# Core1553BBC MIL-STD-1553B Bus Controller

**DirectCore**

## Product Summary

### Intended Use

- 1553B Bus Controller (BC)
- DMA Backend Interface to External Memory

### Key Features

- Supports MIL-STD-1553B
- Interfaces to External RAM
  - Supports up to 128kbytes of Memory
  - Synchronous or Asynchronous Backend Interface
  - Backend Interface Identical to Core1553BRT
- Selectable Clock Rate of 12, 16, 20, or 24 MHz
- Provides Direct CPU Access to Memory
- Interfaces to Standard 1553B Transceivers
- Fully Automated Message Scheduling
  - Frame Support
  - Conditional Branching and Sub-routines
  - Variable Inter-message Gaps and RT Response Times
  - Real Time Clock for Message Scheduling
  - Asynchronous Message Support

### Supported Families

- Fusion
- ProASIC3/E
- ProASICPLUS
- Axcelerator
- RTAX
- SX-A
- RTSX-S

### Core Deliverables

- Netlist Version
  - Compiled RTL Simulation Model, Compliant with the Actel Libero™ Integrated Design Environment (IDE)
  - Compatible with the Actel Designer Place-and-Route Tool
- RTL Version
  - VHDL or Verilog Core Source Code
  - Synthesis Scripts
- Actel-Developed Testbenches, VHDL and Verilog

### Synthesis and Simulation Support

- Synthesis: Synplicity®, Synopsys® (Design Compiler®/ FPGA Compiler™/FPGA Express™), Exemplar™
- Simulation: Vital-Compliant VHDL Simulators and OVI-Compliant Verilog Simulators

### Verification and Compliance

- Actel-Developed Simulation Testbench
- Core Implemented on the 1553B BC Development System
- Third-Party 1553B Compliance Testing of the 1553B Encoder and Decoder Blocks Implemented in an A54SXA32-STD Device

### Development System (Optional)

- Complete 1553B BC Implementation in an SX-A Device
- Includes a PCI Interface for Host CPU Connection
- Includes Transceivers and Bus Termination Components

### Contents

General Description .................................................. 2
Core1553BBC Device Requirements ......................... 4
Core1553BBC Verification and Compliance ............. 4
MIL-STD-1553B Bus Overview .................................. 4
I/O Signal Descriptions ............................................ 6
Bus Transceivers ..................................................... 20
Development System .............................................. 20
Typical BC System ................................................... 22
Specifications .......................................................... 24
Ordering Information ............................................. 28
List of Changes ....................................................... 29
Datasheet Categories .............................................. 29

© 2005 Actel Corporation

# General Description

The Core1553BBC provides a complete, MIL-STD-1553B Bus Controller (BC). A typical system implementation using the Core1553BBC is shown in Figure 1.

Core1553BBC reads message descriptor blocks from the memory and generates messages that are transmitted on the 1553B bus. Data words are read from the memory and transmitted on the 1553B bus. Data received is written to the memory. The core can be configured directly to connect to synchronous or asynchronous memory devices.

The core consists of five main blocks: the 1553B encoder, the 1553B decoder, a protocol controller block, a CPU interface, and a backend interface (Figure 2).

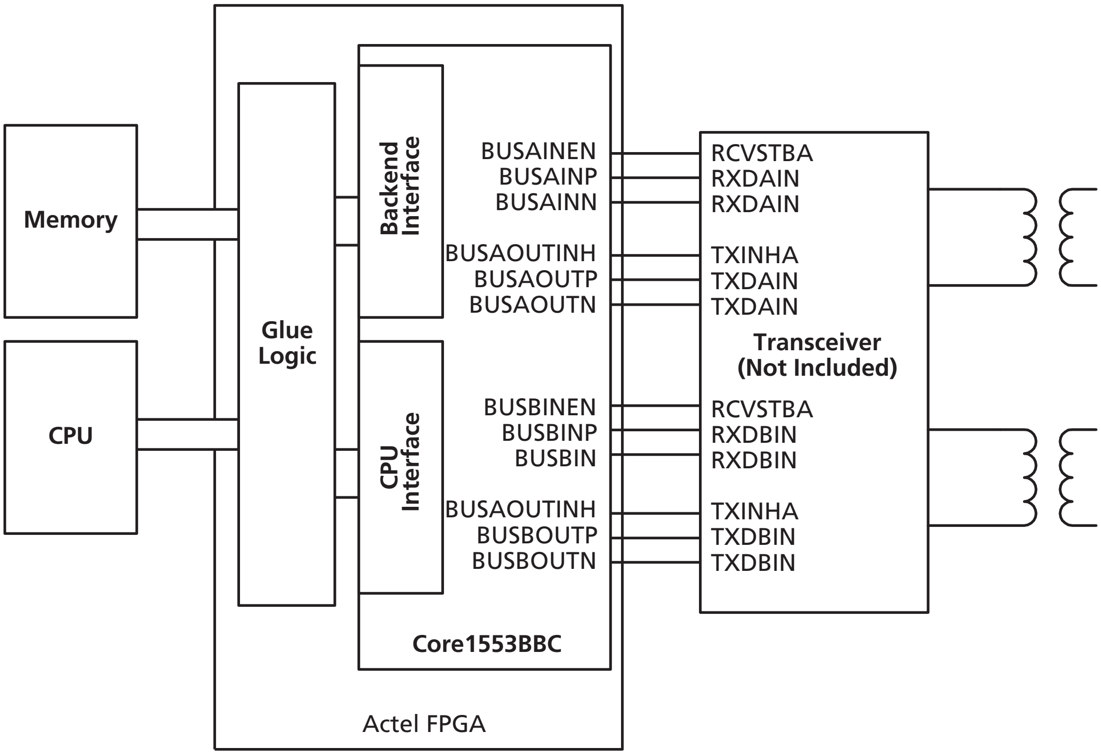
Memory
CPU
Glue Logic
Backend Interface
CPU Interface
BUSAINEN
BUSAINP
BUSAINN
BUSAOUTINH
BUSAOUTP
BUSAOUTN
BUSBINEN
BUSBINP
BUSBIN
BUSAOUTINH
BUSBOUTP
BUSBOUTN
Core1553BBC
Actel FPGA
RCVSTBA
RXDAIN
RXDAIN
TXINHA
TXDAIN
TXDAIN
Transceiver (Not Included)
RCVSTBA
RXDBIN
RXDBIN
TXINHA
TXDBIN
TXDBIN

*Figure 1* • **Typical Core1553BBC System**

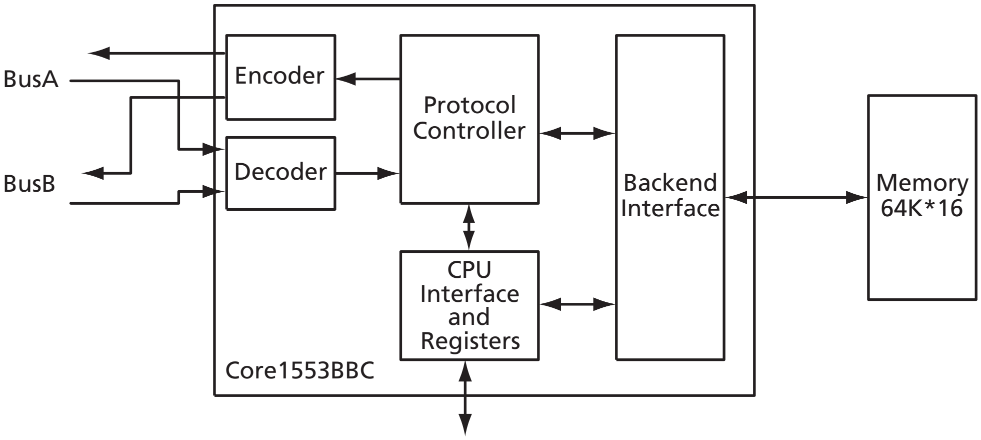
BusA
BusB
Encoder
Decoder
Protocol Controller
CPU Interface and Registers
Backend Interface
Core1553BBC
Memory 64K*16

*Figure 2* • **Core1553BBC BC Block Diagram**

A single 1553B encoder takes each word to be transmitted and serializes it using Manchester encoding. The encoder includes independent logic to prevent the BC from transmitting for greater than the allowed period and to provide loopback fail logic. The loopback logic monitors the received data and verifies that the core has correctly received every word that is transmitted. The encoder output is gated with the bus enable signals to select which buses the RT should be transmitting.

Since the BC knows which bus is in use at any time, only a single decoder is required. The decoder takes the serial Manchester received data from the bus and extracts the received data words. The decoder requires a 12, 16, 20, or 24 MHz clock to extract the data and the clock from the serial stream.

The decoder contains a digital phased lock 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 occurred in the word.

The protocol controller block handles all the message sequencing and error recovery. This is a complex state machine that reads the 1553B message frames from the memory and transmits them on the 1553B bus.

The CPU interface allows the system CPU to access the control registers within the BC. It also allows the CPU to directly access the memory connected to the backend interface. These features can simplify system design.

The backend interface for the Core1553BBC allows a simple connection to a memory device. The backend interface 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 asynchronous 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.

## Core1553BBC Operation

A bus controller is responsible for sending data bus commands, participating in data transfers, receiving status responses, and monitoring the bus system. The system CPU will create message lists in the BC memory, as illustrated in Figure 3.

When started, the BC works its way through the message lists. The Core1553B transmits the specified 1553B command and data words, and receives the 1553B status word and associated data words and writes them to the BC memory. During this process, the BC monitors all possible 1553B error conditions. If an RT does not respond correctly, the BC will retry the message on both the original bus and the alternate bus.

