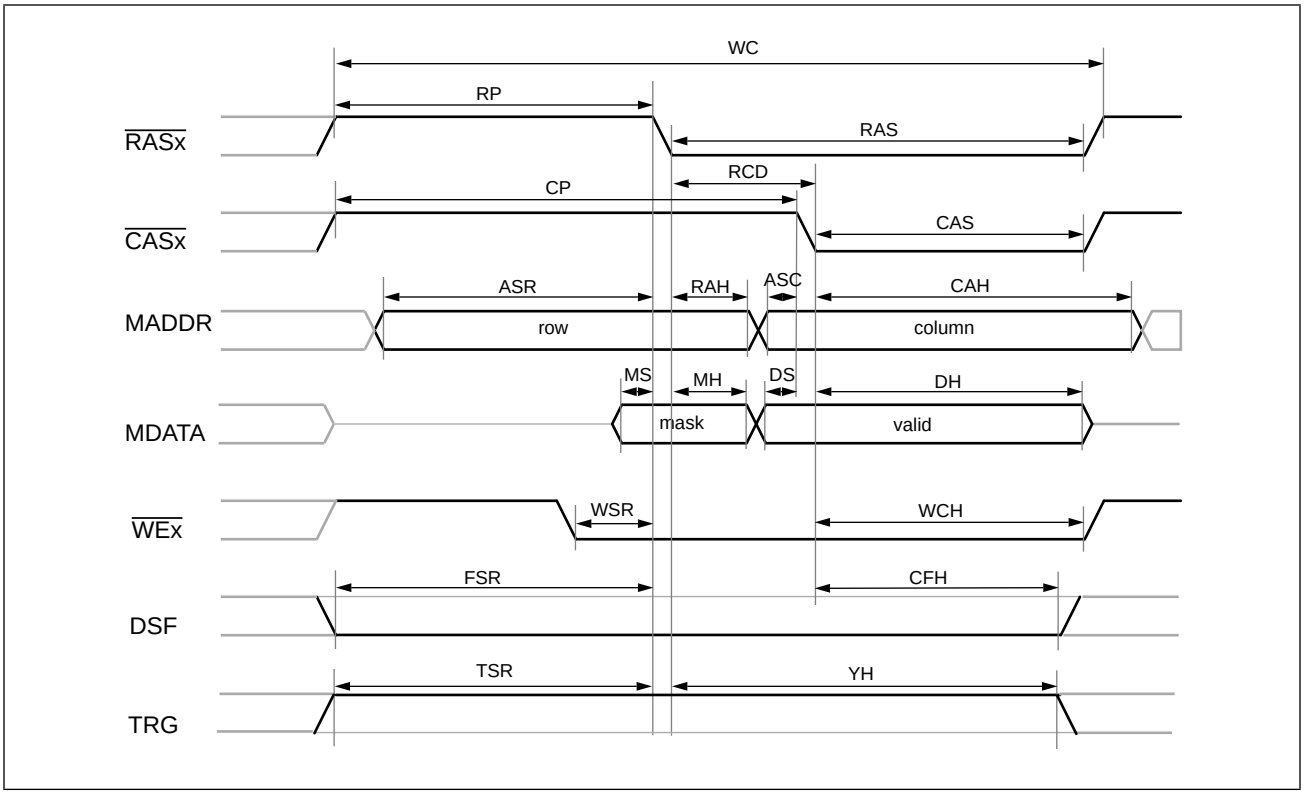


Figure 63. Load Color Register Cycle

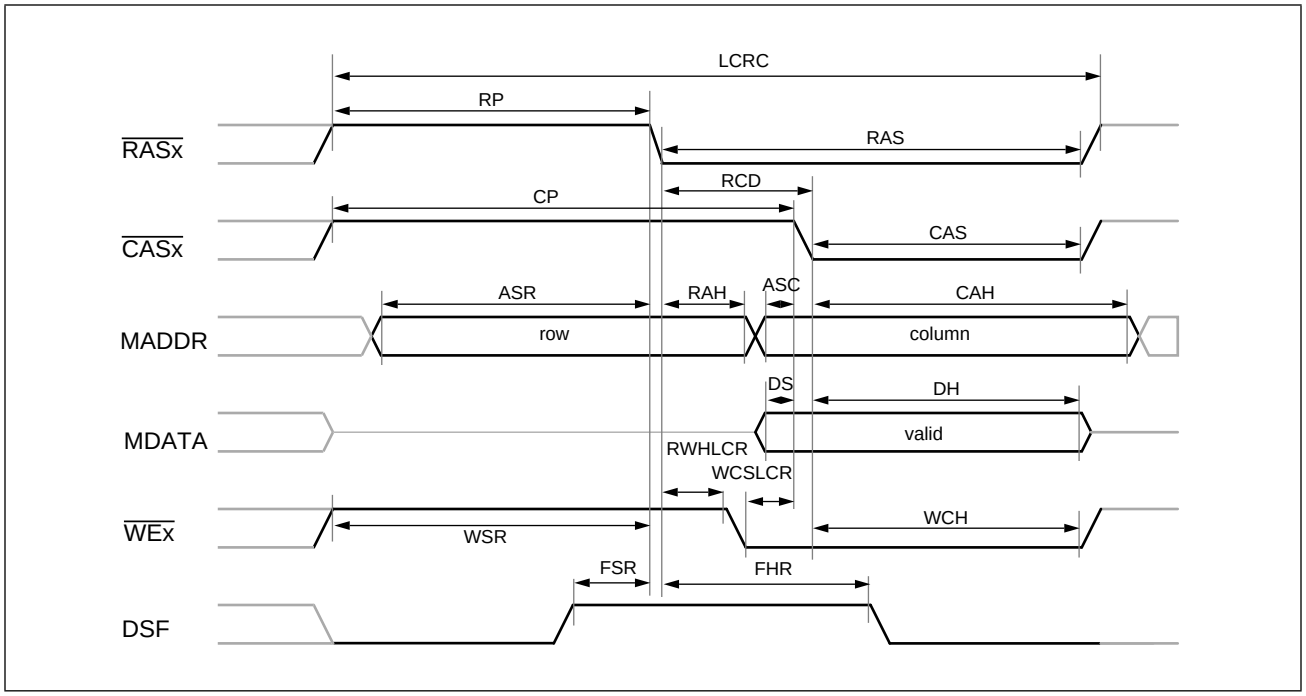


Figure 64. CAS-Before-RAS Refresh Cycles

