Cortex-M1 v3.1 Handbook

Microsemi®

Table of Contents

Introduction

This document describes the architecture and implementation of the 32-bit ARM® Cortex™-M1 microprocessor developed by ARM specifically for use in FPGAs. Cortex-M1 has a three-stage pipeline and runs the ARMv6-M instruction set; it is essentially a functional subset of the Cortex-M3 processor. The streamlined Cortex-M1, developed for use in embedded applications, is designed to balance size and speed when implemented in an FPGA.

Key Features

- 32-bit RISC architecture (ARMv6-M)
- 32-bit AHB-Lite bus interface
- Three-stage pipeline
- 32-bit ALU
- 4-GB memory addressing range (the upper 0.5 GB is reserved)
- Optional real-time debug unit
- JTAG interface

Benefits

- Fully implemented in FPGA fabric
- All microprocessor I/Os available to user
- No license fees or royalties
- Can run all existing Thumb® code
- Upward-compatible with Cortex-M3 processor

Supported Microsemi SoC Products Group FPGA Families

The Cortex-M1 version part number prefix is in parentheses.

- IGLOO® (M1AGL)
- IGLOOe (M1AGLE)
- ProASIC3L (M1A3PxxxxL)
- ProASIC®3 (M1A3P)
- ProASIC3E (M1A3PE)
- Fusion (M1AFS)

Utilization and Performance

Cortex-M1 can be implemented in several Microsemi FPGA devices. Table 1 gives typical utilization figures using standard synthesis tools. The Configuration column of Table 1 uses numbers to identify particular configurations of the core. Refer to Table 1-1 on page 8 for a description of each configuration.

Table 1 • **Cortex-M1 Utilization and Performance Data**

Device	Configuration	Frequency (MHz)	Utilization	
			RAM Blocks	Tiles
M1AFS250	028910	64.08	4	4411
M1AFS600	028910	63.86	4	4411
	028911	62.83	4	7491
M1AFS1500	028910	59.94	4	4411
	028911	61.74	4	7491
	134820	45.93	28	7465
	134821	47.34	28	10881
M1AGL250V2	028910	26.59	4	4411
M1AGL250V5	028910	43.49	4	4411
M1AGL600V2	028910	25.43	4	4411
	028911	25.11	4	7491
M1AGL600V5	028910	42.47	4	4411
	028911	40.45	4	7491
M1AGL1000V2	028910	25.81	4	4411
	028911	25.46	4	7491
M1AGL1000V5	028910	42.36	4	4411
	028911	42.65	4	7491
M1AGLE3000V2	028910	25.82	4	4411
	028911	25.36	4	7491
M1AGLE3000V5	028910	41.29	4	4411
	028911	40.65	4	7491
M1A3P250	028910	66.08	4	4411
M1A3P400	028910	61.63	4	4411
	028911	58.89	4	7491
M1A3P600	028910	66.90	4	4411
	028911	63.47	4	7491

Notes:

1. *See Table 1-1 on page 8 for a description of the numbers in the Configuration column.*
2. *All frequency values are measured at commercial operating range conditions.*

Table 1 • **Cortex-M1 Utilization and Performance Data (continued)**

Device	Configuration	Frequency (MHz)	Utilization	
			RAM Blocks	Tiles
M1A3P1000	028910	66.26	4	4411
	028911	63.65	4	7491
M1A3PE1500	028910	65.61	4	4411
	028911	63.80	4	7491
	134820	46.35	28	7465
	134821	48.38	28	10881
M1A3PE3000	028910	62.34	4	4411
	028911	62.03	4	7491
M1A3P600L	028910	53.42	4	4411
	028911	52.13	4	7491
M1A3P1000L	028910	55.09	4	4411
	028911	52.68	4	7491
M1A3PE3000L	028910	51.03	4	4411
	028911	50.76	4	7491

Notes:

1. *See Table 1-1 on page 8 for a description of the numbers in the Configuration column.*
2. *All frequency values are measured at commercial operating range conditions.*

Utilization of Global Nets

Cortex-M1 configurations that do not include debug logic use two global clock nets within the FPGA for the following signals:

- HCLK
- SYSRESETn

Cortex-M1 configurations that do include debug logic use four global clock nets within the FPGA for the following signals:

- HCLK
- SYSRESETn
- SWCLKTCK
- DBGRESETn

Layout Constraints

Microsemi's SmartDesign tool should be used to instantiate and configure Cortex-M1.

If you configure Cortex-M1 to enable debugging via the UJTAG macro (which is the necessary configuration when debugging with Microsemi's SoftConsole tool), it is good practice to ensure that low skew routing is used for the clock signal output from the UJTAG macro. You may also wish to use low skew routing for the reset signal from the UJTAG macro, but this is a less critical signal. Debugging via UJTAG is enabled by selecting the **Include debug** check box and setting the **Debug interface** option to **JTAG, using UJTAG macro** in the Cortex-M1 configuration window.

The easiest way to ensure that low skew routing is used for the UJTAG clock signal is to select the **Instantiate CLKINT buffer for UJTAG clock signal** check box in the Cortex-M1 configuration window.

Alternatively, you may wish to leave the **Instantiate CLKINT buffer for UJTAG clock signal** check box cleared and instead use a PDC constraint file to ensure that low skew routing is used for the UJTAG clock signal. For example, a constraint such as the following could be used to assign the UJTAG clock signal (UDRCK) to a quadrant clock net in the lower right (LR) quadrant of the device:

```
assign_quadrant_clock -net <hierarchical path to Cortex-M1 instance>/RS/UDRCK -quadrant
LR
```

If your SmartDesign component is the top-level module in your design and you have accepted the default instance name of **CortexM1Top_0** for Cortex-M1, the previous example PDC constraint will have the following form:

```
assign_quadrant_clock -net CortexM1Top_0/RS/UDRCK -quadrant LR
```

The PDC constraint to assign UDRCK to a global clock net would be:

```
assign_global_clock -net CortexM1Top_0/RS/UDRCK
```

1 – Cortex-M1 Overview

Cortex-M1 Processor

Cortex-M1 is a general purpose 32-bit microprocessor that offers high performance and small size in FPGAs.

Cortex-M1 runs a subset of the Thumb-2 instruction set (ARMv6-M) that includes all base 16-bit Thumb instructions and a few Thumb-2 32-bit instructions (BL, MRS, MSR, ISB, DSB, and DMB). This enables writing very tight and efficient processor code, which is ideal for the limited memory typically found in deeply-embedded applications.

Figure 1-1 shows a block diagram of the Cortex-M1 processor available for use in Microsemi M1-enabled devices. The components within the blue box in this diagram are preconfigured and fixed for each available configuration of the core. These components are contained within a black box core data base (CDB) file which includes placement and routing information for these components. At the top level, there is an RTL wrapper surrounding the CDB and this contains reset synchronization logic and some debug related logic.

The main blocks in Cortex-M1 are shown in Figure 1-1 and include the processor core, the Nested Vectored Interrupt Controller (NVIC), the AHB interface, and the debug unit. The processor has 13 general purpose 32-bit registers as well as a Link register (LR), a program counter (PC), a stack pointer (SP) and a Program Status register (xPSR). If the core has been configured with operating system (OS) extensions present, a second stack pointer is available for use. A dedicated memory interface is available for access to Instruction and Data Tightly Coupled Memories (ITCM and DTCM) when the core has been configured with non-zero-sized TCMs. Currently OS extensions and TCMs are not supported on the majority of Microsemi M1 devices but are available when using an M1AFS1500 or M1A3PE1500 device.

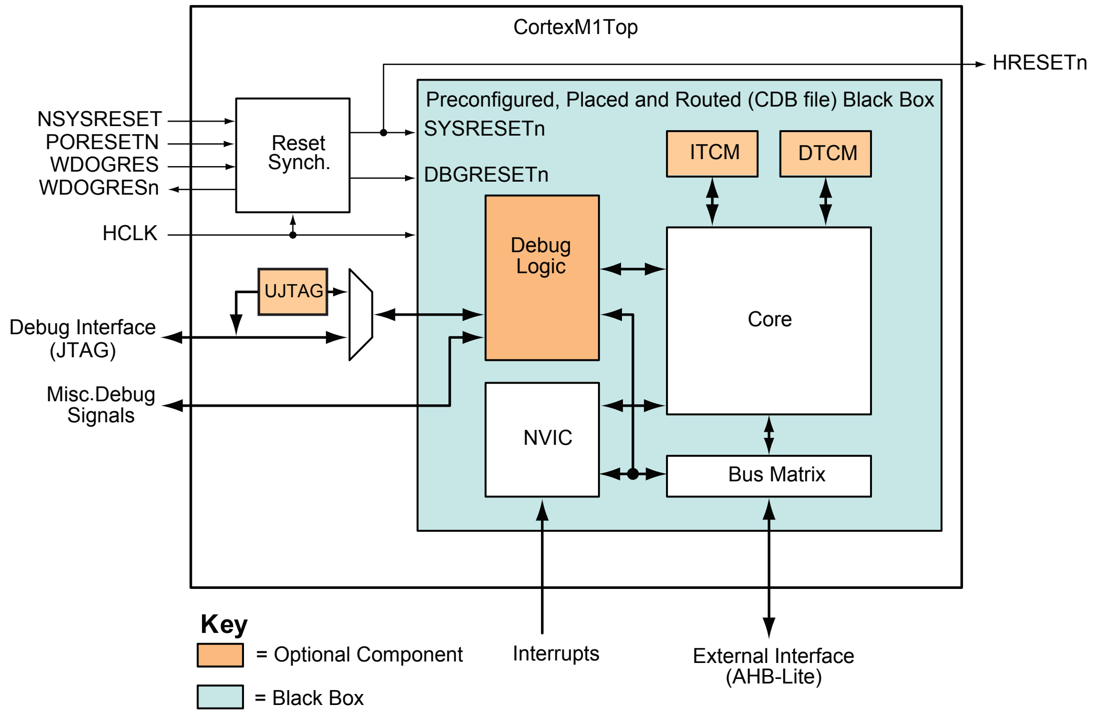


Figure 1-1 • **Cortex-M1 Block Diagram**

The NVIC is closely coupled to the Cortex-M1 core to achieve low-latency interrupt processing. The processor state is automatically saved on interrupt entry and restored on interrupt exit, with no instruction overhead to simplify software development.

The 16-bit length of the Cortex-M1 Thumb instruction allows it to approach twice the code density of the standard 32-bit ARM code, while retaining most of the ARM performance advantages over a traditional 16-bit processor that uses 16-bit registers. This is possible because Thumb code operates on the 32-bit register set in the processor. Thumb code is able to provide up to 65% of the code size of ARM, and 160% of the performance of an equivalent ARM processor connected to a 16-bit memory system.

Detailed information on the Cortex-M1 processor is contained in the Cortex-M1 Technical Reference Manual (TRM). This document is available on the ARM website (www.arm.com) and is also delivered with the core. Right-click on the core in the Libero Catalog pane and select **Open documentation** > **CortexM1_TRM.pdf** to open the document.

The information in the Cortex-M1 TRM applies to the level of hierarchy contained within the blue box shown in Figure 1-1 on page 7; that is, at the level of the black box CDB.

