Core1553BRM Handbook

v2.0

Actel®

Table of Contents

Introduction

Actel Core1553BRM provides a complete MIL-STD-1553B bus controller (BC), remote terminal (RT), or bus monitor terminal (BM or MT). Core1553BRM can be configured to provide all three 1553 functions or any combination thereof. The core is supported in all recent Actel Flash, antifuse, and radiation-tolerant product families. A typical system implementation using Core1553BRM is shown in Figure 1.
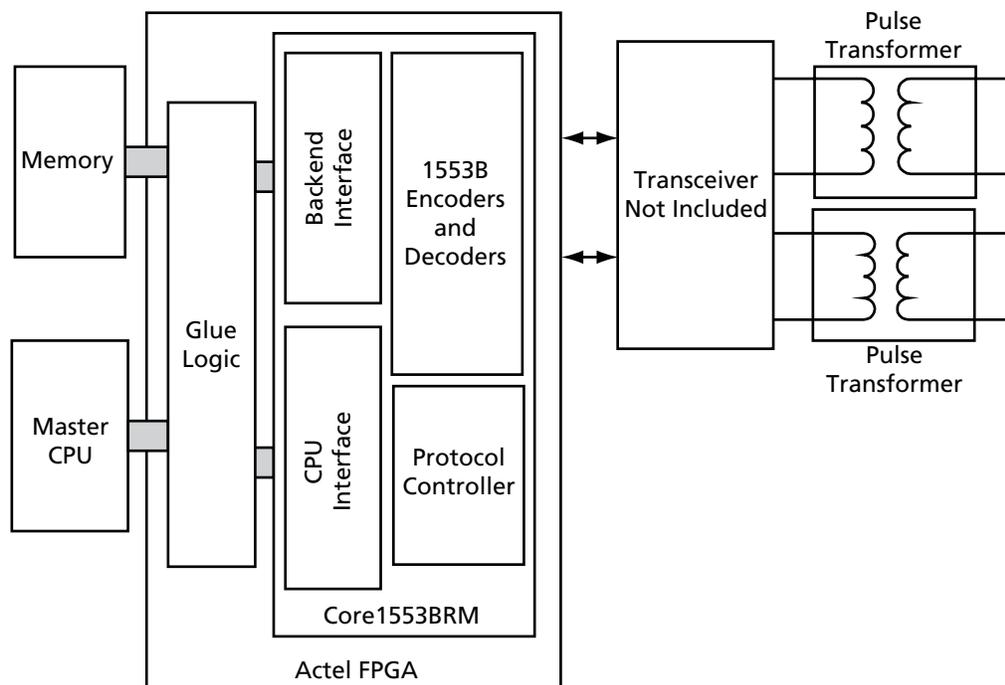
Figure 1 · Typical Core1553BRM Application

A typical Core1553BRM system requires connection to an external CPU, used to set up the core registers and initialize the data tables in memory. To facilitate system integration, Core153BRM is register-compatible with the SuMMIT™ family of 1553B devices from Aeroflex Inc.

The external memory block is used to store the received and transmitted data. This memory can be internal or external to the FPGA, depending upon the family targeted. The core interfaces to the 1553 bus through an external 1553 transceiver and transformer.

Three versions of the core are available:

- An Evaluation version that allows core simulation with Actel Libero® Integrated Design Environment (IDE) or Model*Sim*®
- An Obfuscated version that provides obfuscated RTL and precompiled testbenches
- An RTL version with full access to the source code

Related Documents

MIL–STD–1553 Intellectual Property Products Brochure

Platform8051 Development Kit User's Guide

Core1553BRM Demonstration User Guide

Reference Documents

MIL-STD-1553B, Notices I and II

MIL-HDBK-1553A

Enhanced SuMMIT Family Product Handbook, October 1999, UTMC Microelectronic Systems, Inc.

Version

This handbook applies to Core1553BRM v3.1 and later.

Verification and Compliance

Core1553BRM has been fully verified against the RT Validation Test Plan (MIL-HDBK-1553A, "Verification Tests Carried Out" on page 111). This ensures that the 1553B encoders and decoders are fully compliant with the 1553B specification. Core1553BRM is implemented on the Core1553BRM development system using an APA600 device; this can be purchased from Actel.

Device Requirements

Core1553BRM can be implemented in multiple Actel FPGAs. Table 1 on page 9 through Table 4 on page 10 give typical utilization figures using standard synthesis tools for the complete core. Note that utilization for Fusion and IGLOO™ families is shown in Table 1. The Core column indicates the core configuration as follows:

- B: Bus Controller enabled
- R: Remote Terminal enabled
- M: Bus Monitor enabled
- 0: RT Legalization registers disabled
- 1: RT Legalization registers implemented in logic tiles
- 2: RT Legalization registers implemented using memory
- E: Actel enhanced functions enabled

Table 1 · Device Utilization – ProASIC®3-Based Families

Core	Family	Cells or Tiles			Memory Blocks	Device	Utilization
		Combinatorial	Sequential	Total			
BRM0E		4,470	1,148	5,623	0	A3P600	40.7%
BRM1E		5,047	1,404	6,456	0	A3P600	46.7%
BRM2E		4,511	1,148	5,664	1	A3P600	41.0%
BR0E		3,738	953	4,696	0	A3P600	34.0%
BR1E		4,234	1,209	5,448	0	A3P600	39.4%
BR2E		3,698	953	4,656	1	A3P600	33.7%
RM0E	Fusion	3,555	1,081	4,641	0	A3P600	33.6%
RM1E	IGLOO/e	4,192	1,337	5,534	0	A3P600	40.0%
RM2E	ProASIC®3/E	3,663	1,081	4,749	1	A3P600	34.4%
BME		3,091	979	4,075	0	A3P600	29.5%
BE		2,271	766	3,042	0	A3P600	22.0%
R0		2,612	862	3,474	0	A3P600	25.1%
R1		3,225	1,118	4,343	0	A3P600	31.4%
R2		2,636	862	3,498	1	A3P600	25.3%
M		1,846	779	2,625	0	A3P600	19.0%

Table 2 · Device Utilization – ProASIC Family

Core	Family	Cells or Tiles			Memory Blocks	Device	Utilization
		Combinatorial	Sequential	Total			
BRM0E		5,544	1,185	6,729	0	APA450	54.8%
BRM1E		6,386	1,471	7,857	0	APA450	63.9%
BRM2E		5,586	1,190	6,776	2	APA450	55.1%
BR0E		4,675	984	5,659	0	APA450	46.1%
BR1E		5,456	1,269	6,725	0	APA450	54.7%
BR2E		4,698	988	5,686	2	APA450	46.3%
RM0E		4,476	1,104	5,580	0	APA450	45.4%
RM1E	ProASIC^PLUS ®	5,278	1,390	6,668	0	APA450	54.3%
RM2E		4,511	1,106	5,617	2	APA450	45.7%
BME		3,811	1,003	4,814	0	APA450	39.2%
BE		2,873	788	3,661	0	APA450	29.8%
R0		3,441	887	4,328	0	APA450	35.2%
R1		4,251	1,170	5,421	0	APA450	44.1%
R2		3,439	886	4,325	2	APA450	35.2%
M		2,453	789	3,242	0	APA450	26.4%

Table 3 · Device Utilization – Axcelerator®-Based Families

Core	Family	Cells or Tiles			Memory Blocks	Device	Utilization
		Combinatorial	Sequential	Total			
BRM0E	Axcelerator RTAX-S	2,634	1,162	3,796	0	AX250	89.9%
BRM1E		2,877	1,442	4,319	0	AX1000	23.8%
BRM2E		2,681	1,163	3,844	1	AX250	91.0%
BR0E		2,156	964	3,120	0	AX250	73.9%
BR1E		2,382	1,235	3,617	0	AX250	85.6%
BR2E		2,203	966	3,169	1	AX250	75.0%
RM0E		2,043	1,090	3,133	0	AX250	74.2%
RM1E		2,241	1,367	3,608	0	AX250	85.4%
RM2E		2,023	1,088	3,111	1	AX250	73.7%
BME		1,913	992	2,905	0	AX250	68.8%
BE		1,401	777	2,178	0	AX250	51.6%
R0		1,466	868	2,334	0	AX250	55.2%
R1		1,719	1,135	2,854	0	AX250	67.6%
R2		1,520	866	2,386	1	AX250	56.5%
M		1,231	766	1,997	0	AX250	47.3%

Table 4 · Device Utilization – SX-A–Based Families

Core	Family	Cells or Tiles			Memory Blocks	Device	Utilization
		Combinatorial	Sequential	Total			
BRM0E	SX-A RTSX-S	2,826	1,199	4,025	0	A54SX72A	66.7%
BRM1E		3,083	1,477	4,560	0	A54SX72A	75.6%
BRM2E		Not supported					
BR0E		2,374	1,003	3,377	0	A54SX72A	56.0%
BR1E		2,581	1,276	3,857	0	A54SX72A	63.9%
BR2E		Not supported					
RM0E		2,135	1,106	3,241	0	A54SX72A	53.7%
RM1E		2,418	1,392	3,810	0	A54SX72A	63.1%
RM2E		Not supported					
BME		1,966	1,017	2,983	0	A54SX72A	49.4%
BE		1,544	797	2,341	0	A54SX72A	38.8%
R0		1,560	885	2,445	0	A54SX72A	40.5%
R1		1,859	1,168	3,027	0	A54SX72A	50.2%
R2		Not supported					
M		1,209	774	1,983	0	A54SX72A	32.9%

The Core1553BRM clock rate can be programmed to be 12, 16, 20, or 24 MHz. All the Actel families listed above easily meet the required performance.

Core1553BRM I/O requirements depend on the system requirements and external interfaces. If the core and memory blocks are implemented within the FPGA and the CPU interface has a bidirectional data bus, approximately 67 I/O pins are required. If external memory is used with a bidirectional data bus, the number of I/O pins increases to approximately 110.

External Components

There are three external components required for proper operation of Core1553BRM:

- Memory: Between 1 kbyte and 128 kbytes (16 bits wide) of internal FPGA memory or external memory used for data storage
- Transceivers: Standard 1553B transceiver
- CPU: Used to control the core

The requirements for these three blocks are discussed in "Implementation Hints" on page 101.

MIL-STD-1553B Bus Overview

The MIL-STD-1553B bus is a differential serial bus used in military and space equipment. It comprises multiple redundant bus connections and communicates at 1 Mbps.

The bus has a single active BC and up to 31 RTs. The BC manages all data transfers on the bus using the command and status protocol. The BC initiates every transfer by sending a command word, and data if required. The selected RT will respond with a status word, and data if required.

The 1553B command word contains a 5-bit RT address, transmit or receive bit, 5-bit subaddress and 5-bit word count. This allows for up to 32 RTs on the bus. Normally, only 31 RTs can be connected to the bus, since RT address 31 is used to indicate a broadcast transfer. A broadcast transfer is one where all RTs accept the following data. Each RT has 30 subaddresses reserved for data transfers. The other two subaddresses (0 and 31) are reserved for mode codes used for bus control functions. Data transfers contain up to thirty-two 16-bit data words. Mode code command words are used for bus control functions such as synchronization.

Word Formats

There are only three types of words in a 1553B message: a command word (CW), a data word (DW), and a status word (SW). Each word consists of a 3-bit sync pattern, 16 bits of data, and a parity bit, making up the 20-bit word. The word formats are given in Figure 2.

Bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

CW

Sync	RT Address (5)	T/R (1)	Subaddress (5)	Word Count / Mode Code (5)	P (1)

DW

Sync	Data (16)	P (1)

SW

Sync	RT Address (5)	Message Error (1)	Instrumentation (1)	Service Request (1)	Reserved (3)	Broadcast Received (1)	Busy (1)	Subsystem Flag (1)	Dynamic Bus Acceptance (1)	Terminal (1)	Parity (1)
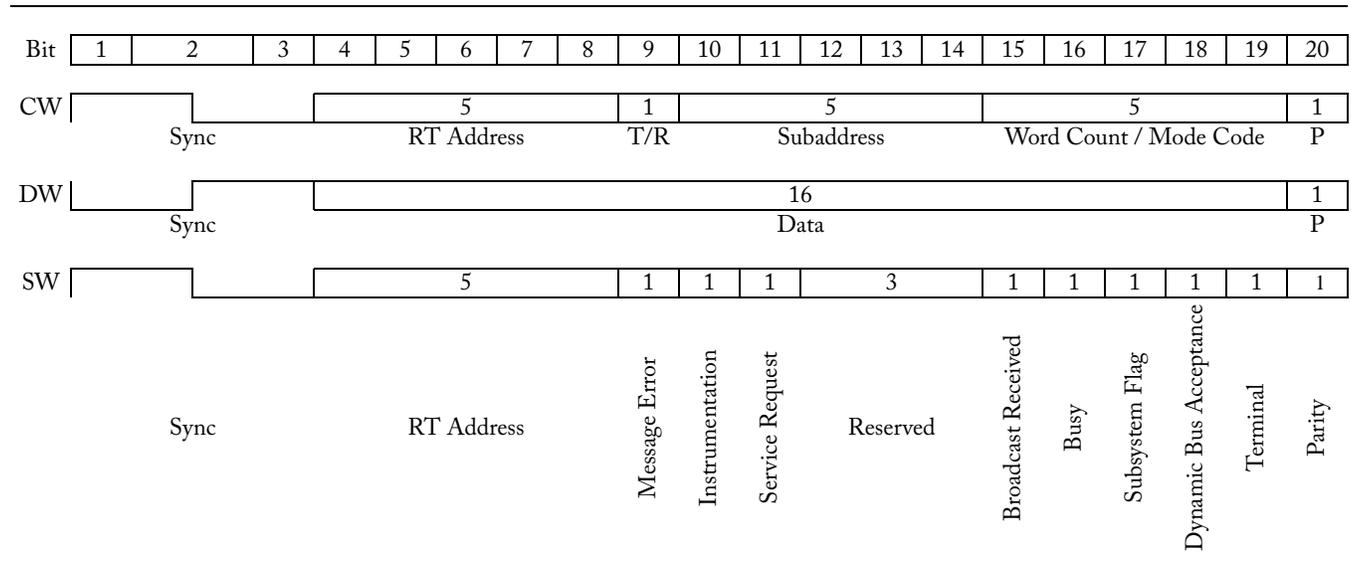
Figure 2 · 1553B Word Formats

Message Types

The 1553B bus supports 10 message transfer types, allowing basic point-to-point, broadcast, and BC-to-RT data transfers, mode code messages, and direct RT-to-RT messages. Figure 3 shows the message formats.

BC-to-RT Transfer

BC					RT		BC
Transmit Command	Data 0	Data …	Data n	Response Time	Status Word	Message Gap	Next Command

RT-to-BC Transfer

BC		RT					BC
Receive Command	Response Time	Status Word	Data 0	Data …	Data n	Message Gap	Next Command

RT-to-BC Transfer

BC		RT					BC
Receive Command	Response Time	Status Word	Data 0	Data …	Data n	Message Gap	Next Command

RT-to-RT Transfer

BC			RT 1					RT2		BC
Receive Command	Transmit Command	Response Time	Status Word	Data 0	Data …	Data n	Response Time	Status Word	Message Gap	Next Command

BC-to-all-RTs Broadcast

BC					BC
Transmit Command	Data 0	Data …	Data n	Message Gap	Next Command

RT-to-All-RTs Broadcast

BC			RT 1					BC
Receive Command	Transmit Command	Response Time	Status Word	Data 0	Data …	Data n	Message Gap	Next Command

Mode Command, No Data

BC		RT		BC
Mode Command	Response Time	Status Word	Message Gap	Next Command

Mode Command, RT Receive Data

BC			RT		BC
Mode Command	Mode Data	Response Time	Status Word	Message Gap	Next Command

Mode Command, RT Transmit Data

BC		RT			BC
Mode Command	Response Time	Status Word	Mode Data	Message Gap	Next Command

Broadcast Mode Command, No Data

BC		BC
Mode Command	Message Gap	Next Command

Broadcast Mode Command with Data

BC			BC
Mode Command	Mode Data	Message Gap	Next Command

Figure 3 · 1553B Message Formats

1

Functional Description

The core consists of six main blocks: a 1553 encoder, 1553 decoders, a protocol controller block, a CPU interface, a command word legality interface, and a backend interface (Figure 1-1).
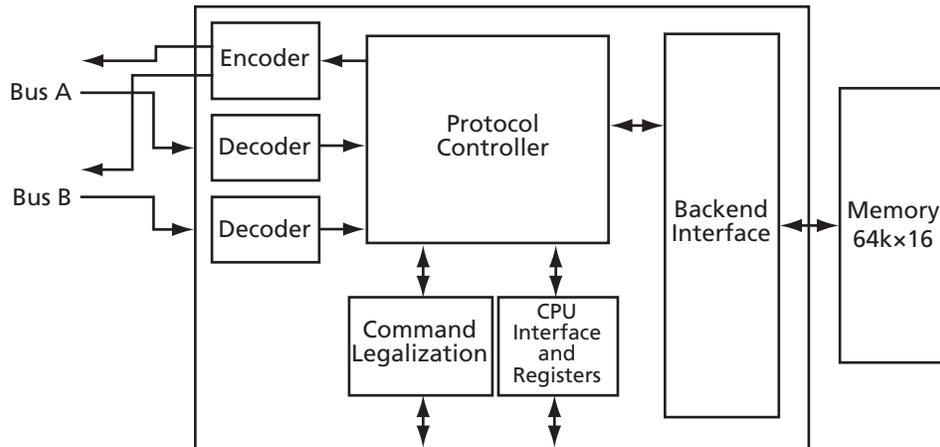
Figure 1-1 · Core1553BRM Block Diagram (all optional blocks included)

The core can be configured to provide all three functions—BC, RT, and MT—or any combination of the three. All core variations use all six blocks except for the command legalization interface, which is only required in RT functions that implement the RT legalization function externally.

A single 1553 encoder takes each word to be transmitted and serializes it using Manchester encoding. The encoder also includes loopback fail logic and independent logic to prevent Core1553BRM from transmitting for longer than the allowed period. The loopback logic monitors the received data and verifies that the core has correctly received every word that it transmits. The output of the encoder is gated with the bus enable signals to select which busses the core should be transmitting on.

Two decoders take the serial Manchester received data from each bus and extract the received data words. The decoder requires a 12, 16, 20, or 24 MHz clock to extract the data and clock from the serial stream.

The decoder contains a digital phase-locked loop (PLL) that generates a recovery clock used to sample the incoming serial data. The data is then deserialized and the 16-bit word decoded. The decoder detects whether a command, status, or data word has been received and checks that no Manchester encoding or parity errors have occurred in the word.

The protocol controller block handles all the message sequencing and error recovery for all three operating modes—BC, RT, and BM. This is a complex state machine that processes messages based on the message tables set up in memory, or reacts to incoming command words. The protocol controller implementation varies depending on which functions are implemented.

The CPU interface allows the system CPU to access the control registers within the core. It also allows the CPU to directly access the memory connected to the backend interface; this can simplify the system design. The core includes thirty-three 16-bit registers. Of the 33 registers, 17 are used for control functions and 16 for RT command legalization. The RT command legalization registers can be omitted from the core, reducing device utilization.

The command legality interface allows an external circuit to legalize command words that the remote terminal will respond to. The external legality checker allows a very small piece of logic to legalize command words down to word-count level, rather than using the sixteen 16-bit command legality registers within the CPU interface.

The memory interface for Core1553BRM allows a simple connection to a memory device. It can be configured to connect to either synchronous or asynchronous memory devices. This allows the core to be connected to synchronous logic or memory within the FPGA or to external memory blocks. The interface supports a standard bus request and grant protocol, and provides a WAIT input, allowing the core to interface to slow memory devices. This allows the core to share system memory rather than have its own dedicated memory block.

Registers

Core1553BRM contains thirty-three 16-bit registers (Table 1-1). One of these is used to enable enhanced Core1553BRM functions. The remaining 32 registers are used to control the core. The Control and Operation registers are used to allow a CPU to set the core operating mode; BC, RT, MT, or combined RT and MT. The function of the other registers varies depending on the operating mode.

Table 1-1 · Registers Address Map

Address	Name
00	Control
01	Operation and Status
02	Current Command
03	Interrupt Mask
04	Pending Interrupt
05	Interrupt Pointer
06	Built-In Test (BIT) Register
07	Time Tag
08	Descriptor Pointer
09	1553B Status Word
10	Initialization Count
11	Monitor Command Pointer
12	Monitor Data Pointer
13	Monitor Block Count
14	Monitor Filter A
15	Monitor Filter B
16–31	RT Command Legalization
32	Enhanced Features

Core Operation

Core1553BRM is designed to be software-compatible with existing 1553B solutions.

It supports the following features:

- Interrupt logs
- Programmable message timeouts
- Circular buffer operation

It does not support the following features:

- Buffer mode operation
- Built-in test functions, although the BIT register and the transmit BIT mode code are supported.
- Auto-initialization of internal registers and memory

Loopback Tests

Core1553BRM performs loopback testing on all of its transmissions; the transmit data is fed back into the receiver, and each transmitted word is compared to the original. If an error is detected, the transmitter shutdown bit is set in one of the core registers. The core also supports internal data loopback that may be used for self-testing without generating any 1553B transmissions.

Bus Transceivers

Core1553BRM needs a 1553B transceiver to drive the 1553B bus. Core1553BRM is designed to interface directly to common MIL-STD-1553 transceivers, such as the DDC BU-63147, Holt HI-1567/1568/1573/1574, or Aeroflex ACT4402. When using ProASIC^{PLUS}, RTAX-S, or Axcelerator FPGAs, level translators are required to connect the 5 V outputs of the 1553B transceivers to the 3.3 V inputs of the FPGA.

In addition to the transceiver, a pulse transformer is required for interfacing to the 1553B bus. Figure 1-2 and Figure 1-4 on page 19 show the connections required from Core1553BRM to the transceivers and then to the bus via the pulse transformers.

Typical System and Memory Requirements

Core1553BRM requires a master CPU to set up the registers and data tables. The CPU needs to able to access the internal core registers as well as the memory. Core1553BRM can be configured in two ways, with CPU shared memory (Figure 1-3 on page 18) and with its own memory (Figure 1-2).
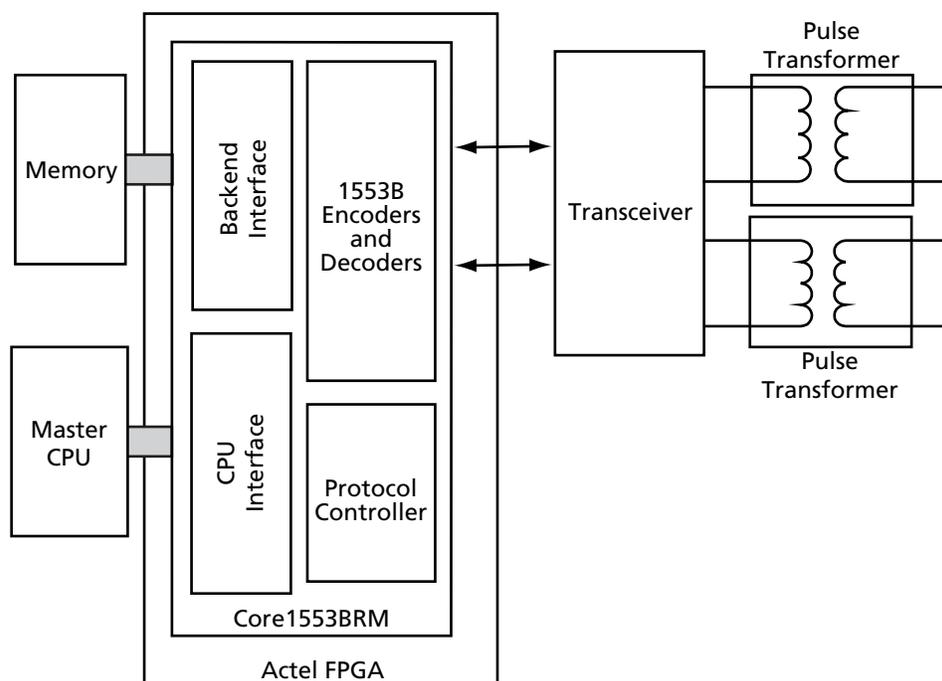
Figure 1-2 · Core1553BRM with Its Own Memory

When configured with its own memory, only the CPU port needs to be connected to the CPU. The CPU accesses the backend memory via Core1553BRM. This configuration also supports using internal FPGA memory connected to the core and removes the need for external bus arbitration on the CPU bus.
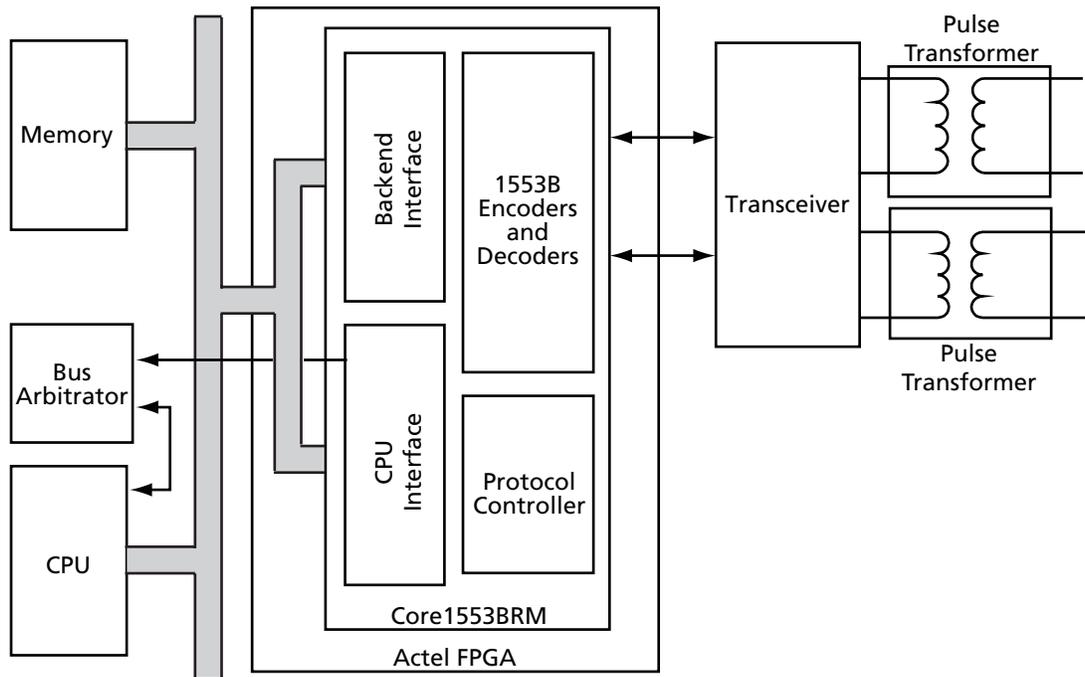
Figure 1-3 · Core1553BRM Using Shared Memory

Alternatively, the core can share CPU memory. In this case, both the backend memory and CPU interfaces are connected to the CPU bus. The core provides control lines that allow the memory and CPU interfaces to share the same top-level I/O pins. When in this configuration and the core needs to read or write the memory, it uses the MEMREQn, MEMGNTn, and MEMACCn signals to arbitrate for the CPU bus before completing the cycle.

Core1553BRM is compatible with legacy 1553B devices that use a single address and data bus when using a shared CPU and memory bus. The core also includes a wrapper file with a functional pinout that matches these legacy devices, allowing direct replacement.

For both shared and own memory systems, the core supports up to 128 kbytes of memory. The amount of memory required depends on the system requirements. A complete BC, RT, and MT could be created with only 1 kbyte of memory. Typical systems will have at least 4 kbytes of memory.