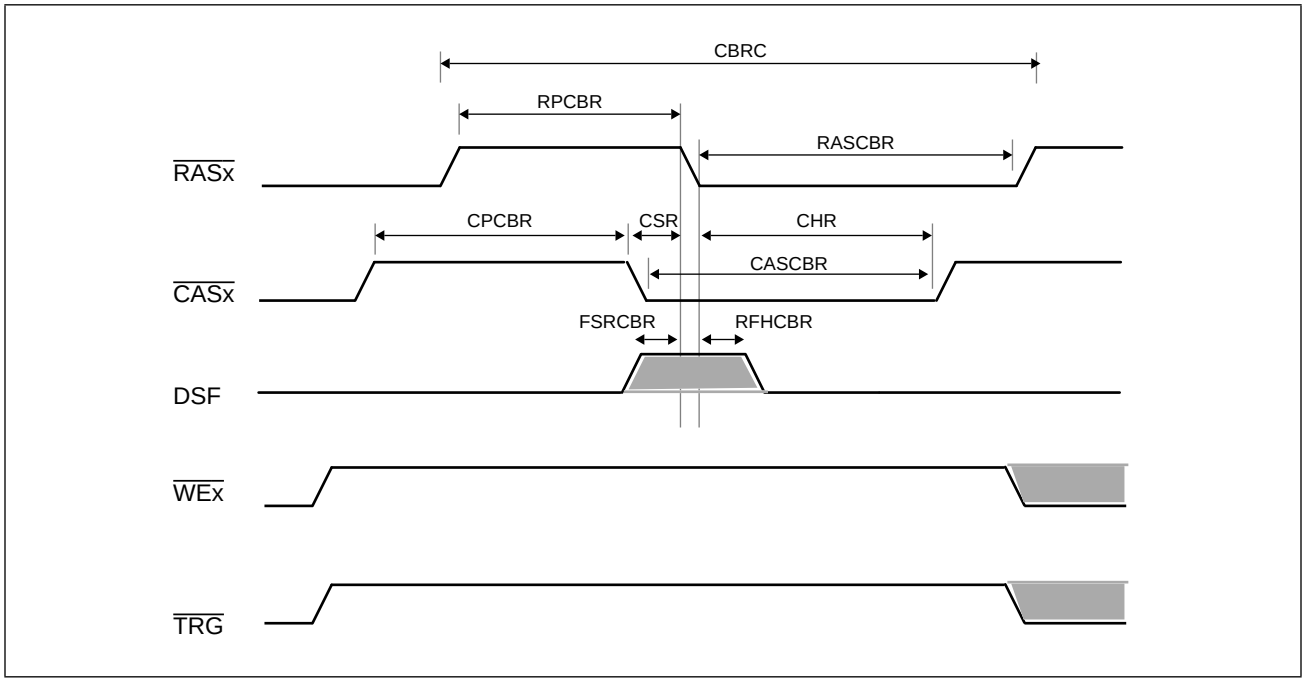


Figure 65. VRAM Non-Masked Block Write Cycle

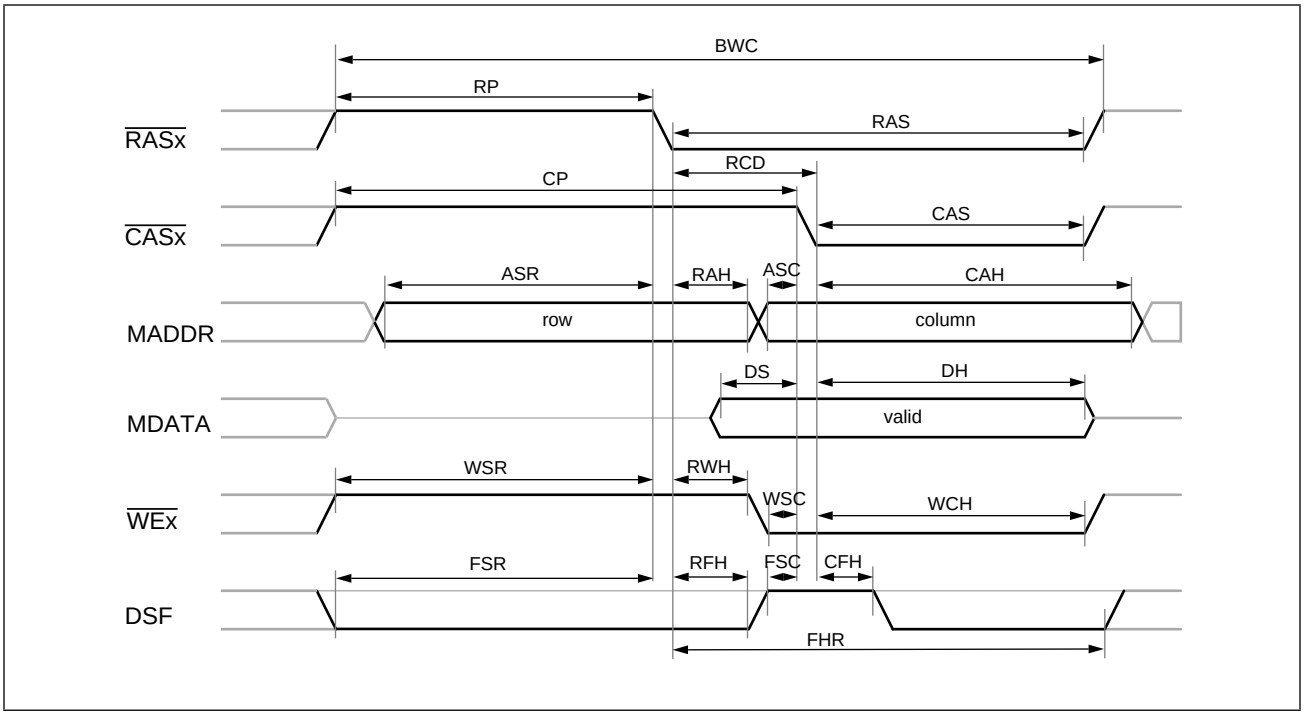