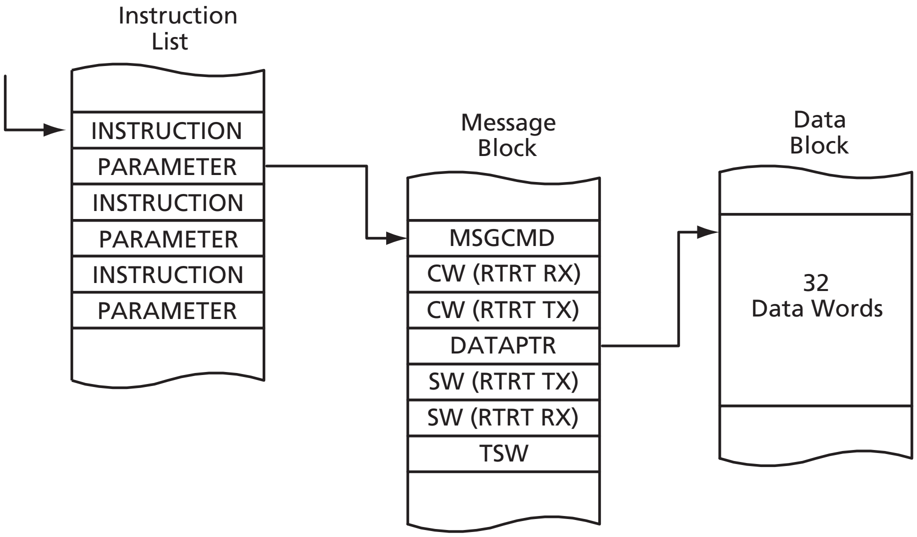

*Figure 3* • **Message Lists**

# Core1553BBC Device Requirements

The Core1553BBC can be implemented in several Actel FPGA devices. Table 1 shows typical utilization figures for the Core1553BBC implemented in these devices.

*Table 1* • **Device Utilization**

| Family | Cells or Tiles | | | Device | Utilization |
|---|---|---|---|---|---|
| | **Combinatorial** | **Sequential** | **Total** | | |
| Fusion | 1773 | 558 | 2331 | AFS600 | 17% |
| ProASIC3/E | 1773 | 558 | 2331 | A3PE600 | 17% |
| ProASICPLUS | 2250 | 560 | 2810 | APA150-STD | 46% |
| Axcelerator | 1072 | 584 | 1656 | AX500-STD | 20% |
| RTAX-S | 1072 | 584 | 1656 | RTAX250-STD | 9% |
| SX-A | 1115 | 589 | 1704 | A54SX32A-STD | 56% |
| RTSX-S | 1098 | 598 | 1696 | RT54SX32S-STD | 57% |

The Core1553BBC clock rate can be programmed to 12, 16, 20, or 24 MHz. All Actel device families listed in Table 1 easily meet this performance requirement.

When implemented in ProASICPLUS or Axcelerator devices, the Core1553BBC can connect directly to the internal FPGA memory blocks, eliminating the need for external memories.

# Core1553BBC Verification and Compliance

Core1553BBC is based upon the Actel Core1553BRT, which has been fully verified against the RT validation Test Plan (MIL-HDBK-1553A, Appendix A). This ensures that the 1553B encoders and decoders are fully compliant to the 1553B specification. The actual bus controller function has been extensively verified in both simulation and hardware. Core1553BBC has been implemented on an A54SX32A-STD part connected to external transceivers and memory.

# MIL-STD-1553B Bus Overview

The MIL-STD-1553B bus is a differential serial bus used in military and space equipment. It is comprised of multiple redundant bus connections and communicates at 1MB per second.

The bus has a single active bus controller (BC) and up to 31 remote terminals (RTs). The BC manages all data transfers on the bus using the command and status protocol. The bus controller 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 five-bit RT address, a transmit or receive bit, a five-bit sub-address and a five-bit word count. This allows for 32 RTs on the bus. However, since RT address 31 is used to indicate a broadcast transfer, only 31 RTs may be connected. Each RT has 30 sub-addresses reserved for data transfers. The other two sub-addresses (0 and 31) are reserved for mode codes. Data transfers contain up to (32) 16-bit data words. Mode code command words are used for bus control functions such as synchronization.

## Message Types

The 1553B bus supports ten message transfer types, allowing basic point-to-point and broadcast BC to RT data transfers, mode code messages, and direct RT-to-RT messages. Figure 4 shows the message formats.

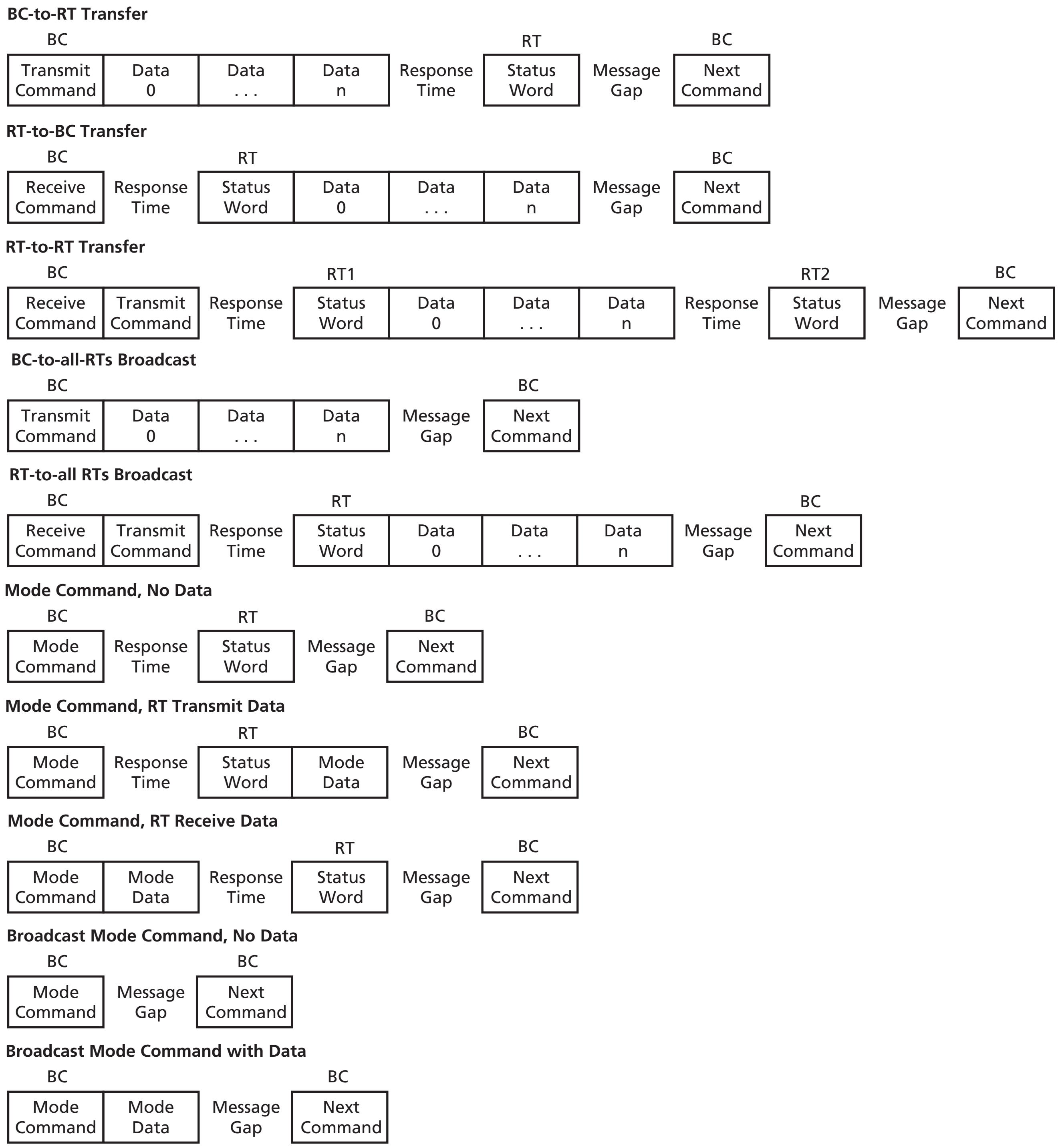
BC-to-RT Transfer
BC
Transmit Command
Data 0
Data . . .
Data n
Response Time
RT
Status Word
Message Gap
BC
Next Command
RT-to-BC Transfer
BC
Receive Command
Response Time
RT
Status Word
Data 0
Data . . .
Data n
Message Gap
BC
Next Command
RT-to-RT Transfer
BC
Receive Command
Transmit Command
Response Time
RT1
Status Word
Data 0
Data . . .
Data n
Response Time
RT2
Status Word
Message Gap
BC
Next Command
BC-to-all-RTs Broadcast
BC
Transmit Command
Data 0
Data . . .
Data n
Message Gap
BC
Next Command
RT-to-all RTs Broadcast
BC
Receive Command
Transmit Command
Response Time
RT
Status Word
Data 0
Data . . .
Data n
Message Gap
BC
Next Command
Mode Command, No Data
BC
Mode Command
Response Time
RT
Status Word
Message Gap
BC
Next Command
Mode Command, RT Transmit Data
BC
Mode Command
Response Time
RT
Status Word
Mode Data
Message Gap
BC
Next Command
Mode Command, RT Receive Data
BC
Mode Command
Mode Data
Response Time
RT
Status Word
Message Gap
BC
Next Command
Broadcast Mode Command, No Data
BC
Mode Command
Message Gap
BC
Next Command
Broadcast Mode Command with Data
BC
Mode Command
Mode Data
Message Gap
BC
Next Command

*Figure 4* • **1553B Message Formats**

## 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 20-bit word consists of a 3-bit sync pattern, 16 bits of data, and a parity bit (Figure 5).