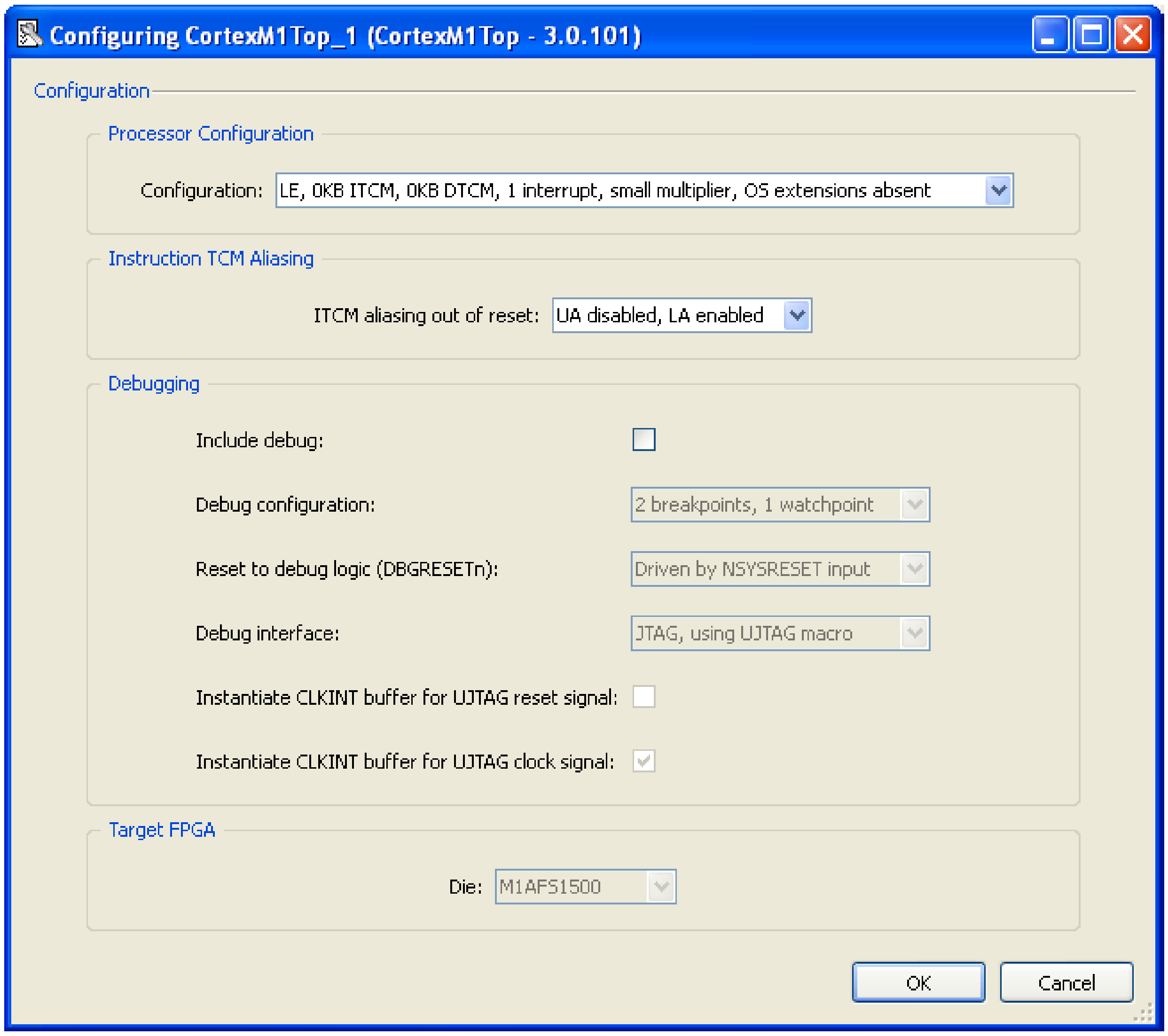
Cortex-M1 Configurations

Due to the fact that much of the logic in the Cortex-M1 is contained within the black box CDB, the extent of configurability available to the user is limited; essentially, canned configurations are presented for use. Currently, configurations which minimize the resource requirements of the core are available for each M1-enabled device. For the M1AFS1500 and M1A3PE1500 devices, a more fully featured configuration can be selected. In all cases the core can be configured to include or exclude debug logic. It is intended that, over time, more configurations will become available across a range of devices. Each configuration is assigned a unique number for ease of reference; Table 1-1 shows the available configurations along with the corresponding configuration numbers.

Table 1-1 **• Cortex-M1 Configurations**

Configuration Number	Configuration Description	Devices on which Configuration is Available
028910	Little endian, 1 interrupt, small multiplier, OS extensions absent, 0 KB ITCM, 0 KB DTCM, debug logic not included	All M1-enabled devices
028911	Little endian, 1 interrupt, small multiplier, OS extensions absent, 0 KB ITCM, 0 KB DTCM, reduced debug with JTAG interface	All M1-enabled devices
134820	Little endian, 16 interrupts, normal multiplier, OS extensions present, 8 KB ITCM, 4 KB DTCM, debug logic not included	M1AFS1500, M1A3PE1500
134821	Little endian, 16 interrupts, normal multiplier, OS extensions present, 8 KB ITCM, 4 KB DTCM, reduced debug with JTAG interface	M1AFS1500, M1A3PE1500

The configurations listed in the "Cortex-M1 Configurations" section on page 8 are relevant to the configuration of the logic within the black box CDB, which accounts for the majority of the Cortex-M1 logic. There is also some configurability associated with the top-level RTL wrapper surrounding the CDB. A GUI configuration window is available for setting configuration options when working with Cortex-M1 within SmartDesign. The configuration options chosen are used to select the appropriate CDB as well as to configure the top-level RTL wrapper. A screenshot of the configuration window is shown in Figure 1-2. For many of the options, tooltips will pop up when the mouse pointer hovers over the option. These tooltips provide more information on the options available and explain abbreviations used in the text of settings.


Configuring CortexM1Top_1 (CortexM1Top - 3.0.101)

Configuration

Processor Configuration

Configuration: LE, 0KB ITCM, 0KB DTCM, 1 interrupt, small multiplier, OS extensions absent

Instruction TCM Aliasing

ITCM aliasing out of reset: UA disabled, LA enabled

Debugging

Include debug: ☐

Debug configuration: 2 breakpoints, 1 watchpoint

Reset to debug logic (DBGRESETn): Driven by NSYSRESET input

Debug interface: JTAG, using UJTAG macro

Instantiate CLKINT buffer for UJTAG reset signal: ☐

Instantiate CLKINT buffer for UJTAG clock signal: ✓

Target FPGA

Die: M1AFS1500

Figure 1-2 • **M1 Configuration Window**

The **Configuration** option is used to select the major aspects of the processor configuration.

The **ITCM aliasing out of reset** option determines the reset or default values of the ITCMUAEN and ITCMLAEN bits of the Cortex-M1 Auxiliary Control register. (See the *Cortex-M1 Technical Reference Manual* for more details on the Auxiliary Control register.) Note that the **Configuration** and **ITCM aliasing out of reset** options may be interdependent and not every combination of settings may be valid. Small yellow warning triangles are displayed when an invalid combination has been selected. If you hover over a warning triangle with your mouse pointer, an information message will appear with an explanation of how to resolve the issue.

Select or clear the **Include debug** check box to include or exclude debug logic. Selecting **Include debug** enables the other debug related configuration options.

The **Reset to debug logic (DBGRESETn)** option is used to determine how the internal DBGRESETn signal is driven. This signal is used to reset some debug components that are clocked by HCLK. Ideally, **Reset to debug logic (DBGRESETn)** should be set to **Driven by PORESETN input** and a power-on reset signal should be connected to the PORESETN input. Alternatively, NSYSRESET can be used as the source for DBGRESETn by selecting **Driven by NSYSRESET input,** in which case PORESETN is not used. The advantage of using a power-on reset signal as the source for DBGRESETn is that this allows NSYSRESET to be asserted during a debug session without losing the debug connection to the processor. NSYSRESET will often be connected to an external reset push-button.

Set the **Debug interface** option to **JTAG, using UJTAG macro** when using Microsemi SoftConsole tool to debug your Cortex-M1 system. When debugging with SoftConsole, the dedicated JTAG pins of the device are used for the debug connection with the UJTAG macro used as a conduit between the dedicated on-chip JTAG controller and the FPGA fabric where the Cortex-M1 resides. Set **Debug interface** to **JTAG, not using UJTAG macro** when using a third party debugger such as ARM/Keil™ RealView® or the IAR Systems® Embedded Workbench for ARM (EWARM) tool. When the **JTAG, not using UJTAG macro** setting has been selected, the JTAG pins (TCK, TMS, TDI, TDO, and nTRST) should be routed to appropriate device pins, which will typically be connected to a debug header. When **JTAG, using UJTAG macro** has been selected, the JTAG pins must still be routed to the top level of your design, but in this case specific pin assignments are not required. Microsemi Designer tool will recognize that the UJTAG macro is in use and make use of the dedicated JTAG pins of the device for the Cortex-M1 debug connection.

When **Debug interface** is set to **JTAG, using UJTAG macro**, two further check boxes are enabled for selecting whether or not to instantiate CLKINT buffers for the UJTAG clock and reset signals. Instantiating a CLKINT buffer is one way of ensuring that low skew routing is used for a signal. Alternatively, design constraints may be used to cause low skew routing to be used. See the "Layout Constraints" section on page 5 for more information on using PDC constraints to cause low skew routing to be used for UJTAG signals. It is good practice to use low skew routing for the UJTAG clock signal, but the use of low skew routing for the UJTAG reset signal is not as critical.

Delivery and Deployment

Cortex-M1 is available through the Libero® Integrated Design Environment (IDE) IP Catalog. It can be downloaded from a remote web-based repository and installed into the user's local vault, ready for use. Once installed in Libero IDE, the core can be instantiated, configured, and generated within SmartDesign for inclusion in your Libero IDE project.

Cortex-M1 I/O Ports

Table 1-2 lists the ports of CortexM1Top, the top level of the core.

***Table 1-2* • Cortex-M1 Port Descriptions**

Name	Width	Type	Description
HCLK	1	Input	Main processor clock
NSYSRESET	1	Input	Active low system reset.
PORESETN	1	Input	Active low power on reset. This input is only used when the **Reset to debug logic (DBGRESETn)** configuration option is set to **Driven by PORESETN input**.
HRESETn	1	Output	Reset output to other components in the system
WDOGRES	1	Input	"Bark" signal from watchdog
WDOGRESn	1	Output	Reset signal to watchdog

Table 1-2 **• Cortex-M1 Port Descriptions (continued)**

LOCKUP	1	Output	Status output which, when asserted, indicates that the processor is in the lock-up state.
HALTED	1	Output	Status output which, when asserted, indicates that the processor is in halting debug mode. This output is only functional when the core has been configured to include debug logic.
NMI	1	Input	Non-maskable interrupt
IRQ0	1	Input	External interrupt 0
IRQ1 to IRQ7		Input	External interrupts 1 to 7. These are only functional when the core has been configured with 8 or more interrupts.
IRQ8 to IRQ15		Input	External interrupts 8 to 15. These are only functional when the core has been configured with 16 or more interrupts.
IRQ16 to IRQ31		Input	External interrupts 16 to 31. These are only functional when the core has been configured with 32 interrupts.
HADDR	32	Output	AHB-Lite address bus
HBURST	3	Output	AHB-Lite burst indication
HPROT	4	Output	AHB-Lite protection control signals
HRDATA	32	Input	AHB-lite read data bus
HREADY	1	Input	AHB-Lite "bus ready" signal
HRESP	2	Input	AHB-Lite response signal; indicates OKAY or ERROR status for each transfer on the bus.
HSIZE	3	Output	AHB-Lite size indication; byte, halfword, word, for example.
HTRANS	2	Output	AHB-Lite transfer type indication. Can be IDLE, BUSY, NONSEQUENTIAL, or SEQUENTIAL.
HWDATA	32	Output	AHB-Lite write data bus
HWRITE	1	Output	AHB-Lite transfer direction indication. High for a write transfer.
HMASTLOCK	1	Output	AHB-Lite signal that indicates if a transfer is part of a locked sequence.
EDBGRQ	1	Input	External debug request. This input is only functional when the core has been configured to include debug logic.
JTAGTOP	1	Output	Indicates state of JTAG controller. This output is only functional when the core has been configured to include debug.
JTAGNSW	1	Output	Indicates whether JTAG or Serial Wire (SW) based debug is in use. High = JTAG, low = SW. This output is only functional when the core has been configured to include debug.
SWDO	1	Output	Serial data output. This output is only functional when the core has been configured to include debug.
SWDOEN	1	Output	Active high serial data output enable. This output is only functional when the core has been configured to include debug.
nTDOEN	1	Output	Active low output enable for JTAG TDO signal. This output is only functional when the core has been configured to include debug.
TCK	1	Input	JTAG clock input. This input is only functional when the core has been configured to include debug.

Table 1-2 • **Cortex-M1 Port Descriptions (continued)**

nTRST	1	Input	JTAG reset signal, active low. This input is only functional when the core has been configured to include debug.
TMS	1	Input	JTAG test mode select. Also serves as serial data input when SW debug is in use. This input is only functional when the core has been configured to include debug.
TDI	1	Input	JTAG data input. This input is only functional when the core has been configured to include debug.
TDO	1	Output	JTAG data output. This output is only functional when the core has been configured to include debug.

2 – Interfaces

The Cortex-M1 processor has an external AHB-Lite interface that can be used to connect to other AMBA components. Internally a private peripheral bus (PPB) is used to facilitate communication between the processor core and the NVIC and debug blocks. When a processor configuration which includes TCMs has been selected, dedicated interfaces are available to provide fast, uncontended access to the ITCM and DTCM.

External Interface

The external interface is an AMBA AHB-Lite bus interface. Descriptions of the bus signals are shown in Table 1-2 on page 10. Processor accesses and debug accesses to external AHB peripherals can occur over this bus interface. Because AHB fetches take two cycles longer than TCM fetches, instructions and data should ideally be contained in TCM where possible.

Figure 2-1 shows the timing of a read without wait states on the external interface. The Address A0 is presented on **HADDR** one cycle later than the internal address is generated, and the returned data D0 is registered again before use in the processor. This enables the AHB peripherals sufficient time to use the address generated.