Figure 66. VRAM Memory-to-Shift Register Transfer Cycle

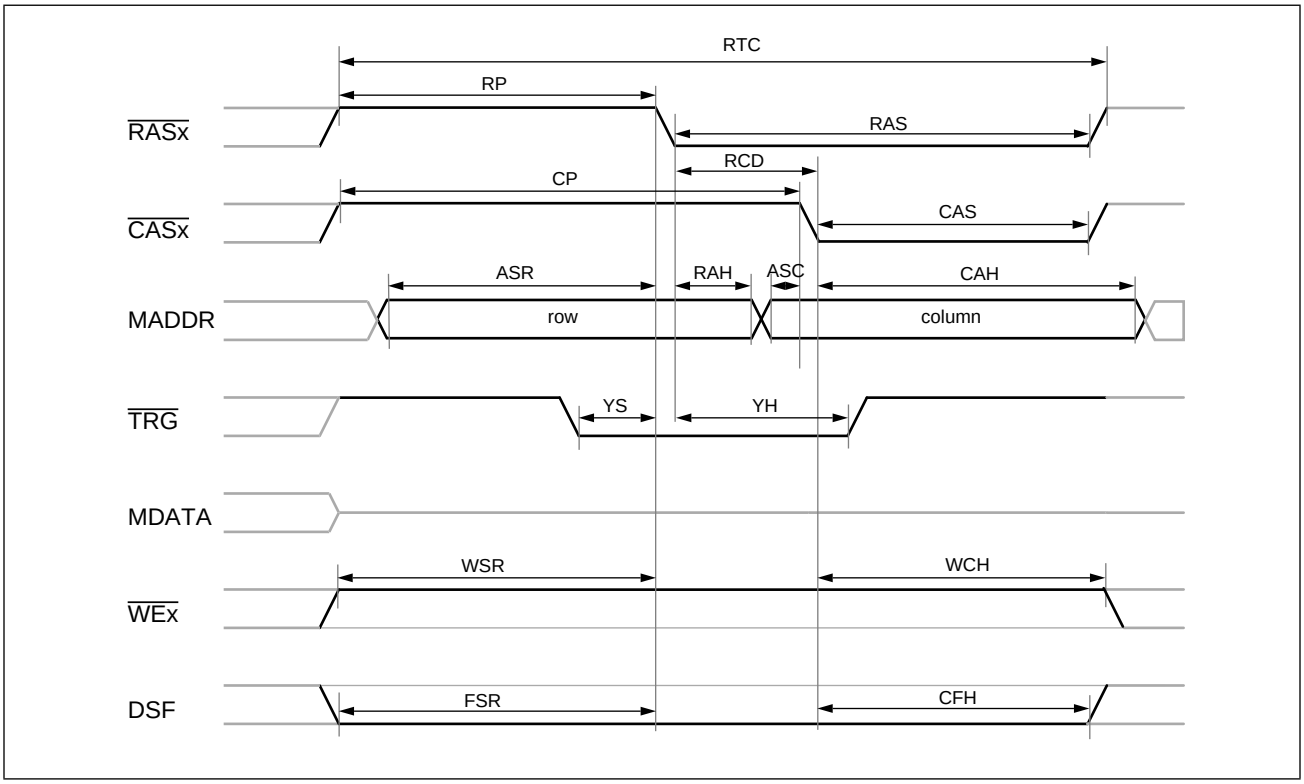


Figure 67. Page Mode Cycles

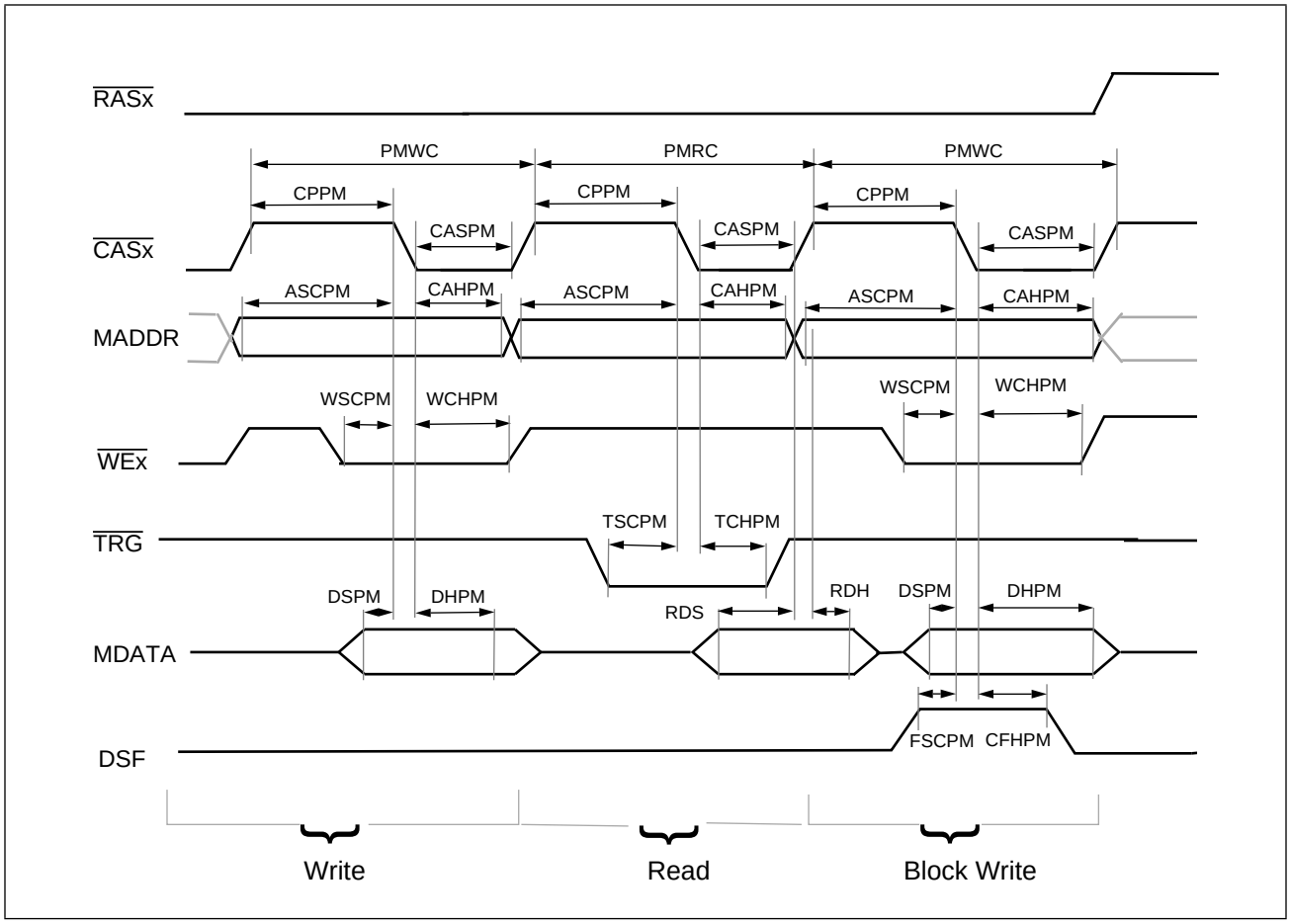