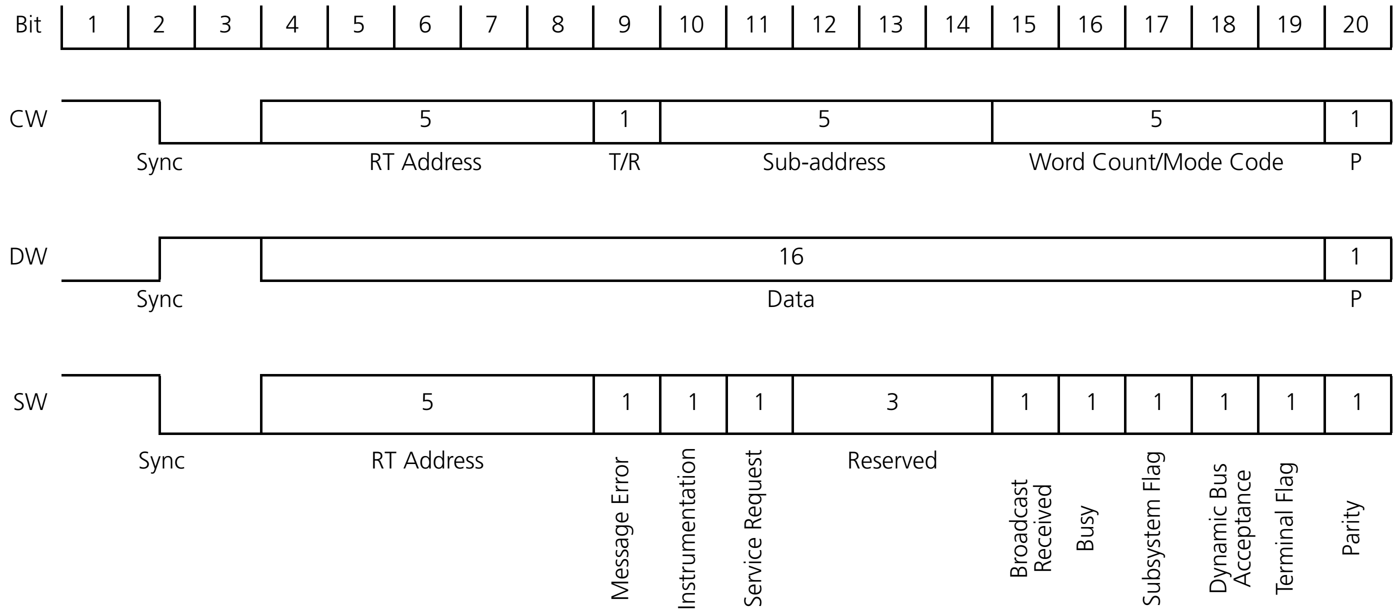
Bit
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
CW
5
1
5
5
1
Sync
RT Address
T/R
Sub-address
Word Count/Mode Code
P
DW
16
1
Sync
Data
P
SW
5
1
1
1
3
1
1
1
1
1
1
Sync
RT Address
Message Error
Instrumentation
Service Request
Reserved
Broadcast Received
Busy
Subsystem Flag
Dynamic Bus Acceptance
Terminal Flag
Parity

*Figure 5* • **1553B Word Formats**

# I/O Signal Descriptions

*Table 2* • **1553B Bus Interface**

| 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 (is held high when no transmission) |
| BUSAOUTN | Out | Negative data output to the bus A transmitter (is held high when no transmission) |
| BUSBOUTIN | Out | Active high transmitter inhibits the B transmitter |
| BUSBOUTP | Out | Positive data output to the bus B transmitter (is held high when no transmission) |
| BUSBOUTN | Out | Negative data output to the bus B transmitter (is held high when no transmission) |

*Table 3* • **Control and Status Signals**

| Name | Type | Description |
|---|---|---|
| CLK | In | Master clock input (either 12 MHz, 16 MHz, 20 MHz, or 24 MHz) |
| RSTINn | In | Reset input (active low) |
| INTOUT | Out | Interrupt Request (active high). The CPU is required to read the internal status register to find the reason for the interrupt. It is cleared by the CPU writing to the interrupt register. |
| MEMFAIL | Out | This goes high if the core fails to read 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. |
| BUSY | Out | This is high when the core is active, i.e. processing a message list. |
| EXTFLAG | In | External flag input used by the condition codes within the bus controller |

## CPU Interface

The CPU interface allows access to the Core1553BBC internal registers and direct access to the backend memory. This interface is synchronous to the clock (Table 4).

*Table 4* • **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] is '0,' data bits [15:8] are written. When CPUWRn[0] is '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.<br>Timing is shown in the Figure 12 on page 24 and Figure 13 on page 25. |
| CPUMEM | In | Selects whether the CPU accesses internal registers or backend memory.<br>'0': Accesses internal registers, register number is specified on CPUADDR[2:0]<br>'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. |

## Backend Interface

The backend interface supports both synchronous operation and asynchronous operation to backend devices. Synchronous operation directly supports the use of internal FPGA memory blocks. Asynchronous operation allows connection to standard external memory devices.

*Table 5* • **Backend Signals**

| Name | Type | Description |
|---|---|---|
| MEMREQn | Out | Memory Request (active low) output. The BC holds MEMREQn active if it requires additional memory access cycles to take place immediately after the current memory cycle. This occurs during the inter-message gap. |
| MEMGNTn | In | Memory Grant (active low) input. This input should be synchronous to CLK and needs to meet the internal register setup time. This input may be held low if the core has continuous access to the RAM. |
| MEMWRn[1:0] | Out | Memory Write (active low). When MEMWRn[1] is '0,' D[15:8] is written. When MEMWRn[0] is '0,' D[7:0] is written.<br>Synchronous mode: This output indicates that data will be written on the rising clock edge. If MEMWAITn is asserted, the MEMWRn pulse will be extended until MEMWAITn becomes inactive.<br>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)<br>Synchronous mode: This output indicates that data is read on the next rising clock edge. If MEMWAITn is active, then 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.<br>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 |
| 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. |

The backend interface must allow the bus controller access to the memory when requested. The memory access time from MEMREQn low to completion of the access cycle MEMRDn and MEMWRn high varies depending on the BC setup. When the CPU is allowed to access the memory through the bus controller (CPUMEMEN active), the memory access time is reduced (Table 6 on page 9).

If the backend fails to allow the bus controller access to the memory in the required time, the bus controller will assert its MEMFAIL output and stop operation.

## Miscellaneous I/O

Several inputs are used to modify the core functionality to simplify integration in the application. These inputs should be tied to logic '0' or logic '1' as appropriate (Table 7).

*Table 6* • **Memory Access Requirements**

| CPUMEMEN | CLK Speed MHz | Memory Access Time |
|---|---|---|
| 0 | 12 | 9.58µs |
| 0 | 16 | 9.68µs |
| 0 | 20 | 9.75µs |
| 0 | 24 | 9.79µs |
| 1 | 12 | 4.58µs |
| 1 | 16 | 4.68µs |
| 1 | 20 | 4.75µs |
| 1 | 24 | 4.79µs |

*Table 7* • **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 buses are connected to the same system bus. |

## Bus Controller Registers

The bus controller has nine internal registers used to control the bus controller operation and provide status information (Table 8).

*Table 8* • **Bus Controller Registers**

| Address | Name | Type | Size | Function |
|---|---|---|---|---|
| 000 | CONTROL | W | [3:0] | Allows the CPU to control the BC |
| 000 | STATUS | R | [15:0] | Provides status information |
| 001 | SETUP | RW | [15:0] | BC setup register |
| 010 | LISTPTR | RW | [15:0] | Current LISTPTR value. The address of the current instruction being executed. At the start of operation, the CPU should set this to the point at the first instruction. This value will automatically step through the BC instruction list. |
| 011 | MSGPTR | R | [15:0] | Current MSGPTR value. Provides the address of the message block being processed. |
| 100 | CLOCK | RW | [15:0] | BC internal clock value<br>This 16-bit value counts up at a 1µs, 4µs, 8µs, or 32µs rate. This gives a maximum timer value of 2 seconds. The CPU may directly load the counter. |
| 101 | ASYNCPTR | RW | [15:0] | Asynchronous list pointer<br>Provides a pointer to a list of messages that will be processed when started by the ASYNC message list bit in the control register. |

*Table 8* • **Bus Controller Registers**

| Address | Name | Type | Size | Function |
|---|---|---|---|---|
| 110 | STACKPTR | RW | [15:0] | BC stack pointer<br>This is the internal stack pointer register; it is used for the CALL and RETURN instructions. When the bus controller is started, the STACKPTR is set to FFFF. The upper eight bits are fixed to FF, and the lower eight bits will count down and up. This allows up to 255 addresses to be stored in the stack memory. |
| 111 | INTERRUPT | RW | [15:0] | Interrupt Register |

*Table 9* • **Setup Register**

| Bits | Name | Type | Reset | Function |
|---|---|---|---|---|
| 15 | FORCEORUN | RW | 0 | '1': If a BC-RT message with a word count between 1 and 31 is carried out, the BC will transmit for greater than 680µs. This will cause the transmitter timer to trigger and the BC to shutdown.<br>'0': Normal operation |
| 14 | CLOCKEN | RW | 0 | Enables the internal CLOCK to count<br>'0': Internal CLOCK will not count<br>'1': Internal CLOCK enabled<br>The clock is automatically enabled by the WAITC instruction. |
| 13:12 | CLKFREQ | RW | 01 | Tells the core what the external clock frequency is<br>00: 12 MHz<br>01: 16 MHz<br>10: 20 MHz<br>11: 24 MHz |
| 11 | RETRYMODE | WR | 0 | Sets how the retry system works<br>'0': Retries on the same bus for the number of times set by the reties setting in the message block, then on the alternate bus for the number of times set by the alternate bus reties in the message block.<br>'1': Reties alternates between the two buses. The total number of retries is the number of reties plus alternative bus retries as set in the message block. |
| 10 | INTENABLE | RW | 0 | Enables the external interrupt pin<br>'1': The INTPENDING bit will drive the INTOUT pin<br>'0': The INTOUT pin is held at a '0' |
| 9 | AUTOCLOCK | RW | 1 | '1': Sets the CLOCK register to 0000 when the BC is started<br>'0': The CLOCK register is not reset when the BC is started |
| 8 | AUTOSTACK | RW | 1 | '1': Sets the STACKPTR register to FFFF when the BC is started<br>'0': The STACKPTR register is not reset when the BC is started. This allows the BC to be restarted when previously stopped. |
| 7:6 | CLKRATE | RW | 00 | Sets the rate at which the TIMER and CLOCK count<br>00: 1µs<br>01: 4µs<br>10: 8µs<br>11: 32µs |
| 5:4 | IMG | RW | 00 | Sets the default minimum inter-message GAP<br>00: 4µs<br>01: 8µs<br>10: 16µs<br>11: 32µs<br>**Note:** The actual inter-message GAP is a function of the memory access times. Typically, six memory accesses need to take place in the inter-message gap. |