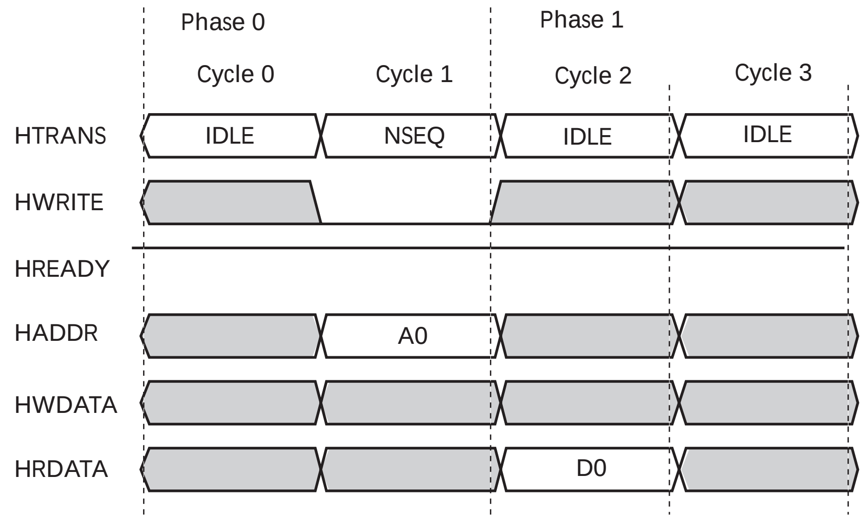


Figure 2-1 **• AHB Read without Wait States**

Processor accesses and debug accesses share the external interface. Debug accesses take priority over processor accesses. Giving the highest priority to debug means that debug cannot be locked out by a continuously executing stream of core instructions. Timing of processor accesses might be changed by the presence of debug accesses. Debug accesses tend to be infrequent, so debug accesses generally do not have a major impact on processor accesses.

Any vendor-specific components with an AHB interface can populate this bus.

Unaligned accesses to this bus are not supported.

Write Buffer

To prevent bus wait cycles from stalling the processor during data stores, buffered stores to the external interfaces go through a one-entry write buffer. If the write buffer is full, subsequent accesses to the bus stall until the write buffer has drained. The write buffer is only used if the bus waits for the data phase of the buffered store; otherwise the transaction is immediately completed on the bus.

The DMB and DSB instructions wait for the write buffer to drain before completing. If an interrupt arrives while DMB/DSB is waiting for the write buffer to drain, the processor returns to the opcode after the DMB/DSB, on completion of the interrupt. This is because interrupt processing is a so-called memory barrier operation. In other words, all reads and writes occurring before the interrupt appear to happen before the interrupt, so the DMB and DSB instructions must appear to have completed before the interrupt.

Tightly Coupled Memory (TCM) Interface

For processor configurations which include non zero sized TCMs, dedicated low latency memory interfaces are available for accessing ITCM and DTCM.

Because reads are speculatively fetched from TCMs, Device and Strongly-Ordered memory types such as FIFOs in TCM space are not supported. The processor does not support wait states for the memory interfaces. Figure 2-2 shows the signal timings for ITCM. The DTCM signal timings are the same as ITCM signal timings.

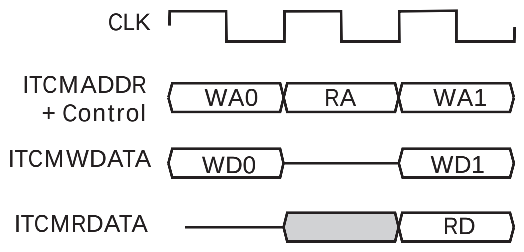


Figure 2-2 • **ITCM Signal Timings**

The write address, WA0, and write data, WD0, are presented in the same cycle. The Read Address (RA) is presented in one cycle, and the memory generates the Read Data (RD) in the next cycle. The sequence that Figure 2-2 shows is only possible on ITCM, where the RA read is an instruction fetch, and WA1 write is the product of a store instruction. The WA0-RA sequence is possible on DTCM.

In situations where there is only one large RAM available, but the user wants to run from RAM (for example, the user is debugging code on an Microsemi Cortex-M1 Development Board), the RAM should be mapped to location 0x6000000. The user should download the program (via debugger) to this location and start execution from this point. The user's software should be written/linked assuming this location during the debug stage. This is the only SRAM area in the memory map that supports both instruction fetches and data accesses. When the debugging has finished, the software needs to be rewritten and linked to assume the starting point of location 0x00000000.

Note: ITCM and DTCM are currently fixed at 0 memory size in the majority of Microsemi M1 devices. TCMs are supported on the M1AFS1500 and M1A3PE1500 devices.

3 – Cortex-M1 Features

This section briefly outlines the main features of the Cortex-M1 processor. More detailed information is available in the Cortex-M1 Technical Reference Manual.

Programmer's Model

The Cortex-M1 processor implements a subset of the Thumb-2 (ARMv7) architecture called ARMv6-M. This includes all of the 16-bit Thumb-2 instructions and some of the 32-bit instructions. The processor does not support ARM instructions.

The Thumb-2 (ARMv7) instruction set architecture (ISA) was developed by ARM for the Cortex family of processors. Offering increased efficiency and performance, the Thumb-2 ISA differs from previous ARM architectures in that it includes both 16- and 32-bit instructions. The previous 16-bit Thumb instruction set and 32-bit ARM instruction set were separate and had to be executed from different modes within the processor. The Thumb-2 ISA gives users all the advantages of the reduced code size of the 16-bit Thumb instructions and the higher performance of the 32-bit ARM instructions. This is achieved in a single ISA that can be executed without requiring any context switching within the processor, increasing the efficiency of the code as it executes and improving the performance and throughput of the Cortex family of processors.

Processor Operating States

The Cortex-M1 processor has two operating states:

- **Thumb state** – This is normal execution, running the set of 16-bit, halfword-aligned Thumb and Thumb-2 instructions; as well as the 32-bit BL, MRS, MSR, ISB, DSB, and DMB instructions.
- **Debug state** – This is the state when halting debug

Processor Operating Modes

The Cortex-M1 processor supports two modes of operation:

- **Thread mode** – Entered on Reset, and can be re-entered as the result of an exception return
- **Handler mode** – Entered as the result of an exception

Main Stack and Process Stack Access

Out of reset, all code uses the main stack with the processor in Thread mode. If the processor is configured with OS extensions present, a second (process) stack can be used in addition to the main stack. The stack pointer (R13) becomes a banked register when OS extensions are present.

Note: OS extensions are not supported on the majority of Microsemi M1 devices. OS extensions are supported on the M1AFS1500 and M1A3PE1500 devices.

Data Types

The processor supports the following data types:

- 32-bit words
- 16-bit halfwords
- 8-bit bytes

Note: Unless otherwise stated, the core can access all regions of the memory map, including the code region, with all data types. To support this, the system must support sub-word writes without corrupting neighboring bytes in that word (i.e., individual byte enables for writes).

Registers

The processor has the following 32-bit registers (shown in Figure 3-1):

- 13 general purpose registers, R0–R12
- Stack Pointer (SP), R13
- Link Register (LR), R14
- Program Counter (PC), R15
- Program status registers, xPSR

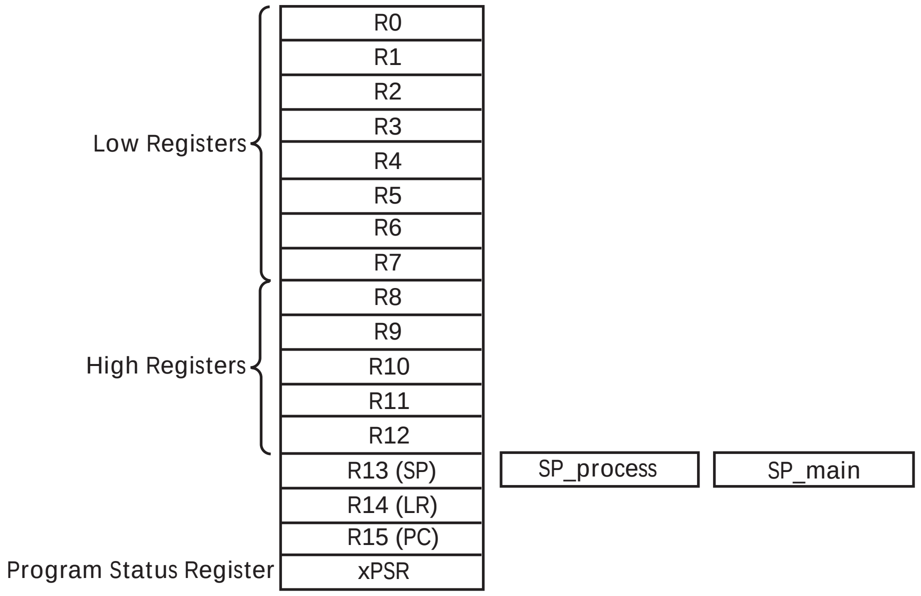


***Figure 3-1* • Cortex-M1 Register Set**

General Purpose Registers

The general purpose registers, R0–R12, have no architecture-specific uses.

- **Low registers** – Registers R0–R7 are accessible by all instructions that specify a general purpose register.
- **High registers** – Registers R8–R12 are accessible by some, but not all 16-bit instructions.

The R13, R14, and R15 registers have the following special functions:

- **Stack pointer** – Register R13 is used as the Stack Pointer (SP). Because the SP ignores writes to bits [1:0], it is auto-aligned to a word (four-byte) boundary. The stack pointer has banked aliases, SP_process and SP_main, when the processor has been configured with OS extensions present. When OS extensions are absent, only a single stack pointer, SP_main, is present.
- **Link register** – Register R14 is the subroutine Link Register (LR). The LR receives the return address from the Program Counter (PC) when a Branch and Link (BL) instruction is executed. The LR is also used for exception returns. At all other times, R14 can be treated as a general purpose register.
- **Program counter** – Register R15 is the Program Counter (PC). Bit [0] is always 0, so instructions are always aligned to 16-bit halfword (two-byte) boundaries.

Special Purpose Program Status Registers (xPSR)

Processor status at the system level is broken into three categories and can be accessed as individual registers, a combination of any two of the three, or a combination of all three using the MRS and MSR instructions.

- **Application PSR (APSR)** – Contains the condition code flags. Before entering an exception, the processor saves the condition code flags on the stack. The APSR can be accessed with the MSR and MRS instructions.
- **Interrupt PSR (IPSR)** – Contains the *Interrupt Service Routine* (ISR) number of the current exception
- **Execution PSR (EPSR)** – Contains the *Thumb state bit* (T-bit). Unless the processor is in Debug state, the EPSR is not directly accessible and all fields read as zero using an MRS instruction. MSR instruction writes are ignored.

On entering an exception, the processor saves the combined information from the three status registers on the stack.

Special Purpose Priority Mask Register

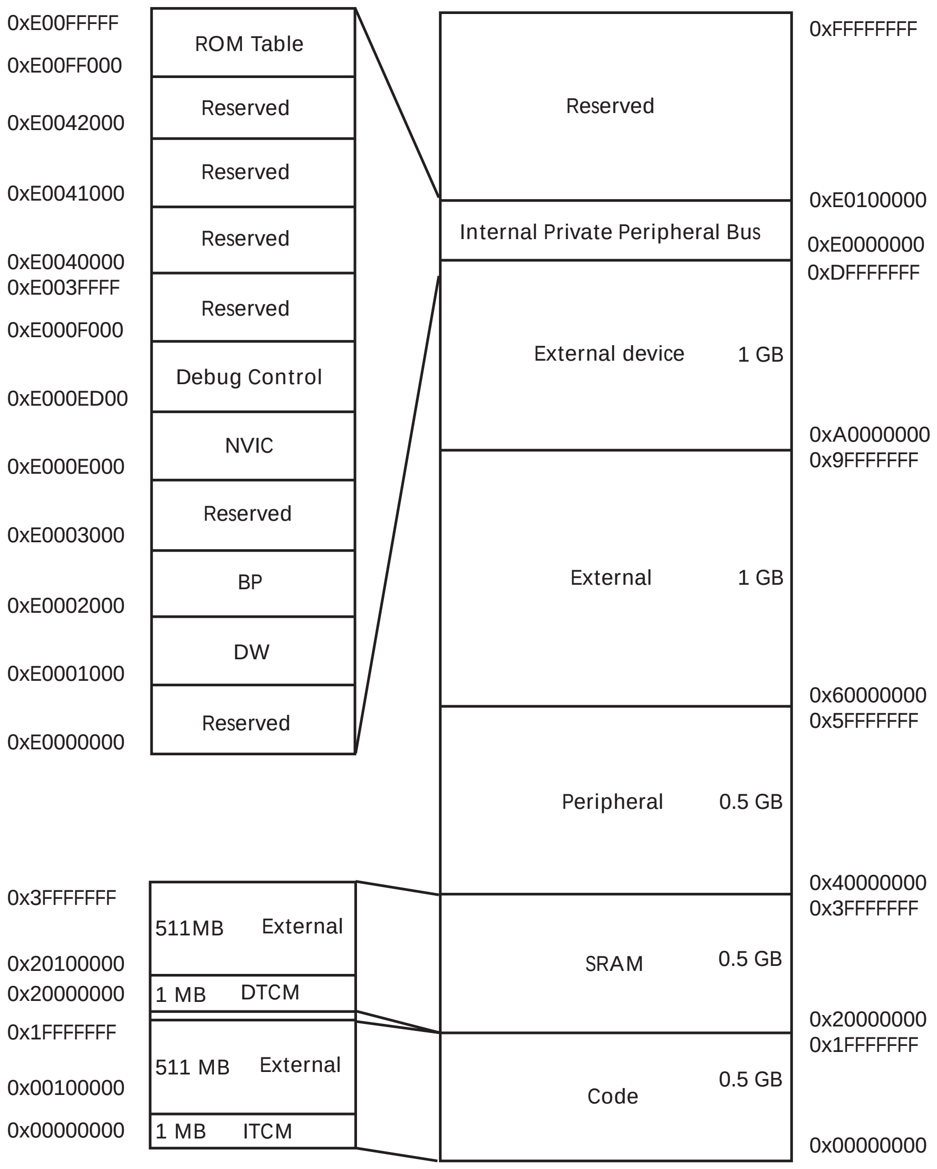
Use the special purpose Priority Mask Register to boost execution priority. The special purpose Priority Mask Register can be accessed by using the MSR and MRS instructions. The CPS instruction to set or clear PRIMASK can also be accessed.