Figure 68. ROM Read Cycle

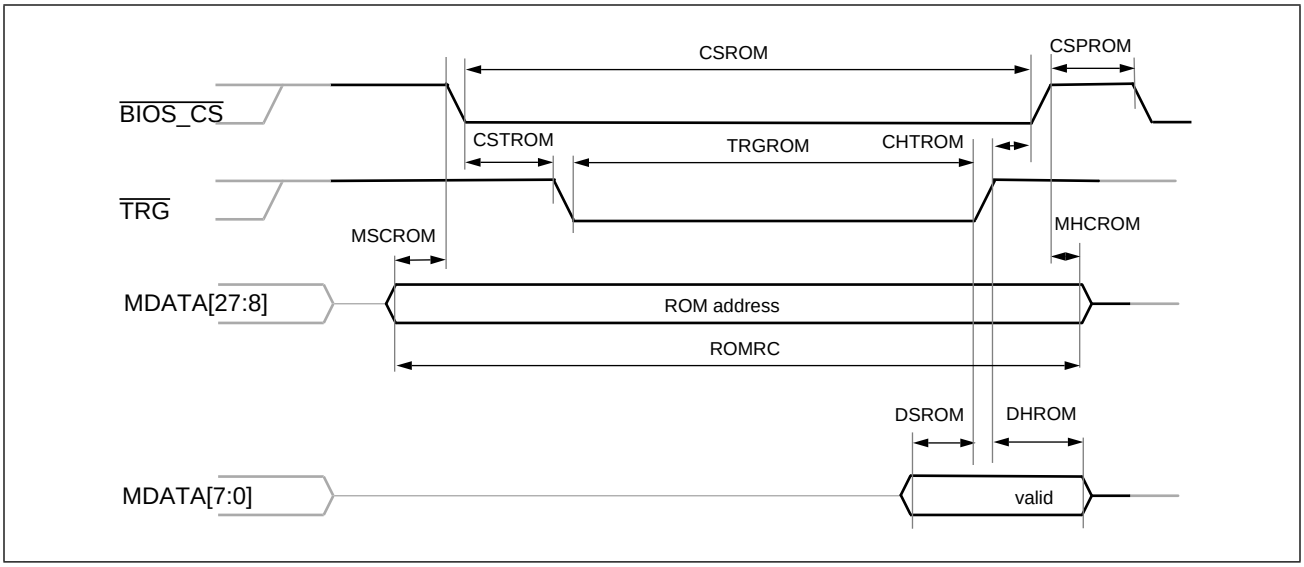


Figure 69. ROM Write Cycle, FLASH ROM

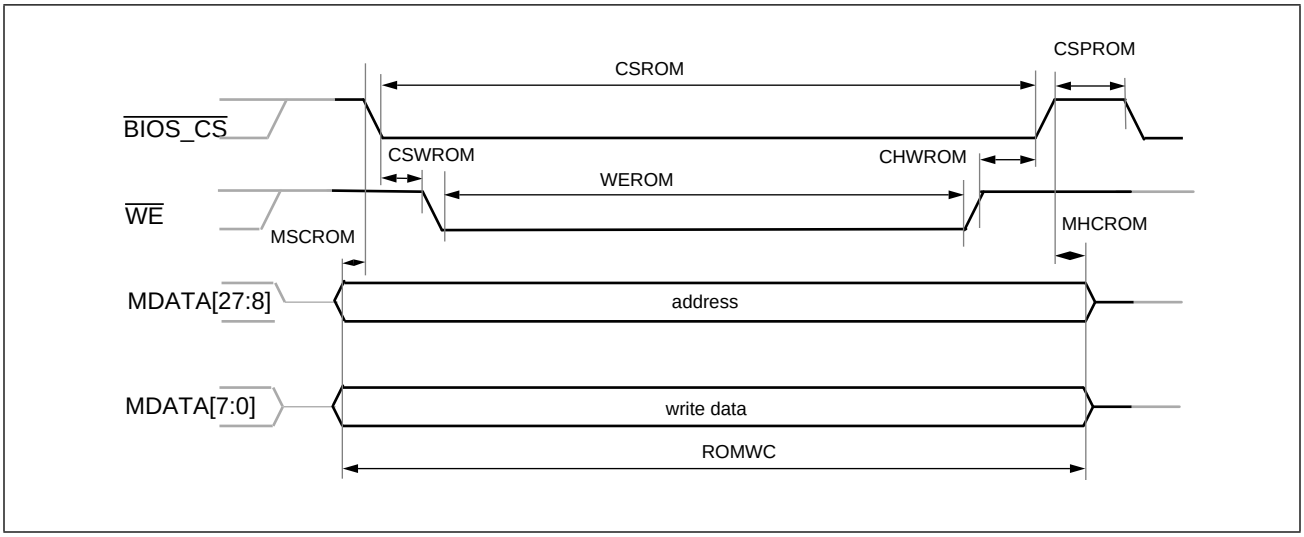


Figure 70. OPL Read Cycle