Development System

A complete 1553B bus controller development system is also available (Actel part number "Core1553BRM-Dev-Kit"). The development system (Figure 1-4) uses an external terminal (PC) using a serial UART link to control two separate Core1553BRM instances implemented in a single ProASIC$^{\text{PLUS}}$ APA600 FPGA.
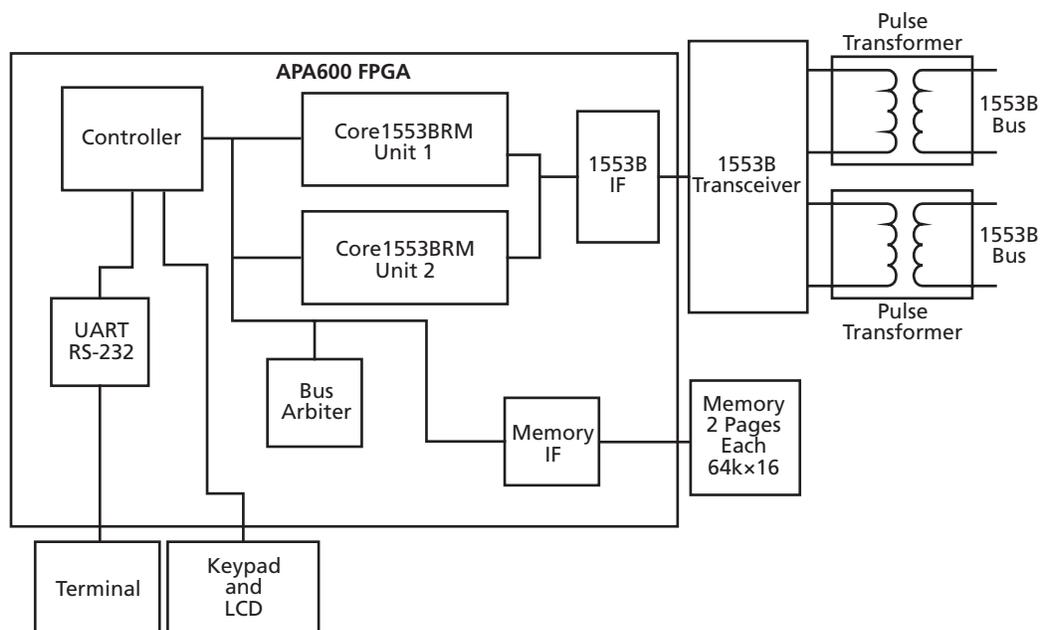
Figure 1-4 · Core1553BRM Development System

The 1553B interface logic allows the two cores to operate with a single transceiver. It can also provide a direct loopback mode, allowing the card to operate without any 1553B bus connections. The bus arbiter allows the controller and two cores to access the memory, which provides 64 k words of memory for each of the cores.

The development kit includes external memory and 1553 transceivers and transformers supporting both direct and transformer bus coupling.

On power-up, the controller will automatically configure one of the cores as a BC and the other as an RT/MT. The BC is programmed to transmit data to and from the RT. These 1553B messages can be monitored using an external bus monitor or using the terminal and monitor function in the RT/MT unit.

Tool Flows

Licenses

Core1553BRM is licensed in three ways; depending on your license, tool flow functionality may be limited.

Evaluation

Precompiled simulation libraries are provided, allowing the core to be instantiated in CoreConsole and simulated within Actel Libero IDE, as described in the "CoreConsole" section. The design may not be synthesized, as source code is not provided.

Obfuscated

Complete RTL code is provided for the core, enabling the core to be instantiated with CoreConsole. Simulation, Synthesis, and Layout can be performed with Libero IDE. The RTL code for the core is obfuscated,[1] and the some of the testbench source files are not provided. They are precompiled into the compiled simulation library instead.

RTL

Complete RTL source code is provided for the core and testbenches.

CoreConsole

Core1553BRM is preinstalled in the CoreConsole Intellectual Property Deployment Platform (IDP). To use the core, click and drag it from the IP core list into the main window. The CoreConsole project may be exported to Libero IDE at this point, providing access to the core only, or other IP blocks can be interconnected, allowing the complete system to be exported from CoreConsole to Libero IDE.

---

1. *Obfuscated means the RTL source files have had formatting and comments removed, and all instance and net names have been replaced with random character sequences.*

The core can be configured using the configuration GUI within CoreConsole, as shown in Figure 2-1. The "Parameters on Core1553BRM" section on page 25 describes the function of each of the parameters shown in Figure 2-1.
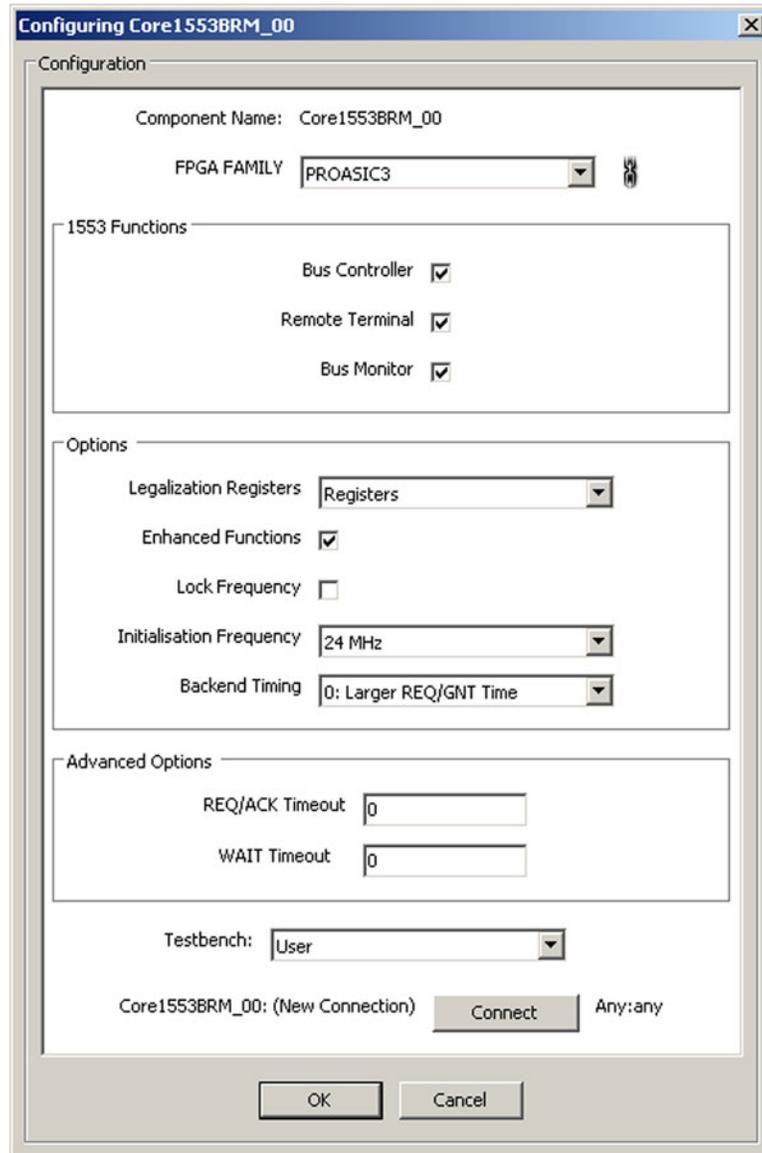
Figure 2-1 · Core1553BRM Configuration within CoreConsole

After configuring the core, Actel recommends you use the top-level Auto Stitch function to connect all the core interface signals to the top level of the CoreConsole project.

Once the core is configured, invoke the **Generate** function in CoreConsole. This will export all the required files to the project directory in the *LiberoExport* directory. This is in the CoreConsole installation directory by default.

Importing into Libero IDE

After generating and exporting the core from CoreConsole, the core can be imported into Libero IDE. Create a new project in Libero IDE and import the CoreConsole project from the *LiberoExport* directory. Libero IDE will then install the core and the selected testbenches, along with constraints and documentation, into its project.

Note: If two or more DirectCores are required, they can both be included in the same CoreConsole project and imported into Libero IDE at the same time.

Simulation Flows

To run simulations, the required testbench flow must be selected within CoreConsole and **Save & Generate** must be run from the Generate pane. The required testbench is selected through the core configuration GUI in CoreConsole. The following simulation environments are supported:

- Full 1553 verification environment (VHDL only)
- Simple testbench (VHDL and Verilog)

When CoreConsole generates the Libero IDE project, it will install the appropriate testbench files. To run the testbenches, simply **set the design root to the Core1553BRM instantiation in the Libero IDE** file manager and click the **Simulation** icon in Libero IDE. This will invoke Model*Sim*® and automatically run the simulation.

Synthesis in Libero IDE

To run Synthesis on the core with parameters set in CoreConsole, **set the design root to the top of the project imported from CoreConsole**. This is a wrapper around the core that sets all the generics appropriately. Click the **Synthesis** icon in Libero IDE. The synthesis window appears, displaying the Synplicity® project. To run Synthesis, click the **Run** icon.

Place-and-Route in Libero IDE

Having set the design route appropriately and run Synthesis, click the **Layout** icon in Libero IDE to invoke Designer. Core1553BRM requires no special place-and-route settings.

3

Interface Descriptions

Parameters on Core1553BRM

Core1553BRM has several top-level parameters (generics) that are used to select the operational modes of the core that are implemented (Table 3-1). Using these parameters allows the size of the core to be reduced when functions are not required.

Table 3-1 · Core Parameters

Name	Values	Description
FAMILY	0 to 21	Must be set to match the supported FPGA family: 8: SX-A 9: RTSX-S 11: Axcelerator 12: RTAX-S 14: ProASIC^{PLUS} 15: ProASIC3 16: ProASIC3E 17: Fusion 20: IGLOO 21: IGLOOe
BCENABLE	0 or 1	When 1, the BC function is implemented.
RTENABLE	0 or 1	When 1, the RT function is implemented.
MTENABLE	0 or 1	When 1, the MT function is implemented.
LEGREGS	0 to 2	This controls the implementation of the RT legalization registers.<table><tr><td>0</td><td>The legalization registers are not implemented. The user must use the external RT legalization interface.</td></tr><tr><td>1</td><td>The legalization logic is implemented using registers within the FPGA.</td></tr><tr><td>2</td><td>The legalization logic is implemented using memory within the FPGA.</td></tr></table>
ENHANCED	0 or 1	When 1, the Enhanced Features (Table 1-1 on page 16) register is implemented. When 0, the enhanced features are disabled and the sixth bit of the CPU address register is ignored.
INITFREQ	12, 16, 20, or 24	Sets the operating frequency of the core. Legal values are 12, 16, 20, and 24 MHz. If the Enhanced Features register is enabled, the operating frequency can be modified by the CPU.
LOCKFREQ	0 to 1	When 1, the core operating frequency is locked to the frequency set by INITFREQ. When 0, the clock frequency bits in the Enhanced Features register ("Register 32 – Enhanced Features Register" on page 80) can be used to change the clock frequency.
BETIMING	0 to 2	Modifies the backend timing requirements. Refer to Table 3-11 on page 32 and Table 3-12 on page 33.
ACKVAL	0 to 255	Specifies the REQ/GNT timer value when BETIMING = 2.
WAITVAL	0 to 255	Specifies the WAIT timer value when BETIMING = 2.

I/O Signal Descriptions

1553B Bus Interface

Table 3-2 · Bus Interface Signals

Name	Type	Description
BUSAINEN	Out	Active high output that enables for the A receiver
BUSAINP	In	Positive data input from the A receiver
BUSAINN	In	Negative data input from the A receiver
BUSBINEN	Out	Active high output that enables for the B receiver
BUSBINP	In	Positive data input from the bus to the B receiver
BUSBINN	In	Negative data input from the bus to the B receiver
BUSAOUTIN	Out	Active high transmitter inhibit for the A transmitter
BUSAOUTP	Out	Positive data output to the bus A transmitter (held HIGH when no transmission)
BUSAOUTN	Out	Negative data output to the bus A transmitter (held HIGH when no transmission)
BUSBOUTIN	Out	Active high transmitter; inhibits the B transmitter
BUSBOUTP	Out	Positive data output to the bus B transmitter (held HIGH when no transmission)
BUSBOUTN	Out	Negative data output to the bus B transmitter (held HIGH when no transmission)

Core Setup Signals

Table 3-3 · Core Setup Signals

Name	Type	Description
LOCKn	In	When 0, prevents the internal registers overriding the RTADDRIN, RTADDRPIN, MSELIN, and ABSTDIN inputs.
RTADDRIN[4:0]	In	Sets the RT address.
RTADDRPIN	In	RT address parity input.
RTADERR	Out	Indicates that the RT address is incorrectly set; active high.
MSELIN[1:0]	In	Sets the operating mode. 00: Bus Controller 01: Remote Terminal 10: Bus Monitor 11: Bus Monitor and Remote Terminal
ABSTDIN	In	Sets which bus standard is supported. 0: MIL-STD-1553-B 1: MIL-STD-1553-A
SSYSFn	In	Controls the subsystem flag bit in the 1553B status word; active low.

All core setup signals, apart from SYSSFn, are latched on the first clock edge after an external or software reset.

Remote Terminal Command Legalization Interface

Table 3-4 · Remote Terminal Command Legalization Interface

Name	Type	Description
CMDVAL[11:0]	Out	Active Command 11: 0 – Non-broadcast; 1 – Broadcast 10: 0 – Receive; 1 – Transmit 9:5: Subaddress 4:0: Word count / mode code These outputs are valid throughout the complete 1553B message. They can be also be used to steer data to particular backend devices. In particular, bit 11 allows non-broadcast and broadcast messaged to be differentiated, as required by MIL-STD-1553B Notice 2.
CMDSTB	Out	A single-cycle active high pulse that occurs just after CMDVAL changes
CMDOK	In	Command word is okay (active high). The external logic must set this within 2 µs of the CMDVAL output changing.
CMDOKOUT	Out	Indicates whether the internal core command word checking logic has validated the command word (active high).

Control and Status Signals

Table 3-5 · Control and Status Signals

Name	Type	Description
CLK	In	Master clock input (either 12, 16, 20, or 24 MHz)
TCLK	In	External time base clock input. Maximum frequency is ¼ of CLK with a 50% duty cycle.
RSTINn	In	Reset input (active low)
OPMODE[1:0]	Out	Indicates the actual operating mode: 00: Bus Controller 01: Remote Terminal 10: Bus Monitor 11: Bus Monitor and Remote Terminal
INTOUTH	Out	Hardware Interrupt Request (active high). This is set whenever bits 15:12 of the interrupt register are set. The CPU is required to read the internal status register to find the reason for the interrupt.
INTACKH	In	Hardware Interrupt Acknowledge (active high). This will clear the INTOUTH output.
INTOUTM	Out	Message Interrupt Request (active high). This is set whenever bits 11:0 of the interrupt register are set. The CPU is required to read the internal status register or interrupt log to find the reason for the interrupt.
INTACKM	In	Message Interrupt Acknowledge (active high). This will clear the INTOUTM output.
INTLEVEL	In	Sets the interrupt operating mode: 0: The INTOUM and INTOUTH outputs pulse active for three clock cycles. 1: The INTOUM and INTOUTH outputs go active and stay active until INTACKH or INTACKM are active.
MEMFAIL	Out	This goes HIGH if the core fails to read data from or write data to the backend interface within the required time. This can be caused by the backend not asserting MEMGNTn fast enough or asserting MEMWAITn for too long. It is cleared by the CPU writing to the interrupt register.
ACTIVE	Out	Indicates the CPU has started the core. For BC operations, this will be HIGH when the BC is processing a message list. For RT and MT operations, it will be HIGH when the RT/MT is either processing a 1553 message or waiting for a message.
BUSY	Out	This is HIGH when the core is processing a message. For BC operations, this will be HIGH when the BC is processing a message list. For RT and MT operations, it will by HIGH when the RT/MT is processing a 1553 message.
MSGSTART	Out	Indicates that the core has started to process a message.
CMDSYNC	Out	This pulses HIGH for a single clock cycle when the core detects the start of a 1553B command word (or status word) on the bus. Provides an early signal that the RT may be about to receive or transmit data or a mode code.
SYNCNOW	Out	This pulses HIGH for a single clock cycle when the RT receives a command to synchronize with or without data mode. The pulse occurs just after the 1553B command word (sync with no data) or data word (sync with data mode code) has been received.
BUSRESET	Out	This pulses HIGH for a single clock cycle whenever the RT receives a reset mode command. The core logic will also automatically reset itself on receipt of this command.
RSTOUTn	Out	Reset output (active low) The core's internal reset uses a global network that is active whenever the RSTINn is active or BUSRESET is active. This allows the rest of the system to make use of the global reset network if required.

The core uses a global resource (CLKINT) to drive the internal reset network.

CPU Interface

The CPU interface (Table 3-6) allows access to the Core1553BRM internal registers and direct access to the backend memory. This interface is synchronous to the clock.

Table 3-6 · CPU Interface Signals

Name	Type	Description
CPUCSn	In	CPU chip select input (active low)
CPUWRn[1:0]	In	CPU write input (active low). Two write inputs are provided for processors that support byte operations. When CPUWRn[1] = 0, data bits 15:8 are written; when CPUWRn[0] = 0, data bits 7:0 are written.
CPURDn	In	CPU read input (active low)
CPUWAITn	Out	CPU wait output (active low) Indicates that the CPU should hold CPURDn or CPUWRn active while the core completes the read or write operation. CPUWAITn is not asserted when the internal CPU registers are accessed. When accessing the backend interface through the core, CPUWAIT will be activated for a minimum of four clock cycles for read operations and three for write operations. CPUWAITn is asserted for extra clock cycles if the backend interface delays asserting MEMGNTn or asserts MEMWAITn. Timing is shown in Figure 4-4 on page 36 and Figure 4-5 on page 36.
CPUMEM	In	Selects whether CPU accesses internal registers or backend memory. 0: Accesses internal registers; register number is specified on CPUADDR[2:0] 1: Accesses the backend memory
CPUADDR[15:0]	In	CPU address input
CPUDOUT[15:0]	Out	CPU data output
CPUDIN[15:0]	In	CPU data input
CPUDEN	Out	Data bus enable (active high). This signal is HIGH when the core is providing data output on the CPUDOUT bus. It is intended for a tristate enable function.

Memory Interface

The memory interface supports both synchronous operation and asynchronous operation to backend devices (Table 3-7). Synchronous operation directly supports the use of internal FPGA memory blocks, and asynchronous operation allows connection to standard external memory devices.

Table 3-7 · Backend Signals

Name	Type	Description
MEMREQn	Out	Memory Request (active low) output Indicates that the core requires access to memory. MEMREQn will stay active until MEMGNTn is asserted.
MEMGNTn	In	Memory Grant (active low) input Indicates that the core has been granted access to the bus. The core will assert its MEMACCn output and start memory accesses. Once MEMACCn has been asserted, the MEMGNTn input can be deasserted. This input should be synchronous to CLK and needs to meet the internal register setup time.
MEMACCn	Out	Memory Access (active low) output The core will assert MEMACCn when MEMGNTn is asserted. It will hold MEMACCn active until it has completed its memory accesses. The core may do multiple memory accesses whilst MEMACCn is asserted.
MEMWRn[1:0]	Out	Memory Write (active low). When MEMWRn[1] = 0, D[15:8] are written; when MEMWRn[0] = 0, D[7:0] are written. Synchronous mode: This output indicates that data is to be written on the rising clock edge. If MEMWAITn is asserted, the MEMWRn pulse will be extended until MEMWAITn becomes inactive. Asynchronous mode: This output will be LOW for a minimum of one clock period and can be extended by the MEMWAITn input. The address and data are valid one clock cycle before MEMWRn is active and held for one clock cycle after MEMWRn goes inactive.
MEMRDn	Out	Memory Read (active low) Synchronous mode: This output indicates that data will be read on the next rising clock edge. If MEMWAITn is active, the data will be sampled on the rising clock edge on which MEMWAITn becomes inactive. This signal is intended as the read signal for synchronous RAMs. Asynchronous mode: This output will be LOW for a minimum of one clock period and can be extended by the MEMWAITn input. The address is valid one clock cycle before MEMRDn is active and held for one clock cycle after MEMRDn goes inactive. The data is sampled as MEMRDn goes HIGH.
MEMCSn	Out	Memory Chip Select (active low). This output has the same timing as MEMADDR.
MEMWAITn	In	Memory Wait (active low) Indicates that the backend is not ready and the core should extend the read or write strobe period. This input should be synchronous to CLK and needs to meet the internal register setup time. It can be permanently held HIGH.
MEMADDR[15:0]	Out	Memory address output
MEMDOUT[15:0]	Out	Memory data output
MEMDIN[15:0]	In	Memory data input

Table 3-7 · Backend Signals (continued)

Name	Type	Description
MEMCEN	Out	Control signal enable (active high). This signal is HIGH when the core is requesting the memory bus and has been granted control. It is intended to enable any tristate drivers that may be implemented on the memory control and address lines.
MEMDEN	Out	Data bus enable (active high). This signal is HIGH when the core is requesting the memory bus, has been granted control, and is waiting to write data. It is intended to enable any bidirectional drivers that may be implemented on the memory data bus.

Miscellaneous I/O

Several inputs are used to modify the core functionality to simplify integration in the application (Table 3-8). These inputs should be tied to logic 0 or logic 1 as appropriate.

Table 3-8 · Miscellaneous I/O

Name	Type	Description
ASYNCIF	In	When 1, the backend interface is in asynchronous mode. When 0, the backend interface is in synchronous mode.
CPUMEMEN	In	When 1, the CPU interface has access to the backend memory. When 0, the CPU cannot access the backend memory through the core. This must be set to 0 if the core shares the CPU memory, i.e., the CPU and memory busses are connected to the same system bus.

Backend Memory Interface Timing

The core may do multiple memory accesses in a single memory access cycle (MEMACCn asserted). The number of memory cycles depends on the state and operating mode of the core. The minimum and maximum number of memory cycles for each of the modes is given in Table 3-9 and Table 3-10 on page 32.

Table 3-9 · Memory Access Burst Lengths

Mode	Minimum Memory Cycles	Maximum Memory Cycles
RT	1	See Table 3-10 on page 32.
BC	1	4
MT	1	6
RT/MT	1	6

Table 3-10 · RT Mode

RT Mode	Number of Read Cycles in Initial Burst Read	Number of Write Cycles in Burst Write
Ping Pong	4	5
Index	4	6
Circular Mode 1	4	6
Circular Mode 2	4	7

The memory interface must allow the core to access memory when requested. When the core asserts MEMREQn, the external memory interface must assert MEMGNTn within the time period specified in Table 3-11 and Table 3-12 on page 33. The core also limits the number of wait cycles that may be inserted and, hence, the width of the MEMRDn and MEMWRn pulses. The core supports two fixed sets of backend timing parameters controlled by the BETIMING parameter, along with user-configurable settings. When BETIMING = 1, the bus request/grant time is reduced and the number of wait cycles per access is increased. BETIMING = 2 allows the user to pick the tradeoff between the REQ/GNT and wait times; "ACKVAL and WAITVAL Settings" on page 117 lists the settings that may be used for the ACKVAL and WAITVAL generics. When the CPU is allowed to access the memory through the core (CPUMEMEN active), the memory access time is reduced.

Table 3-11 · Memory Access Requirements (BETIMING = 0)

CPUMEMEN	CLK Speed in MHz	MEMREQn to MEMGNTn Maximum Delay in μs	Maximum Number of Wait States	Maximum Read/ Write Pulse Width Clocks	Maximum Read/ Write Pulse Width in ns
0	12	4.917	3	4	333.33
0	16	5.063	6	7	437.50
0	20	5.600	7	8	400.00
0	24	6.250	7	8	333.33
1	12	2.750	3	4	333.33
1	16	2.813	6	7	437.50
1	20	3.300	7	8	400.00
1	24	3.792	7	8	333.33

Table 3-12 · Memory Access Requirements (BETIMING = 1)

CPUMEMEN	CLK Speed in MHz	MEMREQn to MEMGNTn Maximum Delay in µs	Maximum Number of Wait States	Maximum Read/ Write Pulse Width Clocks	Maximum Read/ Write Pulse Width in ns
0	12	4.000	5	6	500.00
0	16	4.000	9	10	625.00
0	20	4.250	12	13	650.00
0	24	4.917	13	14	583.33
1	12	2.000	5	6	500.00
1	16	2.000	9	10	625.00
1	20	2.250	12	13	650.00
1	24	2.750	13	14	583.33

Interface Timing

CPU Interface Timing

CPUDOUT is asynchronous to CLK for all reads of registers, except for the RT legalization registers when implemented using memory (LEGREGS = 2). In this case, CPUDOUT is synchronous to the clock. The CPU interface must ensure that the read pulse is long enough to guarantee that one positive clock edge occurs during the read pulse. CPU interface timing is shown in Figure 4-1 through Figure 4-7 on page 36.
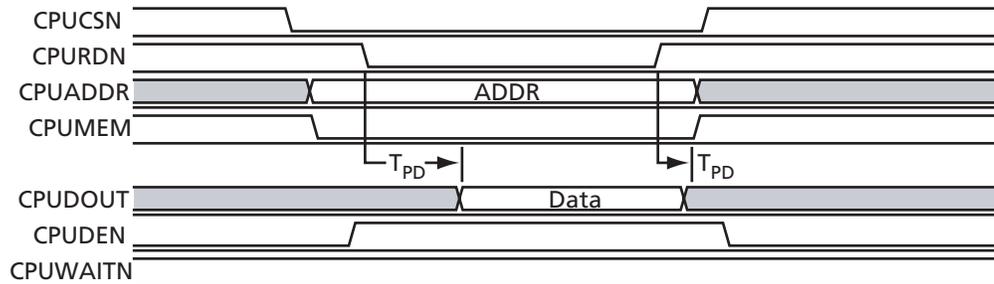
Figure 4-1 · CPU Interface Register Read Cycle
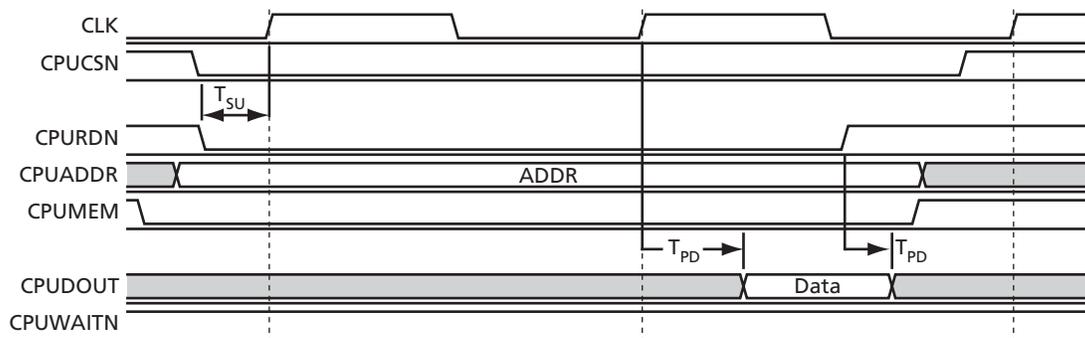
Figure 4-2 · CPU Interface Register Read Cycle – RT Legalization Registers Using Memory
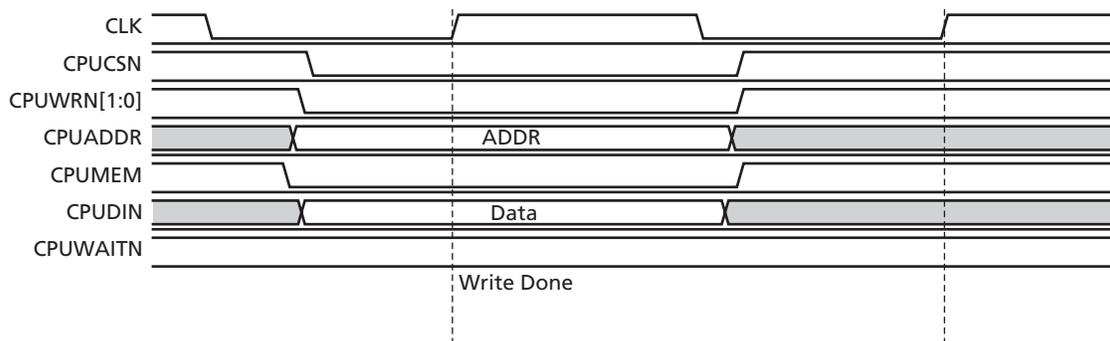
Figure 4-3 · CPU Interface Register Write Cycle
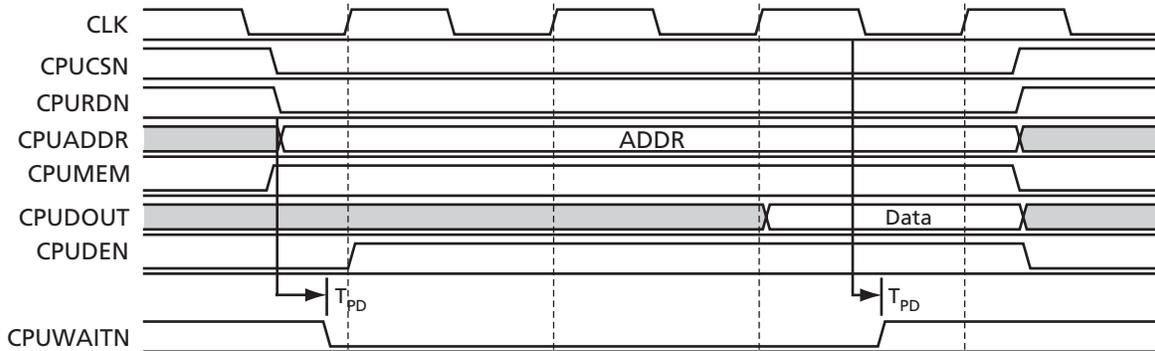
Figure 4-4 · CPU Interface Memory Read Cycle
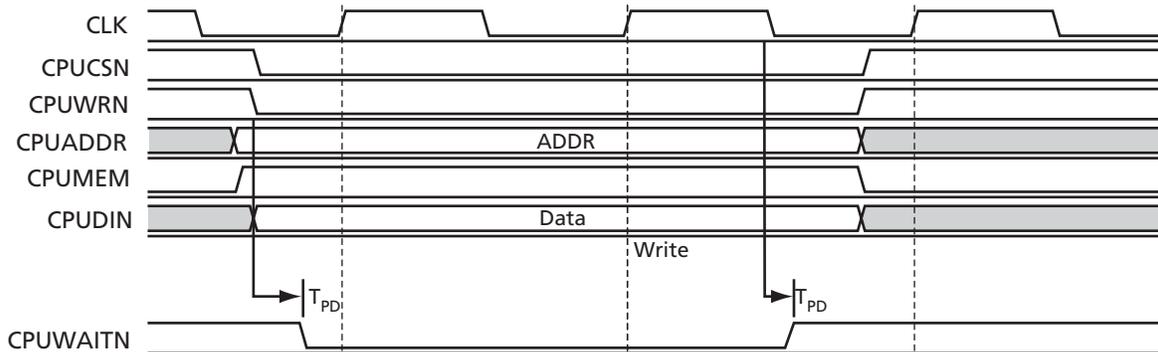
Figure 4-5 · CPU Interface Memory Write Cycle
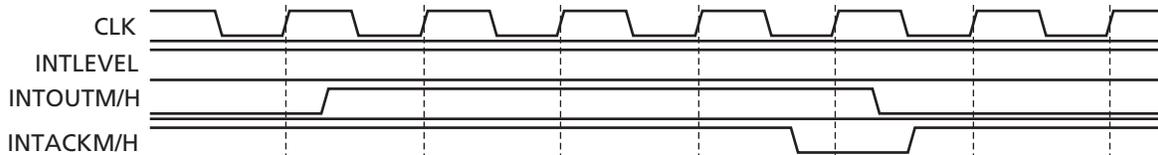
Figure 4-6 · Interrupt Timing (INTLEVEL = 1)
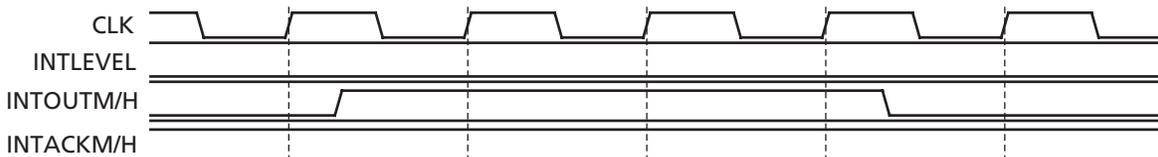
Figure 4-7 · Interrupt Timing (INTLEVEL = 0)

CPUWAITn will be driven LOW for a minimum of three write cycles or four read cycles, plus however many clock cycles the memory backend delays the assertion of MEMGNTn and asserts MEMWAITn for. CPUWAITn is driven LOW by CPURDn/CPUWRn becoming active, and returns HIGH on the falling clock edge after the data is valid.