*Table 9* • **Setup Register (Continued)**

| Bits | Name | Type | Reset | Function |
|---|---|---|---|---|
| 3:2 | RESPTIME | RW | 01 | Sets the maximum time that the BC will wait for an RT to respond<br>00: 12µs<br>01: 16µs<br>10: 20µs<br>11: 24µs |
| 1:0 | Reserved | R | 00 | Reserved, return 00 |

*Table 10* • **Control Register**

| Bits | Name | Type | Function |
|---|---|---|---|
| 3 | ASYNC | W | Writing a '1' causes the bus controller to jump to process the asynchronous instruction list pointed to by the ASYNCPTR register at the end of the current message. When a RETAS instruction is found, the bus controller returns to the original instruction list. An ASYNC instruction can be issued while the bus controller is both active and inactive. |
| 2 | ABORT | W | Writing a '1' stops the bus controller immediately; both normal and asynchronous message operation will be aborted. |
| 1 | STOP | W | Writing a '1' stops the bus controller at the end of the current message. |
| 0 | START | W | Writing a '1' starts the bus controller. The bus controller cannot be started when an asynchronous message is active. |

*Table 11* • **Status Register**

| Bits | Name | Type | Function |
|---|---|---|---|
| 15:8 | VERSION | R | Indicates the Core1553BBC code revision<br>Core release notes provide latest version numbers. |
| 7:6 | Reserved | R | Reserved, set to 00 |
| 5 | LOOPFAIL | R | Indicates that a loopback failure occurred in the current frame |
| 4 | FRAMEOK | R | Indicates that all the messages in the current frame have completed successfully and no system action is required |
| 3 | FLAG | R | Indicates the value of the flag condition stored by the STOREFLAG instruction |
| 2 | ASYNC | R | Asynchronous message requested or in progress. This bit is cleared by the RETAS instruction. When it is active, the bus controller cannot be started. |
| 1 | BUSINUSE | R | Indicates which bus is in use<br>0: Bus A<br>1: Bus B |
| 0 | ACTIVE | R | BC is Active |

*Table 12* • **Interrupt Register**

<table>
<tr><th>Bits</th><th>Name</th><th>Type</th><th colspan="2">Function</th></tr>
<tr><td>15</td><td>INTPENDING</td><td>RW</td><td colspan="2">When set, the BC has an interrupt pending. This bit is set if any of the INTVECT bits are set.</td></tr>
<tr><td rowspan="8">14:8</td><td rowspan="8">INTVECT</td><td rowspan="8">RW</td><td colspan="2">Interrupt Reason<br>The CPU writing a '1' to the bit clears the bit.</td></tr>
<tr><td>14</td><td>BC has completed the message list, HALT instruction executed</td></tr>
<tr><td>13</td><td>INTREQ instruction executed</td></tr>
<tr><td>12</td><td>Memory access failure. This bit also directly drives the MEMFAIL output.</td></tr>
<tr><td>11</td><td>Asynchronous message is completed, RETAS instruction is executed.</td></tr>
<tr><td>10</td><td>Transmitter shutdown is set when the core detects that it has been transmitting continuously on the bus for greater than 700µs. When set, the BC disables its transmitter.</td></tr>
<tr><td>9</td><td>Stack pointer overflow or underflow is set if the BC attempts to push more than 256 return addresses onto the stack or pop of a non-existent address from the stack. The BC stops operation when this occurs.</td></tr>
<tr><td>8</td><td>Corrupt instruction list or data table.<br>Illegal command written to the control register, e.g. start instruction while an asynchronous message is active. The BC stops operation when this occurs.</td></tr>
<tr><td>7:0</td><td>USERVECT</td><td>R</td><td colspan="2">Provides the user-supplied interrupt reason as set by the instruction parameter</td></tr>
</table>

## Bus Controller Operation

After power-up, the bus controller waits while the CPU sets up the bus controller memory and registers. The memory contains an instruction list, message blocks, and data blocks. Once the instruction list, message blocks, and data blocks are setup, the CPU starts the bus controller. The bus controller works its way through all the message blocks until it reaches the end of the instruction list (Figure 6).

The instruction list contains a list of pointers to message blocks. The message block contains the command words transmitted on the 1553B bus and status words received from the 1553B bus. It also contains a pointer to a data block. The data block contains the data transmitted on the 1553B bus, or the data received from the 1553B bus.

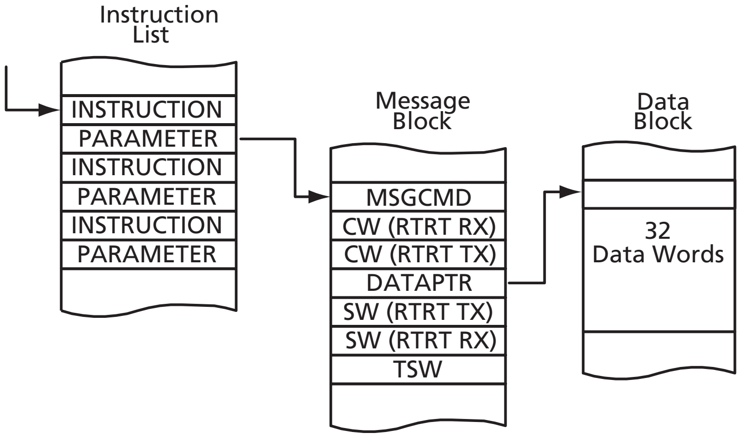

*Figure 6* • **BC Memory Usage**

## Instruction List

The instruction list contains pairs of words: an instruction and a parameter. Core1553BBC supports a broad set of instructions allowing branching and sub-routine calls with condition code support. This allows complex instruction lists to be supported. The instruction contains a 4-bit OPCODE and a 5-bit condition code field (Table 13 and Table 14).

*Table 13* • **Instruction Word**

| 15:13 | 12:8 | 7:4 | 3:0 |
|---|---|---|---|
| Reserved | CONDCODE | Reserved | OPCODE |

All of the OPCODES support the condition code field. If the condition is TRUE, then the OPCODE is carried out; otherwise, the BC continues to the next instruction. For RT-to-RT messages, the condition code will be true if the bit is set in either status word or not set in either status word (Table 15 on page 14).

*Table 14* • **Supported Instructions**

| OPCODE | Function | Condition Code | Parameter | Description |
|---|---|---|---|---|
| 0000 | NOP | N/A | N/A | No operation, jumps to next message |
| 0001 | DOMSG | Yes | Message Block Address | Process the message block |
| 0010 | JUMP | Yes | New Instruction Address | Jumps to the new message list address |
| 0011 | INTR | Yes | User interrupt value (Lower eight bits) | Force a BC interrupt |
| 0100 | HALT | Yes | User interrupt value (Lower eight bits) | Stop the BC |
| 0101 | DELAY | Yes | Timer value (Lower eight bits) | Loads the timer with the parameter and waits until the timer reaches zero |

*Table 14* • **Supported Instructions (Continued)**

| OPCODE | Function | Condition Code | Parameter | Description |
|---|---|---|---|---|
| 0110 | LOADC | Yes | Clock value | Loads the BC clock |
| 0111 | WAITC | Yes | Clock value | Waits until the BC clock reaches the specified value |
| 1000 | CALL | Yes | New Instruction Address | Jumps to the new message list address and pushes the return address onto the stack |
| 1001 | RET | Yes | N/A | Gets a return instruction address from the stack and jumps to it |
| 1010 | RETAS | Yes | N/A | Gets a return instruction address from the stack and jumps to it. Also clears the ASYNC bit in the status register allowing further ASYNC messages to be accepted. |
| 1011 | STOREF | Yes | N/A | Stores the selected flag so that it may be tested at a later stage. The condition code field indicates which flag to store. |
| Others | Illegal | N/A | N/A | Will halt operation and set the illegal OPCODE interrupt |

*Table 15* • **Condition Codes**

| Condition Code | Function | Description |
|---|---|---|
| 00000 | ALWAYS | Always perform the associated instruction |
| 00001 | RESP | Performs the instruction if there was a response to the previous message |
| 00010 | GBR | Performs the instruction if the previous message was successful |
| 00011 | TF | Performs the instruction if the Terminal Flag was set in the last received status word |
| 00100 | DBA | Perform the instruction if the Dynamic Bus Acceptance flag was set in the last received status word |
| 00101 | SSF | Performs the instruction if the Sub-system Flag was set in the last received status word |
| 00110 | BUSY | Performs the instruction if the Busy bit was set in the last received status word |
| 00111 | BR | Performs the instruction if the Broadcast Received bit was set in the last received status word |
| 01000 | SR | Performs the instruction if the Service Request bit was set in the last received status word |
| 01001 | ME | Performs the instruction if the Message Error bit was set in the last received status word |
| 01010 | SWE | Performs the instruction if the RT address field is incorrect, the instrumentation bit is set, or any of the reserved bits was set in the last received status word |
| 01011 | SWAB | Performs the instruction if any bits are set in the last received status word (ignoring the RT address field) |
| 01100 | ASYNC | Performs the instruction if asynchronous message processing is active |
| 01101 | EXT | Performs the instruction if the External flag input is active '1' |
| 01110 | SFLAG | Performs the instruction if the previously stored flag bit was set |
| 10000 | NEVER | Never performs the associated instruction |
| 10001 | NORESP | Performs the instruction if there was no response to the previous message |
| 10010 | NGBR | Performs the instruction if the previous message was unsuccessful |
| 10011 | NTF | Performs the instruction if the Terminal Flag was not set in the last received status word |
| 10100 | NDBA | Performs the instruction if the Dynamic Bus Acceptance flag was not set in the last received status word |
| 10101 | NSSF | Performs the instruction if the Sub-System Flag was not set in the last received status word |
| 10110 | NBUSY | Performs the instruction if the Busy bit was not set in the last received status word |

*Table 15* • **Condition Codes (Continued)**

| Condition Code | Function | Description |
|---|---|---|
| 10111 | NBR | Performs the instruction if the Broadcast Received bit was not set in the last received status word |
| 11000 | NSR | Performs the instruction if the Service Request bit was not set in the last received status word |
| 11001 | NME | Performs the instruction if the Message Error bit was not set in the last received status word |
| 11010 | NSWE | Performs the instruction if the RT address field is correct and the instrumentation bit is not set, and none of the reserved bits are set in the last received status word |
| 11011 | NOSWAB | Performs the instruction if no bits (ignoring the RT address field) are set in the last received status word |
| 11100 | NASYNC | Performs the instruction if asynchronous message processing is not active |
| 11101 | NEXT | Performs the instruction if the External flag input is inactive '0' |
| 11110 | NSFLAG | Performs the instruction if the previously stored flag bit was not set |
| Others 0xxxx | Illegal | Equivalent to NEVER |
| Others 1xxxx | Illegal | Equivalent to ALWAYS |

## Message Block

An 8-word message block controls each message. The BC reads the 1553B command words from the message block and will write the received status words back to message block. Message blocks must be positioned on an 8-word memory boundary (Table 16).