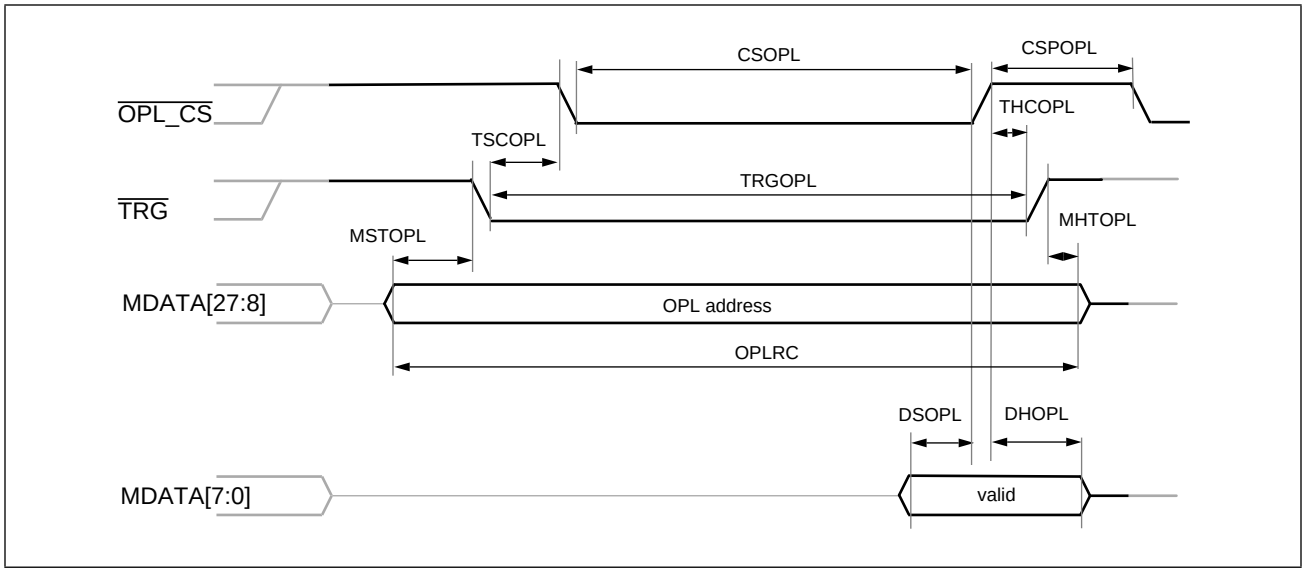


Figure 71. OPL Write Cycle

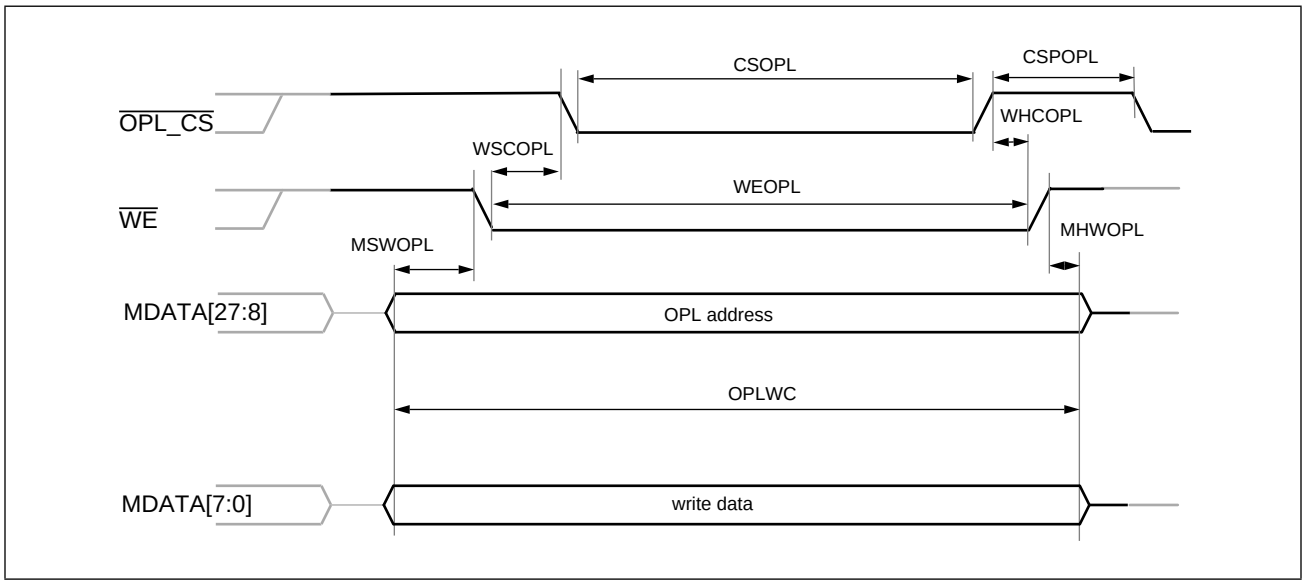
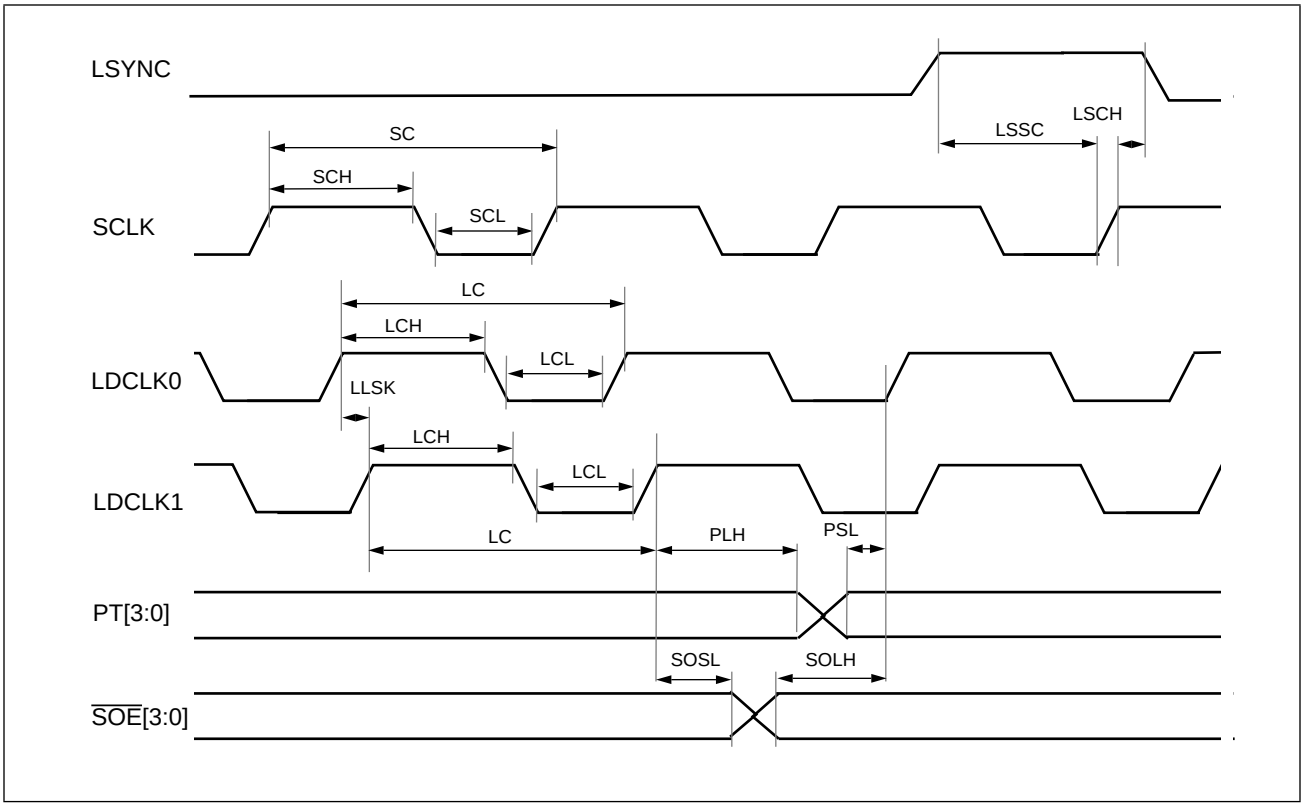