Special Purpose Control Register

The special purpose Control Register identifies the stack pointers usage.

Memory Map

Cortex-M1 has a defined memory map with the various processor interfaces addressed by different memory map regions, as shown in Figure 3-2. The processor can access all regions within the memory map with the exception of the reserved regions. The reserved regions are Execute Never (XN) and instruction accesses are prevented by the processor hardware. The SRAM, Peripheral, External Device, and Private Peripheral Bus regions in the memory map are also XN. Instructions can be executed from the Code and External (not External Device) regions of the memory map.



Note: TCMs shown are maximum size (1,024 kbytes).

Figure 3-2 • **Cortex-M1 Memory Map**

The processor views memory as a linear collection of bytes numbered in ascending order from 0.

For example:

- Bytes 0–3 hold the first stored word

- Bytes 4–7 hold the second stored word

The processor can access data words in memory in little-endian format or big-endian format.

In little-endian format, the byte with the lowest address in a word is the least significant byte of the word. The byte with the highest address in a word is the most significant. The byte at address 0 of the memory system connects to data lines 7–0.

In big-endian format, the byte with the lowest address in a word is the most significant byte of the word. The byte with the highest address in a word is the least significant. The byte at address 0 of the memory system connects to data lines 7–0.

Cortex-M1 always accesses code in little-endian format. Little-endian is the default memory format for ARM processors. The processor contains a configuration option that enables the user to select either the little-endian or big-endian format during implementation. Currently, only little-endian configurations of Cortex-M1 are supported on Microsemi devices.

Subsystem Restrictions

The fixed memory map of the Cortex-M1 places certain restrictions on the processor subsystem.

The Code region encompasses two CoreAHB/CoreAHBLite 256 MB slots (0 and 1). If the Data region of the memory map is being mapped to a device (RAM, for example), this must be mapped to AHB slot 2 or 3.

1. If the user has internal or external RAM mapped to the Data space and is using non-zero DTCM, there will be a wasted area of external RAM. If the user wishes to use non-zero DTCM, the external SRAM should be mapped up to the second SRAM space (0x60000000 and above).
2. Similarly, if the user wishes to run from internal or external SRAM, with non-zero ITCM, this SRAM should be mapped to 0x60000000 and above. The RAM would be wasted if SRAM were remapped to location zero after boot.
3. If the user wishes to run from NVM, with non-zero ITCM, NVM can be left at location 0x00000000. ITCM can be filled from NVM after reset and the ITCM then remapped to 0x00000000 using the Auxiliary Control register. From then on instruction fetches in the ITCM range would go to the ITCM, and instruction fetches above this space (but still within Code space) would go to NVM.

Exceptions

The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. All exceptions are handled in Handler mode. The processor state is automatically stored to the stack on an exception, and automatically restored from the stack at the end of the exception handler Interrupt Service Routine (ISR). The following features enable efficient, low-latency exception handling:

- Automatic state saving and restoring. The processor pushes state registers on the stack before entering the ISR, and pops them after exiting the ISR with no instruction overhead.
- Automatic reading of the vector table entry that contains the ISR address in code memory or data SRAM
- Closely-coupled interface between the processor and the NVIC to enable early processing of interrupts and processing of late-arriving interrupts with higher priority
- One configurable interrupt
- Separate stacks for Handler and Thread modes if OS extensions are implemented
- ISR control transfer using the calling conventions of the C/C++ standard, *Procedure Call Standard for the ARM Architecture* (PCSAA)
- Priority masking to support critical regions

Exception Types

The types of exceptions supported in Cortex-M1 are listed in Table 3-1. A fault is an exception that results from an error condition. Faults can be reported synchronously or asynchronously to the instruction that caused them. In general, faults are reported synchronously. Faults caused by writes over the bus are asynchronous faults. A synchronous fault is always reported with the instruction that caused the fault. An asynchronous fault may vary in how it is reported with respect to the instruction that caused the fault.

Table 3-1 **• Cortex-M1 Exceptions**

Position	Exception Type	Priority	Description	Activated
–	–	–	Stack top is loaded from first entry of vector table on reset.	–
1	Reset	–3 (highest)	Invoked on power-up and warm reset. On first instruction, drops to lowest priority. Thread mode.	Asynchronous
2	Non-maskable Interrupt	–2	Cannot be marked, prevented by activation, by any other exception. Cannot be preempted by any other exception other than Reset.	Asynchronous
3	Hard Fault	–1	All classes of Fault	Synchronous or asynchronous
4–10	–	–	Reserved.	–
11	SVCall	Configurable	System service call with SVC instruction.	Synchronous
12–13	–	–	Reserved	–
14	PendSV	Configurable	Pendable request for system service. This is only pended by software.	Asynchronous
15	SysTick	Configurable	System tick timer has fired.	Asynchronous
16–48	External Interrupt	Configurable	Asserted from outside the processor, IRQ[2^{n-1}:0], and fed through the NVIC (prioritized).	Asynchronous

Exception Priority

In the processor exception model, priority determines when and how the processor handles exceptions. Software priority levels can be assigned to interrupts.

The NVIC supports software-assigned priority levels. A priority level from 0 (highest) to 3 (lowest) can be assigned to an interrupt by writing to the two-bit IP_N field in an Interrupt Priority Register. Hardware priority ranges from –3 (highest), to 3 (lowest). By default, external interrupts have a hardware priority of 3, but hardware priority decreases with increasing interrupt number. The programmable priority level overrides the hardware priority. For example, IRQ(4) would have a default priority lower than IRQ(2), but if IRQ(4) is assigned a software priority of 1 and IRQ(2) is assigned 0, then IRQ(2) has priority over IRQ(4).

Servicing an Exception

When the processor invokes an exception, it automatically pushes the following eight registers in two stages to the stack in the following order:

1. Processor Status Register (*x*PSR)
2. ReturnAddress ()
3. Link Register (LR)
4. R12
5. R3
6. R2
7. R1
8. R0

The SP is decremented by eight words on the completion of the stack push.Figure 3-3 shows the contents of the stack after an exception preempts the current program flow.

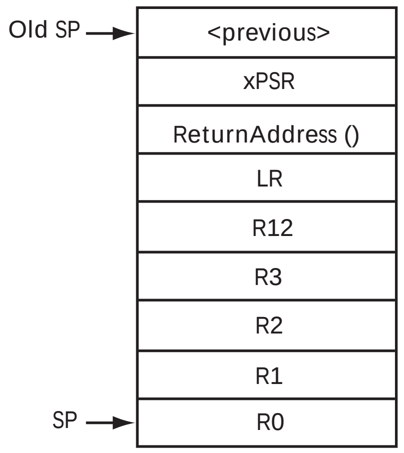


***Figure 3-3* • Stack Contents from an Exception**

When the processor services an exception, it takes the following steps before it enters the exception service routine.

1. It pushes 8 registers: *x*PSR, ReturnAddress (), R0, R1, R2, R3, R12, and LR on the selected stack
2. It reads the vector from the appropriate vector table entry, for example: (0x0) + (exception_number *4). This vector table read is done after all eight registers in the previous step are pushed onto the stack.
3. On Reset only, SP_main is updated from the first entry in the vector table. Other exceptions do not modify SP_main at this time and in this manner.
4. Updates PC with vector table read location. No other late-arriving exceptions can be processed until the first instruction of the exception starts to execute.
5. LR is set to EXC_RETURN to exit from the exception.

Figure 3-4 shows a timing example of an exception entry without wait states.

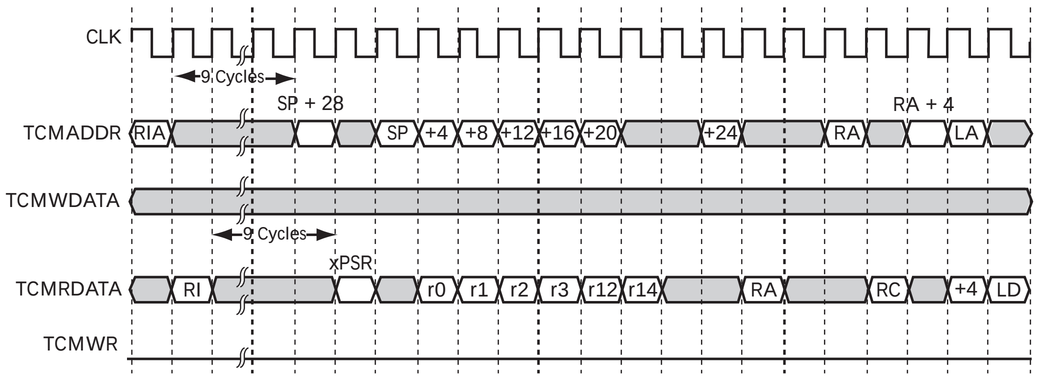


***Figure 3-4* • Exception Entry Without Wait States**

After returning from the exception, the processor automatically pops the eight registers from the stack. The interrupt return value, EXC_RETURN, passes as a data field in the LR, so exception functions can be normal C/C++ functions and do not require a veneer.

Nested Vectored Interrupt Controller

The NVIC facilitates low-latency exception and interrupt handling and implements System Control Registers. The NVIC supports reprioritizable interrupts. The NVIC and the processor core interface are closely coupled, which enables low-latency interrupt processing and efficient processing of late arriving interrupts. The NVIC registers are listed in Table 3-2 and can only be accessed using word transfers. Any attempt to write a halfword or byte individually causes corruption of the register bits. All NVIC registers and system debug registers are little-endian regardless of the endianness state of the processor.

***Table 3-2* • Cortex-M1 NVIC Registers**

Name of Register	Type	Address	Reset Value
Irq 0 to 31 Set Enable Register	R/W	0xE000E100	0x00000000
Irq to 31 Clear Enable Register	R/W	0xE000E180	0x00000000
Irq to 31 Set Pending Register	R/W	0xE000E200	0x00000000
Irq to 31 Clear Pending Register	R/W	0xE000E280	0x00000000
Priority 0 Register	R/W	0xE000E400	0x00000000
Priority 1 Register	R/W	0xE000E404	0x00000000
Priority 2 Register	R/W	0xE000E408	0x00000000
Priority 3 Register	R/W	0xE000E40C	0x00000000
Priority 4 Register	R/W	0xE000E410	0x00000000
Priority 5 Register	R/W	0xE000E414	0x00000000
Priority 6 Register	R/W	0xE000E418	0x00000000
Priority 7 Register	R/W	0xE000E41C	0x00000000

The processor supports both level and pulse interrupts. A level interrupt is held asserted until it is cleared by the ISR accessing the device. A pulse interrupt is a variant of an edge model. The edge must be sampled on the rising edge of the processor clock (HCLK) instead of being asynchronous.

For level interrupts, if the signal is not deasserted before the return from the interrupt routine, the interrupt remains pending and re-activates. This is particularly useful for FIFO and buffer-based devices because it ensures that they drain either by a single ISR or by repeated invocations, with no extra work. This means that the device continues to assert the signal until the device is empty.

A pulse interrupt must be asserted for at least one HCLK cycle to enable the NVIC to latch the pending bit.

A pulse interrupt can be reasserted during the ISR so that the interrupt can be pending and active at the same time. If this occurs, the application design must ensure that a second pulse does not arrive before the first pulse is activated. The second pulse cannot set the pending bit and would have no effect, because the interrupt is already pending. However, if the interrupt is activated for at least one cycle, the NVIC latches the pending bit. After the ISR activates, the pending bit is cleared. After the bit is cleared if the interrupt is asserted again while it is activated, it can latch the pending bit again.