*Table 16* • **Message Block**

| Offset | Contents | Written by | Description | | | |
|---|---|---|---|---|---|---|
| 0 | MSGCMD | CPU | Type of 1553B Message | | | |
| | | | 15:10 | Inter-message GAP (IMG) after this message, 0 to 63µs.<br>When 000000, it uses the inter-message gap as set by the BC setup register. Longer IMG values can be achieved by using the DELAY instruction between messages.<br>The actual IMG may be llonger than when set by this register. The BC needs to perform up to six memory accesses during the IMG period. If the backend memory responds slowly, then the IMG may increase | | |
| | | | 9 | '0': Normal Operation<br>'1': The CLOCK value at the mid-point of the data word sync is used as the data for the synchronize with data mode code. | | |
| | | | 8: | Bus to use<br>'0': Bus A<br>'1': Bus B | | |
| | | | 7:6 | Retries on Alternate Bus 0 to 3 | | |
| | | | 5:4 | Retries on Bus 0 to 3 | | |
| | | | 3:0 | 0000 | 0 | BC-to-RT |
| | | | | 0001 | 1 | RT-to-BC |
| | | | | 0110 | 6 | Mode Code with no data |
| | | | | 0010 | 2 | Mode Code RT RX with data |
| | | | | 0011 | 3 | Mode Code RT TX with data |
| | | | | 0101 | 5 | RT-to-RT |
| | | | | 1000 | 8 | Broadcast BC-to-RT |
| | | | | 1110 | E | Broadcast Mode Code RT with no data |
| | | | | 1010 | A | Broadcast Mode Code RX with data |
| | | | | 1101 | D | Broadcast RT-to-RT |
| | | | | Others | Illegal | |
| 1 | CW | CPU | 1553B Command Word. For RT-to-RT messages, this is the RX command word | | | |
| 2 | RTRT TX CW | CPU | RT-to-RT TX 1553B Command Word | | | |
| 3 | DATAPTR | CPU or BC | Data Messages | Pointer to memory location containing the 32-word data buffer. Must be on a 32-word boundary, i.e. bits 4:0 are 00000 | | |
| | | | Mode Code Messages | Mode code data word | | |
| 4 | SW | BC | Received status word. For RT-to-RT messages this is the status word from the TX RT | | | |
| 5 | RTRT RX SW | BC | Received status word from the RX RT for RT-to-RT messages | | | |

*Table 16* • **Message Block (Continued)**

| Offset | Contents | Written by | Description | |
|---|---|---|---|---|
| 6 | TSW | BC | Message Transfer Status Word. Provides status information on the message block. The CPU should clear this field when setting up the message block | |
| | | | 15 | Message Okay<br>'1': Message completed okay with no errors. May have been retried.<br>'0': Message failed after all retries completed |
| | | | 14 | No response from RT |
| | | | 13 | RT Signaled Message Error |
| | | | 12 | 1553B Error occurred, encoding, parity, too many words, etc. |
| | | | 11:9 | Number of retry attempts to transmit the message (0-6) |
| | | | 8 | Unexpected bit set in the status word |
| | | | 7 | Got the RT-to-RT RX status word |
| | | | 6 | Got the status word, for RT-to-RT got the TX status word |
| | | | 5:0 | Number or data words transmitted or received.<br>Note: '000000' indicates 0 and '100000' indicates 32 |
| 7 | Reserved | – | 15:0 | Not Used |

## Detailed Operation Flow

Table 17 shows the operations the core goes through in processing a message list containing two messages. The first message is a BC-to-RT transfer of three words, and the second is an RT-to-BC transfer of three words.

*Table 17* • **Typical Operation**

| Time | Memory Accesses | Operation | 1553B Activity |
|---|---|---|---|
| ↓ | Read first Instruction code | | |
| | Read first Instruction parameter | | |
| | Read first MSGCMD | | |
| | Read first CW | | |
| | | Wait until the Inter-message gap expired | |
| | Read the DATAPTR | | Transmit the command word |
| | Read the 1st DW | | |
| | Read the 2nd DW | | Transmit the 1st data word |
| | Read the 3rd DW | | Transmit the 2nd data word |
| | | | Transmit the 3rd data word |
| | | Wait for the RT Response Time | RT responds with SW |
| | Write the SW to memory | | |
| | Write the TSW to memory | | |
| | Read second Instruction code | | |
| | Read second instruction parameter | | |
| | Read second MSGCMD | | |
| | Read second CW | | |
| | | Wait until the Inter-message gap expired | |
| | Read the DATAPTR | Wait for the RT Response Time | Transmit the command word |
| | | | RT responds with SW |
| | Write the SW to memory | | RT sends 1st data word |
| | Write the 1st DW to memory | | RT sends 2nd data word |
| | Write the 2nd DW to memory | | RT sends 3rd data word |
| | Write the 2nd DW to memory | | |
| | Write the TSW to memory | | |
| | Read third Instruction code (HALT) | | |
| | | Generate the complete interrupt | |

## Error Conditions

Core1553BBC monitors bus errors and in most cases will perform automatic retry operations if recovery is possible (Table 18).

*Table 18* • **Error Conditions**

| **Error Condition** | | **Action** |
|---|---|---|
| **Group** | **Error** | |
| Signaling | 1553B signaling error, parity, Manchester error, too many or to few words, or incorrect SYNC type | Message is retried |
| | 1553B Loopback Failure. Can occur if an RT responds late, causing the RT response and following command word to corrupt each other on the bus | Message is retried<br>Loopback bit set in BC status<br>BC continues to process messages |
| | Transmitter Overrun. Internal timer detects the BC has transmitted for greater than 688µs. | BC controller aborts and asserts the transmitter shutdown interrupt |
| Memory | Memory Access Failure | BC controller aborts and asserts the memory failure interrupt |
| | Stack Overflow or Underflow | BC controller aborts and asserts the stack overflow interrupt |
| Status Word | Terminal Flag in SW<br>Sub-system Flag in SW<br>Service Request Flag in SW<br>Broadcast bit is SW | Unexpected bit in 1553B status bit set in the TSW. Message is not retried. |
| | Busy Flag in SW<br>Message Error bit in SW | Message is retried |
| | Other SW bit | Message is retried |
| RT Response | No or Late Response | Message is retried |
| Miscellaneous | Corrupt Instruction List<br>Illegal OPCODE<br>Message block MSGCMD message type bits [3:0] mismatch the provided command word | BC controller aborts and asserts the corrupt instruction list interrupt. |
| Retry Fails | Retries do not correct the error | Message Okay bit in TSW not set |
| CPU Interface | Start or second asynchronous message command issued while an asynchronous message is active | Command is ignored and an illegal command interrupt is generated. |

## Loop Back Tests

The Core1553BBC 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 the BC status register.

## Message Sequence Control

Core1553BBC message sequence control enables it to automatically sequence messages without CPU intervention. It supports conditional jumps and sub-routine calls as well as time control functions.

All instructions make use of the condition codes. The condition codes cover error conditions, 1553B status word values, and an external input. Core1553BBC supports CALL and RETURN instructions with the aid of a stack that allows for 255 return addresses to be stored. The stack occupies the top 256 words of memory.

To support message timing and minor/major frame timing, Core1553BBC has a built-in real-time clock (16-bit) and timer (8-bit) that can be used to synchronize message timing. The real time clock and timer have a programmable resolution of 1µs, 4µs, 8µs, or 32µs. Messages can be programmed to be sent at an absolute time or relative to the end of the previous message.

## Asynchronous Messages

Core1553BBC supports asynchronous messages. While idle, or when a normal message list is being processed, the CPU can initiate the core to jump to a secondary (asynchronous) message list and process these messages. When complete, the core will go back to the original message list.

The asynchronous message list can be started directly by the CPU by writing to the control register. When the current message completes, the core pushes the current LISTPTR address on the stack and loads the LISTPTR with the value specified in the ASYNCPTR. It will execute these instructions until a RETAS instruction is found. At this point, the LISTPTR is reloaded from the stack and the bus controller enters the idle state or resumes the original instruction list. While the asynchronous message list is being processed, the START instruction and further asynchronous events are disabled. They are re-enabled by the RETAS instruction.

## Retry Operations

Core1553BBC supports an automatic retry system that retries messages that fail automatically. On detecting an error that can be retried, the BC immediately retries the message. Each message can be retried up to six times. The Core1553BBC can be programmed to retry up to three times on the original bus, then retry up to three times on the alternative bus, or to simply retry initially on the alternative bus and then switch buses after each attempt.

## Inter-Message Gap (IMG) Control

Core1553BBC provides several ways to control the 1553B inter-message gaps. First, a default IMG is programmed into the Core1533BBC control register. Secondly every message block can be programmed with its own inter-message gap; this specifies the delay to the next message. Finally, the WAITC and DELAY instructions can be used to insert extra delays between messages.

The actual IMG gap is also a function of the backend memory access system. There is a six-cycle overhead required between each message to read and write to the message block. These six memory accesses directly effect the inter-message gap. The actual IMG will be the largest of the duration of these six memory cycles or the programmed IMG value.

# Bus Transceivers

Core1553BBC needs a 1553B transceiver to drive the 1553B bus. Core1553BBC is designed to directly interface to common MIL-STD-1553 transceivers, such as the DDC BU-63147 and the Aeroflex ACT4402. When using ProASIC$^{\underline{\text{PLUS}}}$ or Axcelerator, level translators are required to connect the 5V output levels of the 1553B transceivers to the 3.3V input levels of the FPGA.