The CPU interface signals are internally synchronized to the Core1553BRM master clock. If these inputs are asynchronous, CPUCSn, CPUADDR, and CPUDATA should be valid before CPUWRn and remain valid after the CPUWRn pulse. CPUWRn must be active for at least one clock cycle.

Memory Timing
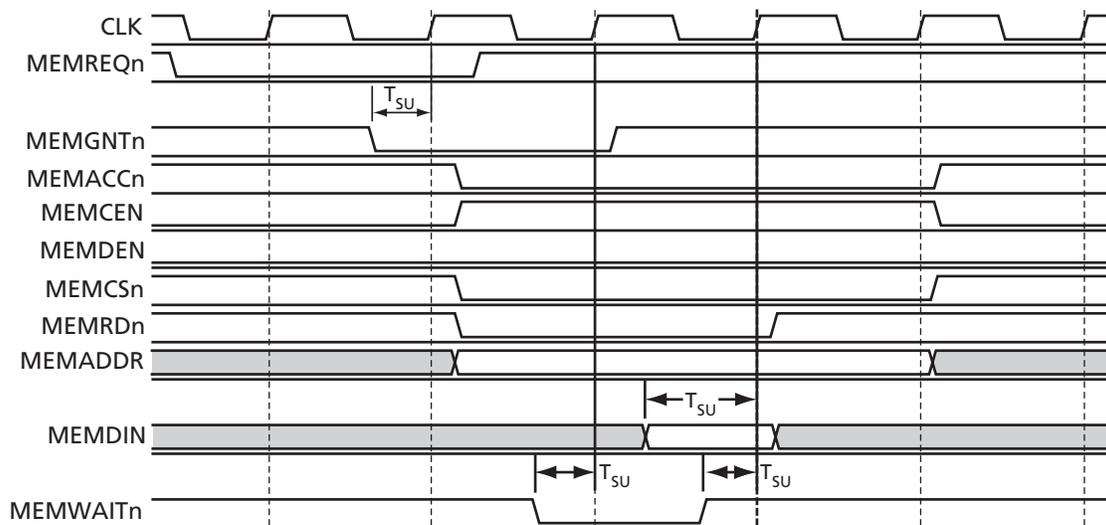
Figure 4-8 · Asynchronous Memory Read Cycle
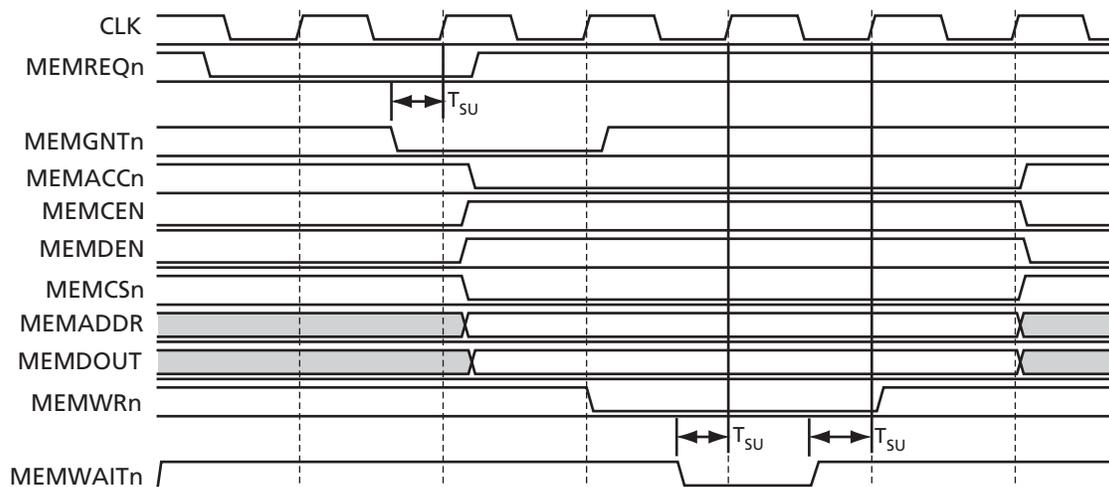
Figure 4-9 · Asynchronous Memory Write Cycle
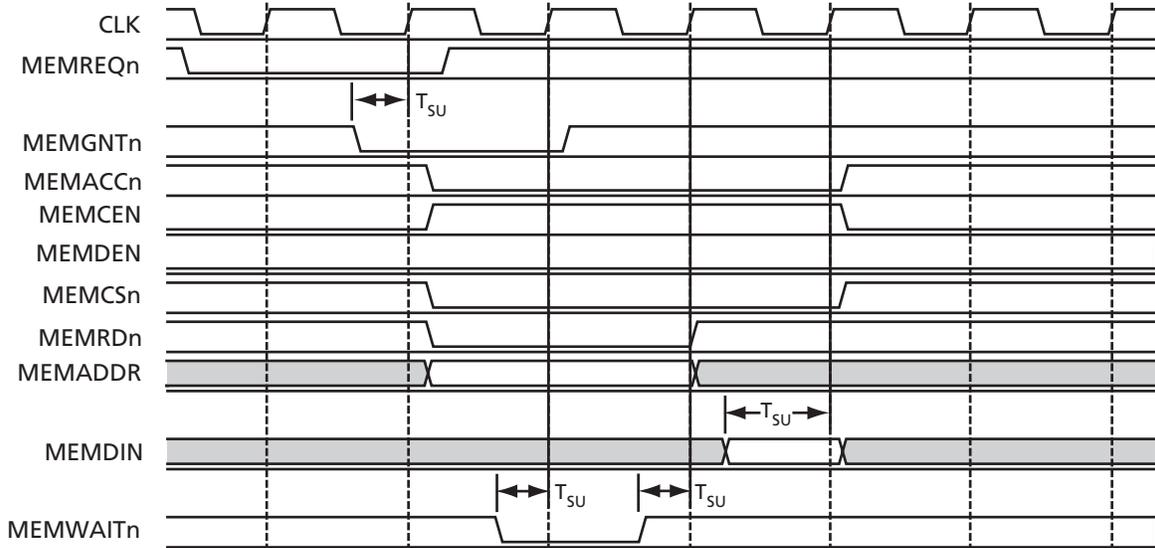
Figure 4-10 · Synchronous Memory Read Cycle
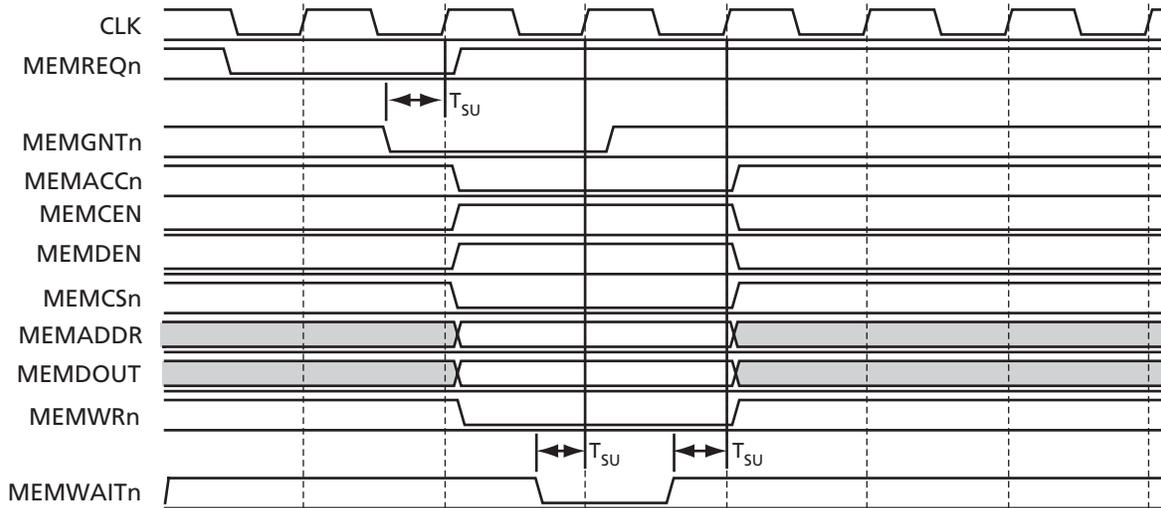
Figure 4-11 · Synchronous Memory Write Cycle
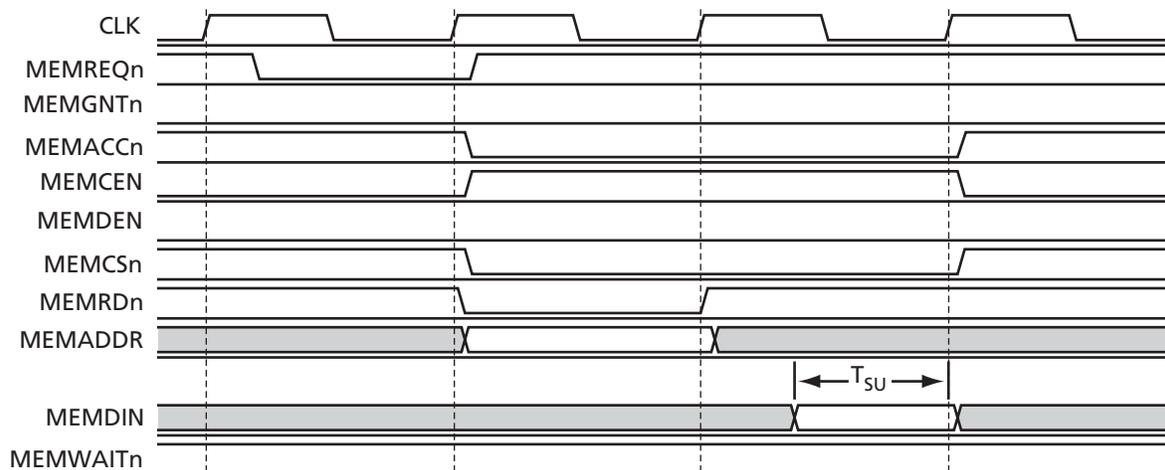
Figure 4-12 · Synchronous Memory Read Cycle with MEMGNTn Active
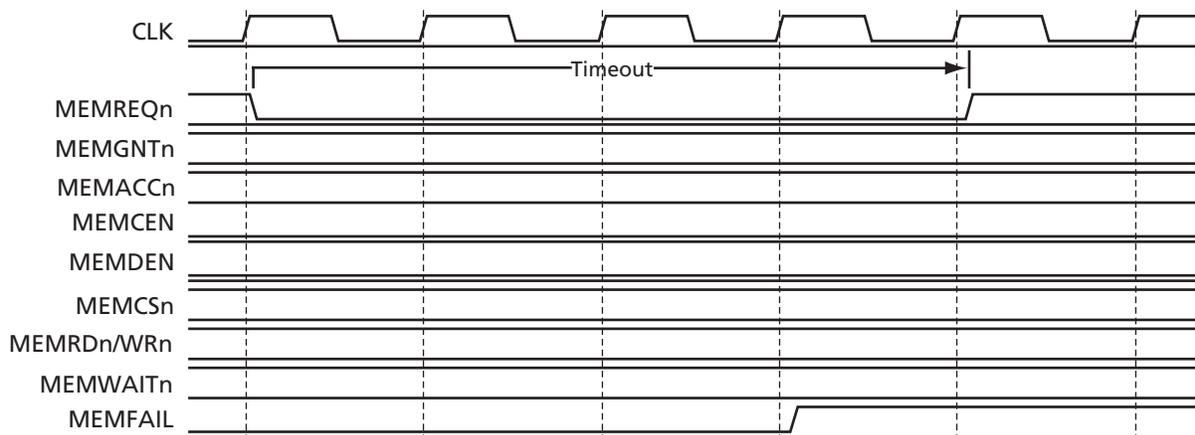
Figure 4-13 · Memory Grant Timeout
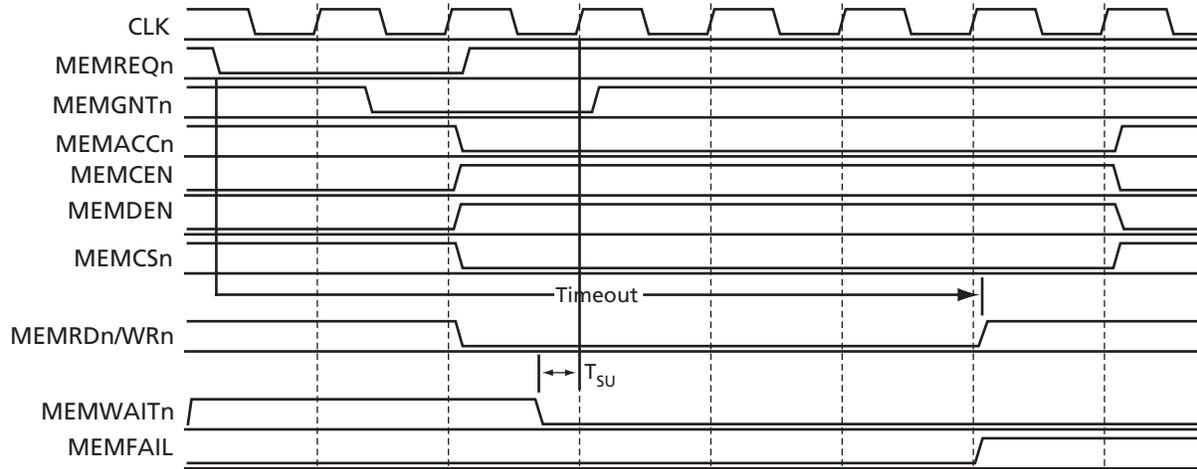
Figure 4-14 · Memory Wait Timeout

Figure 4-15 shows the timing of the external legalization logic interface. External logic has 3 μs to assert the CMDOK input after CMDSTB is asserted. When the internal legalization registers are used, the CMDOKOUT output will be asserted two clock cycles after CMDSTB.
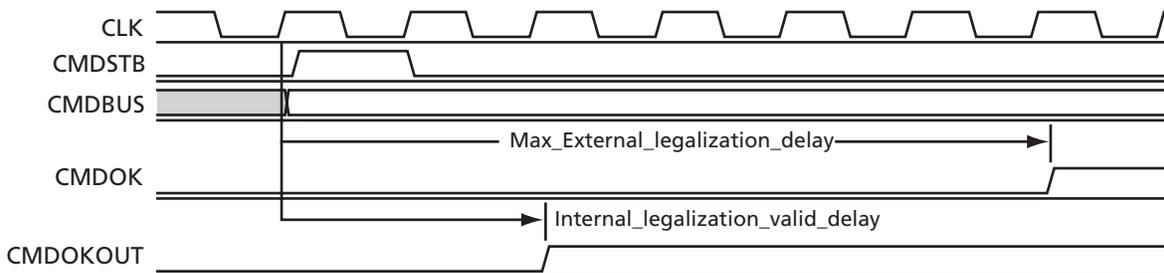
Figure 4-15 · RT Legalization Interface

RT Response Times

RT response time is from the midpoint of the parity bit in the command word to the midpoint of the status word sync (Table 4-1).

RT-to-RT timeout is from the first command word parity bit to the expected sync of the first data word.

Table 4-1 · RT Response Times

Spec	Description	At 12 MHz	At 16 MHz	At 20 MHz	At 24 MHz
T_{rtresp}	RT response time	4.75 to 7.0 μs	4.75 to 7.0 μs	4.75 to 7.0 μs	4.75 to 7.0 μs
T_{rtrtto}	RT-to-RT timeout	57 μs	57 μs	57 μs	57 μs
T_{xxto}	Transmitter timeout	704 μs	668 μs	691 μs	693 μs

Transceiver Loopback Delays

Core1553BRM verifies that all transmitted data words are correctly transmitted. As data is transmitted by the transceiver on the 1553B bus, the data on the bus is monitored by the transceiver and decoded by Core1553BRM. The core requires that the loopback delay, i.e., the time from BUSAOUTP to BUSAINP, be less than the values given in Table 4-2.

Table 4-2 · Transceiver Loopback Requirements

Clock Speed	Maximum Loopback Delay
12 MHz	2.50 µs
16 MHz	2.50 µs
20 MHz	2.45 µs
24 MHz	2.40 µs

The loopback delay is a function of the internal FPGA delay, PCB routing delays, internal transceiver delay, and transmission effects from the 1553B bus. Additional register stages can be inserted in the FPGA on either the 1553B data input or output, providing the loopback delays in Table 4-2 are not violated. This is recommended if additional gating logic is inserted inside the FPGA between the core and transceiver to minimize skew between the differential inputs and outputs.

Clock Requirements

To meet the 1553B transmission bit rate requirements, the Core1553BRM clock input must be 12, 16, 20, or 24 MHz with a tolerance of ± 0.01%.

Core1553BRM Operation as a Bus Controller

Overview

Core153BRM can either be synthesized to function as a BC only, or the entire core can be implemented and then configured to operate only as a BC via signals MSELIN[1:0] or MSEL[1:0] (register 1, bits 9:8). See "Register 01 – Operation and Status" on page 74.

Features

Properly configured, the core can implement a full-featured, fully-MIL-STD-1553A/B-compliant bus controller. In addition, the core provides register compatibility with legacy 1553A/B BCs. The core is designed to operate with little host intervention and offers a number of user-customizable features.

Multiple Message Processing

The core operates using an opcode command set to control the command block flow. In addition, the core provides for chaining of multiple 1553 commands into major and minor frames as needed. This ability to chain commands allows the core to perform complex tasks with little or no host intervention.

Message Scheduling

The core architecture provides for host control of message flow. For example, the core architecture allows the core to perform periodic message transactions with multiple remote terminals.

Polling

The architecture also supports polling, allowing the host to request status word responses from selected RTs. Polling can determine what action, if any, should be taken by the core (generate a specific interrupt, branch to a new message frame, etc.)

Automatic Retry

The core supports automatic message retry, whether due to an error or a specific received status bit. The core can retry sending a message up to four times per command block, on either the primary or the alternate bus.

Control and Message Processing

When Core1553BRM operates as a BC, configuration data for the core is stored in registers, and commands and data are stored in external memory. Details of the memory structure are discussed in this section; the control registers are described both here and in "Registers" on page 44.

Message processing is controlled through the use of command blocks, eight-word, contiguous blocks of memory that contain opcodes for controlling the core as well as 1553 command words and associated data locations in memory. See "Command Blocks" on page 45 for more details.

The core will execute command blocks in a contiguous fashion as long as no "go to," "branch," "call," or "return" opcodes are used. The core reads the command block during minor frame processing (i.e., after assertion of ACTIVE), during which it will arbitrate for control of the memory bus. After completing a read of the command block, the core will surrender control of the bus (i.e., deassert MEMACCn) and begin the acquisition of data words for either transmission or storage.

For 1553 receive commands (BC transmits data), the data pointer determines the location of the data words to be retrieved (see "Command Blocks" on page 45). The core will retrieve data words sequentially from the address specified

by the data pointer. Conversely, for a transmit command (BC receives data), the data pointer determines the memory location for data storage. The core stores data words sequentially starting from this memory location.

After transmission or reception, the core will begin command post-processing. The core will first arbitrate for the memory bus and then perform a DMA burst to update the command block status. An optional interrupt log entry is made after a command block update, during which the core modifies the control word as required.

Configuration of the core as a BC is controlled though the use of 10 registers (out of the 33 defined in the core architecture). The registers contain setup information, command and data locations, and status info.

Registers

The functionality of the core as well as its specific responses to 1553 events is controlled through registers. In addition to the seven control registers common to all core implementations, Core1553BRM, when implementing a BC, has three registers used to control its functions. Table 5-1 shows which bits of the 10 control registers are used by the core in BC mode. See "Core1553BRM Registers" on page 71 for detailed register usage information.

Table 5-1 · Register/Bit Applicability Map for Core153BRM as Bus Controller

Register Address	Name	Bit Locations															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00	Control	✓	✓	✓			✓	✓			✓		✓		✓	✓	
01	Operation and Status						✓	✓	✓	✓	✓			✓		✓	✓
02	Current Command	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
03	Interrupt Mask	✓	✓		✓	✓						✓	✓	✓	✓	✓	
04	Pending Interrupt	✓	✓		✓	✓						✓	✓	✓	✓	✓	
05	Interrupt Pointer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
06	Built-In Test Register	✓	✓			✓	✓			✓	✓	✓	✓	✓	✓	✓	✓
07	Minor Frame Timer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
08	Descriptor Pointer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
10	Initialization Count	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
32	Enhanced Features												✓	✓		✓	✓

Memory Structure

The external memory space (up to 64 k words) can be sized and allocated by the user according to the needs of the application. This memory space is needed to hold command blocks, data, and the interrupt log list. How the memory is allocated is up to the user, within the restrictions listed.

As the number of command blocks needed for the application is known, the user can predetermine the space required for their storage. The command blocks can be stored in contiguous memory locations for ease of operation. However, with the use of "go to," "branch," "call," or "return" opcodes, almost any memory configuration is possible. Command blocks together are referred to as a command frame. If branching is used, smaller collections of command blocks are referred to

as minor frames. The "go to," "branch," "call," and "return" opcodes can be used to link these minor frames together for command processing.

The starting address is set by register 8, the Command Block Pointer. As the first command block is processed, the value of register 8 is updated by the core to reflect the location of the next command block.

Each command block contains a data pointer to indicate where data for the block is stored. It is suggested that the data memory space be allocated to follow the command block space. Since the core can store and fetch a specified number of data words, memory space can be allocated efficiently. In addition, if the same data is to be sent to multiple RTs, this data need only be stored in a single memory location. See "Command Blocks" for more details.

The Interrupt Log List is a 32-word ring buffer that contains information necessary to service interrupts. The memory space for the Interrupt Log List must be allocated on a 32-word boundary. The starting location for the Interrupt Log List is set by register 5, the Interrupt Pointer.

Command Blocks

As stated earlier, command blocks are eight-word, contiguous blocks of memory that contain opcodes for controlling the core as well as 1553 command words and associated data locations in memory. Each command word transmitted over the bus must be associated with a command block. The command block's eight contiguous memory locations are one control word, two command words, a data pointer, two status words, a branch address, and a timer value (Table 5-2).

Table 5-2 · Command Block Architecture

Word	Function
1	Control Word
2	Command Word 1
3	Command Word 2
4	Data Pointer
5	Status Word 1
6	Status Word 1
7	Branch Address
8	Timer Value

Control Word

Located in the first memory location of each command block is the Control Word (Figure 5-1). A Control Word contains the opcode, number of retries, bus definition, RT–RT instruction, condition codes, and block access message error (BAME) necessary to complete a single 1553 command.
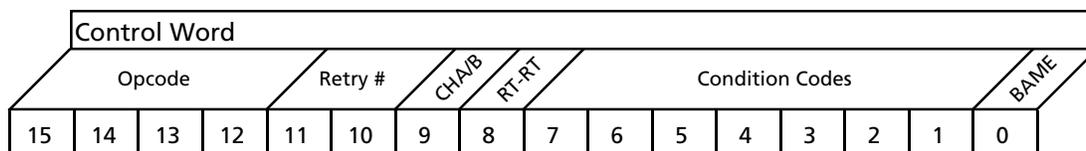
Figure 5-1 · Control Word

Bits 15:12 – Opcode

These bits specify the opcode to be used for the command block being processed. If the opcode does not call for a 1553 function, all remaining bits of the control word are ignored. Table 5-4 on page 47 lists available opcodes and their functions.

Bits 11:10 – Retry # (Retry Number)

The Retry # bits specify the number of times the core will retry each command block, providing that the opcode allows for retrying. The setting for PPEN (register 0, bit 2) will determine on which bus the retry will occur. The settings are shown in Table 5-3.

Table 5-3 · Bits 11:10 Retry Settings

No. of Retries	Bit 11	Bit 10
1	0	1
2	1	0
3	1	1
4	0	0

Bit 9 – CHA/B (Channel A/B)

Setting this bit HIGH selects bus A as the primary bus for 1553 transmissions. LOW selects bus B.

Bit 8 – RT–RT (RT-to-RT Transfer)

The RT–RT bit defines whether the current command block involves an RT-to-RT transfer and should therefore transmit Command Word 2. This bit is active high.

Note: The core will store all data associated with an RT-to-RT transfer.

Bit 7 – Condition Code 7

The Message Error condition will be met if the core detects an error in the response from the RT or if there is no response after the message timeout period has expired.

Bit 6 – Condition Code 6

This condition is met if the core receives a Status Word response from the RT with the Message Error bit set (bit 9 in 1553A mode).

Bit 5 – Condition Code 5

This condition is met if the core receives a Status Word response from the RT with the Busy bit set (bit 16 in 1553A mode).

Bit 4 – Condition Code 4

This condition is met if the core receives a Status Word response from the RT with the Terminal Flag bit set (bit 19 in 1553A mode).

Bit 3 – Condition Code 3

This condition is met if the core receives a Status Word response from the RT with the Subsystem Fail bit set (bit 17 in 1553A mode).

Bit 2 – Condition Code 2

This condition is met if the core receives a Status Word response from the RT with the Instrumentation bit set (bit 10 in 1553A mode).

Bit 1 – Condition Code 1

This condition is met if the core receives a Status Word response from the RT with the Service Request bit set (bit 11 in 1553A mode).

Bit 0 – BAME

When BAME is HIGH, this indicates that a protocol message error occurred in the RT response. The CPU should reset this bit when writing the control word into memory.