Note: The number of external interrupts is currently fixed at 1 in the majority of Microsemi M1 devices. A processor configuration with 16 external interrupts is available for the M1AFS1500 and M1A3PE1500 devices.

Clocking and Resets

Clocks

HCLK is the main clock input and clocks the majority of the logic in the processor. When debug logic is included the TCK input is used to clock logic in the debug access port. TCK is not functional when debug logic is excluded. HCLK is also used for clocking some debug components when these are present.

Resets

Within the CortexM1Top level of hierarchy (see Figure 1-1 on page 7) a reset synchronization block takes in a number of reset signals and produces several reset outputs. The synchronization block ensures that resets which may assert asynchronously are deasserted synchronous to the clock domain to which they are relevant.

NSYSRESET is the main reset input. A power-on reset input, PORESETN, is functional when the core is appropriately configured. Set the **Reset to debug logic (DBGRESETn)** configuration option to **Driven by PORESETN input** if you have a power-on reset signal available in your system. Alternatively, you can set this option to **Driven by NSYSRESET input** if no power-on reset is available. This latter setting is less desirable because it means that if NSYSRESET (which is typically connected to a push-button reset) is asserted during a debug session, then the debug connection will be lost as some of the debug logic will be reset. If you have chosen not to include debug logic, the setting chosen for the **Reset to debug logic (DBGRESETn)** option is of no consequence.

WDOGRES and WDOGRESn ports are provided to facilitate support of a watchdog type component such as CoreWatchdog. WDOGRES is the "bark" signal from the watchdog and WDOGRESn is a reset output to the watchdog.

When debug logic is included, the nTRST reset input is functional and is used to reset logic clocked by TCK within the debug port.

The following paragraphs describe the reset outputs from the reset synchronization block and detail the reset sources for each reset output.

HRESETn

HRESETn internally drives the SYSRESETn input of the processor core and is also an output from the top level for use as a synchronized reset to other components in the design clocked by HCLK. HRESETn is asserted whenever any of the following signals assert:

- NSYSRESET (external push-button reset)
- PORESETN (power-on reset signal)
- SYSRESETREQ (reset request signal from processor core)
- WDOGRES (bark signal from watchdog, if present in the processor subsystem)

DBGRESETn

DBGRESETn resets debug logic which is clocked by HCLK and the source of its assertion is dependent on the setting chosen for the **Reset to debug logic (DBGRESETn)** option. When this option is set to **Driven by NSYSRESET input**, DBGRESETn is asserted when NSYSRESET is asserted. When this option is set to **Driven by PORESETN input**, DBGRESETn is asserted when PORESETN is asserted.

WDOGRESn

WDOGRESn is a reset output suitable for connection to a watchdog component such as CoreWatchdog. WDOGRESn is asserted when any of the following signals assert:

- NSYSRESET (external push button reset)
- PORESETN (power-on reset signal)
- SYSRESETREQ (reset request signal from processor core)

Buffering of Clocks and Resets

Buffers are included within the CortexM1Top level of hierarchy to ensure that low skew routing resources are used for clock and reset signals. One important exception to this is the HCLK signal. HCLK is not buffered within the core; you must ensure that the signal driving the HCLK input to the core is driven by a CLKINT buffer.

4 – Software Development

SoftConsole is the free-of-charge Microsemi software development environment that allows quick turn-around for C and C++ based projects targeting Cortex-M1 and other processor-based platforms available for use in Microsemi devices. SoftConsole allows users to create a project and transparently manages all compilation stages in order to generate a binary file ready to be used with the Cortex-M1 processor. SoftConsole includes a fully integrated debugger that offers easy access to memory contents, registers, and single-instruction execution. Programs developed with SoftConsole can be debugged on a target board or in the tool's simulator using the same uniform interface.

SoftConsole provides a flexible and easy-to-use graphical user interface for managing your software development projects. The tool gives you the ability to quickly develop and debug software programs and to implement them in Microsemi devices. SoftConsole enables users to edit and debug software programs, organize files, and configure settings in a project. This tool provides simultaneous access to multiple tool windows and the ability to quickly switch editing, debug, and synchronization views.

The compilation tools can be used to build C or C++ programs. The following tools are included in SoftConsole:

- SoftConsole Eclipse-based IDE
- GCC Compiler
- GDB Debugger
- Support for program download and debug with FlashPro3

There are other tools available from ARM and third-party companies that can be purchased, including the RealView Development Suite, RealView Microcontroller Development Kit, and embedded workbench tools from IAR. The RealView and IAR tools feature compilers that offer a higher level of efficiency than the GCC compiler included in SoftConsole. If higher code density is required for an application than what can be achieved using GNU, Microsemi recommends that one of these other tools be purchased and used.

5 – Cortex-M1 Design Entry Flow

Libero IDE automatically manages Cortex-M1 through the tool flow when it is instantiated in a SmartDesign project system design. The design can consist of only the Cortex-M1 itself or it can include a range of subsystem and higher-level IP blocks. The overall flow is shown in Figure 5-1.

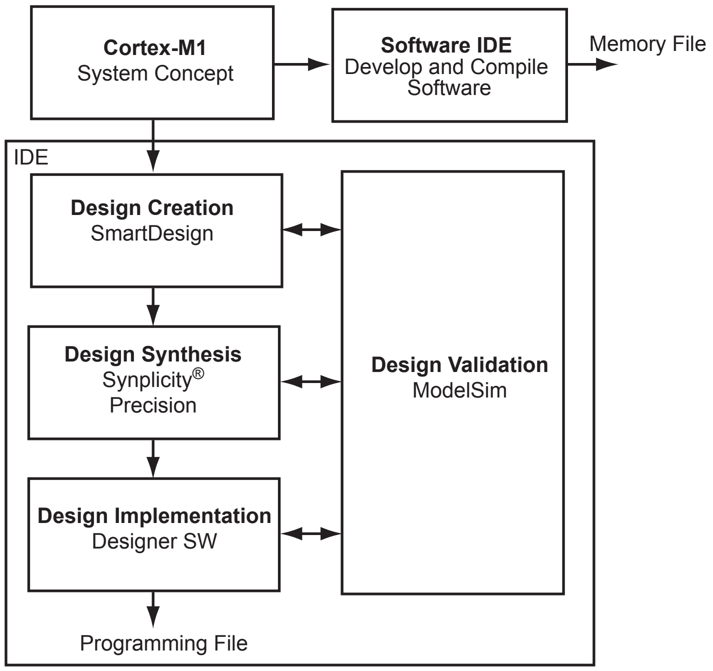


Figure 5-1 • **Cortex-M1 Design Entry Flow**

Use SmartDesign to create a Cortex-M1 based system. The external AHB-Lite interface of Cortex-M1 should be used to master an instance of CoreAHBLite through which the processor can communicate with a range of other AMBA components. The SmartDesign Auto Connect feature can be used to make many of the required connections in a system after you have selected the necessary components and instantiated these on the SmartDesign canvas. A screenshot of an example Cortex-M1 system created in SmartDesign is shown in Figure 5-2 on page 28.

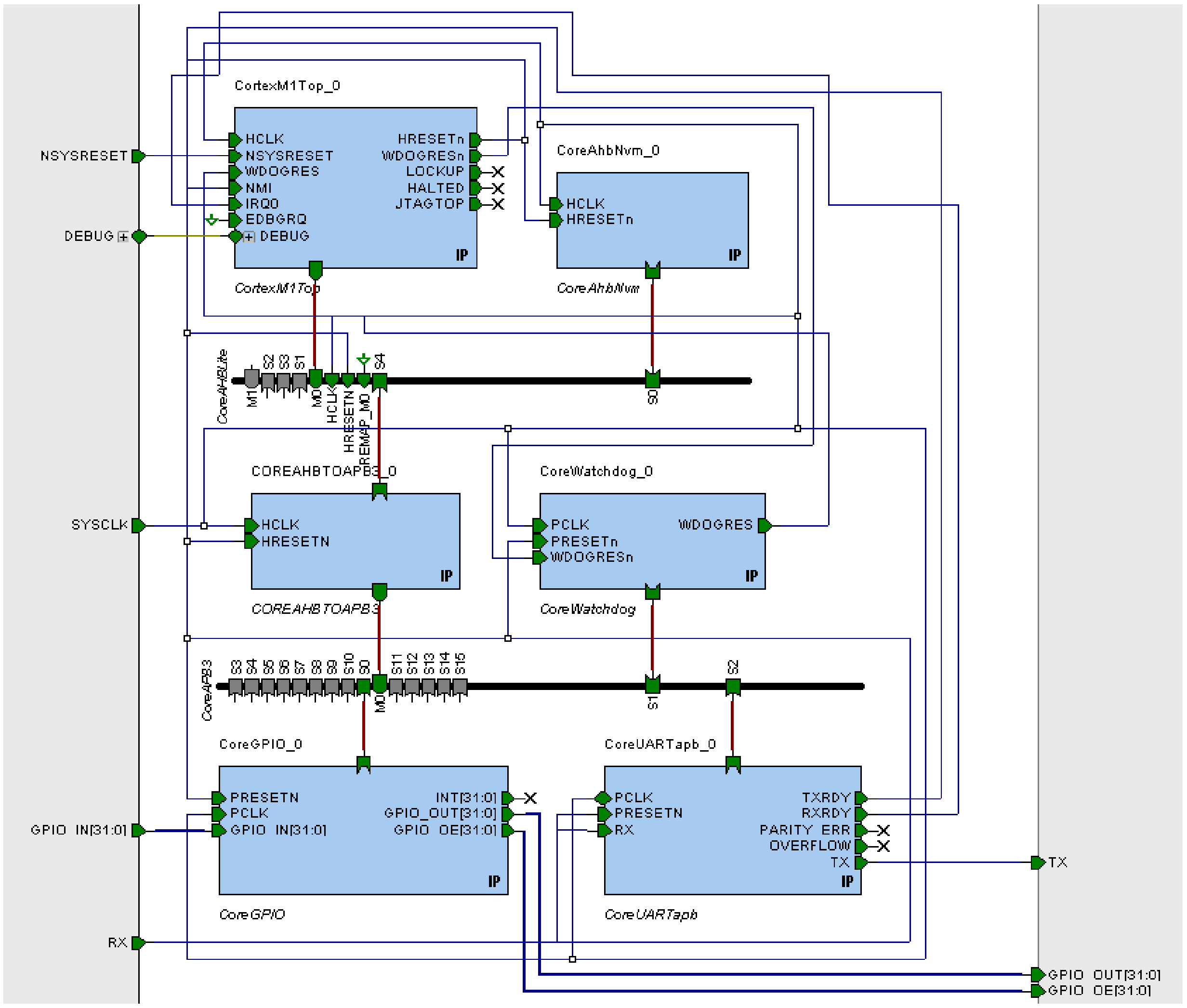


Figure 5-2 • **Cortex-M1 System**

Cortex-M1 Security

The majority of the Cortex-M1 core is contained within a black box CDB file that allows users to access the top level I/O and use the core in Microsemi M1 devices, but not view the contents of the black box. The black box cannot be unlocked and can only be programmed into an Microsemi M1 device. The CDB file includes placement and routing information meaning that the location of the processor is fixed within the FPGA fabric of the target device. Timing shells are included as part of the Cortex-M1 delivery from Microsemi to model the timing at the interface of the CDB. These timing shells are automatically handled by the tools in the Libero IDE and their use is transparent to the Cortex-M1 user.

6 – Bus Functional Model (BFM)

During the development of an FPGA-based SoC, various stages of testing can be performed. This may involve some or all of the following approaches:

- Hardware simulation, using Verilog or VHDL
- Software simulation, using a host-based instruction set simulator (ISS) of the processor
- Hardware and software co-verification, using a full functional model of the processor in Verilog or VHDL form, or using a tool such as Seamless

Due to the rapid prototyping capability of FPGAs, integration of hardware and software often occurs earlier in the development cycle than it would for ASIC targets. Therefore, hardware and software co-verification (which can be very slow) are not as critical an issue for most FPGA-based system-level designs.

SmartDesign provides a means for stitching together IP blocks to create a system. When this system is generated within SmartDesign, a system testbench is also created to aid simulation of the design. In the case of a Cortex-M1 based system, a bus functional model (BFM) of the processor is generated by SmartDesign for use during simulation. Essentially this BFM takes the place of the Cortex-M1 black box during simulation as the black box is not amenable to simulation. SmartDesign also creates script files for controlling the BFM.

This section describes the following aspects of Cortex-M1 BFM:

- Functionality
- BFM usage flow
- BFM script language
- Platforms
- Supported simulation tools
- Example BFM use case

BFM Usage Flow

The BFM is part of an overall system test strategy, so it is helpful to look at the context in which it is used. Figure 6-1 shows the various components within a typical system-level testbench.