In addition to the transceiver, a pulse transformer is required for interfacing to the 1553B bus. Figure 7 on page 21, Figure 8 on page 22, and Figure 9 on page 23 show the connections required from the Core1553BBC to the transceivers and then to the bus via the pulse transformers.

# Development System

A complete 1553B Bus controller development system is also available. The Actel part number is “Core1553BBC Eval Board." The development system implements a PCI to 1553B bus controller on a single PCB using an Actel A54SX32A FPGA.

The PCI target interface uses the Actel CorePCI66 PCI target interface core.

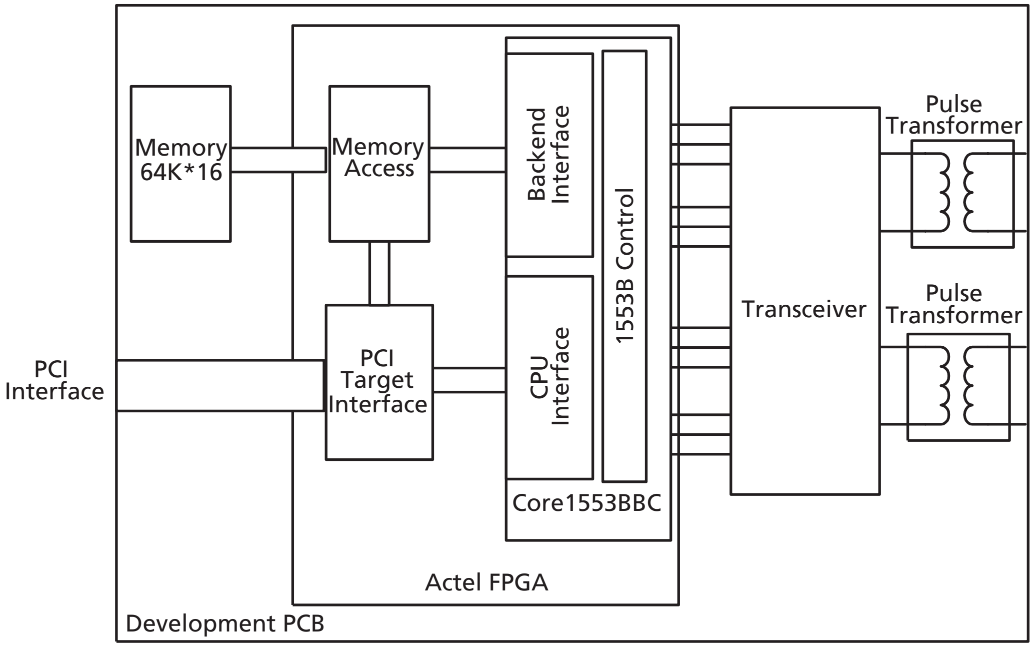
Memory
64K*16
Memory
Access
Backend
Interface
1553B Control
Pulse
Transformer
PCI
Interface
PCI
Target
Interface
CPU
Interface
Transceiver
Pulse
Transformer
Core1553BBC
Actel FPGA
Development PCB

*Figure 7* • **Core1553BBC Development System**

# Typical BC System

Core1553BBC requires a master CPU to set up the data tables. The CPU needs to be able to access the internal core registers as well as the backend memory. Core1553BBC can be configured in two ways with the CPU shared memory and 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 Core1553BBC. This configuration also supports using an internal FPGA memory connected to the core and removes the need for external bus arbitration on the CPU bus.

Alternatively, the core can share the CPU memory as shown in Figure 9 on page 23. 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, the core needs to read or write the memory it uses MEMREQn and MEMGNTn signals to arbitrate for the CPU bus before completing the cycle.

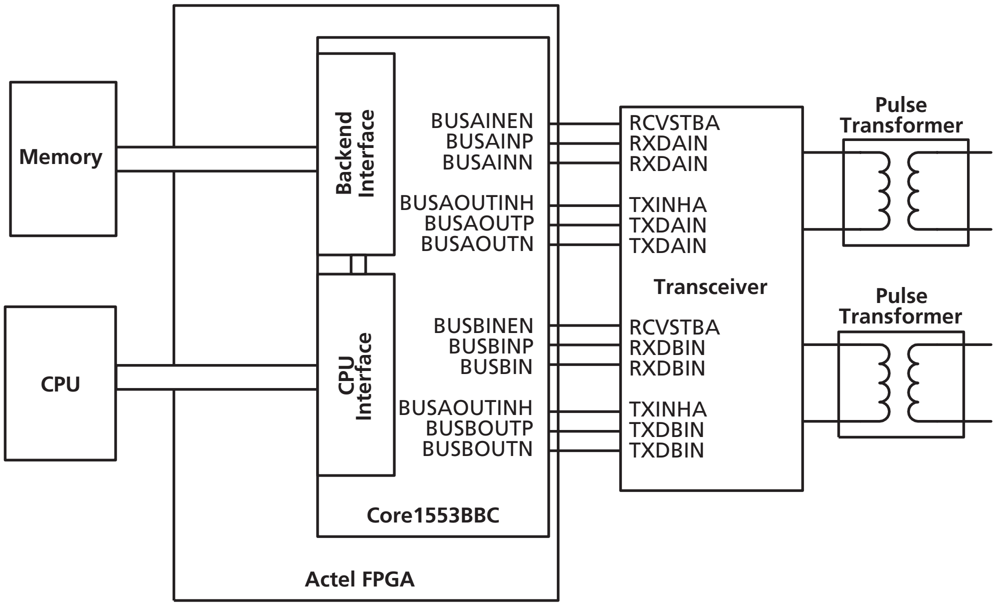
Memory
CPU
Backend Interface
CPU Interface
BUSAINEN
BUSAINP
BUSAINN
BUSAOUTINH
BUSAOUTP
BUSAOUTN
BUSBINEN
BUSBINP
BUSBIN
BUSAOUTINH
BUSBOUTP
BUSBOUTN
Core1553BBC
Actel FPGA
RCVSTBA
RXDAIN
RXDAIN
TXINHA
TXDAIN
TXDAIN
Transceiver
RCVSTBA
RXDBIN
RXDBIN
TXINHA
TXDBIN
TXDBIN
Pulse Transformer
Pulse Transformer

*Figure 8* • **Core1553BBC with Its Own Memory**

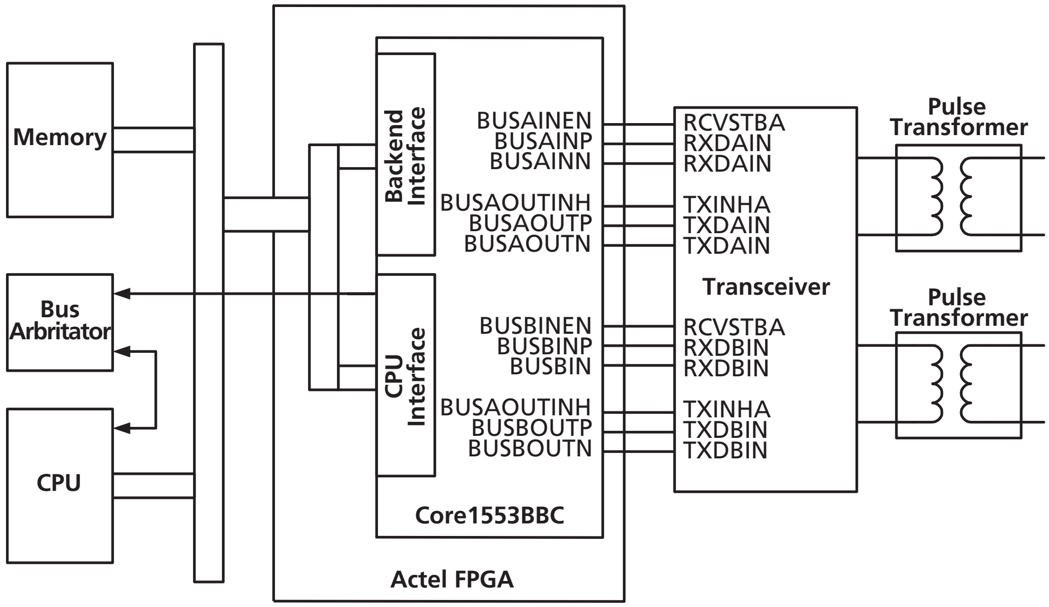
Memory
Bus Arbitrator
CPU
Backend Interface
CPU Interface
BUSAINEN
BUSAINP
BUSAINN
BUSAOUTINH
BUSAOUTP
BUSAOUTN
BUSBINEN
BUSBINP
BUSBIN
BUSAOUTINH
BUSBOUTP
BUSBOUTN
Core1553BBC
Actel FPGA
RCVSTBA
RXDAIN
RXDAIN
TXINHA
TXDAIN
TXDAIN
Transceiver
RCVSTBA
RXDBIN
RXDBIN
TXINHA
TXDBIN
TXDBIN
Pulse Transformer
Pulse Transformer

*Figure 9* • **Core1553BBC Using Shared Memory**

# Specifications

## CPU Interface Timing

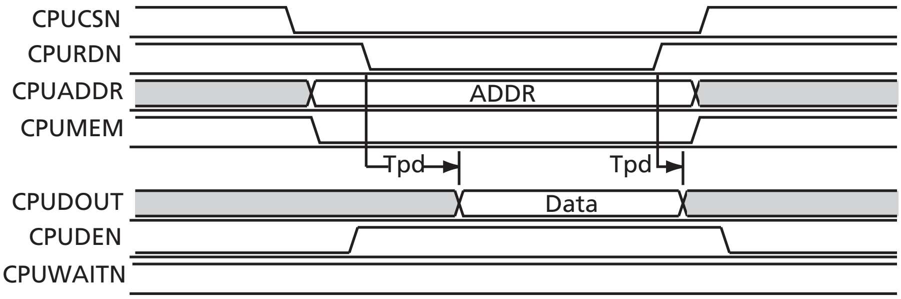
CPUCSN
CPURDN
CPUADDR
ADDR
CPUMEM
Tpd
Tpd
CPUDOUT
Data
CPUDEN
CPUWAITN