Table 5-4 · Opcodes

Opcode	Name	1553 Command Processing	Function
0000	End of List		Instructs the core that the last command block has been encountered. Command processing stops and the EOL interrupt (register 4, bit 5) is generated if enabled.
0001	Skip		Instructs the core to load the message-to-message timer with the value stored in Timer Value (command block, word 8). The value sets the length of time the core will wait before proceeding to the next command block.
0010	Go To		Instructs the core to branch to the command block starting at the address located in Branch Address (command block, word 7). The GOTO instruction also supports asynchronous operation when enhanced functions are enabled (see "Bus Controller GOTO Enhancements" on page 89).
0011	BIT		Core1553BRM does not implement built-in self-test. This command will clear any error conditions set in the BIT word (register 6), and the core will jump to the next command block.
0100	Execute Block – Continue	✓	Instructs the core to execute the current command block and proceed to the next upon completion.
0101	Execute Block – Branch	✓	Instructs the core to execute the current command block and branch unconditionally to the command block starting at the address located in Branch Address (command block, word 7).
0110	Execute Block – Branch on Condition	✓	Instructs the core to execute the current command block and to branch to the command block starting at the address located in Branch Address (command block, word 7) if the conditions listed in bits 7:1 of the Control Word are met. If the condition is not met, this opcode will function as opcode 0100.
0111	Retry on Condition	✓	Instructs the core to retry a message the number of times indicated in bits 11:10 of the Control Word if the conditions in bits 7:1 are met. If the conditions are not met, this opcode will function as opcode 0100.

Table 5-4 · Opcodes (continued)

Opcode	Name	1553 Command Processing	Function
1000	Retry on Condition – Branch	✓	Instructs the core to retry a message the number of times indicated in bits 11:10 of the Control Word if the conditions in bits 7:1 are met. Once the specified number of retries has been executed, the core branches to the command block starting at the address located in Branch Address (command block, word 7). If the condition is not met, this opcode will function as opcode 0100.
1001	Retry on Condition – Branch if All Retries Fail	✓	Instructs the core to retry a message the number of times indicated in bits 11:10 of the Control Word if the conditions in bits 7:1 are met. If all message retries fail, the core branches to the command block starting at the address located in Branch Address (command block, word 7). If these conditions are not met, this opcode will function as opcode 0100.
1010	Interrupt – Continue		Instructs the core to generate an interrupt and continue with the next command block.
1011	Call		Instructs the core to branch to the command block starting at the address located in Branch Address (command block, word 7). The core stores the address of the next command block for use by opcode 1100. The CALL instruction also supports asynchronous operation when enhanced functions are enabled (see "Bus Controller GOTO Enhancements" on page 89).
1100	Return to Call		Instructs the core to return the command block at the address stored by opcode 1011.
1101	Reserved		The core will generate an Illegal Opcode interrupt (register 4, bit 3), if enabled, and terminate execution.
1110	Load Minor Frame Timer		Instructs the core to load the minor frame timer with the value stored in Timer Value (Control Block, word 8). The core will load the time when the previously set timer value is decremented to zero. Once the timer has been loaded, the core will process the next command block.
1111	Return to Branch		Instructs the core to return to the command block at the address saved during opcodes 0101 or 0110.

Command Words

Located in the second and third memory locations of each command block are 1553 command words. For most 1553 messages, only the first command word is needed. During RT-to-RT transfers, the first command word is the receive command and the second is the transmit command.

Data Pointer

Located in the fourth memory location of each command block is the data pointer, indicating the first location in memory where data associated with the command word(s) is to be stored or fetched from.

Status Word

Command block words 5 and 6 are for 1553 status word storage. The core will store the RT's responding status after a 1553 command. For an RT-to-RT transfer, the status word from the transmitting RT will be stored in word 5, and the status word from the receiving RT will be stored in word 6.

Branch Address

Word 7 of the command block contains the starting address of the command block that is the destination of a branch opcode.

Timer Value

This word of the command block contains the value for setting one of two timers: either the minor frame timer (opcode 1110) or the message-to-message timer (opcode 0001).

Note: The minor frame timer can be driven either from the TCLK pin or an internal 15.625 kHz clock. The message-to-message timer is clocked by the core clock input (12, 16, 20, or 24 MHz).

MIL-STD-1553A Operation

Core1553BRM can be configured to operate compliant with MIL-STD-1553A. Taking input signal ABSTDIN HIGH configures the core for MIL-STD-1553A-compliant operation (taking this signal LOW actives MIL-STD-1553B mode). An alternate method for configuring the core is to use bit 7, A/B STD (1553A or 1553B Support). When configured for MIL-STD-1553A BC operation, the core will do the following:

• Expect a response from the RT within 9 μs after a message is sent

• Define all mode codes without data

• Define subaddress 00000b as a valid mode code

6

Core1553BRM Operation as a Remote Terminal

Overview

Core1553BRM can either be synthesized to function as a remote terminal only, or the entire core can be implemented and then configured to operate only as an RT via signals MSELIN[1:0] or MSEL[1:0] (register 1, bits 9:8). See "Register 01 – Operation and Status" on page 74.

Features

Indexing

The core, when configured as an RT, can support bulk data transfer, buffering up to 256 messages per subaddress. Once a specified number of messages has been received, the core can signal the host subsystem via an interrupt.

Buffer Ping Pong

The core supports the use of dual buffers per subaddress for data processing. This allows the core to process messages using the primary buffer while the host or subsystem can access the secondary buffer to store new data for transmission or process previously received data. The core will switch back and forth (ping pong) between the two buffers when a message is received or transmitted.

Circular Buffers

To simplify the software servicing of the RTs during periodic or bulk data transfers, the core supports the use of circular buffers. The user can select between two circular buffer modes at start-up or rely on the default operation.

Broadcast

The core architecture allows the user to choose whether or not data received from broadcast commands is to be segregated from data received from non-broadcast commands.

Interrupt History

The core architecture supports a programmable interrupt structure that can store up to 16 interrupts, and the subaddress or command block that generated each interrupt, in a 32-word buffer before servicing.

Message Information

Along with message data words, the core also writes a Message Information Word (MIW) and Time Tag for each processed message. The MIW contains information on the type of message transacted, the word count, and any message errors.

Control and Message Processing

When Core1553BRM is configured as an RT, its configuration data is stored in registers, and commands and data are stored in external memory. Details of the memory structure are discussed in this section; the control registers are described both here and in "Registers" on page 53.

Control

Control of the core operating as an RT is accomplished through the use of control words stored in descriptor blocks, and mode codes and subaddresses sent in 1553 messages. Control word information allows the core to generate interrupts, buffer messages, and control message processing. Moreover, the descriptor block contains pointers to data buffers where mode codes and subaddresses to be used by the host or subsystem in further message processing are stored.

For receive commands, the core processes each incoming message for correct format, word count, and contiguous data. If a message error is detected, the core will stop processing the remainder of the message, suppress status word transmission, and set the message error bit (ME, bit 9) of the status word. The core will track the message until the end of the message is detected.

During RT-to-RT transfers, the core will ensure that the terminal address in the incoming status message matches that of the transmitting RT as specified in the command word. The core will set the message error bit in the MIW and prevent transmission of the status word in the case of a mismatch.

Core Interface

The core communicates with the 1553 bus through dual Manchester II encoders/decoders. These encoders/decoders electrically interface with the bus via 1553 bus transceivers. The core receives all message traffic from the bus via either the primary or secondary decoder. Each decoder checks the incoming signal for the proper sync pulse and Manchester waveform, edge skew, number of bits, and parity.

During transmission, the encoded (transmitted) word is repeated back through the core's decoder (loopback). This allows the core to constantly monitor transmissions for possible encoder errors. Should the encoder word and reflected word not match, the WRAPF bit (register 6, bit 14) is set and INTOUTH is generated (if enabled).

In addition to the loopback compare test, the core will terminate a transmission greater than 800 μs by the assertion of Fail-Safe Timer (TIMERONAn or TIMERONBn). This timer is reset upon receipt of another command.

Remote Terminal Address

When the core is operating as an RT, the terminal address can be set one of two ways: via input signals RTADDRIN[4:0] and RTADDRPIN or through the Operation and Status register (register 1, bits 15:10). In all cases, the terminal address must have odd parity, which can be achieved by correctly setting the input signal RTADDRPIN or bit RTPTY (register 1, bit 10). If parity is not correct, the core will set TAPF (register 1, bit 2). TAPF will be valid after the rising edge of RSTINn.

If the terminal address and parity are set from external signals, taking RSTINn LOW will latch the address set. A new address will not be latched until RSTINn is taken HIGH for a minimum of one clock cycle and then LOW with LOCKn set LOW. The core will check the terminal address and parity at power-up.

If the terminal address and parity are set via register 1, bits 15:10 (with LOCKn set HIGH), the core will load the address and check parity once the register 1 write is complete.

Note: Setting BCEN (register 0, bit 4) LOW reserves address 31 (11111b) for use as an RT address.

Reset

The core can be reset in one of three ways:

- Input signal RSTINn
- Via the host or subsystem using SRST (register 0, bit 13)
- Through a 1553 message using Reset Remote Terminal (mode code 01000b).

Using SRST will act as a master reset of the core and will terminate current command processing. This reset will occur immediately. The core must then be reinitialized by the host or subsystem.

If mode code Reset Remote Terminal is used, the core will partially reset after transmission of a status word. During this reset, the encoders/decoders will be cleared, Time Tag will be reset, both busses will be enabled, and the Terminal Flag

will be enabled. The core will remain configured and continue to operate as an RT. The CPU interface registers are not reset by the Reset Remote Terminal mode code.

Registers

The functionality of the core, as well as its specific responses to 1553 events, is controlled though registers. In addition to the seven control registers common to all core implementations, Core1553BRM, when implementing an RT, has 18 registers used to control its functions. Table 6-1 shows which bits of the 18 control registers are used by the core in RT mode. See "Core1553BRM Registers" on page 71 for detailed register usage information.

Table 6-1 · Register/Bit Applicability Map for Core1553BRM as Remote Terminal

Register Address	Name	Bit Locations															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00	Control	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
01	Operation and Status	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
02	Current Command	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
03	Interrupt Mask	✓	✓	✓	✓	✓	✓	✓	✓	✓							
04	Pending Interrupt	✓	✓	✓	✓	✓	✓	✓	✓	✓							
05	Interrupt Pointer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
06	Built-In Test Register	✓	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓	✓	✓
07	Time Tag	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
08	Command Block Pointer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
09	Status Word	✓						✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
16–31	RT Illegalization Registers	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
32	Enhanced Features													✓	✓	✓	✓

Memory Structure

The host or subsystem controlling the core must allocate 512 consecutive words of memory for storing the subaddress and mode code descriptor tables (Figure 6-1). The descriptor table is composed of multiple descriptor blocks (each descriptor block is four words). The top of the descriptor table can be set at any address location within the system memory. The descriptor table space is defined and initialized by the host, with the starting address defined by the Descriptor Table Pointer (register 8).
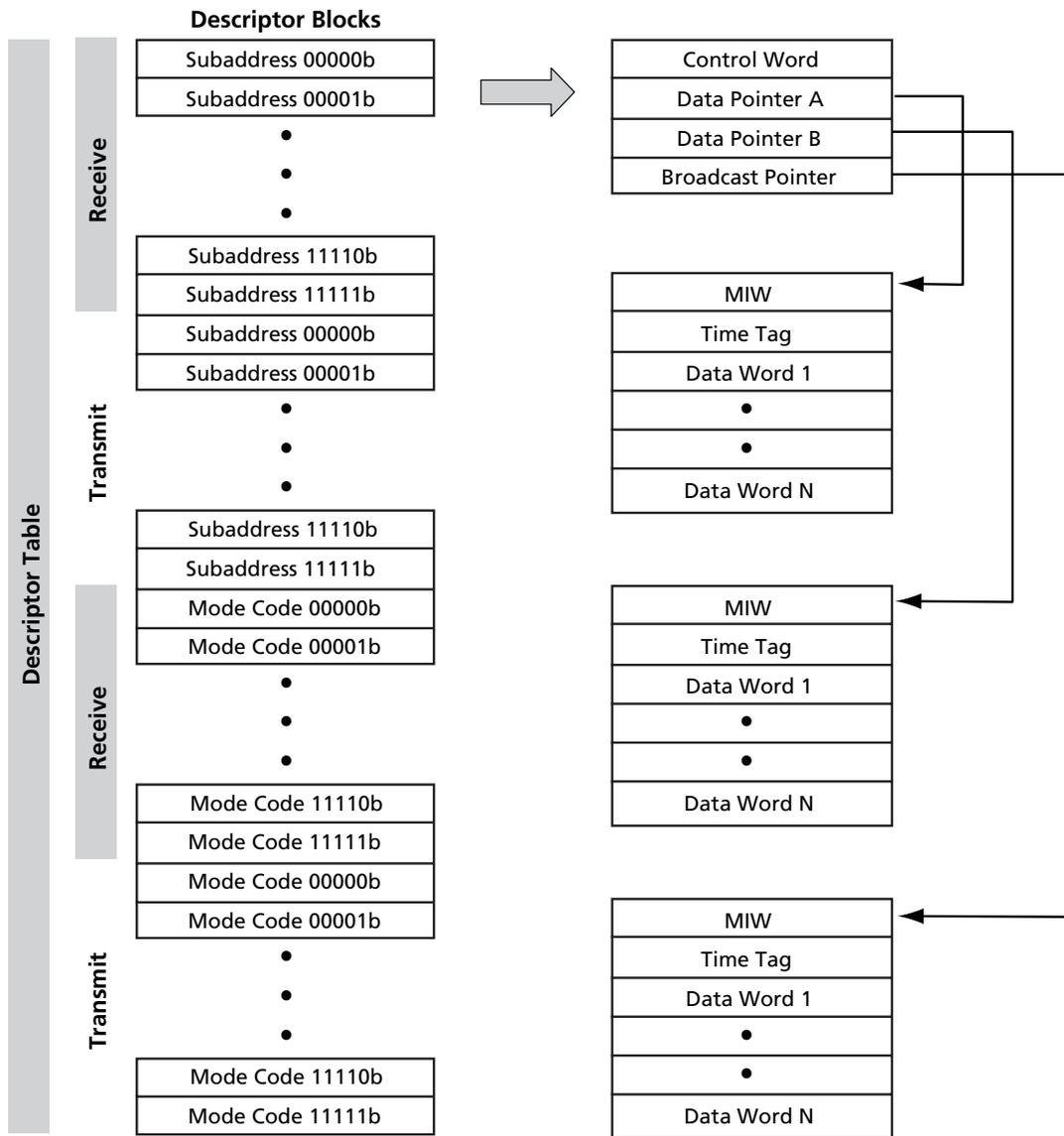

Figure 6-1 · Remote Terminal Memory Map

Descriptor blocks are stored sequentially in the descriptor table space starting with the receive subaddress descriptor blocks and followed by the transmit subaddress descriptor blocks, receive mode code, and transmit descriptor blocks.

Table 6-2 and Table 6-3 on page 59 give the starting address offset for each descriptor block, starting from the address location specified in the Descriptor Table Pointer register.

Table 6-2 · Descriptor Block Starting Addresses

Descriptor	T/R Bit	Starting Address Offset
Subaddress	0	(Subaddress No. × 4) + 0
Subaddress	1	(Subaddress No. × 4) + 128
Mode Code	0	(Mode Code No. × 4) + 256
Mode Code	1	(Mode Code No. × 4) + 384

Descriptor Blocks

To process messages, the core accesses a descriptor block at the beginning and end of command processing. A descriptor block is composed of four words:

- Control Word
- Data Pointer A
- Data Pointer B
- Broadcast Data Pointer

A unique descriptor block is assigned for each subaddress and mode code used for both receive and transmit commands. Collectively, all of the descriptor blocks are referred to as the Descriptor Table. The contents and configuration of the Descriptor Table are controlled and entered into memory by the host or subsystem.

The Control Word contains information to allow the core to generate interrupts, buffer messages, and control message processing.

Data pointers give the starting address where data is stored for receive commands, or read from for transmit commands. The core will store data sequentially starting from the top of the data buffer with a two-address location offset. This two-address offset is to allow space for the MIW (top word location) and the Time Tag (second location).

The Broadcast Data Pointer allows for the segregation of broadcast from non-broadcast data storage. The host or subsystem can control this feature via NII (Control Word register, bit 0). If data segregation is not enabled, the broadcast data is stored starting at the appropriate data pointer location.

Note: Broadcast data segregation applies only to receive commands.

During command processing, the core reads the descriptor block after assertion of ACTIVE. The core then arbitrates for the memory bus and then reads the Control Word and the three data pointers. After reading the descriptor block, the core surrenders control of the bus (negate MEMACCn).

Next, the core starts the acquisition of data words for either transmission or storage. The core begins command post-processing once data acquisition is complete.

During command post-processing, the core again arbitrates for the memory bus. During the descriptor update, the core does the following:

- Modifies the Control Word index field and bits 4, 2, and 1, if required
- Updates Data Pointer A if no message errors occurred during the message transaction (the Broadcast Data Pointer is updated if no errors occurred during broadcast message reception)

None of the data pointers will be updated if the core has ping pong mode enabled. See "Ping Pong Buffer Operation" on page 61 for more details.

Note: An optional interrupt log entry is performed after a descriptor update.

Control Word

The Control Word (Figure 6-2) is used by the core in message processing and is initialized by the host or subsystem. The core updates the Control Word during command post-processing to provide the host or subsystem details about the transaction.
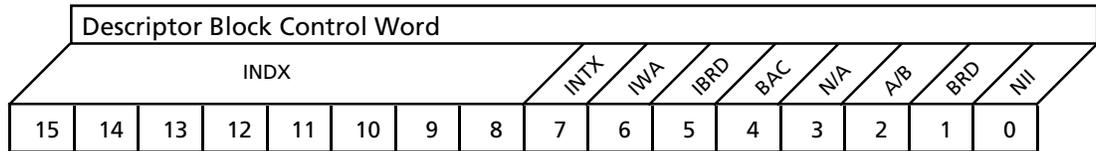
Figure 6-2 · Descriptor Block Control Words

Bits 15:8 – INDX (Index)

Received message processing: The INDX bits define the depth of the core's multiple-message buffer. The value can range from 0 to 255. As the core does not buffer messages in ping pong mode, the setting of INDX is invalid and should be initialized to 0. Each time a message is transacted with no errors (and the particular mode code or subaddress has not been illegalized), the value of INDX is decremented by 1. If enabled by INTX, the core will generate an interrupt as INDX is decremented from 1 to 0.

Transmit message processing: Not used.

Bit 7 – INTX (Interrupt Equals Zero)

Received message processing: If set HIGH, the core will generate an interrupt as INDX is decremented from 1 to 0. The interrupt will be entered into the Pending Interrupt register (register 4) if not masked. The output signal INTOUTM will be taken HIGH after message processing.

Transmit message processing: Not used.

Bit 6 – IWA (Interrupt when Accessed)

Setting this bit enables an interrupt when the core receives a valid subaddress or mode code command. The interrupt will be entered into the Pending Interrupt register (register 4) if not masked. The output signal INTOUTM will be taken HIGH after message processing.

Bit 5 – IBRD (Interrupt Broadcast Received)

Setting this bit enables an interrupt when the core receives a valid subaddress or mode code broadcast command. The interrupt will be entered into the Pending Interrupt register (register 4) if not masked. The output signal INTOUTM will be taken HIGH after message processing.

Bit 4 – BAC (Block Accessed)

The core will set BAC at the end of message processing to indicate processing status to the host or subsystem. The host or subsystem must initialize this bit to 0.

Bit 3 – LA/B (Last A or B Buffer)

In enhanced mode, indicates which buffer was last used (see "Bus Controller GOTO Enhancements" on page 89).

Bit 2 – A/B (A or B Buffer)

If buffer ping pong is enabled, the host can set this bit to indicate which buffer is the primary; otherwise, A/B is not used. A logic 1 designates buffer A as the primary; logic 0 designates buffer B.

Bit 1 – BRD (Broadcast)

The core sets this bit to indicate reception of a valid broadcast command.

Bit 0 – NII (Notice II)

Received message processing: If set HIGH, NII enables data segregation of broadcast and non-broadcast data by enabling the use of a Broadcast Data Pointer. If set LOW, broadcast data is stored using Data Pointer A.

Transmit message processing: Not used.

Data Pointer A and B

Both data pointers A and B (Figure 6-3) contain the starting address for the storage or retrieval of message data words. At the top of each data buffer is the message information word and the Time Tag. Actual data words are stored sequentially after the MIW and Time Tag. The data pointers point to the location of the MIW and not to where the data words are stored.

In index mode, Data Pointer A is read by the core and used to initialize an internal counter that is incremented after the receipt of each new data word. During post-processing, the core will update Data Pointer A to the next MIW unit the Control Word index decrements to 0.

In non-index mode, the core sequentially stores or retrieves data words starting at the Data Pointer A address plus a two-location offset. Data Pointer A is never updated during post-processing. Non-index mode can also be used with ping pong buffer mode, where data is stored or retrieved alternately from Data Buffer A or B (indicated by Data Pointer A and B, respectively).
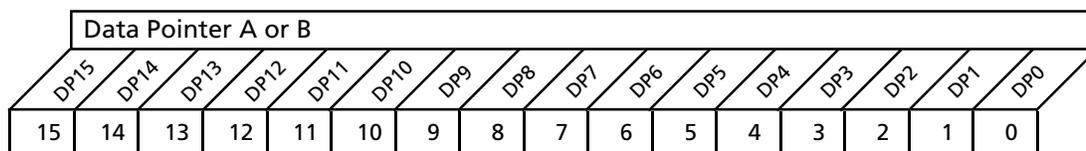
Figure 6-3 · Data Pointer A or B

Bits 15:0 – DP[15:0] (Data Pointer)

These bits contain the starting address of either Data Buffer A or B, depending on its location in the descriptor block.

Broadcast Data Pointer

If broadcast data segregation is selected (NII set HIGH), the broadcast data pointer operation (Figure 6-4) is identical to that of either Data Pointer A or B.
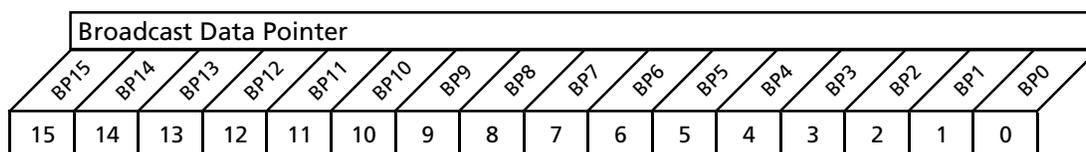
Figure 6-4 · Broadcast Data Pointer

Note: When a broadcast command is followed by a Transmit Last command or Transmit Status Word mode code, the core will transmit a status word with bit 15 of the status word (Broadcast Command Received) set to a logic 1. The Broadcast Command Received bit is cleared by reception of the next valid non-broadcast command.

Bits 15:0 – BP[15:0] (Broadcast Data Pointer)

These bits contain the starting address of the Broadcast Data Buffer, if broadcast data segregation has been enabled.

Data Buffer Structure

Each data buffer, whether data buffer A, B, or broadcast data, is composed of three parts. In the first address location is the MIW. In the second address location is the Time Tag. Up to 32 data words are located in the third and higher locations, consecutively. Each buffer can be located anywhere in memory, but the MIW, Time Tag, and data words must be in consecutive locations. In case of an error condition, all data words in the buffer are considered invalid.

Note: All data pointers have the address for the location of the MIW of that buffer (i.e., a data pointer indicates the location of the MIW, not the data words).

Message Information Word

The MIW (Figure 6-5) contains information on the received or transmitted command: word count, mode codes, status info, and any error conditions.

Message Information Word															
WC/MC4	WC/MC3	WC/MC2	WC/MC1	WC/MC0	N/A	CHA/B	RTRT	ME	N/A	N/A	ILL	TO	OVR	PRTY	MAN
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
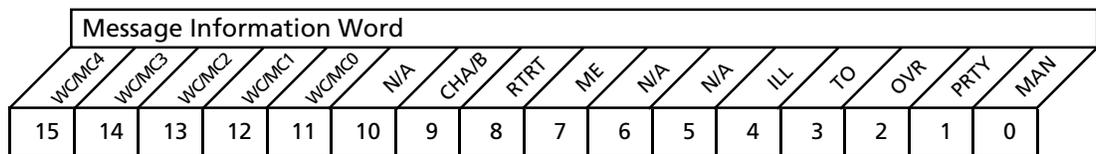
Figure 6-5 · Message Information Word

Bits 15:11 – WC/MC[4:0] (Word Count / Mode Code)

For subaddresses, WC contains the word count of the received or transmitted command. For mode codes, MC contains the receive or transmit mode code. In both cases, this info comes from bits 15:10 of the 1553 command word.

Bit 10

Not used.

Bit 9 – CHA/B (Bus A or B)

CHA/B set HIGH indicates that the message was received on bus A; LOW indicates bus B.

Bit 8 – RTRT (RT-to-RT Transfer)

Receive only: RTRT set HIGH indicates that the command processed is an RT-to-RT transfer.

Bit 7 – ME (Message Error)

This bit set HIGH indicates that an error condition was encounter during processing. Bits 4:0 give details of the error.

Bit 5 – BC (Broadcast)

Bit 6

Not used.

Bit 4 – ILL (Illegal Command)

ILL set indicates that the error was an illegal received command.

Bit 3 – TO (Timeout)

Receive only: This bit set indicates that the number of words received was less than that specified by the word count or mode code.

Bit 2 – OVR (Overrun)

OVR set indicates that the core received either too many words or a data word when none was expected (e.g., a data word with a transmit command).

Bit 1 – PRTY (Parity)

Receive only: PRTY set indicates that the core encountered a parity error in the received data words.

Bit 0 – MAN (Manchester)

Receive only: This bit set HIGH indicates that the core encountered a Manchester decoding error in the received data words.

Time Tag

The Time Tag field is set to the value of the internal timer (register 7) when the 1553 command word has been received and validated.

Mode Codes

Mode codes allow the BC to communicate commands to the RT. Mode codes may or may not have an associated data word (mode codes for MIL-STD-1553A are defined without a data word).

For all mode codes, the command word is stored within the RT protocol controller and can be accessed via register 2 (except mode code 10010b, Transmit Last Command), and a status word is transmitted. Table 6-3 lists all the mode codes available for use with Core1553BRM. All mode codes can be legalized or illegalized using the RT legalization registers (registers 16 to 31).

Table 6-3 · Mode Codes

Mode Code	Function	T/R Bit	Data Word Stored	Data Word Transmitted	Additional Operation
00000:01111	Undefined (without data)	0			
10000	Undefined (with data)	0	✓		
10001	Synchronize (with data)	0	✓		Time Tag counter load with data word value
10010:10011	Undefined	0	✓		
10100	Selected Transmitter Shutdown	0	✓		
10101	Override Selected Transmitter Shutdown	0	✓		
10110:11111	Reserved	0	✓		
00000	Dynamic Bus Control	1			Dynamic Bus Acceptance bit set in outgoing status word if enabled in the Control Register
00001	Synchronize	1			Time Tag counter reset to zero
00010	Transmit Status Word	1			Status word cleared after master reset; core updates status word if illegalized.
00011	Initiate Self-Test	1			This mode code is ignored by Core1553BRM.

Table 6-3 · Mode Codes (continued)

Mode Code	Function	T/R Bit	Data Word Stored	Data Word Transmitted	Additional Operation
00100	Transmitter Shutdown	1			Alternate bus disabled
00101	Override Transmitter Shutdown	1			Alternate bus disabled (if enabled in Control register); Reset Remote Terminal mode code clears the transmitter shutdown.
00110	Inhibit Terminal Flag Bit	1			Terminal flag bit set to zero and assertion disabled
00111	Override Inhibit Terminal Flag Bit	1			Terminal flag bit enabled for assertion
01000	Reset Remote Terminal	1			Core reset
01001:01111	Reserved	1			
10000	Transmit Vector Word	1		✓	Service Request bit in status word set; SRQ (register 9, bit 8) is cleared.
10001	Reserved	1		✓	
10010	Transmit Last Command	1	✓		Command word not stored; last command word transmitted; transmitted data word is zero after reset. Note: The core will store this mode code if legalized and will update the status word.
10011	Transmit BIT Word	1	✓	✓	The core will transmit the Core1553BRM BIT word (register 6).
10100:10101	Undefined (with data)	1		✓	
10110:11111	Reserved	1		✓	

Data Buffer Operation

As stated earlier, at the top of each data buffer is the MIW and the Time Tag. Actual data words are normally stored sequentially after the MIW and Time Tag. There are several possible schemes for data buffering when the core is operating in RT, indexed, ping pong, or circular mode.

Indexed

In indexed mode, received data is written to the buffer sequentially. Data Pointer A sets the start of the buffer. The MIW, Time Tag, and data words are written sequentially into memory. At the end of every received message, Data Pointer A is updated to point to the next memory location, and the INDX value in the subaddress Control Word is decremented. When the INDX field transitions from 1 to 0, an interrupt is generated. Thus, the host must allocate the correct amount of memory and set the initial INDX value correctly. If the INDX value is set to 10, at least 340 words of memory should be allocated (each message can contain an MIW, Time Tag, and 32 data words)

When in indexed mode, transmit data is always transmitted from the location pointed to by Data Buffer A plus two. The first two locations contain the MIW and Time Tag values.