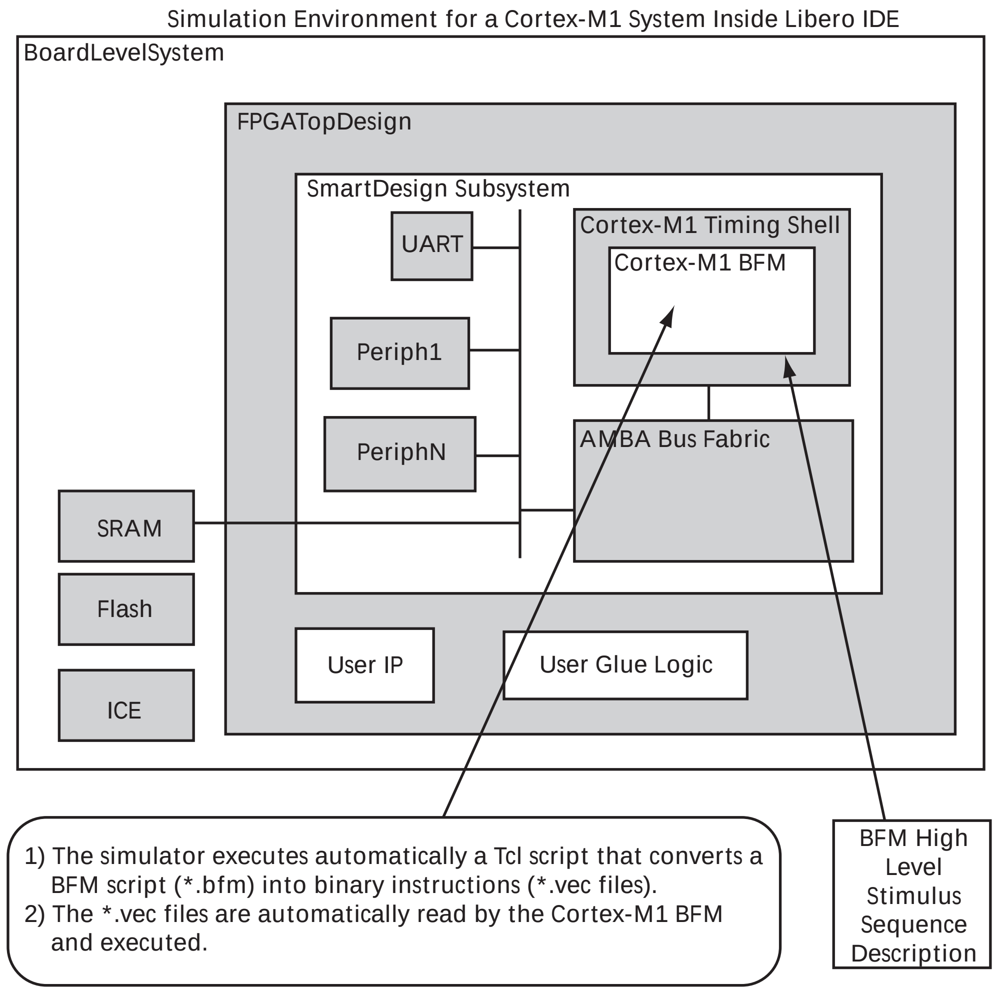

Simulation Environment for a Cortex-M1 System Inside Libero IDE
BoardLevelSystem
FPGATopDesign
SmartDesign Subsystem
UART
Periph1
PeriphN
Cortex-M1 Timing Shell
Cortex-M1 BFM
AMBA Bus Fabric
SRAM
Flash
ICE
User IP
User Glue Logic
1) The simulator executes automatically a Tcl script that converts a BFM script (*.bfm) into binary instructions (*.vec files).
2) The *.vec files are automatically read by the Cortex-M1 BFM and executed.
BFM High Level Stimulus Sequence Description

***Figure 6-1* • BFM Simulation Environment**

In Figure 6-1 it is assumed that the processor subsystem has been specified by selecting the processor, bus fabric, IP blocks, and the memory system using SmartDesign.

Based on the bus connections made, SmartDesign can build up a memory map for the Cortex-M1 system. SmartDesign will generate the following outputs:

- Verilog/VHDL model of SoC subsystem
- Verilog/VHDL models of IP cores
- Cortex-M1 BFM
- BFM test script
- System-level skeleton testbench

The BFM acts as a pin-for-pin replacement for the Cortex-M1 in the project system. It initiates cycle-accurate bus transactions on the native Cortex-M1 bus. It has no knowledge, however, of real Cortex-M1 instructions.

At this point, the BFM may be used to run a basic test of the system, using the skeleton system with the BFM script serving as stimulus for the simulation. This script does a write to and/or read from all accessible locations. It has knowledge of whether registers are read-only, read/write, clear-on-read, or write-only. From this it can decide what the expected data should be on reads.

The system Verilog/VHDL can be edited to add new design blocks in the above diagram. The system-level testbench can be edited to include tasks that test any newly added functionality, or for adding stubs to allow more complex system testing involving the IP cores. The BFM input scripts may also be manually enhanced, so that you can test access to register locations in newly added logic. In this way, stimuli can be provided to the system from the inside (via the Cortex-M1 BFM), as well as from the outside (via testbench tasks).

Functionality

This section describes the specific functionality of the Cortex-M1 BFM. The BFM models transactions on the external (AHB-Lite) bus of Cortex-M1.

Cortex-M1 Pin Compatibility

The BFM model is pin-for-pin compatible with the Cortex-M1. This allows the model to be dropped into the space that would be occupied by the processor core in the system testbench.

Cortex-M1 Bus Cycle Accuracy

The bus cycle timings for the Cortex-M1 external bus signals are specified in the *Cortex-M1 Technical Reference Manual*. The Cortex-M1 BFM models these bus cycles exactly.

Scripting

In order to provide a simple and extensible mechanism for providing stimuli to the BFM, a BFM scripting language is defined (see "BFM Script Language" on page 31). This allows initiating writes to system resources, reads from system resources (with or without checking of expected data), and waiting for interrupt events.

Self-Checking

The BFM gives a pass/fail indication at the end of a test run. This is based on whether or not any of the expected data read checks failed.

Endianness

The BFM supports both big and little-endian memory configurations. For byte and halfword transfers, it reads and writes data from/to the appropriate data lanes.

Interrupt Support

The BFM has the ability to wait for the Cortex-M1 interrupt lines to be triggered before proceeding with the remainder of the test script.

Log File Generation

The BFM generates output messages to the console of the simulation tool, and also generates an HTML log file. The messages in this file are color-coded so that any errors can be easily identified.

BFM Script Language

The following script commands are defined for use by the BFM:

Write

The write command causes the BFM to perform a write to a specified offset, within the memory map range of a specified system resource.

Syntax

```
write width resource_name byte_offset data;
```

Width

This takes on the enumerated values of W, H or B, for word, halfword, or byte respectively.

resource_name

This is a string containing the user-friendly instance name of the resource being accessed.

byte_offset

This is the offset from the base of the resource, in bytes. It is specified as a hexadecimal value.

Data

This is the data to be written. It is specified as a hexadecimal value.

Example

```
write W videoCodec 20 11223344;
```

Read

The read command causes the BFM to perform a read of a specified offset, within the memory map range of a specified system resource.

Syntax

```
read width resource_name byte_offset;
```

Width

:This takes on the enumerated values of W, H or B, for word, halfword, or byte respectively.

resource_name

This is a string containing the user-friendly instance name of the resource being accessed.

byte_offset

This is the offset from the base of the resource, in bytes. It is specified as a hexadecimal value.

Example

```
read W videoCodec 20;
```

Readcheck

The readcheck command causes the BFM to perform a read of a specified offset, within the memory map range of a specified system resource and to compare the read value with the expected value provided.

Syntax

```
readcheck width resource_name byte_offset data;
```

Width

This takes on the enumerated values of W, H or B, for word, halfword, or byte respectively.

resource_name

This is a string containing the user-friendly instance name of the resource being accessed.

byte_offset

This is the offset from the base of the resource, in bytes. It is specified as a hexadecimal value.

Data

This is the expected read data. It is specified as a hexadecimal value.

Example

```
readcheck W videoCodec 20 11223344;
```

poll

This command continuously reads a specified location until a requested value is obtained. This command allows one or more bits of the read data to be masked out. This allows, for example, poll waiting for a ready bit to be set, while ignoring the values of the other bits in the location being read.

Syntax

poll width resource_name byte_offset data_bitmask;

width

This takes on the enumerated values of W, H, or B, for word, halfword, or byte.

resource_name

This is a string containing the user-friendly instance name of the resource being accessed.

byte_offset

This is the offset from the base of the resource, in bytes. It is specified as a hexadecimal value.

bitmask

The bitmask is ANDed with the read data and the result is then compared to the bitmask itself. If equal, then all the bits of interest are at their required value and the poll command is complete. If not equal, then the polling continues.

wait

This command causes the BFM script to stall for a specified number of clock periods.

Syntax

wait num_clock_ticks;

num_clock_ticks

This is the number of Cortex-M1 clock periods, during which the BFM stalls (doesn't initiate any bus transactions).

Timing Shell

A timing shell is provided for each Cortex-M1 variant wrapped around the BFM itself. The BFM is bus cycle accurate and performs setup/hold checks to model output propagation delays.

Example BFM Use Case

This section goes through an example use case of the Cortex-M1 BFM. In this system, the developer requires two additional Microsemi IP cores: the Core10/100 and the CoreUART.

SPIRIT IP-XACT Attributes

SmartDesign has access to a database of Microsemi IP cores and a list of attributes for each core. These attributes are organized according to the SPIRIT IP-XACT specification in XML. For example, in the case of the CoreUART, the attributes would indicate that there are three registers, as shown in Table .

***Table 6-1* • SPIRIT IP-XACT Attributes**

Offset	Register	Read/Write	Width
0	UART Status Register	R	Byte
1	UART Tx Data	W	Byte
2	UART Rx Data	R	Byte

Based on these attributes, SmartDesign can determine when generating the BFM script that there are three locations corresponding to the UART, which may be accessed. In this case, none of the registers are RW, so there will not be any self-checking that can be performed for the UART. Nevertheless, the bus transactions do take place and the cycles can be viewed in a waveform of the simulator.

Automatic BFM Script

After generating a design in SmartDesign, a BFM script is available. This would look similar to the following:

```
read B uart 0;
write B uart 4 bb;
read B uart 8;
write B mac 30 11;
readcheck B mac 11;
```

Run BFM

The developer may run the BFM with the automatic script, or edit the script to put in bus transactions to/from any new logic that has been added to the SoC. For example, transactions to/from the registers in a new block could be added.

The skeleton system-level testbench, generated by SmartDesign can also be modified to add some external resources (for example, models of SSRAM and FLASH) and some high-level tasks.

When running the system simulation, messages appear in the console window of the simulation tool:

```
# read B uart 0;
Reading offset 0 of uart - data = 0x1c
# write B uart 4 bb;
Writing 0xbb to offset 4 of uart
# read B uart 8;
Reading offset 8 of uart - data = 0x28
# write B mac 30 11;
Writing 0x11 to offset 30 of mac
# readcheck B mac 11;
Reading offset 0 of mac
Error: Expected data = 0x11, Actual data = 0x22
Test Failed, with 1 error
```

7 – Debug

The ARM Debug Architecture uses a protocol converter box to allow the debugger to talk via a JTAG port directly to the core.

About Debug

Debug facilitates the following:

- Core halt
- Core stepping
- Core register access
- Read/Write to TCMs
- Read/Write to AHB address space
- Breakpoints
- Watchpoints

The main debug components are as follows:

- Debug control registers to access and control debugging of the core
- BreakPoint Unit (BPU) to implement breakpoints
- Data Watchpoint (DW) unit to implement watchpoints and trigger resources
- Debug memory interfaces to access external ITCM and DTCM
- ROM table

All debug components exist on the internal Private Peripheral Bus (PPB), 0xE000ED30 to 0xE000EEFF. Access to the debug components, debug control, and configuration are only available when the debug extension is present. You can never access the debug components from the processor, even when debug is present.

Debug control and data access occur through the Advanced High-Performance Bus-Access Port (AHB-AP).

Access includes the following:

- The AHB-PPB. Through this bus, the debugger can access debug, including the following:
  - debug control
  - DW unit
  - BPU unit
  - The ROM Table
- The AHB address space. The AHB slaves in the debug system always expect 32-bit AHB transfers. If a byte or halfword access is created from the DAP, the transfer is extended to a 32-bit access and all 32 bits in the register are accessed. Figure 1-1 on page 7 shows an overview of how the debug system interacts with the rest of the processor.

JTAG Debug Interface

The Microsemi FlashPro3 programmer, which is used to program the FPGA and debug the Cortex-M1 core using SoftConsole, uses a standard 10-pin JTAG interface, shown in Figure 7-1. Other debuggers, including those in the RealView and IAR tools, use a 20-pin, 2.54 mm pitch IDC connector (Figure 7-2 on page 36). The cable can be used to mate with a keyed box header on the target.