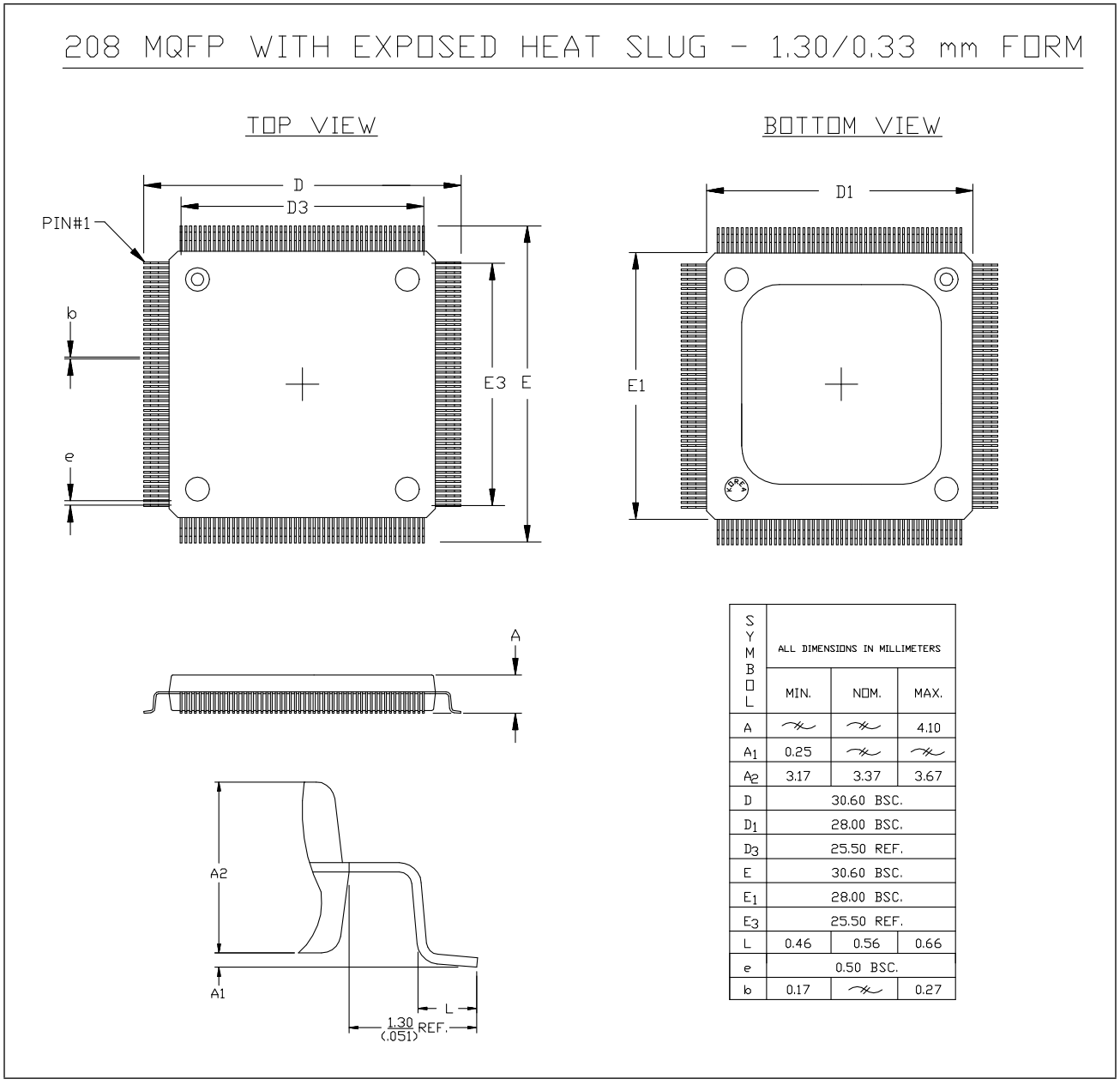
Figure 72. PACDAC Controller Interface



Packaging Specifications

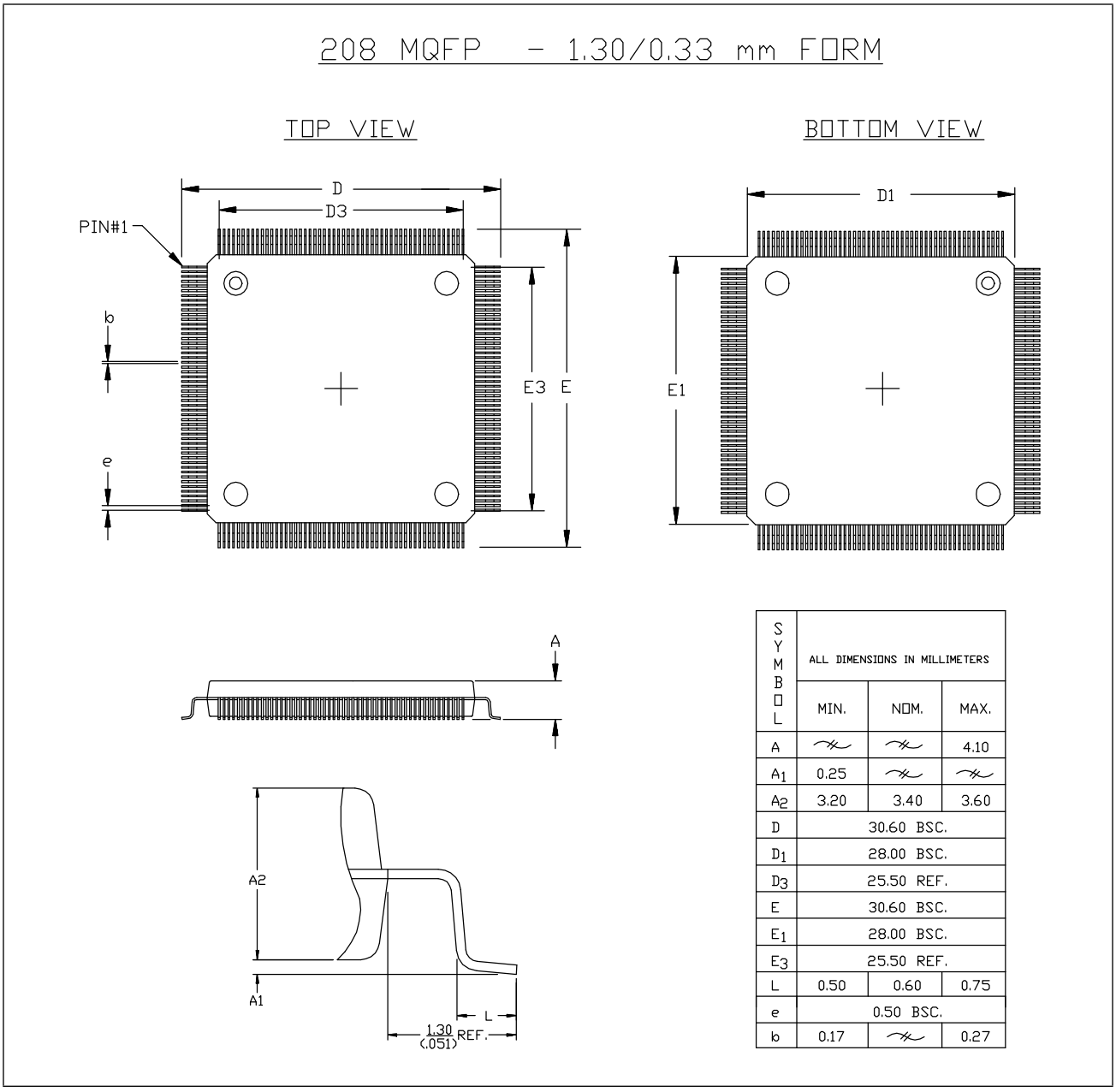
Figure 73 shows model BtV2115KSF: power quad package with exposed heat slug. Figure 74 shows model BtV2115AHF: plastic QFP with an internal heat spreader.

Figure 73. BtV2115KSF Packaging Diagram



SYMBOL	ALL DIMENSIONS IN MILLIMETERS		
	MIN.	NOM.	MAX.
A			4.10
A_1	0.25		
A_2	3.17	3.37	3.67
D		30.60 BSC.	
D_1		28.00 BSC.	
D_3		25.50 REF.	
E		30.60 BSC.	
E_1		28.00 BSC.	
E_3		25.50 REF.	
L	0.46	0.56	0.66
e		0.50 BSC.	
b	0.17		0.27