*Figure 10* • **CPU Interface Register Read Cycle**

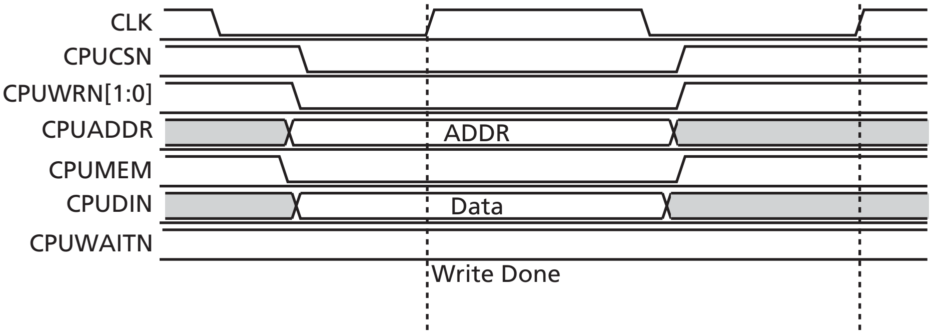
CLK
CPUCSN
CPUWRN[1:0]
CPUADDR
ADDR
CPUMEM
CPUDIN
Data
CPUWAITN
Write Done

*Figure 11* • **CPU Interface Register Write Cycle**

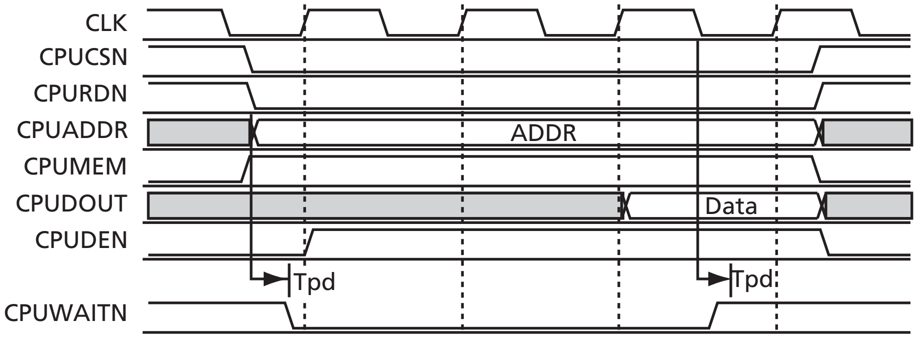
CLK
CPUCSN
CPURDN
CPUADDR
ADDR
CPUMEM
CPUDOUT
Data
CPUDEN
Tpd
Tpd
CPUWAITN

*Figure 12* • **CPU Interface Memory Read Cycle**

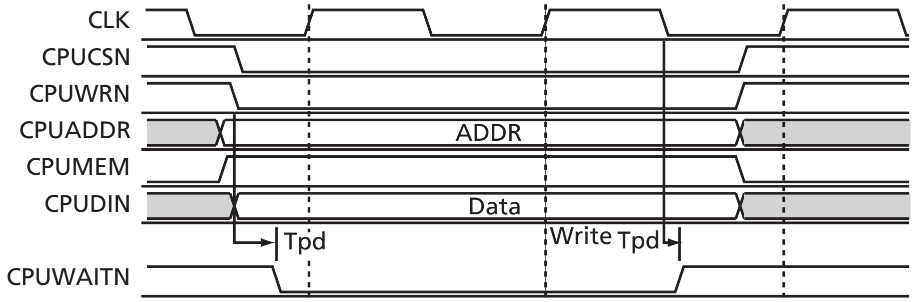
CLK
CPUCSN
CPUWRN
CPUADDR
ADDR
CPUMEM
CPUDIN
Data
Tpd
Write
Tpd
CPUWAITN

*Figure 13* • **CPU Interface Memory Write Cycle**

CPUWAITn will be driven low for a minimum of three (3) clock cycles for write cycles, four (4) for read cycles, and the number of clock cycles the memory backend delays the assertion of MEMGNTn and asserts MEMWAITn. CPUWAITn is driven low by CPURDn/CPUWRn becoming active and returns high on the falling clock edge after data is valid.

The CPU interface signals are internally synchronized to the Core1553BBC master clock. If these inputs are asynchronous, then CPUCSn, CPUADDR, and CPUDATA should be valid/invalid before CPUWRn, and remain valid after CPUWRn. CPUWRn must be active for at least one clock cycle.

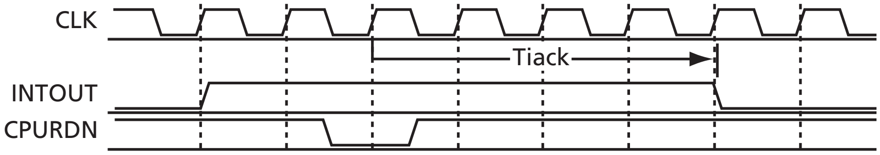
CLK
Tiack
INTOUT
CPURDN

*Figure 14* • **Interrupt Timing**

## Memory Timing

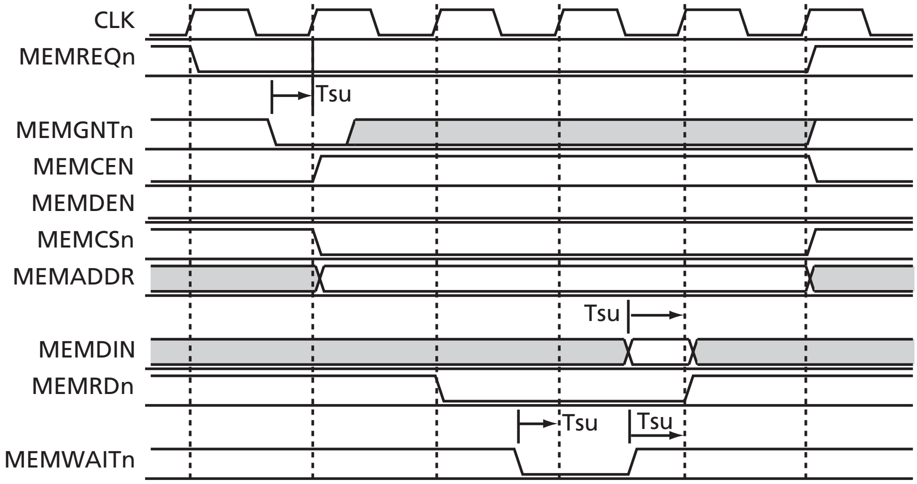
CLK
MEMREQn
Tsu
MEMGNTn
MEMCEN
MEMDEN
MEMCSn
MEMADDR
Tsu
MEMDIN
MEMRDn
Tsu
Tsu
MEMWAITn

*Figure 15* • **Asynchronous Memory Read Cycle**

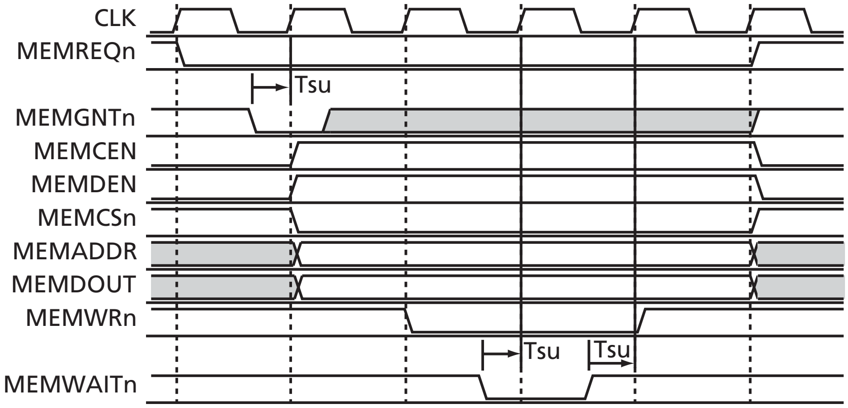
CLK
MEMREQn
Tsu
MEMGNTn
MEMCEN
MEMDEN
MEMCSn
MEMADDR
MEMDOUT
MEMWRn
Tsu
Tsu
MEMWAITn

*Figure 16* • **Asynchronous Memory Write Cycle**

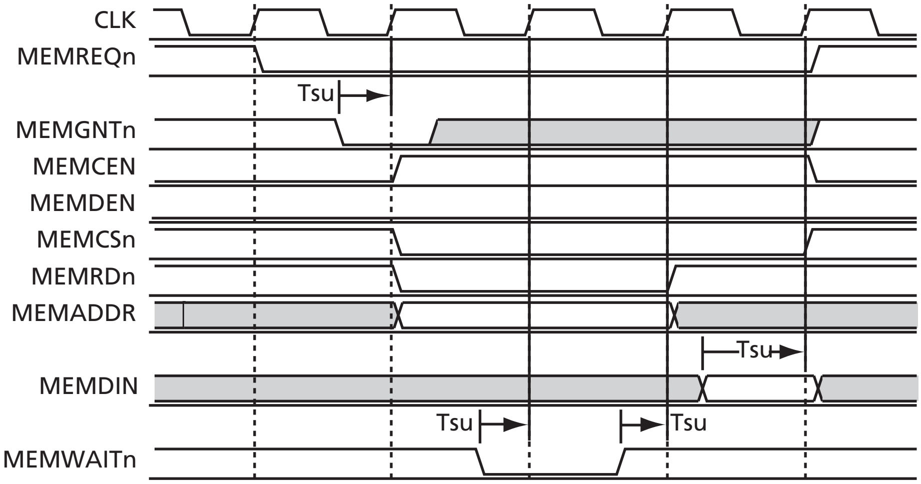
CLK
MEMREQn
Tsu
MEMGNTn
MEMCEN
MEMDEN
MEMCSn
MEMRDn
MEMADDR
Tsu
MEMDIN
Tsu
Tsu
MEMWAITn