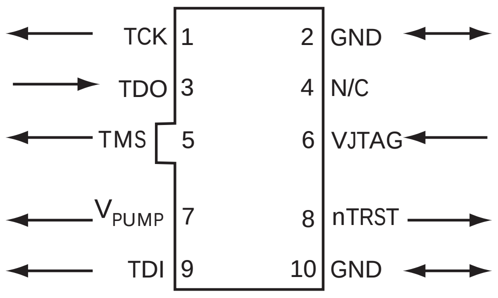


Figure 7-1 **• JTAG 10-Pin Connector Pinout**

The signals on the 10-pin JTAG interface are shown in Table 7-1.

Table 7-1 **• JTAG 10-Pin Connector Signals**

Signal	Description
V_{PUMP}	3.3 V Programming voltage
GND	Signal reference
TCK	JTAG clock
TDI	JTAG data input to device
TDO	JTAG data output from device
TMS	JTAG mode select
nTRST	Programmable output pin may be set to Off, Toggle, Low, or High Level
VJTAG	Reference voltage from the target board
N/C	Programmer does not connect to this pin

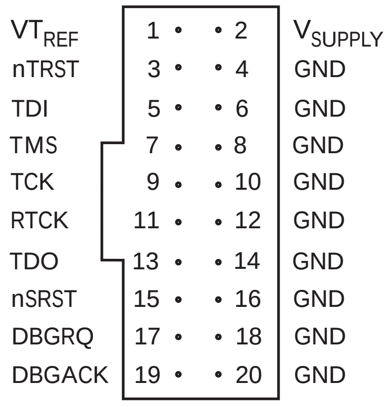


Figure 7-2 **• JTAG 20-Pin Connector Pinout**

The signals on the 20-pin JTAG interface are shown in Table 7-2.

***Table 7-2* • JTAG 20-Pin Connector Signals**

Signal	I/O	Description
DBGACK	–	This pin is connected in the RealView ICE run control unit, but is not supported in the current release of the software. It is reserved for compatibility with other equipment to be used as a debug acknowledge signal from the target system. Microsemi recommends pulling this signal LOW on the target.
DBGRQ	–	This pin is connected in the RealView ICE run control unit, but is not supported in the current release of the software. It is reserved for compatibility with other equipment, to be used as a debug request signal to the target system. This signal is tied LOW. When applicable, RealView ICE uses the core's scanchain 2 to put the core in debug state. Microsemi recommends pulling this signal LOW on the target.
GND	–	Ground
nSRST	Input/ output	Open collector output from RealView ICE to the target system reset. This is also an input to RealView ICE so that a reset initiated on the target can be reported to the debugger. This pin must be pulled HIGH on the target to avoid unintentional resets when there is no connection.
nTRST	Output	Open collector output from RealView ICE to the Reset signal on the target JTAG port. This pin must be pulled HIGH on the target to avoid unintentional resets when there is no connection.
RTCK	Input	Return Test Clock signal from the target JTAG port to RealView ICE. Some targets must synchronize the JTAG inputs to internal clocks. To assist in meeting this requirement, a returned, and retimed, TCK can be used to dynamically control the TCK rate. RealView ICE provides Adaptive Clock Timing, which waits for TCK changes to be echoed correctly before making further changes. Targets that do not have to process TCK can simply ground this pin.
TCK	Output	Test Clock signal from RealView ICE to the target JTAG port. Microsemi recommends pulling this pin LOW on the target.
TDI	Output	Test Data In signal from RealView ICE to the target JTAG port. Microsemi recommends pulling this pin HIGH on the target.
TDO	Input	Test Data Out from the target JTAG port to RealView ICE. Microsemi recommends pulling this pin HIGH on the target.
TMS	Output	Test Mode signal from RealView ICE to the target JTAG port. This pin must be pulled HIGH on the target to avoid adverse effects from any spurious TCKs when there is no connection.
V_{SUPPLY}	Input	This pin is not connected in the RealView ICE run control unit. It is reserved for compatibility with other equipment to be used as a power feed from the target system.
VT_{REF}	Input	This is the target reference voltage. It indicates that the target has power, and It must be at least 0.628 V. VT_{REF} is normally fed from V_{DD} on the target hardware and might have a series resistor (though this is not recommended). There is a 10 kB pull-down resistor on VT_{REF} in RealView ICE.

JTAGTOP

JTAGTOP (JTAG TAP Operation) indicates the state/operation of the JTAG TAP controller state machine contained within the debug access port of the Cortex-M1 processor.

When JTAGTOP is asserted, it implies that the TAP controller is in one of the following four states:

1. Test Logic Reset
2. Run Test/Idle
3. Select DR Scan
4. Select IR Scan

When JTAGTOP is de-asserted, it implies that the TAP controller is in a state other than the four listed above. There are 12 other possible states like Capture DR, Shift DR, Capture IR, Shift IR etc. JTAGTOP indicates the level of activity of the JTAG port. In other words, JTAGTOP signal indicates the internal state of the JTAG TAP controller. This indication of state will be valid whether the debug port is active or not. However, when the debug port is inactive, it is likely that JTAGTOP will be asserted since the state machine will probably be in one of the four states listed above.

The purpose of the JTAGTOP signal is to allow reuse of the TDI and TDO ports for other signals when possible. When the JTAG TAP controller is in one of the four states listed above, the TDI and TDO signals are not in use and so other signals could be multiplexed on these ports at these times, while JTAGTOP used to control the multiplexers. In truth, it's probably unlikely that a user would want to make use of the JTAGTOP signal in one of our devices since there will normally be a substantial number of I/Os available for use on the device. It's also true to say that even where a user did want to make use of JTAGTOP, this could really be done only when the "Debug interface" configuration option is set to "JTAG, not using UJTAG macro".

JTAGTOP is really intended for use when the processor core is used in a dedicated microcontroller type device where pins are likely to be at a premium and it may be very beneficial to be able to reuse some of the debug pins in certain circumstances. As already mentioned, this is less of an issue in our devices since there is more scope to assign pins as needed. ARM cores often provide another output, named JTAGNSW, which indicates whether the debug access port is operating in JTAG or Serial Wire mode. In Serial Wire mode, only two external pins are required; one for a clock signal and one for bi-directional data. Typically the SW pins are overlaid on the JTAG pins and so, similar to the JTAGTOP signal, the JTAGNSW signal can be used to indicate when it is possible to reuse some of the (unused) JTAG pins for other purposes. At present, only JTAG based debug is supported in our releases of the Cortex-M1 and so the JTAGNSW signal does not feature.

8 – AC Parameters

This section gives the AC timing parameters of the Cortex-M1 black box across all M1-enabled devices. These numbers describe the timing at the ports of the black box; that is, the logic contained within the CDB file. The AC parameters are valid for v3.1 of Cortex-M1, operating under worst-case commercial conditions. In the following tables. all times are in nanoseconds.

Note: All AC parameters are measured at commercial range operating conditions.

The parameter definitions are as follows:

- PERIOD is the minimum supported period of the clock signal specified.
- SETUP is the minimum time for which the specified input signal must be valid before the rising edge of the specified clock.
- HOLD is the minimum time for which the specified input signal must remain valid after the rising edge of the specified clock.
- CLOCK2OUT is the maximum propagation delay from the rising edge of the specified clock to the specified output signal to be guaranteed valid.

The AC parameters are shown in Table 8-1 on page 40 through Table 8-6 on page 45.

AC Parameters

***Table 8-1* • AC Parameters for Cortex-M1 Black Box on ProASIC3 Devices**

			Device/Configuration						
			M1A3P250	**M1A3P400**	**M1A3P400**	**M1A3P600**	**M1A3P600**	**M1A3P1000**	**M1A3P1000**
Parameter	**Clock**	**Signal**	**028910**	**028910**	**028911**	**028910**	**028911**	**028910**	**028911**
PERIOD	HCLK	–	15.133	16.226	16.982	14.948	15.755	15.093	15.711
PERIOD	SWCLKTCK	–	15.133	16.226	16.228	14.948	11.928	15.093	12.548
SETUP	HCLK	HRDATA	2.305	1.574	5.625	1.649	5.462	1.594	5.73
SETUP	HCLK	HREADY	5.146	5.861	7.415	5.123	6.757	5.262	6.654
SETUP	HCLK	HRESP	4.597	5.518	6.644	4.513	6.461	4.661	5.816
SETUP	HCLK	IRQ	2.138	1.726	4.469	1.395	3.692	1.889	4.121
SETUP	HCLK	NMI	3.995	3.028	2.831	4.145	3.23	3.124	2.565
SETUP	HCLK	SYSRESETn	0.277	0.29	0.294	0.264	0.235	0.286	0.239
CLOCK2OUT	HCLK	HADDR	2.942	3.709	6.191	2.882	5.657	3.588	6.056
CLOCK2OUT	HCLK	HPROT	2.567	2.445	4.727	3.095	5.248	2.366	4.908
CLOCK2OUT	HCLK	HSIZE	1.136	1.133	4.741	1.203	4.902	1.234	4.838
CLOCK2OUT	HCLK	HTRANS	4.836	5.741	6.783	4.915	6.619	4.892	6.66
CLOCK2OUT	HCLK	HWDATA	2.882	2.622	3.852	3.098	3.719	3.012	4.606
CLOCK2OUT	HCLK	HWRITE	2.414	2.456	4.607	2.536	4.712	2.511	4.694
CLOCK2OUT	HCLK	LOCKUP	3.24	2.722	3.831	2.71	3.889	2.434	3.08
CLOCK2OUT	HCLK	SYSRESETREQ	1.772	1.797	3.383	1.791	2.588	1.912	1.899
SETUP	HCLK	EDBGRQ	–	–	-0.149	–	-0.137	–	-0.272
SETUP	HCLK	DBGRESETn	–	–	0.267	–	0.279	–	0.262
CLOCK2OUT	HCLK	HALTED	–	–	4.769	–	4.573	–	3.253
SETUP	SWCLKTCK	DBGRESETn	–	–	0.267	–	0.252	–	0.255
SETUP	SWCLKTCK	SWDITMS	–	–	2.941	–	3.419	–	4.112
SETUP	SWCLKTCK	TDI	–	–	2.64	–	3.128	–	4.273
SETUP	SWCLKTCK	nTRST	–	–	3.155	–	2.53	–	3.901
CLOCK2OUT	SWCLKTCK	JTAGTOP	–	–	2.887	–	3.006	–	3.309
CLOCK2OUT	SWCLKTCK	TDO	–	–	1.103	–	1.072	–	1.173
CLOCK2OUT	SWCLKTCK	nTDOEN	–	–	1.111	–	1.155	–	1.174

***Table 8-2* • AC Parameters for Cortex-M1 Black Box on ProASIC3E Devices**

			Device/Configuration					
			M1A3PE1500	**M1A3PE1500**	**M1A3PE1500**	**M1A3PE1500**	**M1A3PE3000**	**M1A3PE3000**
Parameter	**Clock**	**Signal**	**028910**	**028911**	**134820**	**134821**	**028910**	**028911**
PERIOD	HCLK	–	15.242	15.674	21.573	20.671	16.042	16.121
PERIOD	SWCLKTCK	–	15.242	15.808	21.573	12.904	16.042	15.77
SETUP	HCLK	HRDATA	1.123	4.735	1.588	6.023	1.412	5.027
SETUP	HCLK	HREADY	5.545	6.7	5.265	6.626	5.558	6.487
SETUP	HCLK	HRESP	5.106	6.412	4.938	7.677	4.481	5.948
SETUP	HCLK	IRQ	1.134	4.044	4.962	4.117	1.278	3.773
SETUP	HCLK	NMI	2.815	3.379	2.623	2.892	2.532	2.092
SETUP	HCLK	SYSRESETn	0.27	0.242	0.249	0.24	0.268	0.25
CLOCK2OUT	HCLK	HADDR	3.593	5.326	3.339	5.11	4.133	5.778
CLOCK2OUT	HCLK	HPROT	2.487	4.599	2.683	4.377	2.852	4.85
CLOCK2OUT	HCLK	HSIZE	1.378	4.954	1.425	4.402	1.789	5.264
CLOCK2OUT	HCLK	HTRANS	5.272	6.555	4.792	6.444	5.686	7.133
CLOCK2OUT	HCLK	HWDATA	3.392	3.604	3.707	5.759	3.38	4.834
CLOCK2OUT	HCLK	HWRITE	3.663	4.441	2.785	4.727	3.067	5.483
CLOCK2OUT	HCLK	LOCKUP	2.863	3.852	2.912	3.138	3.322	3.645
CLOCK2OUT	HCLK	SYSRESETREQ	2.084	3.072	2.067	2.023	2.595	2.368
SETUP	HCLK	EDBGRQ	–	–0.427	–	–0.408	–	–0.726
SETUP	HCLK	DBGRESETn	–	0.27	–	0.272	–	0.277
CLOCK2OUT	HCLK	HALTED	–	4.445	–	3.45	–	3.488
SETUP	SWCLKTCK	DBGRESETn	–	0.241	–	0.312	–	0.308
SETUP	SWCLKTCK	SWDITMS	–	3.917	–	3.526	–	5.532
SETUP	SWCLKTCK	TDI	–	3.691	–	2.78	–	3.061
SETUP	SWCLKTCK	nTRST	–	3.918	–	3.75	–	2.849
CLOCK2OUT	SWCLKTCK	JTAGTOP	–	3.289	–	3.503	–	4.882
CLOCK2OUT	SWCLKTCK	TDO	–	1.316	–	1.329	–	1.64
CLOCK2OUT	SWCLKTCK	nTDOEN	–	1.316		1.311	–	1.658