Ping Pong Buffer Operation

The core architecture supports a dual-buffer operating mode. The core can process messages using the primary buffer while the host or subsystem can use the secondary buffer to store new data for transmission or process previously received data. The core will switch back and forth (ping pong) between the two buffers on a message-by-message basis.

The core will determine the active buffer at the beginning of each message processed. At the end of processing each message, the core will complement invert the PPEN bit to select the alternate buffer on the next message.

For the host or subsystem to effectively use the double-buffering scheme, care needs to be taken that the host or subsystem does not try to access the active buffer currently in use by the core. The host or subsystem can prevent a collision condition by temporarily restricting the core to a single buffer while the host accesses the secondary buffer.

To properly implement buffer servicing while the core is using the ping pong buffering scheme, the host or subsystem needs to do the following:

- Disable the ping pong mode by setting PPEN (register 00, bit X) LOW
- Verify that ping pong mode has been disabled by querying bit 9, MSGTO (Message Timeout)
- Determine the active buffer by querying bit 2, A/B (A or B buffer), of the current descriptor Control Word
- Service the secondary buffer
- Re-enable ping pong mode (setting PPEN HIGH)
- Verify that ping pong mode has been enabled by querying MSGTO

Circular Buffer Operation

To conserve memory, the user has the option of using circular buffers for data storage and retrieval. There are two modes of circular buffer operation, Mode 1 and Mode 2. Mode 1 uses the same structure for data storage as indexed, non-indexed, and ping pong operation, i.e., MIW, Time Tag, and data words are stored in a single buffer. Mode 2 segregates the MIW and Time Tag info into a Message Information Buffer (MIB) and data into a separate data buffer.

Note: Both modes use a custom version of the descriptor block. In addition, ping pong mode is disabled when using circular buffers; bit 2, PPEN (Ping Pong Enable), is ignored.

Mode 1 – Combined Storage

Mode 1 uses the default data buffer structure, i.e., MIW, Time Tag, and data words stored sequentially. However, the descriptor block and Control Word format are altered. The Mode 1 descriptor block's four parts are as follows:

- Control Word
- Buffer Top Address
- Current Address Pointer
- Buffer Bottom Address

The Mode 1 control word is identical to the default Control Word, except that bits 15:8, 2, and 0 (INDX, A/B, and NII) are not used.

The Buffer Top Address is used to define the starting address for the top of the circular buffer, and the Buffer Bottom Address is used to define the bottom of the circular buffer.

The Current Address Pointer is initially set equal to the Buffer Top Address. This pointer indicates the starting address (plus two address locations) for data storage and retrieval. After message processing, the core will write the MIW and Time Tag into the two reserved word spaces above the data words and update the value of the Current Address Pointer to the next available data space. If the Current Address Pointer is greater than the Buffer Bottom Address, it is reset to equal the Buffer Top Address.

Note: If the Current Address Pointer will result in data storage beyond the Buffer Bottom Address, the core will read or write the data beyond the Buffer Bottom Address. This condition needs to be anticipated in allocating the system memory.

The core will generate buffer empty and full flags. When the core reaches the end of the buffer, the core will set bit 7, INTX (Interrupt Enable). When the core starts a new message at the top of the buffer, the core will set bit 8, IXEQ0 (Index Equal Zero Interrupt). Either of these interrupts will be accompanied by the output signal INTOUTH going HIGH.

The core generates a circular buffer empty/full interrupt when the buffer reaches the end (i.e., CA16 greater than BA16) and begins a new message at the top of the buffer. Bit 8 of the Mask register and bit 7 of the Descriptor Control Word mask enable the generation of the full/empty interrupt. On the occurrence of either interrupt, the INTOUTH output asserts.

Mode 2 – Segregated Storage

In Mode 2 operation, message information (MIW and Time Tag) are stored in a MIB separate from the associated data words. Similar to Mode 1, the descriptor block and Control Word format are altered. The Mode 2 descriptor block's four parts are as follows:

- Control Word
- Buffer Top Address
- Current Data Address Pointer
- MIB Base Address and Pointer

The Mode 1 Control Word is identical to the default Control Word, except that bits 15:8 define the MIB length (maximum value is 256) and bits 2 and 0 (A/B and NII) are not used. This allows up to 128 MIW and Time Tag pairs to be stored.

Current Data Address Pointer

The Current Data Address Pointer is initially set equal to the Buffer Top Address. This pointer indicates the starting address (no two-address offset) for data storage and retrieval. After message processing, the core will write the MIW and Time Tag into the MIB and update the value of the Current Data Address Pointer to the next available data space. When the MIB is full, the Current Data Address Pointer is reset to equal the Buffer Top Address (i.e., the data buffer size must be large enough to contain the data from the number of messages allocated to the MIB; it does not have a fixed size).

The MIB Base Address and Pointer

The MIB Base Address and Pointer word defines the base address for the MIB as well as the MIB Current Data Address Pointer (offset) for message information storage. The most significant bits define the base address, and the least significant, the current address pointer. Since the length of the MIB can vary, so can the number of bits used to define both the Base Address and Current Data Address Pointer (Table 6-4 on page 63).

Note: The Current Data Address Pointer must be set on even word boundaries.

Table 6-4 · MIB Base Address and Pointer Format

Length of MIB Buffer	Control Word Bits 15:8	MIB Base Address and Pointer Word	
		Base Address Bits	MIB Pointer Bits
1	01h	Bits 15:1	Bit 0
2	03h	Bits 15:2	Bits 1:0
4	07h	Bits 15:3	Bits 2:0
8	0Fh	Bits 15:4	Bits 3:0
16	1Fh	Bits 15:5	Bits 4:0
32	3Fh	Bits 15:6	Bits 5:0
64	7Fh	Bits 15:7	Bits 6:0
128	FFh	Bits 15:8	Bits 7:0

Note: The host or subsystem can determine the number of messages processed by querying the MIB Current Data Address Pointer.

After message processing, the core will write the MIW and Time Tag into the MIB and update the value of the MIB Current Data Address Pointer to the next available message space. Once the MIB pointer is equal to the MIB length, the core will reset the pointer to zero and set the Current Data Address Pointer equal to the Buffer Start Address.

Flags

Similar to Mode 1, the core will generate buffer empty and full flags. When the core reaches the end of the buffer, the core will, if enabled, generate the IXEQ0 interrupt.

MIL-STD-1553A Operation

Core1553BRM can be configured to operate compliant with MIL-STD-1553A. Taking input signal ABSTD HIGH configures the core for MIL-STD-1553A-compliant operation (taking this signal LOW activates MIL-STD-1553B mode). An alternate method for configuring the core is to use bit 7, A/B STD (1553A or 1553B support).

In addition, setting XMTSW (register 0, bit 0) will enable the core to execute the Transmit Status Word mode code when in MIL-STD-1553A mode.

When configured for MIL-STD-1553A BC operation, the core will do the following:

• Respond with a status word within 7 µs
• Define all mode codes without data
• Ignore the T/R bit setting
• Define subaddress 00000b as a valid mode code—Dynamic Bus Control
• Allow broadcast of all mode codes (except Dynamic Bus Control and Transmit Status Word, if enabled)

Note: When the core is configured for MIL-STD-1553A BC operation, note the following:

• All mode codes use mode code transmit control and information words.
• Only status bits ME and TF are defined; the rest are programmable.
• Both receive and transmits versions of the same mode code need to be legalized.
• The user needs to correctly program the legalization registers for MIL-STD-1553A operation. These registers are initialized for MIL-STD-1553B operation as defined in the datasheet.

7

Core1553BRM Operation as a Bus Monitor

Overview

Core1553BRM can either be synthesized to function as a BM only, or the entire core can be implemented and then configured to operate only as a BM via signals MSELIN[1:0] or MSEL[1:0] (register 1, bits 9:8). See "Register 01 – Operation and Status" on page 74.

Features

Message Information

For every message transacted on the bus, the BM will store a message information word containing general message info and any error codes. (This MIW has a different format than the RT MIW.)

Combined RT/BM Operation

The core can be configured to operate as both an RT and a BM, allowing the core to communicate on and monitor the bus. In this configuration, the BM cannot monitor its own transactions as an RT.

Control and Message Processing

When configured as a BM, Core1553BRM configuration data is stored in registers, and commands and data are stored in external memory. Details of the memory structure are discussed later; the control registers are described both here and in "Registers" on page 66.

As 1553 messages are pulled from the bus, the BM stores the information (command, status, and data locations) in monitor blocks, eight-word locations in memory. Associated data words are also stored in memory. During processing, the core generates a message information word that can give the host detailed information on each message received. The monitor block and data format is similar to the formats used by the bus controller, so it is very simple for the core to switch from BM to BC and then retransmit the messages.

The BM can be configured to monitor-specific terminal addresses. Terminal addresses the BM should monitor are set via registers 14 and 15 (monitor filters A and B).

If the core detects an error in the command word, data word, or RT status, the associated data will not be stored, and the MIW will be updated to reflect the error condition.

If the core is configured to operate as a combined RT/BM, the BM can monitor traffic for a specified terminal address, but not for its own terminal address. Moreover, RT activity takes priority over BM activity. For example, if a message destined for the RT is detected, all BM processing will cease (even mid-message) until the RT has completed message processing. For an RT-to-RT transfer that involves the terminal address of the RT/BM, the RT will process the entire message regardless of which terminal address on the bus has been issued the command first.

Registers

The functionality of the core as well as its specific responses to 1553 events is controlled through registers. In addition to the seven control registers common to all core implementations, Core1553BRM, when implementing a BM, has seven additional registers used to control its functions. Table 7-1 shows which bits of the 14 control registers are used by the core in BM mode. See "Core1553BRM Registers" on page 71 for detailed register usage information.

Table 7-1 · Register/Bit Applicability Map for Core1553BRM as Bus Monitor

Register Address	Name	Bit Locations															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00	Control	✓	✓	✓			✓				✓	✓	✓			✓	
01	Operation and Status							✓	✓	✓	✓			✓			✓	✓
02	Current Command	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
03	Interrupt Mask	✓			✓	✓											✓
04	Pending Interrupt	✓			✓	✓											✓
05	Interrupt Pointer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
06	Built-In Test Register	✓				✓	✓			✓	✓	✓	✓	✓	✓	✓	✓
07	Time Tag	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
11	Monitor Block Pointer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
12	Monitor Data Pointer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
13	Monitor Block Counter	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
14	Monitor Filter A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
15	Monitor Filter B	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
32	Enhanced Features															✓	✓

Memory Structure

The external memory space (up to 64 k words) can be sized and allocated by the user according to the needs of the application. This memory space is needed to hold monitor blocks, data, and the Interrupt Log List. How the memory is allocated is up to the user.

As the number of monitor blocks needed for the application is known (set by register 13, Monitor Block Count), the user can predetermine the space required for their storage. The monitor blocks can be stored in contiguous memory locations for ease of operation.

The starting address is set by register 11, Monitor Command Pointer. As monitor blocks are stored, the value of Monitor Block Count is decremented to 0 to the end of the memory space allocated. When the next monitor block is to be stored, the counter is reset to the initial value and the incoming monitor block is stored in the top location, and the cycle continues.

Each monitor block contains a data pointer to indicate where data for that block is to be stored. Data storage for all monitor blocks starts at the location defined by register 12, Monitor Data Pointer. See "Monitor Blocks" on page 67 for more details.

The Interrupt Log List is a 32-word ring buffer that contains information necessary to service interrupts. The memory space for the Interrupt Log List must be allocated on a 32-word boundary. The starting location for the Interrupt Log List is set by register 5, Interrupt Pointer (see "Interrupts" on page 87 for more details).

Monitor Blocks

As the BM receives each 1553 message for a terminal address to be monitored, information regarding the message is stored in a monitor block (similar in structure to a command block), an eight-word contiguous block. The monitor block's eight contiguous memory locations are one MIW, two command word locations, a data pointer, two status word locations, and a Time Tag. The last location is not used (Table 7-2).

Table 7-2 · Monitor Block Structure

Word	Function
1	Message Information Word
2	Command Word 1
3	Command Word 2
4	Data Pointer
5	Status Word 1
6	Status Word 1
7	Time Tag
8	Not Used

Message Information Word

Location one of a monitor block contains the MIW (Figure 7-1), which holds information about the type of message stored and any errors. For RT-to-RT transfers, the MIW applies to the complete message.
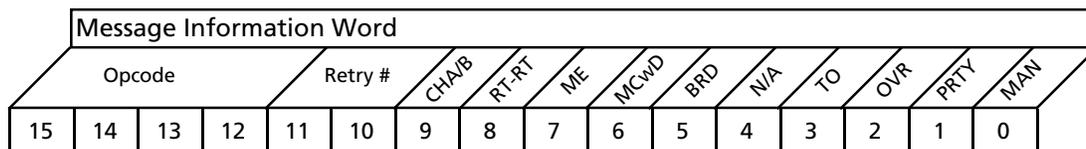
Figure 7-1 · Message Information Word

Bits 15:11 – Opcode

Since the BM must be able to function as a BC, these bits are set to the Execute and Continue opcode (0100b).

Bits 11:10 – Retry # (Retry Number)

Again, since the BM must be able to function as a BC, the Retry # bits are set to 00b.

Bit 9 – CHA/B (Channel A/B)

Setting this bit HIGH indicates that the message was received on bus A; LOW indicates bus B.

Bit 8 – RT–RT (RT-RT Transfer)

The RT–RT bit indicates whether the current message involves an RT-to-RT transfer. This bit will be set if the BM is configured to monitor the receive or transmit terminal address.

Bit 7 – ME (Message Error)

This bit set HIGH indicates that an error condition was encountered during processing. Bits 6:0 give details of the error.

Bit 6 – MCwD (Mode Code without Data)

This bit will be set HIGH to indicate that the core has processed a mode code without an associated data word.

Bit 5 – BRD (Broadcast)

This bit will be set HIGH to indicate that the core has processed a broadcast message.

Bit 4

Reserved.

Bit 3 – TO (Timeout)

This bit set indicates that the number of words monitored was less than that specified by the word count or mode code.

Bit 2 – OVR (Overrun)

OVR set indicates that the core received either too many words or a data word when none was expected (e.g., a data word with a transmit command).

Bit 1 – PRTY (Parity)

PRTY set indicates that the core encountered a parity error in the monitored data words.

Bit 0 – MAN (Manchester)

This bit set HIGH indicates that the core encountered a Manchester decoding error in the monitored data or status words.

Command Words

Located in the second and third memory locations of each monitor block are 1553 Command Words. For most 1553 messages, only the first Command Word contains data. During RT-to-RT transfers, the first Command Word is the Receive command and the second is the Transmit command.

Data Pointer

Located in the fourth memory location of each monitor block is the Data Pointer, indicating the first location in memory where data associated with the Command Word(s) is to be stored. Data is stored contiguously from the Data Pointer location. For RT-to-RT transfers, the pointer is used to store transmitted data.

Status Words

Monitor block words 5 and 6 are for 1553 status word storage. The core will store the RT's responding status after a 1553 command. For an RT-to-RT transfer, the status word from the transmitting RT will be stored in word 5, and the status word from the receiving RT will be stored in word 6.

Time Tag

Word 7 of the monitor block contains the Time Tag for the stored message. The value contains the value of the internal timer (register 7) when the command word is received and validated. It is stored at the end of message processing.

Note: Word 8 of the Monitor Block is not used.

MIL-STD-1553A Operation

Core1553BRM can be configured to operate compliant with MIL-STD-1553A. Taking input signal ABSTD HIGH configures the core for MIL-STD-1553A-compliant operation (taking this signal LOW activates MIL-STD-1553B mode). An alternate method for configuring the core is to use bit 7, A/B STD (1553A or 1553B support). When configured for MIL-STD-1553A BM operation, the core will do the following:

- Expect a response from the RT within 9 μs after a message is sent
- Define all mode codes without data
- Define subaddress 00000b as a valid mode code

8

Core1553BRM Registers

Regardless of whether Core1553BRM is used to implement a BC, RT, BM, or combination of the three, functionality of the core is controlled via register configuration. The CPU interface to the core allows the system CPU to read and write all the control registers. The CPU can directly access the memory connected to the backend interface as well.

The core includes thirty-three 16-bit registers. Of the 33 registers, 17 are used for control functions and 16 for RT command illegalization. Use of the RT command illegalization registers is optional and can be omitted from the core implementation, thus reducing the required logic. Table 8-1 details each of these 33 registers as well as their applicability.

Table 8-1 · Core1553BRM Registers

Register Address	Name	Applicability		
		RT	BC	BM
00	Control	✓	✓	✓
01	Operation and Status	✓	✓	✓
02	Current Command	✓	✓	✓
03	Interrupt Mask	✓	✓	✓
04	Pending Interrupt	✓	✓	✓
05	Interrupt Pointer	✓	✓	✓
06	Built-In Test register	✓	✓	✓
07	Time Tag / Minor Frame Timer	✓	✓	✓
08	Descriptor/Command Block Pointer	✓	✓	
09	1553A/B Status Word	✓		
10	Initialization Count			
11	Monitor Command Pointer			✓
12	Monitor Data Pointer			✓
13	Monitor Block Count			✓
14	Monitor Filter A			✓
15	Monitor Filter B			✓
16–31	RT Legalization Registers	✓		
32	Enhanced Features	✓	✓	✓

At reset, all registers are set to value 0000h, except those registers directly controlled via input signals to the core.

Of the 17 control registers shown in Table 8-1, eight have identical functions in all three core implementations: register addresses 00 through 06 and address 32. These common registers are described below. The remaining registers are covered under the BC, RT, and BM detailed implementation register sections (BC: "Bus Controller–Specific Registers" on page 81, RT: "Remote Terminal–Specific Registers" on page 82, BM: "Bus Monitor–Specific Registers" on page 85).

Common Control Registers

Register 00 – Control

The Control register (Figure 8-1) is used to set core configuration. The STEX bit must be taken LOW prior to writing to this register.

Register 00 – Control															
STEX	SBIT	SRST	BAEN	BBEN	ETCE	MSGTO		BUFM		N/A	BMC	BCEN	DYNBC	PPEN	INTEN	XMTSW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
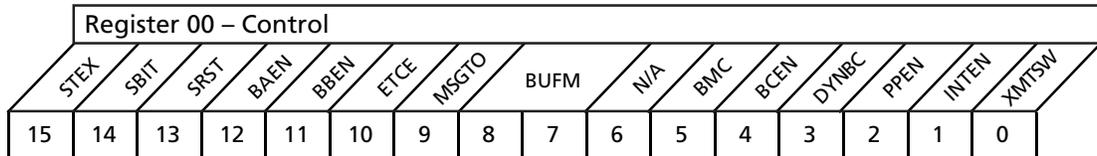
Figure 8-1 · Register 00 – Control

Bit 15 – STEX (Start Execution)

Taking this bit HIGH initiates core operation. If operation is to be halted, STEX can be taken LOW.

BC operation: Taking STEX LOW will halt core operation after completing the current opcode. Prior to halting, the core determines the next command block pointer address and loads the value into register 8. For an EOL command block, register 8 is not updated.

RT operation: An RT address parity error will stop core operation regardless of how this bit is set. If an RT address parity error occurs, register 1, bit 3 (EX) will be set LOW and bit 2 (TAPF) will be set HIGH.

BM operation: Taking STEX LOW will halt core operation after processing the current 1553 message.

Bit 14 – SBIT (Start BIT)

The core does not support BIT. Writing to this bit has no effect.

Bit 13 – SRST (Software Reset)

When SRST is taken HIGH, the core is reset immediately. SRST will clear all internal registers. The core will automatically clear this bit as it resets itself.

Bit 12 – BAEN (Bus A Enable)

RT operation only (ignored by BC and BM implementation): Taking BAEN HIGH enables Bus A. Set LOW, the core will ignore all commands sent over Bus A.

Bit 11 – BBEN (Bus B Enable)

RT operation only (ignored by BC and BM implementation): Taking BBEN HIGH enables Bus B. Set LOW, the core will ignore all commands sent over Bus B.

Bit 10 – ETCE (External Timer Clock Enable)

Assertion of ETCE will force the core to use the external timer clock source.

RT and BM operation: ETCE controls the clock source for the internal Time Tag counter.

BC operation: ETCE controls the clock source used for minor frame timing.

Note: The clock frequency must be set prior to starting core operation.

Bit 9 – MSGTO (Message Timeout)

BC and BM operation: MSGTO sets the RT no response timeout period. During MIL-STD-1553B operation, the programmable timeout occurs at either 14 μs or 30 μs. In MIL-STD-1553A mode, timeout occurs at either 9 μs or 21 μs.

RT operation: When ping pong buffer mode is enabled (bit 2), bit 9 set HIGH serves to acknowledge to the host that ping pong mode has been enabled; set LOW, it acknowledges that this mode has been disabled.

Bits 8:7 – BUFM[1:0] (Buffer Mode)

RT operation only: BUFM sets whether the core will use standard or circular buffer modes (see "Circular Buffers" on page 51 for more details). BUFM bits are set as shown in Table 8-2 (note the reversed bit order):

Table 8-2 · Buffer Modes

BUFM[1:0]	Buffer Operation
00	Non-circular – Mode 0
01	Circular buffer – Mode 1
10	Not used
11	Circular buffer – Mode 2

Bit 6

Not used.

Bit 5 – BMC (Bus Monitor Control)

BM operation only: If BMC is set LOW, the core will monitor all RTs on the bus. If set HIGH, the core will monitor only the RTs specified in Monitor Filter registers 14 and 15.

Bit 4 – BCEN (Broadcast Enable)

Setting BCEN HIGH enables 1553 broadcast mode. Setting BCEN LOW reserves RT address 31 (11111b) for use as an RT address.

Bit 3 – DYNBC (Dynamic Bus Control Acceptance)

RT operation only: Setting DYNBC HIGH allows the core to respond to a Dynamic Bus Control mode code with status word bit 18 set HIGH. When set LOW, the core will not set the Dynamic Bus acceptance bit in the status word.

Bit 2 – PPEN (Ping Pong Enable)

RT operation: If PPEN is set HIGH, ping pong buffer mode is enabled; taking PPEN LOW enables the message indexing features.

BC operation: If PPEN is set HIGH, the core will alternate between Bus A and Bus B on message retries. If set LOW, the core will retry only on the programmed bus as defined in the command block Control Word.

Bit 1 – INEN (Interrupt Log List Enable)

When INEN is set HIGH, interrupt logging is enabled.

Bit 0 – XMTSW (Transmit Word Status)

RT operation only: Setting XMTSW HIGH enables the core to execute the Transmit Status Word mode code when in MIL-STD-1553A mode.

Register 01 – Operation and Status

The Operation and Status register (Figure 8-2) reflects pertinent status information for the core. This register is not cleared on RSTINn but will reflect the actual stimulus applied to input pins RTA[4:0], RTPTY, MSEL[1:0], A/B STD, and LOCKn. Taking LOCKn LOW prevents writes to the remote terminal address, mode selects, and A or B standard bits. When the core is operational (STEX = 1), this register cannot be written.

Register 01 – Operation and Status															
RTA4	RTA3	RTA2	RTA1	RTA0	RTPTY	MSEL1	MSEL0	A/B STD	LOCK	N/A	SSYSF	EX	TAPF	READY	TERACT
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
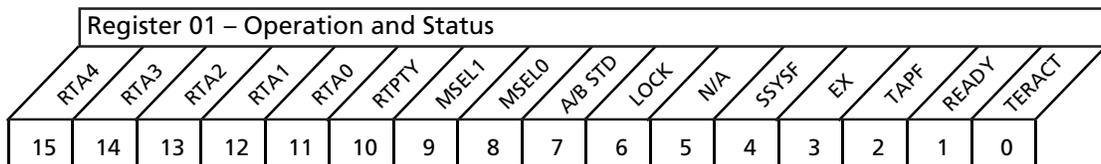
Figure 8-2 · Register 01 – Operation and Status

Bits 15:11 – RTA[4:0] (Remote Terminal Address)

RT operation only: Setting these bits determines the RT address for the core. When LOCKn is active, these bits are read-only.

Bit 10 – RTPTY (RT Address Parity)

RT operation only: This bit is set to provide odd parity for the RT address set in RTA[4:0]—required for proper core operation. When LOCKn is active, this bit is read-only. This bit value is latched on the rising edge of RSTINn.

Bit 9:8 – MSEL[1:0] (Mode Select)

MSEL is used to set the operating mode of the core, BC, RT, BM, or BM/RT. The settings are shown in Table 8-3.

Table 8-3 · Mode Select Settings

MSEL[1:0]	Core1553BRM Operation
00	Bus Controller
01	Remote Terminal
10	Bus Monitor
11	Bus Monitor and Remote Terminal

When LOCKn is active, these bits are read-only. Values written to these bits are latched on the rising edge of RSTINn.

Bit 7 – A/B STD (1553A or 1553B Support)

A/B STD is set LOW for MIL-STD-1553B operation and HIGH for MIL-STD-1553A. When LOCKn is active, this bit is read-only. This bit value is latched on the rising edge of RSTINn.

RT operation only: Setting this bit for MIL-STD-1553A operation also enables the use of XMTSW (register 0, bit 0).

Bit 6 – LOCK (LOCK Status)

LOCK is a read-only bit indicating the inverted status of the input signal LOCKn, i.e., LOCK = 1 when the core is locked and LOCK = 0 when the core is unlocked. This bit value is latched on the rising edge of RSTINn.

Bit 5

Not used.

Bit 4 – SSYSF (SSYSF Status)

RT operation only: SSYSF is a read-only bit indicating the inverted status of the input signal SSYSFn.

Bit 3 – EX (Core Executing)

EX is a read-only bit indicating the operational status of the core: HIGH, the core is executing; LOW, the core is idle.

Bit 2 – TAPF (RT Address Parity Fail)

RT operation only: When this read-only bit is HIGH, it indicates that there is a parity error between bits 15:11 and bit 10 of this same register.

Bit 1 – READY (READY Status)

READY is a read-only bit indicating the inverted status of the output signal READYn. This bit value is cleared at reset.

Bit 0 – TERACT (Terminal Active)

TERACT is a read-only bit indicating the inverted status of the output signal ACTIVE, indicating that the core is currently processing a message. This bit value is cleared at reset.

Register 02 – Current Command

This read-only register, shown in Figure 8-3, contains the current command, either received or transmitted by the core.

Register 02 – Current Command															
CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
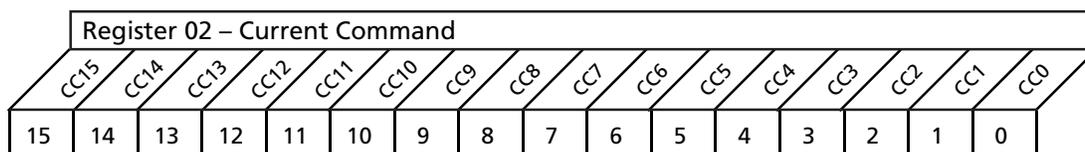
Figure 8-3 · Register 02 – Current Command

Bits 15:0 – CC[15:0] (Current Command)

RT and BM operation: This register contains the last valid command received by the core.

BC operation: When transmission of the command word begins, this register contains the command being transmitted by the core. The value is updated with the execution of each command block. During RT-to-RT transfers, this register will reflect the last valid command being received.

Register 03 – Interrupt Mask

The Core1553BRM architecture allows for the masking of all interrupts. An interrupt is masked if the corresponding bit of this register (Figure 8-4) is set to LOW, allowing the host or subsystem to temporarily disable the service of interrupts. While masked, interrupt notification does not occur. The unmasking of an interrupt after the event occurs does not generate an interrupt for that event. (See "Register 04 – Pending Interrupt" on page 77 for more details on interrupt definitions.)

Register 03 – Interrupt Mask															
DMAF	WRAPF	TAPF	BITF	MERR	SUBAD	BDRCV	IXEQ0	ILLCMD	N/A	EOL	ILLCMD	ILLOP	RTF	CBA	MBC
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
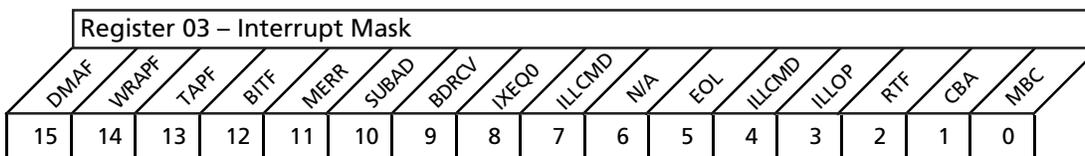
Figure 8-4 · Register 03 – Interrupt Mask

Bit 15 – DMAF (DMA Fail Interrupt)

For all operating modes.

Bit 14 – WRAPF (Wrap Fail Interrupt)

For BC and RT operating modes only.

Bit 13 – TAPF (Terminal Address Parity Fail Interrupt)

For RT operating mode only.

Bit 12 – BITF (BIT Fail Interrupt)

For all operating modes.