*Figure 17* • **Synchronous Memory Read Cycle**

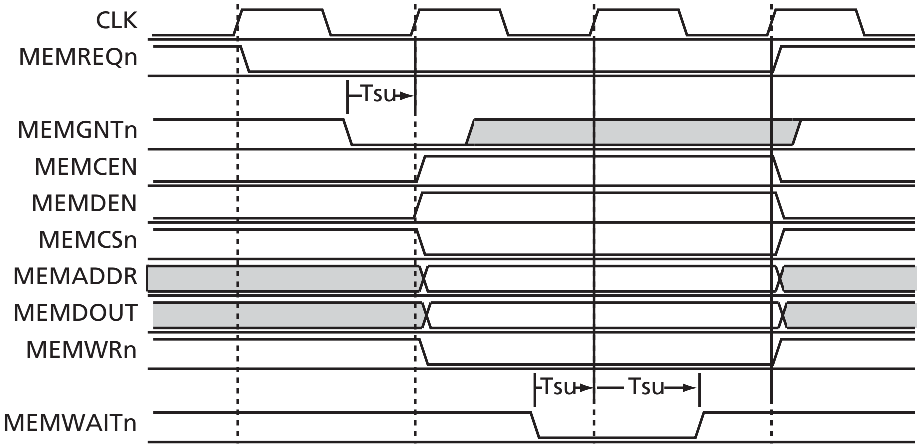
CLK
MEMREQn
Tsu
MEMGNTn
MEMCEN
MEMDEN
MEMCSn
MEMADDR
MEMDOUT
MEMWRn
Tsu
Tsu
MEMWAITn

*Figure 18* • **Synchronous Memory Write Cycle**

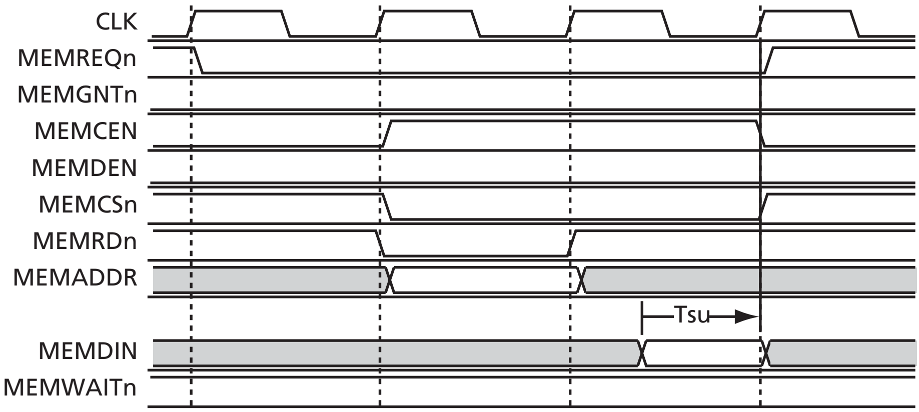
CLK
MEMREQn
MEMGNTn
MEMCEN
MEMDEN
MEMCSn
MEMRDn
MEMADDR
Tsu
MEMDIN
MEMWAITn

*Figure 19* • **Synchronous Memory Read Cycle with MEMGNTn Active**

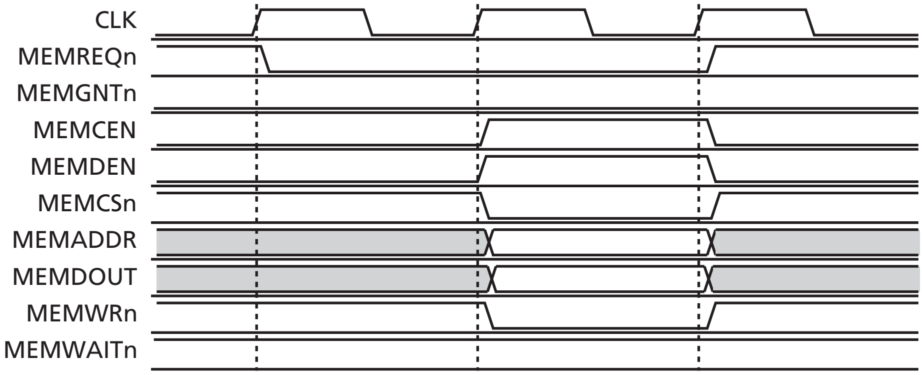
CLK
MEMREQn
MEMGNTn
MEMCEN
MEMDEN
MEMCSn
MEMADDR
MEMDOUT
MEMWRn
MEMWAITn

*Figure 20* • **Synchronous Memory Write Cycle with MEMGNTn Active**

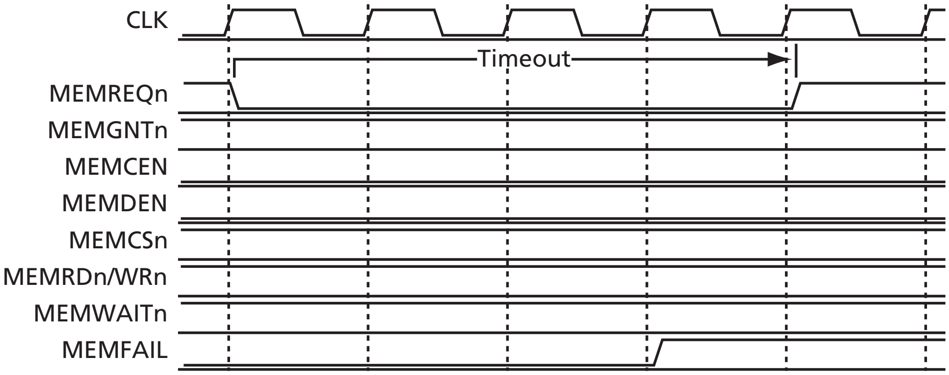
CLK
Timeout
MEMREQn
MEMGNTn
MEMCEN
MEMDEN
MEMCSn
MEMRDn/WRn
MEMWAITn
MEMFAIL

*Figure 21* • **Memory Grant Time-out**

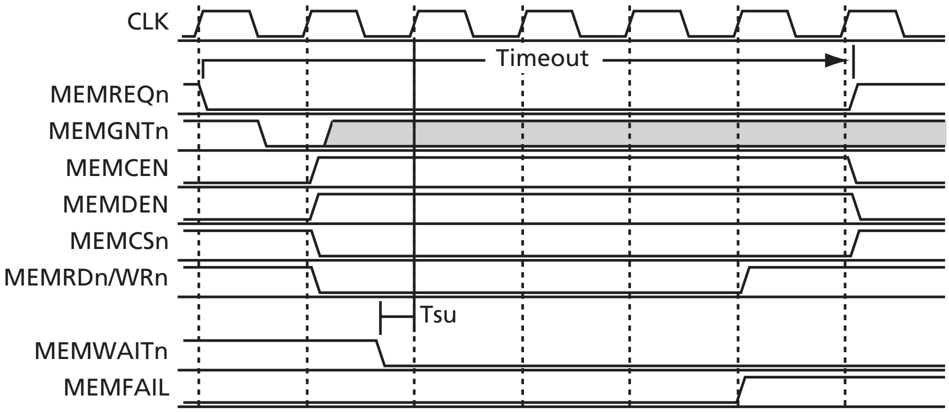

*Figure 22* • **Memory Wait Time-out**

## Clock Requirements

To meet the 1553B transmission bit rate requirements, the Core1553BBC clock input must be 12, 16, 20, or 24 MHz with a tolerance of ±0.01%.

# Ordering Information

Core1553BBC can be ordered through your local Actel sales representative. It should be ordered using the following number scheme: Core1553BBC-XX, where XX is (Table 19):

*Table 19* • **Ordering Codes**

| XX | Description |
|---|---|
| EV | Evaluation Version |
| SN | Netlist for single-use on Actel devices |
| AN | Netlist for unlimited use on Actel devices |
| SR | RTL for single-use on Actel devices |
| AR | RTL for unlimited use on Actel devices |
| UR | RTL for unlimited use and not restricted to Actel devices |

The Evaluation board can also be ordered using the order code "Core1553BBC Eval Board."

# List of Changes

The following table lists critical changes that were made in the current version of the document.

| Previous Version | Changes in Current Version (v4.0) | Page |
|---|---|---|
| v3.0 | The "Supported Families" section has been updated to include Fusion. | 1 |
| | Table 1 was updated to include Fusion data. | 4 |
| v2.0 | The "Supported Families" section has been updated to include ProASIC3/E. | 1 |
| | Table 1 was updated to include ProASIC3/E data. | 4 |

# Datasheet Categories

In order to provide the latest information to designers, some datasheets are published before data has been fully characterized. Datasheets are designated as "Product Brief," "Advanced," and "Production." The definition of these categories are as follows:

## Product Brief

The product brief is a summarized version of an advanced or production datasheet containing general product information. This brief summarizes specific device and family information for unreleased products.

## Advanced

This datasheet version contains initial estimated information based on simulation, other products, devices, or speed grades. This information can be used as estimates, but not for production.

## Unmarked (production)

This datasheet version contains information that is considered to be final.

Actel and the Actel logo are registered trademarks of Actel Corporation.
All other trademarks are the property of their owners.

www.actel.com

**Actel Corporation**

2061 Stierlin Court
Mountain View, CA
94043-4655 USA
**Phone** 650.318.4200
**Fax** 650.318.4600

**Actel Europe Ltd.**

Dunlop House, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
**Phone** +44 (0) 1276 401 450
**Fax** +44 (0) 1276 401 490

**Actel Japan**
www.jp.actel.com

EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
**Phone** +81.03.3445.7671
**Fax** +81.03.3445.7668

**Actel Hong Kong**
www.actel.com.cn

Suite 2114, Two Pacific Place
88 Queensway, Admiralty
Hong Kong
**Phone** +852 2185 6460
**Fax** +852 2185 6488

# AMEYA360

Components Supply Platform

## Authorized Distribution Brand :

ROHM SEMICONDUCTOR | Sunlord EXPERT IN PASSIVE PARTS | SSM SUSUMU | AVERLOGIC | NXP | Elprotronic

2Pai Semi | Ambarella | CanaanTek | Firstohm | GigaDevice

## Website :

Welcome to visit www.ameya360.com

## Contact Us :

### ➢ Address :

401 Building No.5, JiuGe Business Center, Lane 2301, Yishan Rd
Minhang District, Shanghai , China

### ➢ Sales :

| | |
|---|---|
| Direct | +86 (21) 6401-6692 |
| Email | amall@ameya360.com |
| QQ | 800077892 |
| Skype | ameyasales1 ameyasales2 |

### ➢ Customer Service :

| | |
|---|---|
| Email | service@ameya360.com |

### ➢ Partnership :

| | |
|---|---|
| Tel | +86 (21) 64016692-8333 |
| Email | mkt@ameya360.com |