Figure 74. BtV2115AHF Packaging Diagram

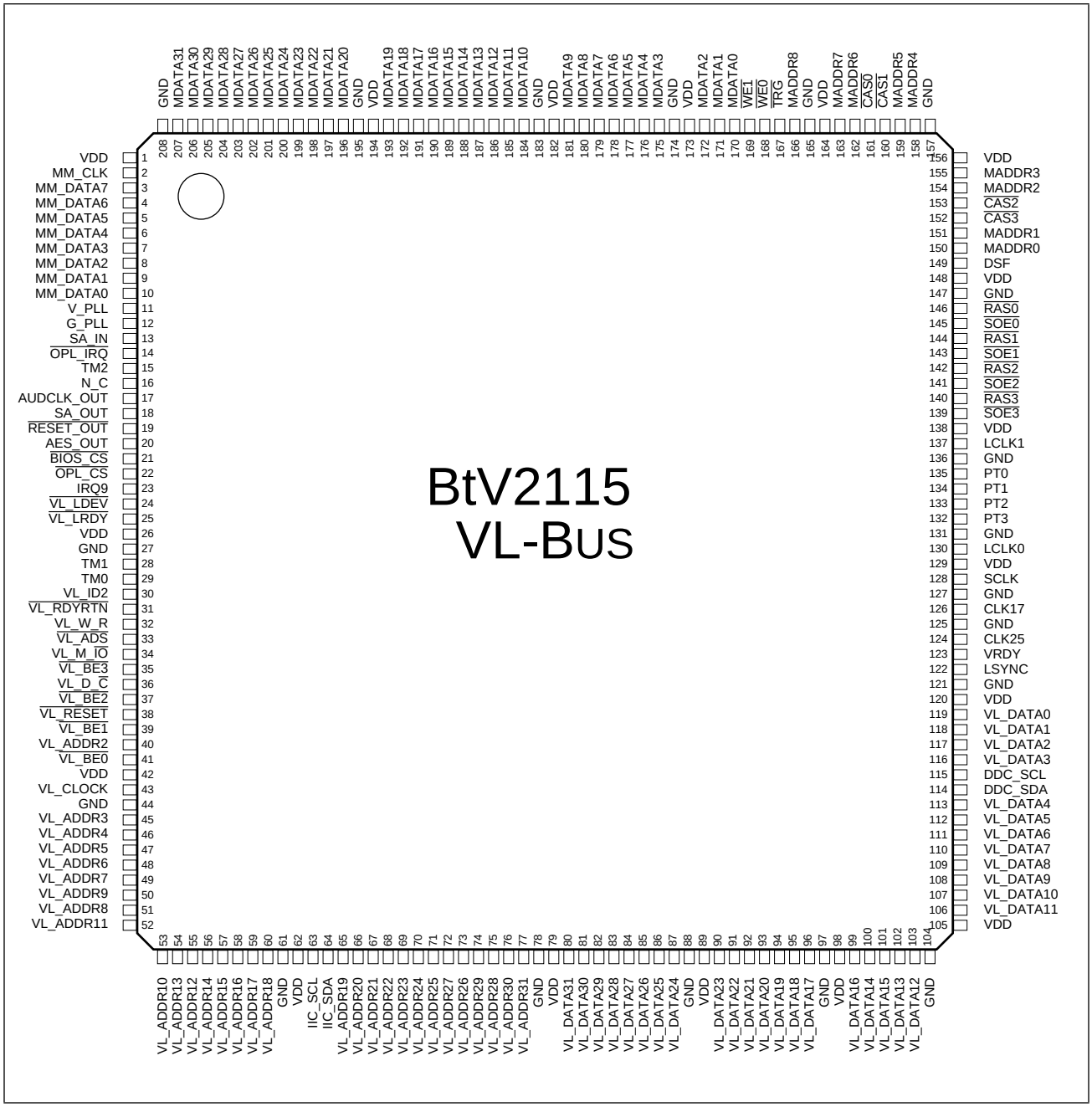


SYMBOL	ALL DIMENSIONS IN MILLIMETERS		
	MIN.	NOM.	MAX.
A			4.10
A_1	0.25		
A_2	3.20	3.40	3.60
D		30.60 BSC.	
D_1		28.00 BSC.	
D_3		25.50 REF.	
E		30.60 BSC.	
E_1		28.00 BSC.	
E_3		25.50 REF.	
L	0.50	0.60	0.75
e		0.50 BSC.	
b	0.17		0.27

BtV2115 VL-Bus Pin Layout

Figure 75 shows the pin assignments for VL-bus interface to the BtV2115. Refer to Table 1 on page 11 for a descriptive summary of each pin.