Bit 11 – MERR (Message Error Interrupt)

For all operating modes.

Bit 10 – SUBAD (Subaddress Accessed Interrupt)

For RT operating mode only.

Bit 9 – BDRCV (Broadcast Command Received Interrupt)

For RT operating mode only.

Bit 8 – IXEQ0 (Index Equal Zero Interrupt)

For RT operating mode only.

Bit 7 – ILLCMD (Illegal Command Interrupt)

For RT operating mode only.

Bit 6

Not used.

Bit 5 – EOL (End of List Interrupt)

For BC operating mode only.

Bit 4 – ILLCMD (Illogical Command Interrupt)

For BC operating mode only.

Bit 3 – ILLOP (Illogical Opcode Interrupt)

For BC operating mode only.

Bit 2 – RTF (Retry Fail Interrupt)

For BC operating mode only.

Bit 1 – CBA (Command Block Accessed Interrupt)

For BC operating mode only.

Bit 0 – MBC (Monitor Block Counter Interrupt)

For BM operating mode only.

Register 04 – Pending Interrupt

This register (Figure 8-5) identifies interrupt events. The Pending Interrupt register is cleared at the end of a read of or write to any other core register. If a bit in the range 15:12 is set, the signal INTOUTH is driven HIGH. If a bit in the range 11:0 is set, the signal INTOUTM is driven HIGH (see "Interrupts" on page 87 for more details on interrupts).

Register 04 – Pending Interrupt															
DMAF	WRAPF	TAPF	N/A	MERR	SUBAD	BDRCV	IXEQ0	ILLCMD	N/A	EOL	ILLCMD	ILLOP	RTF	CBA	MBC
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 8-5 · Register 04 – Pending Interrupt

Bit 15 – DMAF (DMA Fail Interrupt)

All operating modes: To allow the core to correctly transmit and receive on the 1553 bus, all memory accesses must complete within a specified time. The core datasheet details the memory access requirements. When the core accesses memory, an internal timer is started. If the memory access is not completed by the time the counter decrements to 0, this interrupt is generated. If DMAF occurs, current command processing ends, and the core will remain online.

During RT operation: The current cycle terminates, and the bus is released.

Bit 14 – WRAPF (Wrap Fail Interrupt)

BC and RT operating modes only: The core automatically compares the transmitted word (encoder word) to the reflected decoder word via the continuous loopback feature. If the encoder word and reflected word do not match, the WRAPF bit is set, both here and in Built-In Test (register 6, bit 14).

Bit 13 – TAPF (Terminal Address Parity Fail Interrupt)

RT operating mode only: This bit is set HIGH to indicate an RT address parity error. When a parity error occurs, the core will not begin operation (STEX bit forced to LOW), and Bus A and B are not enabled. The TAPF bit is also set in Built-In Test (register 6, bit 13).

Bit 12

Not used.

Bit 11 – MERR (Message Error Interrupt)

All operating modes: If the core detects errors in Manchester, sync field, word count (too many or too few), MIL-STD-1553 word parity, bit count (too many or too few), or protocol, this bit is set.

During RT operation: This bit is always set when the core asserts bit 9 of the status word (e.g., illegal commands, invalid data word, etc.).

Bit 10 – SUBAD (Subaddress Accessed Interrupt)

RT operating mode only: SUBAD is set when a preselected subaddress has transacted a message. To preselect a subaddress, the IWA bit (bit 6) in the subaddress control word is set. The host must query the interrupt log Interrupt Address Word (IAW) to determine which subaddress generated the interrupt.

Bit 9 – BDRCV (Broadcast Command Received Interrupt)

RT operating mode only: When the core receives a valid broadcast command, BDRCV is set and the core suppresses status word transmission.

Bit 8 – IXEQ0 (Index Equal Zero Interrupt)

RT operating mode only: The core sets IXEQ0 to indicate the completion of a predefined number of commands by the core. This interrupt is generated in indexed mode when the INDX value in the subaddress control word decrements from 1 to 0 or when, in circular buffer mode, a buffer wraps back to the start. When this interrupt occurs, the host or subsystem must update the subaddress descriptor block to prevent potential loss of data.

Bit 7 – ILLCMD (Illegal Command Interrupt)

RT operating mode only: When the core receives an illegal command, ILLCMD is set and responds with a status word only. Bit 9 of the status word is set. A command is determined legal or illegal by the legalization registers (registers 16 to 31).

Bit 6

Not used.

Bit 5 – EOL (End Of List Interrupt)

BC operating mode only: EOL is set when the core reaches the End of List command.

Bit 4 – ILLCMD (Illogical Command Interrupt)

BC operating mode only: The core checks for RT–RT terminal address field match, RT–RT transmit/receive bit mismatch and correct order, and broadcast transmit commands. When such an error is detected, the core sets this bit and will halt execution.

Bit 3 – ILLOP (Illogical Opcode Interrupt)

BC operating mode only: If a reserved opcode occurs in a command block, the core will set this bit and halt operation.

Bit 2 – RTF (Retry Fail Interrupt)

BC operating mode only: The core sets this bit when all programmed retries have failed.

Bit 1 – CBA (Command Block Accessed Interrupt)

BC operating mode only: The core sets CBA when a command block is accessed (opcode 1010b).

Bit 0 – MBC (Monitor Block Counter Interrupt)

BM operating mode only: When the core's monitor block counter reaches 0, MBC is set.

Note: The core does not discriminate between messages with or without errors.

Register 05 – Interrupt Pointer

The Interrupt Pointer register (Figure 8-6) contains the starting base address and pointer location of the Interrupt Log List within the 64 k words of system memory. The Interrupt Log List is a 32-word ring buffer that contains information necessary to service interrupts. The most significant 11 bits designate the base address of the ring buffer (which occurs on a 32-word boundary, i.e., the host must initialize the five least significant bits to 00000b). The core controls the five least significant bits to indicate the pointer location. The host or subsystem reads these five bits to determine the location and number of interrupts within the Interrupt Log List.

| Register 05 – Interrupt Pointer															
INTA15	INTA14	INTA13	INTA12	INTA11	INTA10	INTA9	INTA8	INTA7	INTA6	INTA5	INTA4	INTA3	INTA2	INTA1	INTA0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
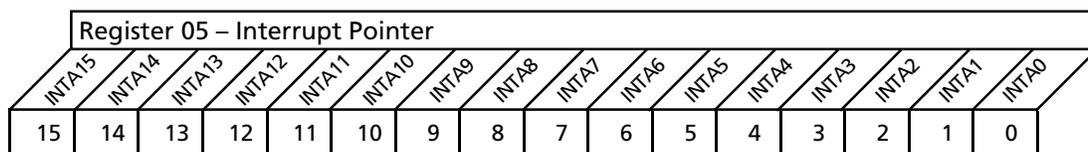
Figure 8-6 · Register 05 – Interrupt Pointer

Bits 15:0 – INTA[15:0] (Interrupt Pointer)

Interrupt Log List base address and location pointer.

Register 06 – Built-In Test Register

The BIT register (Figure 8-7) contains the status of the automatic health monitoring of Core1553BRM. The core does not support the control register BIT function (register 0, bit 14). All of the bits may be set and cleared by the CPU writing to the BIT register.

Register 06 – Built-In Test Register															
DMAF	WRAPF	TAPF	N/A	CHAF	CHBF					N/A					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
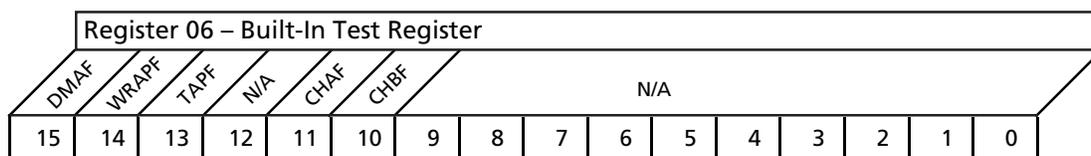
Figure 8-7 · Register 06 – Built-In Test Register

Bits 7:0 are readable and writable by the CPU; their value will be included in the BIT word when it is transmitted by the RT, after power-up reset bits 7:0 are initialized with a version number. Version numbers are provided in the core release notes.

Bit 15 – DMAF (DMA Fail Interrupt)

All operating modes: To allow the core to correctly transmit and receive on the 1553 bus, all memory accesses must complete within a specified time. The core datasheet details the memory access requirements. When the core accesses memory, an internal timer is started. If the memory access is not completed by the time the counter decrements to 0, this interrupt is generated.

Bit 14 – WRAPF (Wrap Fail Interrupt)

BC and RT operating modes only: The core automatically compares the transmitted word (encoder word) to the reflected decoder word via the continuous loopback feature. If the encoder word and reflected word do not match, the WRAPF bit is set.

Bit 13 – TAPF (Terminal Address Parity Fail Interrupt)

RT operating mode only: This bit is set HIGH to indicate an RT address parity error. When a parity error occurs, the core will not begin operation (STEX bit forced to LOW), and Bus A and B are not enabled.

Bit 12

Not used.

Bit 11 – CHAF (Channel A Failure)

CHAF is set when a transmitter timeout occurs on Bus A.

Bit 10 – CHBF (Channel B Failure)

CHBF is set when a transmitter timeout occurs on Bus B.

Register 32 – Enhanced Features Register

This register, shown in Figure 8-8, enables various additional features of Core1553BRM.
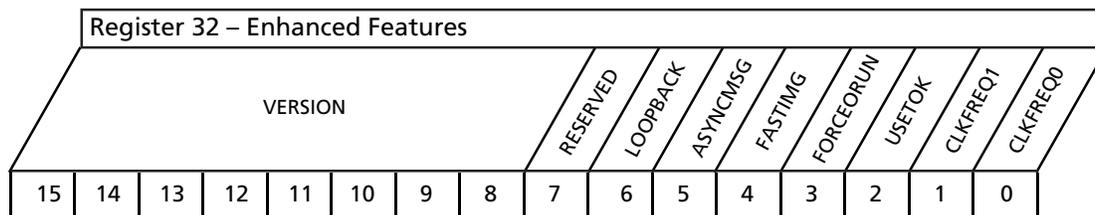
Figure 8-8 · Register 32 – Enhanced Features

Bits 15:8 – VERSION

These bits indicate the version number of the core. The release notes provided with the core detail the values currently in use.

Bits 7 – Reserved

This bit is reserved for controlling possible future enhancements to the core. It may be set LOW, but a HIGH must not be written to ensure compatibility with future core releases. This bit is set LOW at reset.

Bit 6 – LOOPBACK (Loopback Enable)

When set, this bit will loop back the 1553 busses; the receive data input will be connected to the transmit data output, and the external transmit data outputs will be held inactive. If the core is configured as a BC and a broadcast transmit data message is transmitted, the core should transmit and verify its transmissions and report no errors; if a normal transmit data command is used, the core should report a no-response error condition. This bit is LOW at reset.

Bit 5 - ASYNCMSG (Enabled Asynchronous Message)

When set, this bit enables the asynchronous message option on the BC GOTO instruction (see "Bus Controller GOTO Enhancements" on page 89). This bit is LOW at reset.

Bit 4 – FASTIMG (Fast Inter-Message Gap)

BC operation only: When set LOW, the core operates with a minimum inter-message gap of 28 μs. When set HIGH, the minimum inter-message gap is reduced to 6 μs. The inter-message gap may be longer if the backend logic delays asserting the MEMGNTn signal. This bit is set LOW at reset.

Bit 3 – FORCEORUN (Force Overrun)

When set, the core will transmit more than 32 data words (actually the message word count plus 32), causing the internal transmission overrun timer to trigger. This bit is set LOW at reset and should not be set HIGH in normal operation. It is intended to allow the transmission timers to be tested.

Bit 2 – USEXTOK (Use External Verification)

RT operation only: When set LOW, the core uses the internal register settings to verify command words. When set HIGH, the core uses the external command word validation logic input CMDOK.

Bits 1:0 – CLKFREQ (Clock Frequency)

CLKFREQ sets the core operating frequency and can be selected to be 12, 16, 20, or 24 MHz (Table 8-4 on page 81). The reset value of the registers is set by the INITFREQ parameter. If the LOCKFREQ parameter is set, these bits cannot be changed.

Table 8-4 · Clock Frequencies

CLKFREQ[1:0]	Core Operating Frequency
00	12 MHz
01	16 MHz
10	20 MHz
11	24 MHz

Bus Controller–Specific Registers

In addition to the seven common control registers, Core1553BRM, when implementing a BC, has three registers used to control its functions. These are registers 7, 8, and 10.

Register 07 – Minor Frame Timer Register

This read-only register, shown in Figure 8-9, is loaded via the Minor Frame Timer (MFT) opcode (1110b). For user-defined resolution, use TCLK. This register resets to zero any time operation halts.

Register 07 – Minor Frame Timer Register															
MFT15	MFT14	MFT13	MFT12	MFT11	MFT10	MFT9	MFT8	MFT7	MFT6	MFT5	MFT4	MFT3	MFT2	MFT1	MFT0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
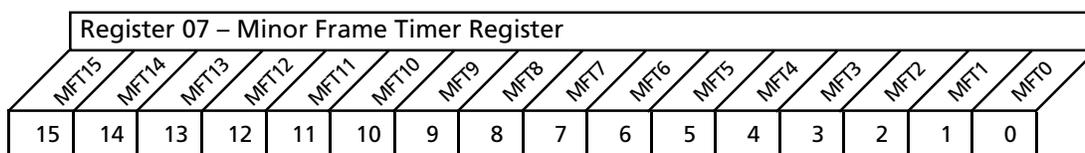
Figure 8-9 · Register 07 – Minor Frame Timer Register

Bits 15:0 – MFT[15:0] (Minor Frame Timer)

These bits contain the value of the Minor Frame Timer.

Register 08 – Command Block Pointer

This register (Figure 8-10) contains the location at which to start the command blocks. After execution begins, this register is automatically updated with the address of the next block.

Register 08 – Command Block Pointer															
CBA15	CBA14	CBA13	CBA12	CBA11	CBA10	CBA9	CBA8	CBA7	CBA6	CBA5	CBA4	CBA3	CBA2	CBA1	CBA0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
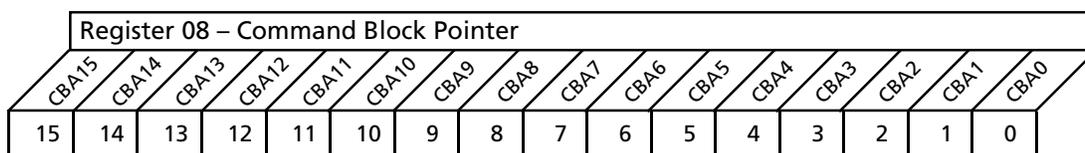
Figure 8-10 · Register 08 – Command Block Pointer

Bits 15:0 – CBA[15:0] (Command Block Address)

These bits contain the starting address of the command block.

Register 10 – Not Implemented

Remote Terminal–Specific Registers

In addition to the seven common control registers, Core1553BRM, when implementing a remote terminal, has three additional registers (7, 8, and 9) used for control and 16 registers (16:31) used for command legalization.

Register 07 – Time Tag Register

This register (Figure 8-11) contains the current value for the 16-bit, free-running counter contained within the core. The default resolution of this timer is 64 ms/bit, or the user can alter this resolution via the input signal TCLK. The timer begins counting on the rising edge of RSTINn or within 64 ms after one of the following events:

• Receipt of a valid Reset Remote Terminal mode code

• Receipt of a valid Synchronize with/without Data mode code

The timer is automatically reset when the core receives a valid Synchronize without Data mode code. If the core receives a valid Synchronize with Data mode code, the Time Tag register is loaded with the associated data. If the core should be halted (STEX = 0), the timer will continue to run. The Time Tag value is captured at command word validation.

Register 07 – Time Tag Register															
TT15	TT14	TT13	TT12	TT11	TT10	TT9	TT8	TT7	TT6	TT5	TT4	TT3	TT2	TT1	TT0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
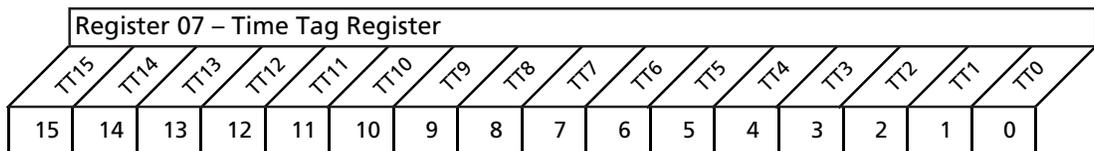
Figure 8-11 · Register 07 – Time Tag Register

Bits 15:0 – TT[15:0] (Time Tag)

These bits contain the value of the Time Tag counter.

Register 08 – Descriptor Pointer

Each 1553B RT has a reserved location in memory for storing information on how to process various subaddresses and mode codes. The memory space is referred to as the Descriptor Table. The Descriptor Pointer register (Figure 8-12) contains the address that points to the top of this reserved memory space.

The core uses the T/R bit, subaddress /mode code field, and mode code to select one block within the Descriptor Table needed for message processing. The value of the Descriptor Pointer Register remains static during message processing.

Register 08 – Descriptor Pointer															
DP15	DP14	DP13	DP12	DP11	DP10	DP9	DP8	DP7	DP6	DP5	DP4	DP3	DP2	DP1	DP0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
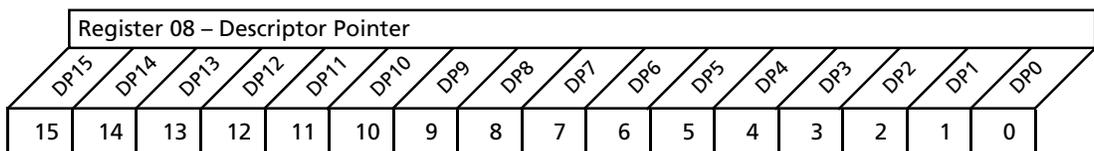
Figure 8-12 · Register 08 – Descriptor Pointer

Bits 15:0 – DP[15:0] (Descriptor Pointer)

These bits contain the value of the Descriptor Pointer.

Register 09 – 1553A/B Status Word Register

For both MIL-STD-1553A and B applications, this register (Figure 8-13 and Figure 8-14) contains the value for the status word. The host or subsystem accesses this register to control the outgoing MIL-STD-1553 status word by setting the various status bits. If the Immediate Clear function is enabled (via IMCLR, bit 15), the status bits are automatically cleared after status word transmission. The Immediate Clear function does not alter the operation of the Transmit Status word and Transmit Last Command Word mode codes.

Register 09 – Status Word Register (for MIL-STD-1553A)															
IMCLR	N/A					SB10	SB11	SB12	SB13	SB14	SB15	SB16	SB17	SB18	SB19
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
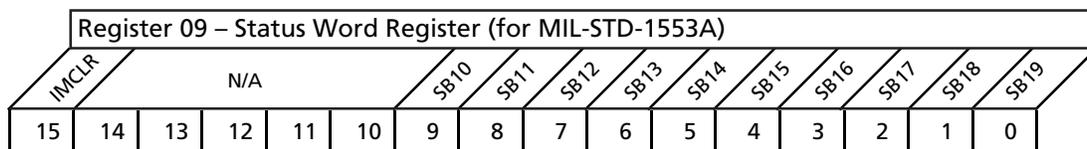
Figure 8-13 · Register 09 – Status Word Register (for MIL-STD-1553A)

Register 09 – Status Word Register (for MIL-STD-1553B)															
IMCLR	N/A					INS	SRQ	N/A				BUSY	SSYSF	N/A	TF
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
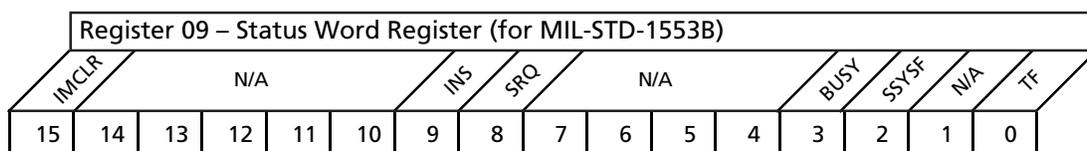
Figure 8-14 · Register 09 – Status Word Register (for MIL-STD-1553B)

Bit 15 – IMCLR (Immediate Clear Enable)

Setting this bit enables the Immediate Clear function, where the INS, BUSY, TF, SRQ, and/or SUBF bits are cleared immediately after a message is completed.

Bits 14:10

Not used.

Bit 9 – INS (Instrumentation)

Setting this bit asserts status word bit 10 (Instrumentation bit).

Bit 8 – SRQ (Service Request)

Setting this bit asserts status word bit 11 (Service Request bit).

Bits 7:4

Not used.

Bit 3 – BUSY (Busy)

Setting this bit asserts status word bit 16 (Busy bit). Setting this bit prevents memory access.

Bit 2 – SSYSF (Subsystem Flag)

Setting this bit asserts status word bit 17 (Subsystem Flag bit). This bit can also be set via the SSYSF input pin.

Bit 1

Not used.

Bit 0 – TF (Terminal Flag)

Setting this bit asserts status word bit 19 (Terminal Flag bit). The Inhibit Terminal Flag mode code prevents host or subsystem assertion.

Bit 15 – IMCLR (Immediate Clear Enable)

Setting this bit enables the Immediate Clear function, where status word bits 19:10 are cleared immediately after a status word transmission.

Note: Exercise caution when using this bit, as once set, it will remain set (Immediate Clear function enabled) until cleared.

Bits 14:10

Not used.

Bits 9:0 – SB(10:19)

Sets 1553A status word bits 10:19.

Register 16:31 – RT Legalization Registers

The core legalization registers are used by the RT to determine which valid, received commands are legal. A command is determined to be illegal if it is supported neither by the standard nor by additional system requirements. Table 8-5 lists the registers used to legalize each set of commands. It also shows the value of the registers after reset. A '1' illegalizes a command and a '0' legalizes a command.

Table 8-5 · Command Illegalization Registers

Register	Function	Reset Value LEGREGS = 1
16	Receive Subaddress 15 to 0	0000
17	Receive Subaddress 31 to 16	0000
18	Transmit Subaddress 15 to 0	0000
19	Transmit Subaddress 31 to 16	0000
20	Broadcast Receive Subaddress 15 to 0	0000
21	Broadcast Receive Subaddress 31 to 16	0000
22	Broadcast Transmit Subaddress 15 to 0	FFFF
23	Broadcast Transmit Subaddress 31 to 16	FFFF
24	Receive Mode Code 15 to 0	FFFF
25	Receive Mode Code 31 to 16	FFFD
26	Transmit Mode Code 15 to 0	FE01
27	Transmit Mode Code 31 to 16	FFF2
28	Broadcast Receive Mode Code 15 to 0	FFFF
29	Broadcast Receive Mode Code 31 to 16	FFFD
30	Broadcast Transmit Mode Code 15 to 0	FE05
31	Broadcast Transmit Mode Code 31 to 16	FFFF

Depending on the core parameter settings used during the design phase, the RT command legalization registers may be handled as follows:

- Implemented in FPGA registers
- Implemented in FPGA memory blocks
- Controlled through the legalization interface using combinatorial logic

When implemented in registers, the values are initialized at reset (external to software) to the values shown in Table 8-5 on page 84. When implemented using memory blocks, these registers are not initialized.

Each command is assigned a specific bit location. For example, the most significant bit of register 16 controls the illegalization of subaddress 15 (01111b), decrementing down to the least significant bit, which controls illegalization of subaddress 0 (00000b). Each bit setting of each register determines whether a specific command is found to be legal or illegal (0 = legal, 1 = illegal).

Bus Monitor–Specific Registers

When Core1553BRM implements bus monitor functions, there are five additional registers used for control (registers 10 through 15). These are in addition to the seven common control registers.

Register 11 – Monitor Command Pointer

The Monitor Command Pointer register (Figure 8-15) contains the starting address for the monitor blocks. This value should not be altered during monitor execution (when EX, register 1, bit 3 is HIGH).

Register 11 – Monitor Comand Pointer															
MCA15	MCA14	MCA13	MCA12	MCA11	MCA10	MCA9	MCA8	MCA7	MCA6	MCA5	MCA4	MCA3	MCA2	MCA1	MCA0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
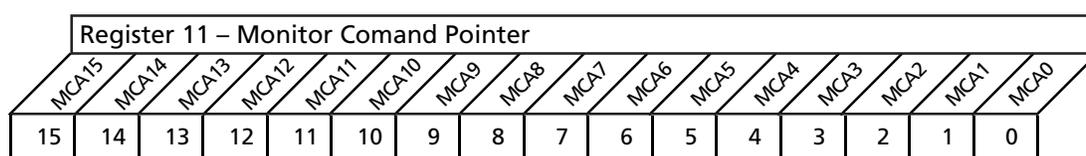
Figure 8-15 · Register 11 – Monitor Command Pointer

Bits 15:0 – MCA[15:0] (Monitor Command Address)

These bits contain the value of the starting address for monitor commands.

Register 12 – Monitor Data Pointer

The Monitor Data Pointer register (Figure 8-16) contains the starting address for the monitor data. This value should not be altered during monitor execution (when EX, register 1, bit 3 is HIGH).

Register 12 – Monitor Data Pointer															
MDA15	MDA14	MDA13	MDA12	MDA11	MDA10	MDA9	MDA8	MDA7	MDA6	MDA5	MDA4	MDA3	MDA2	MDA1	MDA0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
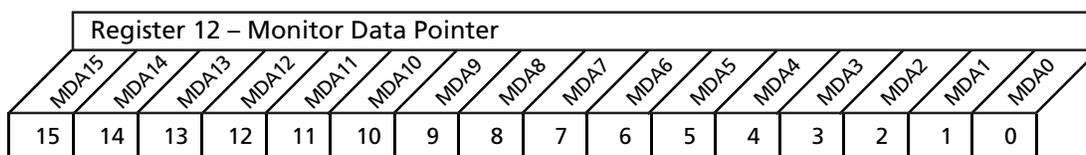
Figure 8-16 · Register 12 – Monitor Data Pointer

Bits 15:0 – MDA[15:0] (Monitor Data Address)

These bits contain the value of the starting address for monitor data.

Register 13 – Monitor Block Count

This register (Figure 8-17) is used to set the number of monitor blocks to be logged. Once execution begins, the value contained in the register will be decremented. Upon reaching 0, an interrupt is generated (MBC—register 4, bit 0). The core will restart at the initial address specified in registers 11 and 12.

Register 13 – Monitor Block Count															
MBC15	MBC14	MBC13	MBC12	MBC11	MBC10	MBC9	MBC8	MBC7	MBC6	MBC5	MBC4	MBC3	MBC2	MBC1	MBC0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
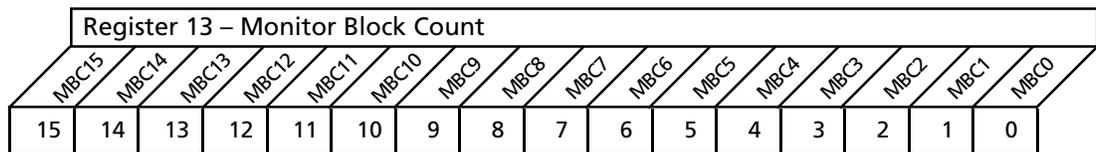
Figure 8-17 · Register 13 – Monitor Block Count

Bits 15:0 – MBC[15:0] (Monitor Block Counter)

These bits contain the value for the number of monitor blocks to be logged.

Register 14 – Monitor Filter A

This register (Figure 8-18) sets which RTs (from the range 31 to 16) will be monitored, indicated by setting the appropriate bit HIGH. Initial value is 0000h.

Register 14 – Monitor Filter A															
MRT31	MRT30	MRT29	MRT28	MRT27	MRT26	MR25	MR24	MR23	MR22	MRT21	MR20	MRT19	MRT18	MRT17	MRT16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
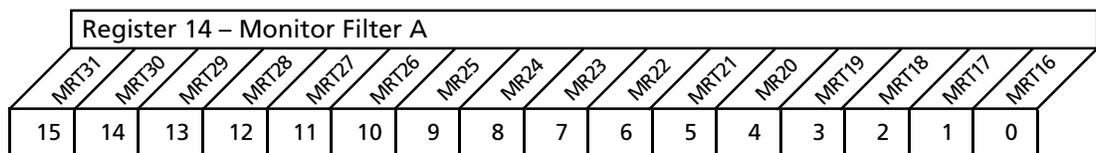
Figure 8-18 · Register 14 – Monitor Filter A

Bits 15:0 – MRT[31:16] (Monitor RT)

These bits select the RTs that should be monitored by the core.

Register 15 – Monitor Filter B

This register (Figure 8-19) sets which RTs (from the range 15 to 0) will be monitored, indicated by setting the appropriate bit HIGH. Initial value is 0000h.