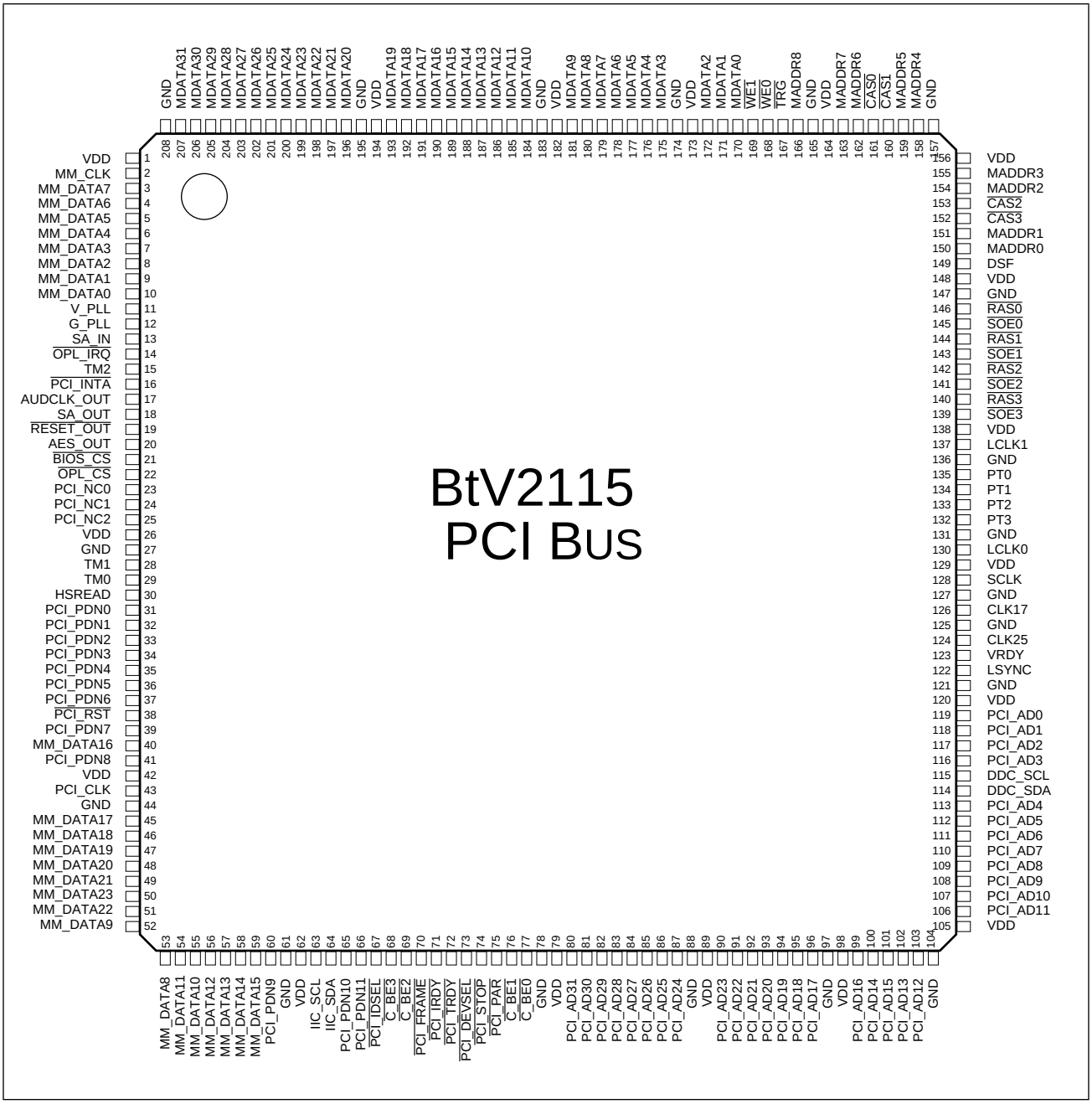
Figure 75. VL-Bus Pin Layout



BtV2115 PCI-Bus Pin Layout

Figure 76 shows the pin assignments for the PCI bus interface to the BtV2115. Refer to Table 2 on page 15 for a descriptive summary of each pin.

Figure 76. PCI-Bus Pin Layout



Revision History

Table 209. BtV2115 Datasheet Revision History

Revision	Date	Change	Description
A	9/20/94	Initial Release	
B	12/16/94		
C	2/3/95	Initial Release	
C	8/30/95	Update	