Register 15 – Monitor Filter B															
MRT15	MRT14	MRT13	MRT12	MRT11	MRT10	MRT9	MRT8	MRT7	MRT6	MRT5	MRT4	MRT3	MRT2	MRT1	MRT0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
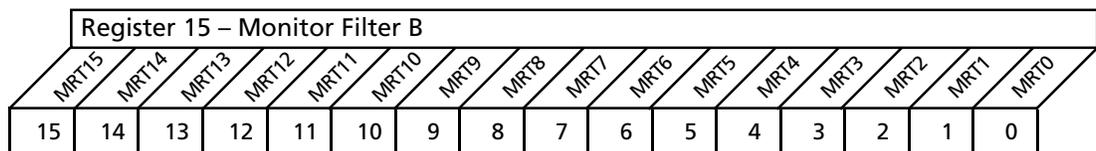
Figure 8-19 · Register 15 – Monitor Filter B

Bits 15:0 – MRT[15:0] (Monitor RT)

These bits select which RTs should be monitored by the core.

Interrupts

Core1553BRM incorporates an interrupt system to allow the host or subsystem to correctly identify the type of interrupt that has occurred and determine its cause. Interrupts are broken into two classes: hardware and message interrupts. Hardware interrupts need to be serviced as soon as they occur, whereas message interrupts are stored for later investigation.

All interrupts are stored in register 4, Pending Interrupt, depending on the settings of register 3, Interrupt Mask. The most significant four bits are classed as hardware interrupts, the lower bits as message interrupts.

When a hardware interrupt occurs, the core will set the appropriate bit in register 4 and alert the host or subsystem via INTOUTH. When alerted, the host or subsystem should service the interrupt immediately, as hardware interrupts are not stored by the core, and until the interrupt is cleared no further hardware interrupts will be signaled.

When a message interrupt occurs, the core will set the appropriate bit in register 4 and alert the host or subsystem via INTOUTM. If enabled, these interrupts are also stored in the Interrupt Log List, a 32-word ring buffer. Each interrupt is stored using two words of information:

- Interrupt Information Word (IIW) – a 16-bit word with format identical to that of register 4 (with the four most significant bits masked)
- Interrupt Address Word (IAW) – a 16-bit word that identifies the source of the interrupt (content varies with core configuration)

With each message interrupt, the system will store the interrupt in the Interrupt Log List based on the setting of register 5, Interrupt Pointer, with the first IIW stored at address offset 00000b, the first IAW stored at address offset 00001b, and so on until the buffer wraps while storing the 17th message interrupt (the core updates the value of the least-significant five bits of register 5 while storing each interrupt word).

IAW format depends upon how the core is configured for operation. When operating as a BC, the IAW contains the location of the command block being processed when the interrupt occurred. When the core is configured as an RT, the IAW contains the subaddress or mode code descriptor that generated the interrupt. During BM operation, the IAW contains the current command block being processed. (This behavior is identical to the SuMMIT device.)

Note: When the core is configured to operate as a combined RT/BM, the host must determine which operating mode generated the interrupt. Determination can be done by examining the IIW or by decoding the IAW address to see whether the address matches an RT descriptor block or a monitor command block.

9

Enhanced Operation

Bus Controller GOTO Enhancements

The Call and GOTO instructions have been enhanced to support asynchronous message operation. This feature is enabled when bit 5 of the Enhanced Features register (address 32) is set. When enabled, bits 11 and 10 in the bus controller control word affect the Call and GOTO instructions as shown in Table 9-1.

This allows the CPU to initially create a message frame that repeats and does not execute the Call/GOTO instruction (bits 11:10 = '11'). While the frame is active, the CPU can then set the bits to '10'. The next time the core executes the instruction, it will perform the Call/GOTO instruction and, on completion, modify the two bits to '11' again, preventing the Call/GOTO from being repeated.

Table 9-1 · Effect of Bits 11 and 10 on Call and GOTO Instructions

Bit	Name	Function
11	ENABLE_ASYNC	0: The instruction will be executed as normal. 1: The instruction is only executed when bit 10 is set to 0.
10	DONE_ASYNC	0: Asynchronous message not yet done 1: Asynchronous message done

Remote Terminal Ping Pong Operation

In addition to setting bit 2 in the RT control word (refer to "Control Word" on page 56) to indicate which buffer will be used next, Core1553BRM also sets bit 3 to indicate the last buffer used.

- Bit 2: A/B – Indicates the next buffer that is about to be used
- Bit 3: LA/B – Indicates the last buffer that was used

When ping pong is on, the A/B and LA/B bits are normally the inverse of each other. Should ping pong be disabled when a message is received, the core will not ping pong, and the two bits will be the same, indicating that the next received message was placed in the same buffer. This can occur if a second message is received while the host has disabled ping pong to service the previous message. The additional LA/B bit allows this case to be detected.

Table 9-2 gives the significance of these two bits as regards the buffers.

Table 9-2 · Significance of Bits 2 and 3 of RT Descriptor Word

LA/B	A/B	State of Buffers
0	0	Last data was placed in Buffer A; next data will go in A.
0	1	Last data was placed in Buffer A; next data will go in B.
1	0	Last data was placed in Buffer B; next data will go in A.
1	1	Last data was placed in Buffer B; next data will go in B.

Memory Access Sequence

The protocol controller state machine within Core1553BRM accesses memory depending on its operational mode and 1553 activity. The actual sequence of operations is very complex. Figure 9-1 and Figure 9-2 on page 91 show the sequence of operations for a two-word RT–BC transfer followed by a two-word BC–RT transfer, for the three possible core operating modes.

1553B Activity		Bus Controller Memory Activity	Remote Terminal Memory Activity	Monitor Terminal Memory Activity
		Read OpCode Read CW Read DPTR		
CW				
			Read DPTR0 Read DPTR1 Read DPTR2 Read DPTR3 Read Data 1	Write DPTR
SW			Read Data 2	
		Write SW		Write SW
DW 1				
		Write Data 1		Write Data 1
DW 2				
		Write Data 2	Write MINFO	Write Data 2

Figure 9-1 · Memory Access Sequence

1553B Activity		Bus Controller Memory Activity	Remote Terminal Memory Activity	Monitor Terminal Memory Activity
		Write OpCode Status Write IIW Write IAW Read OpCode Read CW	Write Time Tag* Write MIW Write DPTR Write IIW Write IAW	Write Message Information Write Command Word Write Time Tag Write IIW Write IAW
CW		Read DPTR Read Data 1		
DW 1		Read Data 2	Read DPTR0 Read DPTR1 Read DPTR2 Read DPTR3	Write DPTR
DW 2			Write Data 1	Write Data 1
			Write Data 2	Write Data 2
SW				
		Write SW Write OpCode Status Write IIW Write IAW Read OpCode	Write MINFO Write Time Tag Write MIW Write DPTR Write IIW Write IAW	Write SW Write Message Information Write Command Word Write Time Tag Write IIW Write IAW

Note: *The Time Tag for both RT and MT mode is written at the end of message processing. The Time Tag written is the value of the Time Tag at the end of the 1553B command word, not the value when it is actually written.

Figure 9-2 · Memory Access Sequence (continued)

10

Testbench Operation and Modification

Three testbenches are provided with Core1553BRM:

- **Verification** – A complex testbench that verifies core operation. This testbench exercises all the features of the core. It is not recommended that this testbench be modified.
- **VHDL User** – A simple-to-use testbench written in VHDL and intended for customer modification
- **Verilog User** – A simple-to-use testbench written in Verilog and intended for customer modification

Verification Testbench

Actel has developed a 1553 verification testbench that you can use to verify the core performance per the 1553 specification. The testbench is coded in VHDL and includes several Core1553BRM cores connected to a 1553 bus and backend interfaces. A procedural testbench controls the various blocks and implements the tests (Figure 10-1). The source code is not made available with Obfuscated core licenses.
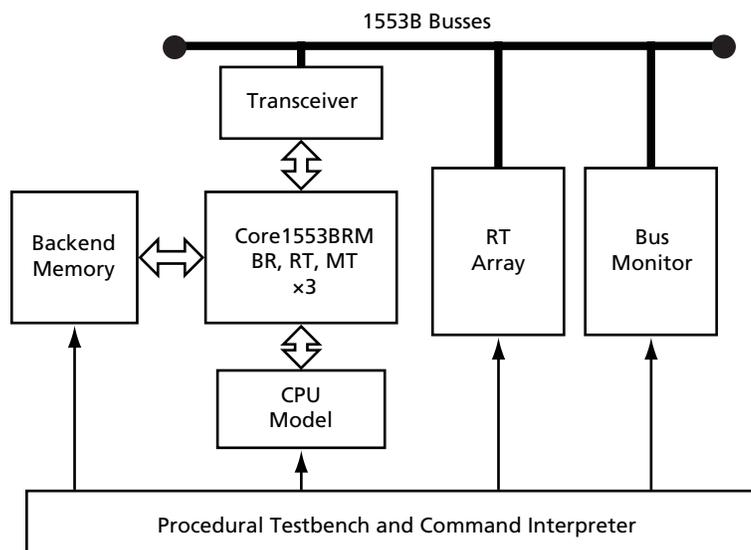
Figure 10-1 · Verification Testbench

The testbench contains the following blocks:

- BRM – 3 Core1553BRM devices, allowing one to be used as a bus controller, one as a remote terminal, and the other as a monitor
- TRANSCEIVER – Models the 1553B transceivers
- CPU – Models the CPU interface to the core
- BACKEND – Provides the backend memory connected to the bus controller. This can operate in both asynchronous and synchronous modes, with programmable access times.
- RT ARRAY – 2 test RTs (16 and 17). These RTs have the ability to create error conditions.
- BUS MONITOR – A bus monitor that monitors 1553 activity and detects error conditions
- INTERPRETER – Processes user input or command files and runs the simulations

The Core1553BRM verification testbench uses a command interpreter to apply high-level stimuli to the BRM core. This allows the user to directly set the BRM memory and registers.

When started, the simulation will initialize and wait for user input. A simple command file is shown below.

```
# Example Script

UNIT 0                  ! Access BRM unit 0


REG 0 #0416             ! ETCE BCAST int log enabled, PPONG
REG 1 #0000             ! BC mode
REG 3 #FFFF             ! Enable all interrupts
REG 5 #F000             ! Set tnterrupt log at F000
INTINIT #F000           ! Point testbench interrupt handler to log


MEM #0000 #4200  1.0.1.4   0.0.0.0  #0100 0 0 0 0   ! BC to RT
MEM #0008 #4200  1.1.1.3   0.0.0.0  #0200 0 0 0 0   ! RT to BC
MEM #0010 #0000 0 0 0 0 0 0 0 ! EOL


# Data Tables
FILLMEM #0100 #1000 32 1
FILLMEM #0200 #0000 32 0


# Now start the Bus Controller
START #0000             ! Start the BRM at address 0000
INTWAIT 1
INTCMP 1 #0020 #0010  ! Vector pointer, expected  IIW=0020  IAW=0010
STATS                  ! Display bus statistics
```

This command file sets up a message list that processes two messages, a BC-to-RT message and an RT-to-BC message. Having programmed the memory, the BRM is started by writing to the control register. The testbench then waits for the interrupt to be generated by the EOL instruction and verifies the interrupt IAW and IIW values.

Supported Commands

The command interrupter supports the commands below. More detailed information can be found by using the HELP command when the simulation is running.

DEMO	: Run *demo.txt*
DOALL	: Run the complete verification simulation
UNIT [N]	: Set unit number
MONITORS CPU [BUS] [BRMn BRM] [RT]	: Turn monitors on/off
VERSION	: Display version information
PAUSE	: Pause simulation
START [ADDR]	: Start the core, set the block pointer
RESET	: Reset the core
RUNBC UNIT ADDR [WAIT]	: Start a bus controller at address
STARTU [UNIT]	: Start the core

STOPU [UNIT]	: Stop the core
REG [ADDR DATA]	: Display or set registers
REGCMP ADDR DATA [MASK]	: Compare register
REGBIT ADDR BIT VALUE	: Set or clear register bit
DISPLAY ADDR [N]	: Display memory
DISPRT RT ADDR [N]	: Display test RT memory
MEM ADDR DATA [DATA]	: Set memory
MEMBYTE UL ADDR DATA	: Do a memory byte write UL = 10/01
REGBYTE UL ADDR DATA	: Do a register byte write UL = 10/01
FILLMEM ADDR DATA [N] [INC]	: Fill core memory
MEMCMP ADDR DATA [MASK]	: Verify memory
FILLCMP ADDR DATA [N] [INC]	: Verify memory
BC SUM PARA	: Set up core controls
RT RTn SUM PARA	: Set up the test RT
CLRCNT	: Clear the bus word counters
CMPCNT BUSA BUSB	: Check the bus word counters
MEMTEST ADDR SIZE LOOP [FAIL]	: Memory test
INTINIT #ADDR	: Initialize interrupt handler
INTWAIT IMH [MIN TIME MAX TIME]	: Wait for interrupt μs
INTCMP IMH [#ADDR] [#REAS]	: Check interrupt value
RTINIT MODE	: Initialize memory for RT operations
PARAMS [1 2 3 4 5 6 7]	: Set the $parameters for future use
DO filename	: Run commands from file
DOLOG filename	: Run commands from file and create CPU log file
CPULOG 0|1	: Enable or disable the CPU log file
WAIT [X]	: Run simulation for X μs, default 20 μs
STATS	: Display simulation statistics
JUMP LABEL	: Ignore commands until LABEL matches
LABEL LABEL	: Label for JUMP instruction
ECHO [0|1]	: Turn command echo on or off
HELP	: Display help information
#	: Comment; will be echoed to the simulation log
#*	: Comment; will NOT be echoed to the log
.	: Repeat the last command
QUIT	: Quit

Data for the commands can be entered in several forms:

1234	: Decimal
#1234	: Hexadecimal
A123	: Automatically switches to hexadecimal
1.0.23.12	: 1553B Command Word, RT = 1, TX = 0, SA = 23, WC = 12
#1F.#1.#1F.#01	: 1553B Command Word with hexadecimal values
$[1 2 3 4 5 6 7]	: Use one of the values set with the PARAMS command

The HELP command provides additional information on the command operations.

Command Files

Actel supplies a set of command files[1] that are used to verify the core (Table 10-1). These command files provide 100% code coverage for the Core1553BRM RTL source code. A detailed list of the tests each of these command files performs is provided in "Verification Tests Carried Out" on page 111.

Table 10-1 · Command Files

File	Function
doall	Runs all of the command files below
demo	Simple demo of 1553 operation
bcbasic	BC basic message transfers
bcopcodes	BC operation codes and flags
bcopcodes2	BC extended operation codes and flags
bcrterrors	BC operation with RTs inserting errors
bcretries	BC retry operation
bcregs	BC register operation
bctimers	BC timers
rtindex	RT operation in indexed mode
rtppong	RT operation in ping pong mode
rtcirc1	RT operation in circular buffer mode 1
rtcirc2	RT operation in circular buffer mode 2
rtmcodes	RT mode codes
rtmcodesbc	RT broadcast mode codes
rterrors	RT error conditions

---

1. *The command files are scrambled in the Evaluation release of the core. They are provided as plain text with the Obfuscated and RTL versions of the core.*

Table 10-1 · Command Files (continued)

File	Function
rtstatus	RT status word settings
rtmisc	Miscellaneous RT tests
rtlegal	RT legalization logic
rtstatus	RT status bits
mtbasic	MT operation
mttests	MT operation
mtandrt	Combined RT and MT operation
mterrors	MT error conditions
mtrtrt	MT RT-to-RT messages
bc1553ab	1553A and 1553B operational differences
memory	Memory interface and timeouts
misc	Miscellaneous tests

Alternatively, command files can be created by the user and invoked using the *include* command.

CPU Logging Function

If enabled (DOLOG command), the CPU block will create a log file recording all the read and write transactions between the testbench and Core1553BRM for both core register and memory accesses. This log file can be directly applied to the Core1553BRM development system. This is fully described in the Core1553BRM Development Kit documentation.

VHDL User Testbench

Actel provides an example testbench that you can use as the starting point for design verification of the core in your design. A block diagram of the testbench is shown in Figure 10-2; the blocks are described in Table 10-2.
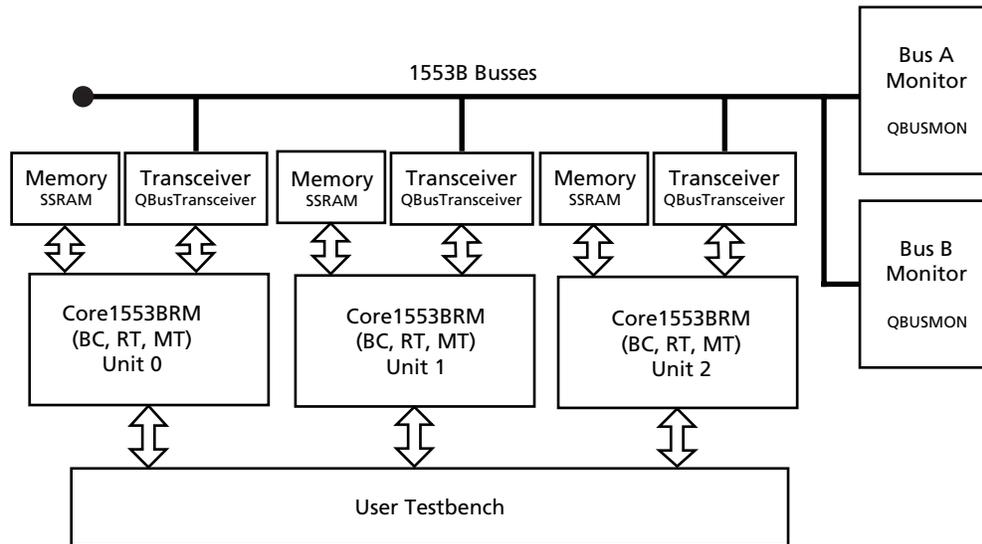
Figure 10-2 · User VHDL Testbench

Table 10-2 · User VHDL Functional Blocks

Block	Description
Core1553BRM	Three Core1553BRM cores are instantiated in the testbench. This allows one of the cores to be configured as the bus controller, and the other two as combined monitors and remote terminals. This allows all three functions to be demonstrated and RT-to-RT messages to be carried out.
QBUSTRANSCEIVER	This block implements a single-channel 1553 transceiver. It connects directly to the transceiver interface on Core1553BRM and the 1553 bus.
QBUSMON	This block monitors the 1553 bus and displays the bus traffic. It connects directly to the 1553 busses.
SSRAM	This block is synchronous memory that can be connected directly to the Core1553BRM backend interface. It implements a 64k×16 memory.

The main process in the testbench writes to the Core1553BRM CPU interface and can program the Core1553BRM registers as well as the memory. To simplify the testbench, the following procedure calls are provided:

```
procedure cpu_write_reg(address: integer ; data : integer);

procedure cpu_write_mem(address, data : integer);

procedure cpu_read_reg(address: integer ; data : out integer);

procedure cpu_read_mem(address : integer; data : out integer);

procedure cpu_write_mblk(address,data0,data1,data2,data3,data4,data5,data6,data7 :
  integer);
```

The first four procedures provide simple read and write functions to Core1553BRM registers or the memory. The fifth procedure allows MSGBLK to be programmed with a single call. The eight data values set the MSGBLK parameters (MSGTYPE, CW1, CW2, DATAPTR, SW1, SW2, BRANCH, TIMER).

Study of the *Usertbench.vhd* file provided in the source directory is recommended to fully understand how this testbench operates.

Verilog User Testbench

Actel provides an example testbench that you can use as the starting point for design verification of the core in your design. A block diagram of the testbench is shown in Figure 10-3; the blocks are described in Table 10-3.
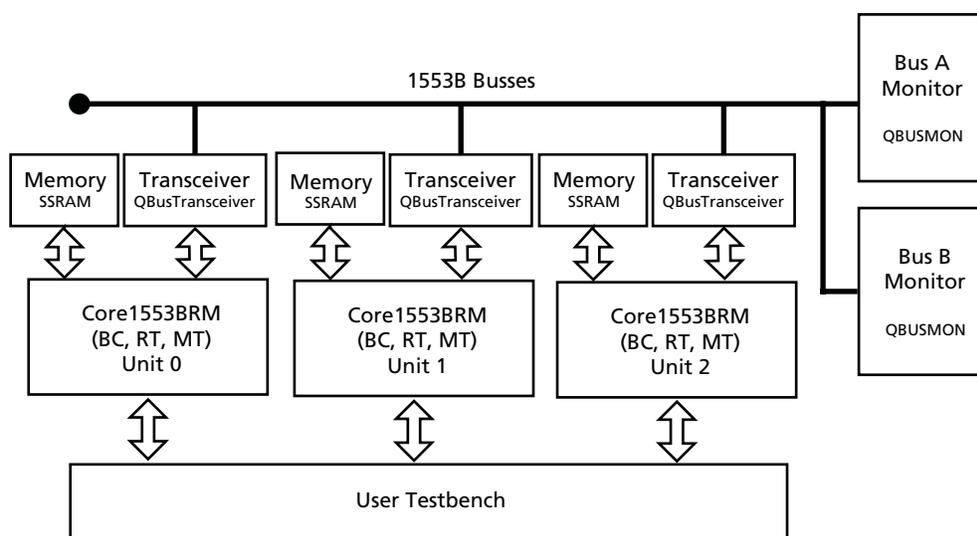
Figure 10-3 · User Verilog Testbench

Table 10-3 · User Verilog Functional Blocks

Block	Description
Core1553BRM	Three Core1553BRM cores are instantiated in the testbench. This allows one core to be configured as the bus controller and the other two to be configured as combined monitors and remote terminals. This allows all three functions to be demonstrated, and RT-to-RT messages can be carried out.
QBUSTRANSCEIVER	This block implements a 1553 transceiver. It connects directly to the transceiver interface on Core1553BRM and the 1553B bus.
QBUSMON	This block monitors the 1553 bus and displays the bus traffic. It connects directly to the 1553 busses.
SSRAM	This block is a synchronous memory that can be connected directly to the Core1553BRM backend interface. It provides a 64k×16 memory.

The main process in the testbench writes to the Core1553BRM CPU interface and can program the Core1553BRM registers as well as the memory. To simplify the testbench, the following tasks are provided:

```
task cpu_write_reg;
 input [15:0] address;
 input [15:0] data;


task cpu_write_mem;
 input [15:0] address;
 input [15:0] data;


task cpu_read_reg;
 input [15:0] address;
 output [15:0] data;


task cpu_read_mem;
 input  [15:0] address;
 output [15:0] data;


task cpu_write_mblk;
 input [15:0] address;
 input [15:0] data0;
 input [15:0] data1;
 input [15:0] data2;
 input [15:0] data3;
 input [15:0] data4;
 input [15:0] data5;
 input [15:0] data6;
 input [15:0] data7;
```

The first four tasks above provide simple read and write functions to Core1553BRM registers or the memory. The fifth procedure allows MSGBLK to be programmed with a single call; the eight data values set the MSGBLK parameters (MSGTYPE, CW1, CW2, DATAPTR, SW1, SW2, BRANCH, TIMER).

Study of the *usertbench.v* file provided in the source directory is recommended to fully understand how this testbench operates.

11

Implementation Hints

Clock and Reset Networks

The core requires that a clock buffer be inserted to drive the CLK input. This should be done automatically by the synthesis tool.

The core gates the external RSTINn input to generate the internal interrupt. The core instantiates a global buffer internally to drive this reset network. It is not required to use a global network buffer to bring in the RSTINn signal.

RT Legalization Registers

The core requires sixteen 16-bit registers to implement the RT legalization registers. These registers can be implemented using logic resources or memory within the FPGA, or via external hardware using a direct decode of the 1553 command words, removing the need for the logic resources and memory. The implementation is controlled by the LEGREGS parameter in the source code (Table 11-1 and Table 11-2).

Table 11-1 · LEGREGS Parameter

LEGREGS	Description
0	The legalization registers are not implemented. The user must use the external RT legalization interface.
1	The legalization logic is implemented in the registers within the FPGA.
2	The legalization logic is implemented in the memory blocks.

Table 11-2 · RT Legalization Registers Implementations

	Advantages	Disadvantages
External Hardware (0)	Requires minimal logic resources. Does not require initialization. Can implement legalization down to the word count level, e.g., a subaddress can be set to accept only 12-word messages.	Not software compatible with legacy devices. Cannot be modified in-system. Subaddress legality needs to be defined in hardware, not software.
Registers (1)	Registers are auto-initialized so that only the supported mode codes are legalized. Legacy devices auto-initialize, so this implementation allows for legacy compatibility. Registers can be read when the RT is operational.	Uses a large amount of logic resources to implement this function, up to 512 logic cells.
Memory (2)	Reduces required logic resources.	Registers are NOT auto-initialized. The CPU must initialize these registers before the core is started. Registers cannot be read when the RT is operational.

Actel recommends that *Registers (1)* be used to implement the legalization registers if logic resources are available, as this provides full software compatibility with legacy devices. Otherwise, memory blocks should be used to implement this function.

Shared versus Own Memory

Core1553BRM requires connection to a memory block to function. Core1553BRM allows the memory to be connected to the core in two modes—shared memory and own memory.

Shared Memory

In this mode (Figure 11-1), the core shares the CPU memory. This is compatible with the SuMMIT device. Core1553BRM will assert its MEMREQ output and, when granted by the bus arbiter, assume control of the memory and complete its memory access cycle.
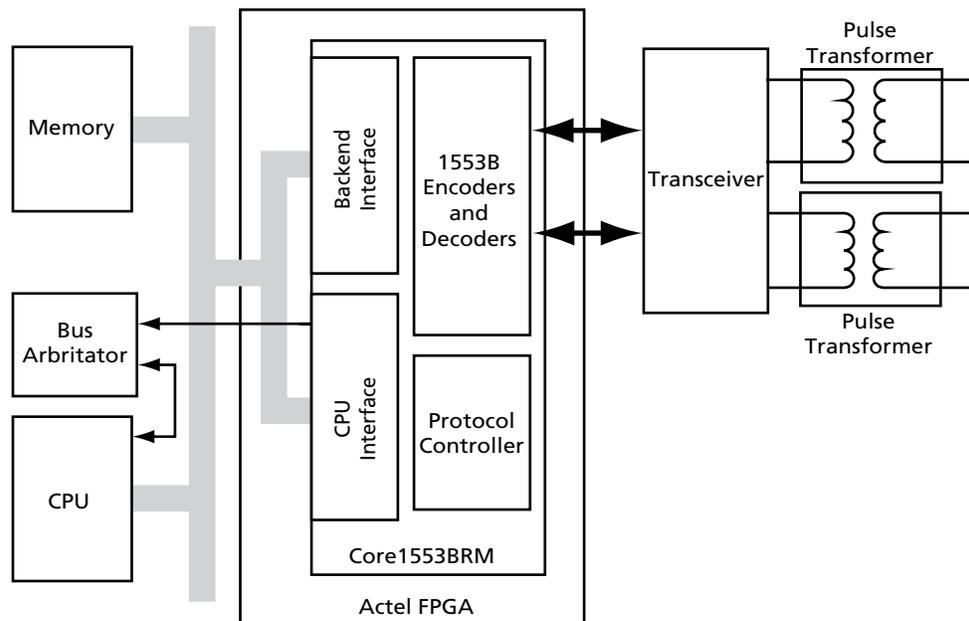
Figure 11-1 · Core1553BRM with Shared Memory

Shared memory implementations can reduce overall cost, as no special memory block needs to be implemented for Core1553BRM, but the core requires direct access to the CPU memory bus, and bus arbitration logic is required.

In shared memory systems, the CPUMEMEN input should be tied LOW.

Own Memory

In this mode (Figure 11-2), the core has its own memory block. The CPU accesses the memory though the CPU interface of the core. The core provides the arbitration function, allowing both the 1553 logic and the CPU interface to access the memory block.
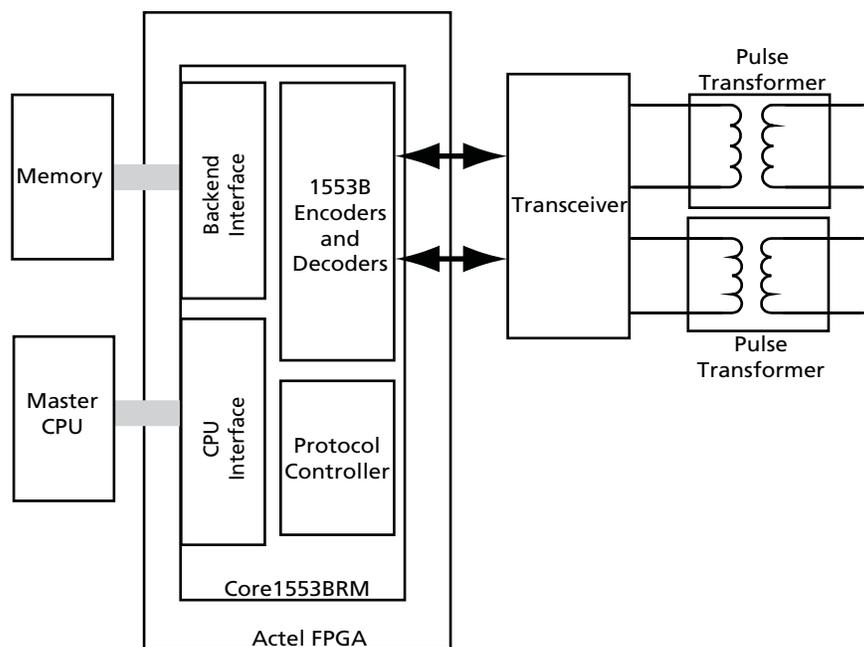
Figure 11-2 · Core1553BRM with Its Own Memory

This implementation is recommended for FPGA devices that have on-chip RAM, where the core backend interface can be directly connected to the FPGA synchronous memory block.

When Core1553BRM has its own memory, the CPUMEMEN input should be tied HIGH.

Transceivers

Core1553BRM needs a 1553B transceiver to drive the 1553B bus. It is designed to interface directly to common MIL-STD-1553 transceivers, such as DDC BU-63147, Holt HI1567/1568/1573/1574, and Aeroflex ACT4402. When using ProASIC3-based families, ProASIC$^{\text{PLUS}}$, or Axcelerator devices, level translators are required to connect the 5 V outputs of the 1553B transceivers to the 3.3 V inputs of the FPGA.

In addition to the transceiver, a pulse transformer is required for interfacing to the 1553B bus. Figure 11-1 on page 102 and Figure 11-2 show the connections required from Core1553BRM to the transceivers and then to the bus via the pulse transformers. Here, the 1553 interface signal bus is connected to the transceiver, which, in turn, is connected to the pulse transformer.

12

Legacy Mode Operation

Core Operation

Core1553BRM is designed to be software-compatible with existing 1553B solutions.

It supports the following features:

- Interrupt logs
- Programmable message timeouts
- Circular buffer operation

It does not support the following features:

- Buffer mode operation
- Built-in test functions, although the BIT register and the transmit BIT mode code are supported
- Auto-initialization of internal registers and memory

Legacy Mode

Core1553BRM is software-compatible with the UTMC 69151 (SuMMIT) device. The hardware interface of the core is designed to simplify integration within an FPGA device and provides separate control (CPU) and memory busses. Using separate busses can simplify integration within the FPGA, especially when FPGA memory is used.

A VHDL wrapper file provided when the user testbench is exported from CoreConsole, *summit.vhd*, creates a top-level design with a single CPU/memory data bus and renames the interface signals to match the SuMMIT device (Figure 12-1 on page 106). This wrapper layer includes a small amount of control logic to multiplex the CPU and memory busses. The wrapper does not use bidirectional address and data busses; instead, separate inputs, outputs, and enables are provided. This allows internal FPGA memory to be used if required. When bidirectional ports are used, they must be directly connected to FPGA I/O pins and can easily be added by the user if required (comments in the wrapper file provide instructions on how to use bidirectional ports).
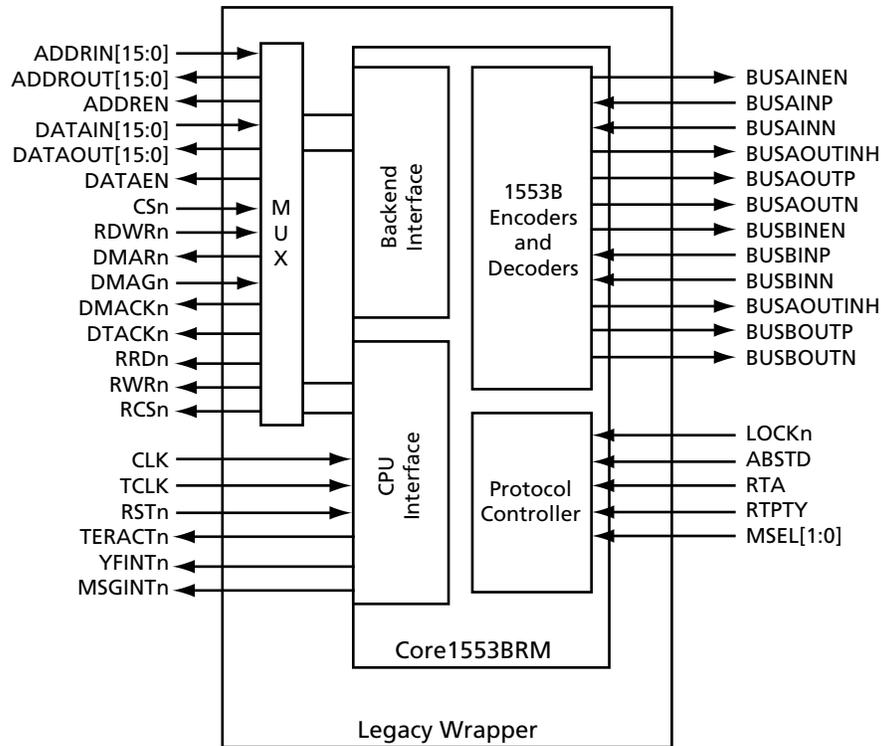
Figure 12-1 · Legacy Mode Wrapper

Table 12-1 gives the mapping between the legacy mode wrapper file and the Core1553BRM signals. Throughout this document, the legacy signal names are used to assist designers who are familiar with the legacy device.

As stated earlier, Core1553BRM is designed to be software-compatible with the actual behavior of SuMMIT 1553 devices. Table 12-2 on page 108 details known differences, either with the SuMMIT datasheet or the SuMMIT device.

Table 12-1 · Legacy Mode Wrapper Signal Assignment

Legacy Signal Name	Core1553BRM Signal Assignment
CLK	CLK
TCLK	TCLK
RSTn	RSTINn
RSTOUTn	RSTOUTn
BUSAINEN	BUSAINEN
BUSAINP	BUSAINP
BUSAINN	BUSAINN
BUSBINEN	BUSBINEN

Table 12-1 · Legacy Mode Wrapper Signal Assignment (continued)

Legacy Signal Name	Core1553BRM Signal Assignment
BUSBINP	BUSBINP
BUSBINN	BUSBINN
BUSAOUTIN	BUSAOUTIN
BUSAOUTP	BUSAOUTP
BUSAOUTN	BUSAOUTN
BUSBOUTIN	BUSBOUTIN
BUSBOUTP	BUSBOUTP
BUSBOUTN	BUSBOUTN
ADDRIN	CPUADDR
ADDROUT	ADDROUT
ADDREN	ADDREN
DATAIN	MEMDIN MUXed with CPUDIN
DATAOUT	DATAOUT
DATAEN	MEMDEN or CPUDEN
CSn	CPUWRn <= not (not CSn and not RDWRn)
RDWRn	CPUWn <= not (not CSn and not RDWRn)
DMARn	MEMREQn
DMAGn	MEMGNTn
DMACKn	MEMACCn
DTACKn	MEMWAITn
RRDn	MEMRDn
RWRn	MEMWRn[0]
RCSn	MEMCSn
ROMENn	1
YFINTn	not INTOUTH
MSGINTn	not INTOUTM
AUTOENn	Not used
LOCKn	LOCKn
ABSTD	ABSTDIN

Table 12-1 · Legacy Mode Wrapper Signal Assignment (continued)

Legacy Signal Name	Core1553BRM Signal Assignment
MSEL	MSELIN
RTA	RTADDRIN
RTPTY	RTADDRPIN
RTADDERR	RTADERR
TERACTn	not BUSY
READYn	READYn
SSYFn	SSYSFn

Table 12-2 · Core1553BRM Behavior vs. SuMMIT Operation

Symptom	Core1553BRM Behavior
RT legalization initialization	Core1553BRM initializes the RT legalization registers only when the LEGREGS parameter is set to 0.
RT information words	Core1553BRM sets bit 5 in the transmit and receive information words when a broadcast message is transmitted or received.
Reset RT mode code	When a Reset RT mode code is received, Core1553BRM will do the following: • Reset the 1553B decoder • Reset the Time Tag register (register 7) • Enable both channels, overriding the Transmitter Shutdown mode code • Enable the Terminal Flag bit, overriding the Inhibit Terminal Flag mode code • Pulse the RSTOUTn output The core will continue to operate in RT mode, i.e., the STEX bit will stay active.

Table 12-2 · Core1553BRM Behavior vs. SuMMIT Operation (continued)

Symptom	Core1553BRM Behavior
Monitor operation	When programmed to capture N messages, Core1553BRM will capture N messages and then generate the Monitor Block Count interrupt. At this point, the monitor descriptor pointer is NOT reset. The following message will be captured to the next descriptor address. After this message has been captured, the monitor descriptor pointer is reset to the initial value. For example, if Core1553BRM is programmed to capture four messages with the initial monitor descriptor set to 2000 hex, the core will do the following: • Capture message 1 to 2000 hex • Capture message 2 to 2008 hex • Capture message 3 to 2010 hex • Capture message 4 to 2018 hex • Generate an interrupt • Capture message 5 to 2020 hex • Reset the monitor descriptor pointer • Capture message 6 to 2000 hex • Capture message 7 to 2008 hex Core1553BRM sets the Interrupt Address Word (IAW) to point to the last monitor descriptor processed. In the example shown above, the IAW would be 2018 hex.
BIT operations	Core1553BRM has automatic health monitoring but does not include the control register BIT function. It will set the BIT word (register 6) bits as below: 15: DMAF – Set when a memory access fails 14: WRAPF – Set when a 1553B loop back failure is detected 13: TAPF – Set when a terminal address parity error occurs 12: BITF – Core1553BRM does not set this bit. 11: CHAF – Set when a transmitter time-out occurs on Bus A 10: CHAF – Set when a transmitter time-out occurs on Bus B 9:0: UDB – Core1553BRM does not set these bits. All of the bits (including 12 and 9:0) can be set and cleared by the CPU writing to the BIT register. Bits 9:0 of the BIT register at reset indicate the version of the core. The settings are provided in the core release notes or datasheet.
Buffer mode	Core1553BRM does not support buffer mode (control register bit 6). The core writes/reads data as required directly to/from memory.
Auto-initialization	Core1553BRM does not support auto-initialization. It is assumed that the local CPU will initialize the core.

Verification Tests Carried Out

The provided command files perform the tests given in Table A-1.

Table A-1 · Files and Tests Performed

File	Tests Performed
doall	Runs all of the command files
bcsetuprts	Sets up the RTs so the BC can be tested
bcbasic	BC basic message transfers
bcopcodes	BC opcodes All opcodes
bcopcodes2	BC opcodes and flags Flag operation
bcrterrors	BC operation with RTs inserting errors Parity error in DW Manchester error in SW Manchester error in DW Inverted SYNC on SW Inverted SYNC on DW Word counts none, +1, −1, 33 Mode code, extra data Mode code, no data No response SW incorrect RT field RTRT no response TX RT RTRT no response RX RT RTRT SWs wrong Message error settings Message error settings RTRT Transmitter loopback tests
bcretries	BC retry operations
bcregs	BC register operation Read/write of control register STOP and RESET instructions Broadcast enable Interrupts
rtindex	RT operation in indexed mode
rtppong	RT operation in ping pong mode
rtcirc1	RT operation in circular buffer mode 1

Table A-1 · Files and Tests Performed (continued)

File	Tests Performed
rtcirc2	RT operation in circular buffer mode 2
rtstatus	RT status word settings in MIL1553A and MIL1553B mode
rtmode	RT mode codes
rtmodebc	RT broadcast mode codes
rtlegal	RT legalization logic
mtbasic	MT operation
mtandrt	Combined RT and MT operation
mterrors	MT error conditions
mtrbrt	RT-to-RT monitor operations Normal and error condition
bc1553ab	1553A and 1553B operational differences
memory	Memory interface and timeouts
misc	Word count errors Transmit timer overrun RT address error logic Enhanced modes

The *doall* script invokes all the tests listed above.

SuMMIT Differences

Table B-1 lists the known differences between Core1553BRM and the Aeroflex SuMMIT device.

Table B-1 · Core1553BRM Behavior vs. SuMMIT Operation

Symptom	Core1553BRM Behavior
RT legalization initialization	Core1553BRM initializes the RT legalization registers as configured by the LEGREGS parameter.
RT information words	Core1553BRM sets bit 5 in the transmit and receive information words when a broadcast message is transmitted or received.
Reset RT mode code	When a Reset RT mode code is received, Core1553BRM will do the following: • Reset the 1553B decoder • Reset the Time Tag register (register 7) • Enable both channels, overriding the Transmitter Shutdown mode code • Enable the Terminal Flag bit, overriding the Inhibit Terminal Flag mode code • Pulse the RSTOUTn output The core will continue to operate in RT mode, i.e., the STEX bit will stay active.
Monitor operation	When programmed to capture N messages, Core1553BRM will capture N messages and then generate the Monitor Block Count interrupt. At this point, the monitor descriptor pointer is NOT reset. The following message will be captured to the next descriptor address. After this message has been captured, the monitor descriptor pointer is reset to the initial value. For example, if Core1553BRM is programmed to capture four messages with the initial monitor descriptor set to 2000 hex, then the core will: • Capture message 1 to 2000 hex • Capture message 2 to 2008 hex • Capture message 3 to 2010 hex • Capture message 4 to 2018 hex • Generate an interrupt • Capture message 5 to 2020 hex • Reset the monitor descriptor pointer • Capture message 6 to 2000 hex • Capture message 7 to 2008 hex Core1553BRM sets the IAW to point to the last monitor descriptor processed. In the example shown above, the IAW would be 2018 hex.

Table B-1 · Core1553BRM Behavior vs. SuMMIT Operation (continued)

Symptom	Core1553BRM Behavior
BIT operations	Core1553BRM has automatic health monitoring but does not include the control register BIT function. It will set the BIT word (register 6) bits as below: • 15: DMAF – Set when a memory access fails • 14: WRAPF – Set when a 1553B loopback failure is detected • 13: TAPF – Set when a terminal address parity error occurs • 12: BITF – Core1553BRM does not set this bit. • 11: CHAF – Set when a transmitter timeout occurs on Bus A • 10: CHAF – Set when a transmitter timeout occurs on Bus B • 9:0 UDB – Core1553BRM does not set these bits. All of the bits (including 12 and 9:0) can be set and cleared by the CPU writing to the BIT register. Bits 9:0 of the BIT register at reset indicate the version of the core. The settings are provided in the core release notes or datasheet.
Buffer mode	Core1553BRM does not support buffer mode (Control register bit 6). The core writes/reads data as required directly to/from memory.
Auto-initialization	Core1553BRM does not support auto-initialization. It is assumed that the local CPU will initialize the core.

Table B-2 · Legacy Mode Wrapper Signal Assignment

Legacy Signal Name	Core1553BRM Signal Assignment
CLK	CLK
TCLK	TCLK
RSTn	RSTINn
RSTOUTn	RSTOUTn
BUSAINEN	BUSAINEN
BUSAINP	BUSAINP
BUSAINN	BUSAINN
BUSBINEN	BUSBINEN
BUSBINP	BUSBINP
BUSBINN	BUSBINN
BUSAOUTIN	BUSAOUTIN
BUSAOUTP	BUSAOUTP
BUSAOUTN	BUSAOUTN
BUSBOUTIN	BUSBOUTIN
BUSBOUTP	BUSBOUTP
BUSBOUTN	BUSBOUTN

Table B-2 · Legacy Mode Wrapper Signal Assignment (continued)

Legacy Signal Name	Core1553BRM Signal Assignment
ADDRIN	CPUADDR
ADDROUT	ADDROUT
ADDREN	ADDREN
DATAIN	MEMDIN MUXed with CPUDIN
DATAOUT	DATAOUT
DATAEN	MEMDEN or CPUDEN
CSn	CPUWRn <= not (not CSn and not RDWRn)
RDWRn	CPUWn <= not (not CSn and not RDWRn)
DMARn	MEMREQn
DMAGn	MEMGNTn
DMACKn	MEMACCn
DTACKn	MEMWAITn
RRDn	MEMRDn
RWRn	MEMWRn[0]
RCSn	MEMCSn
ROMENn	1
YFINTn	not INTOUTH
MSGINTn	not INTOUTM
AUTOENn	Not used
LOCKn	LOCKn
ABSTD	ABSTDIN
MSEL	MSELIN
RTA	RTADDRIN
RTPTY	RTADDRPIN
RTADDERR	RTADERR
TERACTn	not BUSY
READYn	READYn
SSYFn	SSYSFn

ACKVAL and WAITVAL Settings

Table C-1 to Table C-8 on page 125 give the possible ACKVAL and WAITVAL settings for different clock speeds and the CPUMEM input setting.

For instance, if the system is operating at 24 MHz with CPUMEM = 0 (Table C-4 on page 120) and it is known that the maximum number of inserted wait states will be four, then by setting ACKVAL = 167 and WAITVAL = 4, the allowed MEMREQn-to-MEMGNTn delay will be 6.958 μs. Alternatively, if the MEMREQn-to-MEMGNTn delay is less than 0.083 μs (e.g., MEMGNTn is tied LOW), ACKVAL can be set to 2 and WAITVAL to 34, allowing a read/write pulse width of up to 1,458 ns.

Table C-1 · Backend Timing, CPUMEM = 0, CLOCK = 12 MHz

MEMREQn to MEMGNTn Maximum Delay in μs	Maximum Number of Wait States	Maximum Read/Write Pulse Width Clocks	Maximum Read/Write Pulse Width in ns	ACKVAL	WAITVAL
6.250	0	1	83.33	75	0
5.833	1	2	166.66	70	1
5.333	2	3	250.00	64	2
4.916	3	4	333.33	59	3
4.416	4	5	416.66	53	4
4.000	5	6	500.00	48	5
3.500	6	7	583.33	42	6
3.000	7	8	666.66	36	7
2.583	8	9	750.00	31	8
2.166	9	10	833.33	26	9
1.666	10	11	916.66	20	10
1.250	11	12	1000.00	15	11
0.750	12	13	1083.33	9	12
0.333	13	14	1166.66	4	13

Table C-2 · Backend Timing, CPUMEM = 0, CLOCK = 16 MHz

MEMREQn to MEMGNTn Maximum Delay in μs	Maximum Number of Wait States	Maximum Read/Write Pulse Width Clocks	Maximum Read/Write Pulse Width in ns	ACKVAL	WAITVAL
7.062	0	1	62.50	113	0
6.750	1	2	125.00	108	1
6.375	2	3	187.50	102	2
6.062	3	4	250.00	97	3
5.687	4	5	312.50	91	4
5.375	5	6	375.00	86	5
5.000	6	7	437.50	80	6
4.687	7	8	500.00	75	7
4.312	8	9	562.50	69	8
4.000	9	10	625.00	64	9
3.625	10	11	687.50	58	10
3.312	11	12	750.00	53	11
2.937	12	13	812.50	47	12
2.625	13	14	875.00	42	13
2.250	14	15	937.50	36	14
1.937	15	16	1000.00	31	15
1.562	16	17	1062.50	25	16
1.250	17	18	1125.00	20	17
0.875	18	19	1187.50	14	18
0.562	19	20	1250.00	9	19
0.187	20	21	1312.50	3	20

Table C-3 · Backend Timing, CPUMEM = 0, CLOCK = 20 MHz

MEMREQn to MEMGNTn Maximum Delay in μs	Maximum Number of Wait States	Maximum Read/Write Pulse Width Clocks	Maximum Read/Write Pulse Width in ns	ACKVAL	WAITVAL
7.550	0	1	50.00	151	0
7.300	1	2	100.00	146	1
7.000	2	3	150.00	140	2
6.750	3	4	200.00	135	3
6.450	4	5	250.00	129	4
6.200	5	6	300.00	124	5
5.900	6	7	350.00	118	6
5.650	7	8	400.00	113	7
5.350	8	9	450.00	107	8
5.100	9	10	500.00	102	9
4.800	10	11	550.00	96	10
4.550	11	12	600.00	91	11
4.250	12	13	650.00	85	12
4.000	13	14	700.00	80	13
3.700	14	15	750.00	74	14
3.450	15	16	800.00	69	15
3.150	16	17	850.00	63	16
2.900	17	18	900.00	58	17
2.600	18	19	950.00	52	18
2.350	19	20	1000.00	47	19
2.050	20	21	1050.00	41	20
1.800	21	22	1100.00	36	21
1.500	22	23	1150.00	30	22
1.250	23	24	1200.00	25	23
0.950	24	25	1250.00	19	24
0.700	25	26	1300.00	14	25
0.400	26	27	1350.00	8	26
0.150	27	28	1400.00	3	27

Table C-4 · Backend Timing, CPUMEM = 0, CLOCK = 24 MHz

MEMREQn to MEMGNTn Maximum Delay in µs	Maximum Number of Wait States	Maximum Read/Write Pulse Width Clocks	Maximum Read/Write Pulse Width in ns	ACKVAL	WAITVAL
7.875	0	1	41.66	189	0
7.625	1	2	83.33	183	1
7.416	2	3	125.00	178	2
7.208	3	4	166.66	173	3
6.958	4	5	208.33	167	4
6.750	5	6	250.00	162	5
6.500	6	7	291.66	156	6
6.250	7	8	333.33	150	7
6.041	8	9	375.00	145	8
5.833	9	10	416.66	140	9
5.583	10	11	458.33	134	10
5.375	11	12	500.00	129	11
5.125	12	13	541.66	123	12
4.916	13	14	583.33	118	13
4.666	14	15	625.00	112	14
4.458	15	16	666.66	107	15
4.208	16	17	708.33	101	16
4.000	17	18	750.00	96	17
3.750	18	19	791.66	90	18
3.541	19	20	833.33	85	19
3.291	20	21	875.00	79	20
3.041	21	22	916.66	73	21
2.833	22	23	958.33	68	22
2.625	23	24	1000.00	63	23
2.375	24	25	1041.66	57	24
2.166	25	26	1083.33	52	25
1.916	26	27	1125.00	46	26
1.708	27	28	1166.66	41	27
1.458	28	29	1208.33	35	28

Table C-4 · Backend Timing, CPUMEM = 0, CLOCK = 24 MHz (continued)

MEMREQn to MEMGNTn Maximum Delay in μs	Maximum Number of Wait States	Maximum Read/Write Pulse Width Clocks	Maximum Read/Write Pulse Width in ns	ACKVAL	WAITVAL
1.250	29	30	1250.00	30	29
1.000	30	31	1291.66	24	30
0.791	31	32	1333.33	19	31
0.541	32	33	1375.00	13	32
0.333	33	34	1416.66	8	33
0.083	34	35	1458.33	2	34

Table C-5 · Backend Timing, CPUMEM = 1, CLOCK = 12 MHz

MEMREQn to MEMGNTn Maximum Delay in μs	Maximum Number of Wait States	Maximum Read/Write Pulse Width Clocks	Maximum Read/Write Pulse Width in ns	ACKVAL	WAITVAL
3.833	0	1	83.33	46	0
3.500	1	2	166.66	42	1
3.166	2	3	250.00	38	2
2.833	3	4	333.33	34	3
2.416	4	5	416.66	29	4
2.083	5	6	500.00	25	5
1.750	6	7	583.33	21	6
1.333	7	8	666.66	16	7
1.000	8	9	750.00	12	8
0.666	9	10	833.33	8	9
0.250	10	11	916.66	3	10

Table C-6 · Backend Timing, CPUMEM = 1, CLOCK = 16 MHz

MEMREQn to MEMGNTn Maximum Delay in μs	Maximum Number of Wait States	Maximum Read/Write Pulse Width Clocks	Maximum Read/Write Pulse Width in ns	ACKVAL	WAITVAL
4.500	0	1	62.50	72	0
4.250	1	2	125.00	68	1
3.937	2	3	187.50	63	2
3.687	3	4	250.00	59	3
3.437	4	5	312.50	55	4
3.125	5	6	375.00	50	5
2.875	6	7	437.50	46	6
2.625	7	8	500.00	42	7
2.312	8	9	562.50	37	8
2.062	9	10	625.00	33	9
1.812	10	11	687.50	29	10
1.500	11	12	750.00	24	11
1.250	12	13	812.50	20	12
1.000	13	14	875.00	16	13
0.687	14	15	937.50	11	14
0.437	15	16	1000.00	7	15
0.187	16	17	1062.50	3	16

Table C-7 · Backend Timing, CPUMEM = 1, CLOCK = 20 MHz

MEMREQn to MEMGNTn Maximum Delay in µs	Maximum Number of Wait States	Maximum Read/Write Pulse Width Clocks	Maximum Read/Write Pulse Width in ns	ACKVAL	WAITVAL
4.850	0	1	50.00	97	0
4.650	1	2	100.00	93	1
4.450	2	3	150.00	89	2
4.200	3	4	200.00	84	3
4.000	4	5	250.00	80	4
3.800	5	6	300.00	76	5
3.550	6	7	350.00	71	6
3.350	7	8	400.00	67	7
3.150	8	9	450.00	63	8
2.900	9	10	500.00	58	9
2.700	10	11	550.00	54	10
2.500	11	12	600.00	50	11
2.250	12	13	650.00	45	12
2.050	13	14	700.00	41	13
1.850	14	15	750.00	37	14
1.600	15	16	800.00	32	15
1.400	16	17	850.00	28	16
1.200	17	18	900.00	24	17
0.950	18	19	950.00	19	18
0.750	19	20	1000.00	15	19
0.550	20	21	1050.00	11	20
0.300	21	22	1100.00	6	21
0.100	22	23	1150.00	2	22

Table C-8 · Backend Timing, CPUMEM = 1, CLOCK = 24 MHz

MEMREQn to MEMGNTn Maximum Delay in µs	Maximum Number of Wait States	Maximum Read/Write Pulse Width Clocks	Maximum Read/Write Pulse Width in ns	ACKVAL	WAITVAL
5.083	0	1	41.66	122	0
4.916	1	2	83.33	118	1
4.750	2	3	125.00	114	2
4.583	3	4	166.66	110	3
4.375	4	5	208.33	105	4
4.208	5	6	250.00	101	5
4.041	6	7	291.66	97	6
3.833	7	8	333.33	92	7
3.666	8	9	375.00	88	8
3.500	9	10	416.66	84	9
3.291	10	11	458.33	79	10
3.125	11	12	500.00	75	11
2.916	12	13	541.66	70	12
2.750	13	14	583.33	66	13
2.583	14	15	625.00	62	14
2.375	15	16	666.66	57	15
2.208	16	17	708.33	53	16
2.041	17	18	750.00	49	17
1.833	18	19	791.66	44	18
1.666	19	20	833.33	40	19
1.500	20	21	875.00	36	20
1.291	21	22	916.66	31	21
1.125	22	23	958.33	27	22
0.958	23	24	1000.00	23	23
0.750	24	25	1041.66	18	24
0.583	25	26	1083.33	14	25
0.416	26	27	1125.00	10	26
0.208	27	28	1166.66	5	27
0.041	28	29	1208.33	1	28

D

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650.318.8044**

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Actel Technical Support

Visit the Actel Customer Support website (www.actel.com/custsup/search.html) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/contact/offices/index.html.

Index

1553
  bus signals 26
  command words 43
  events 66
  functions 7
  messages 48, 65
  status word 68

A

ACKVAL settings 117
Actel
  electronic mail 127
  telephone 128
  web-based technical support 127
  website 127
antifuse FPGAs 7
Auto Stitch 22
automatic retry 43

B

backend 30
  memory interface timing 31
  timing settings 117
block diagram 15
broadcast commands 51
broadcast data pointer 57
buffers
  circular 51, 61
  data 58
  ping pong 51, 61
bulk data transfer 51
bus controller (BC) 7, 43
  control and message processing 43
  GOTO enhancements 89
  MIL-STD-1553A operation 49
  registers 44, 81
bus monitor (BM) 7, 65
  functions 85
  MIL-STD-1553A operation 69
  registers 66, 85

C

circular buffers 51, 61
clocks
  frequency 72
  networks 101

requirements 41
combined storage 61
command blocks 43, 44, 45
  architecture 45
  status 44
command files
  testbenches 96
command frame 44
command illegalization registers 71
command legality interface 15
command legalization interface 27
command words 48, 68
commands
  chaining 43
compatibility 16, 18
components
  external 11
contacting Actel
  customer service 127
  electronic mail 127
  telephone 128
  web-based technical support 127
control logic 105
control words 44, 45, 56
core reset 52
core versions
  Evaluation 7, 21
  Obfuscated 7, 21
  RTL 7, 21
CoreConsole 21
CPU 11, 18
  interface 15, 29
  interface timing 35
  logging function, testbench 97
  memory 18
current address pointer 62
customer service 127

D

data buffers 58, 60
  structure 58
data memory space 45
data pointers 43, 45, 48, 57, 68, 85
  broadcast 57
decoders 15
delays
  transceiver loopback 41

50200091-0 /3.07

Actel®