Table 8-3 **• AC Parameters for Cortex-M1 Black Box on ProASIC3L Devices**

			Device/Configuration					
			M1A3P600L	**M1A3P600L**	**M1A3P1000L**	**M1A3P1000L**	**M1A3PE3000L**	**M1A3PE3000L**
Parameter	**Clock**	**Signal**	**028910**	**028911**	**028910**	**028911**	**028910**	**028911**
PERIOD	HCLK	–	18.721	19.181	18.151	18.984	19.594	19.698
PERIOD	SWCLKTCK	–	18.721	17.05	18.151	17.928	19.594	14.284
SETUP	HCLK	HRDATA	1.725	6.704	1.334	7.321	1.58	7.138
SETUP	HCLK	HREADY	7.395	8.586	7.725	9.31	6.459	8.018
SETUP	HCLK	HRESP	6.701	7.719	8.742	7.318	5.67	6.75
SETUP	HCLK	IRQ	1.319	4.477	1.805	4.129	1.311	4.151
SETUP	HCLK	NMI	3.513	4.737	3.92	3.384	3.084	2.666
SETUP	HCLK	SYSRESETn	0.339	0.314	0.346	0.323	0.32	0.325
CLOCK2OUT	HCLK	HADDR	4.456	6.398	4.834	7.618	5.716	7.357
CLOCK2OUT	HCLK	HPROT	3.098	6.067	2.989	5.361	5.156	5.985
CLOCK2OUT	HCLK	HSIZE	1.427	5.597	1.526	6.136	2.205	6.2
CLOCK2OUT	HCLK	HTRANS	6.783	8.316	6.863	8.502	7.148	7.936
CLOCK2OUT	HCLK	HWDATA	3.428	4.601	3.611	6.154	4.178	5.857
CLOCK2OUT	HCLK	HWRITE	3.726	5.011	3.458	7.676	4.07	5.364
CLOCK2OUT	HCLK	LOCKUP	3.527	4.733	3.36	3.647	4.538	4.226
CLOCK2OUT	HCLK	SYSRESETREQ	2.328	2.949	2.345	2.358	3.448	4.174
SETUP	HCLK	EDBGRQ	–	-0.148	–	-0.232	–	-0.785
SETUP	HCLK	DBGRESETn	–	0.341	–	0.339	–	0.398
CLOCK2OUT	HCLK	HALTED	–	5.487	–	3.475	–	4.441
SETUP	SWCLKTCK	DBGRESETn	–	0.33	–	0.317	–	0.383
SETUP	SWCLKTCK	SWDITMS	–	6.123	–	6.979	–	5.656
SETUP	SWCLKTCK	TDI	–	5.242	–	4.281	–	4.903
SETUP	SWCLKTCK	nTRST	–	3.632	–	3.423	–	3.836
CLOCK2OUT	SWCLKTCK	JTAGTOP	–	4.051	–	3.931	–	4.484
CLOCK2OUT	SWCLKTCK	TDO	–	1.324	–	1.446	–	2.002
CLOCK2OUT	SWCLKTCK	nTDOEN	–	1.324	–	1.444	–	2.002

Table 8-4 • **AC Parameters for Cortex-M1 Black Box on Fusion Devices**

			Device/Configuration						
			M1AFS250	M1AFS600	M1AFS600	M1AFS1500	M1AFS1500	M1AFS1500	M1AFS1500
Parameter	**Clock**	**Signal**	**028910**	**028910**	**028911**	**028910**	**028911**	**134820**	**134821**
PERIOD	HCLK		15.606	15.659	15.917	16.684	16.197	21.773	21.124
PERIOD	SWCLKTCK		15.606	15.659	11.472	16.684	11.692	21.773	13.54
SETUP	HCLK	HRDATA	1.535	1.077	5.422	1.147	5.885	1.793	5.064
SETUP	HCLK	HREADY	5.513	5.363	6.43	4.532	6.21	5.194	6.337
SETUP	HCLK	HRESP	5.097	4.894	6.383	5.575	5.121	4.569	6.919
SETUP	HCLK	IRQ	1.569	1.577	3.313	1.546	3.118	5.02	5.366
SETUP	HCLK	NMI	3.737	3.266	2.377	3.171	2.032	2.68	3.289
SETUP	HCLK	SYSRESETn	0.292	0.293	3.751	0.282	0.26	0.25	0.249
CLOCK2OUT	HCLK	HADDR	2.934	3.72	5.428	3.316	5.562	3.881	4.886
CLOCK2OUT	HCLK	HPROT	2.552	2.505	4.069	3.133	4.77	2.765	3.746
CLOCK2OUT	HCLK	HSIZE	1.174	1.167	4.365	1.429	4.839	1.463	3.912
CLOCK2OUT	HCLK	HTRANS	4.709	4.883	6.336	4.698	6.191	5.172	6.217
CLOCK2OUT	HCLK	HWDATA	2.789	3.131	3.728	3.862	4.295	4.075	4.172
CLOCK2OUT	HCLK	HWRITE	2.303	2.391	4.91	2.735	4.409	2.815	4.384
CLOCK2OUT	HCLK	LOCKUP	3.76	2.758	3.23	3.173	2.92	2.515	3.732
CLOCK2OUT	HCLK	SYSRESETREQ	1.805	1.786	1.849	3.006	2.002	2.108	2.841
SETUP	HCLK	EDBGRQ	–	–	-0.189	–	-0.405	–	-0.465
SETUP	HCLK	DBGRESETn	–	–	0.256	–	0.279	–	0.269
CLOCK2OUT	HCLK	HALTED	–	–	2.32	–	3.166	–	3.203
SETUP	SWCLKTCK	DBGRESETn	–	–	0.257	–	0.24	–	0.273
SETUP	SWCLKTCK	SWDITMS	–	–	3.751	–	3.081	–	4.216
SETUP	SWCLKTCK	TDI	–	–	3.032	–	4.354	–	1.884
SETUP	SWCLKTCK	nTRST	–	–	5.953	–	5.35	–	4.613
CLOCK2OUT	SWCLKTCK	JTAGTOP	–	–	3.574	–	3.969	–	3.78
CLOCK2OUT	SWCLKTCK	TDO	–	–	1.12	–	1.334	–	1.351
CLOCK2OUT	SWCLKTCK	nTDOEN	–	–	1.12	–	1.334	–	1.331

Table 8-5 • **AC Parameters for Cortex-M1 Black Box on IGLOO Devices (Part 1)**

Parameter	Clock	Signal	Device/Configuration						
			M1AGL250V2	**M1AGL250V5**	**M1AGL600V2**	**M1AGL600V2**	**M1AGL600V5**	**M1AGL600V5**	**M1AGL1000V2**
			028910	**028910**	**028910**	**028911**	**028910**	**028911**	**028910**
PERIOD	HCLK		37.608	22.994	39.325	39.822	23.545	24.724	38.746
PERIOD	SWCLKTCK		37.608	22.994	39.325	31.132	23.545	17.229	38.746
SETUP	HCLK	HRDATA	3.694	2.056	3.271	16.731	2.022	10.279	2.241
SETUP	HCLK	HREADY	17.98	10.417	17.788	20.205	12.001	13.526	16.978
SETUP	HCLK	HRESP	13.902	9.377	16.121	18.644	10.444	10.372	14.138
SETUP	HCLK	IRQ	9.631	2.622	3.943	8.61	2.683	7.738	5.802
SETUP	HCLK	NMI	8.399	6.827	8.374	6.056	4.496	3.603	5.86
SETUP	HCLK	SYSRESETn	0.296	0.291	0.287	11.615	0.265	0.282	0.348
CLOCK2OUT	HCLK	HADDR	8.764	5.994	9.144	17.564	7.245	9.104	8.907
CLOCK2OUT	HCLK	HPROT	5.535	4.494	5.686	10.998	5.075	5.849	6.237
CLOCK2OUT	HCLK	HSIZE	2.913	1.736	2.979	12.006	1.757	5.594	3.048
CLOCK2OUT	HCLK	HTRANS	17.068	6.906	13.472	21.567	8.125	10.831	14.747
CLOCK2OUT	HCLK	HWDATA	6.94	4.221	8.04	11.529	4.607	7.865	7.475
CLOCK2OUT	HCLK	HWRITE	6.112	5.07	7.225	9.771	4.46	6.647	6.441
CLOCK2OUT	HCLK	LOCKUP	9.044	5.633	7.111	8.535	3.942	4.944	6.967
CLOCK2OUT	HCLK	SYSRESETREQ	4.418	2.734	4.502	4.536	2.669	2.726	4.593
SETUP	HCLK	EDBGRQ	–	–	–	-0.136	–	-0.102	–
SETUP	HCLK	DBGRESETn	–	–	–	0.322	–	0.304	–
CLOCK2OUT	HCLK	HALTED	–	–	–	7.938	–	6.234	–
SETUP	SWCLKTCK	DBGRESETn	–	–	–	0.369	–	0.333	–
SETUP	SWCLKTCK	SWDITMS	–	–	–	11.615	–	5.374	–
SETUP	SWCLKTCK	TDI	–	–	–	9.786	–	5.104	–
SETUP	SWCLKTCK	nTRST	–	–	–	9.324	–	4.319	–
CLOCK2OUT	SWCLKTCK	JTAGTOP	–	–	–	8.488	–	4.695	–
CLOCK2OUT	SWCLKTCK	TDO	–	–	–	2.59	–	1.636	–
CLOCK2OUT	SWCLKTCK	nTDOEN	–	–	–	2.645	–	1.636	–

Table 8-6 • **AC Parameters for Cortex-M1 Black Box on IGLOO Devices (Part 2)**

Parameter	Clock	Signal	Device/Configuration						
			M1AGL1000V2	M1AGL1000V5	M1AGL1000V5	M1AGLE3000V2	M1AGLE3000V2	M1AGLE3000V5	M1AGLE3000V5
			028911	028910	028911	028910	028911	028910	028911
PERIOD	HCLK		39.274	23.607	23.445	38.726	39.434	24.218	24.6
PERIOD	SWCLKTCK		30.178	23.607	20.852	38.726	33.866	24.218	18.202
SETUP	HCLK	HRDATA	13.548	1.939	9.385	3.496	12.563	1.891	10.127
SETUP	HCLK	HREADY	20.22	9.939	13.222	15.702	21.521	9.069	13.768
SETUP	HCLK	HRESP	19.906	9.766	9.648	12.77	14.312	8.886	12.371
SETUP	HCLK	IRQ	9.769	2.482	5.556	6.78	7.304	1.529	6.026
SETUP	HCLK	NMI	6.841	4.667	3.745	5.657	8.017	4.92	2.601
SETUP	HCLK	SYSRESETn	0.305	0.281	0.267	0.378	0.283	0.276	0.243
CLOCK2OUT	HCLK	HADDR	16.207	5.869	9.944	10.736	17.055	7.147	10.949
CLOCK2OUT	HCLK	HPROT	12.706	3.774	7.08	4.92	11.627	4.533	7.806
CLOCK2OUT	HCLK	HSIZE	10.82	1.883	7.696	3.567	10.145	2.342	7.062
CLOCK2OUT	HCLK	HTRANS	21.62	9.502	14.296	15.246	24.032	11.452	13.245
CLOCK2OUT	HCLK	HWDATA	11.638	4.403	7.757	9.997	10.235	6.325	8.733
CLOCK2OUT	HCLK	HWRITE	13.404	3.715	8.51	7.514	10.148	5.001	8.526
CLOCK2OUT	HCLK	LOCKUP	7.78	4.698	4.511	7.663	9.927	4.294	6.666
CLOCK2OUT	HCLK	SYSRESETREQ	4.528	2.84	3.586	5.271	5.129	3.266	3.239
SETUP	HCLK	EDBGRQ	-0.248	–	-0.133	–	-0.759	–	-0.617
SETUP	HCLK	DBGRESETn	0.335	–	0.274	–	0.346	–	0.317
CLOCK2OUT	HCLK	HALTED	6.482	–	4.098	–	10.081	–	4.695
SETUP	SWCLKTCK	DBGRESETn	0.342	–	0.283	–	0.426	–	0.339
SETUP	SWCLKTCK	SWDITMS	9.942	–	7.386	–	8.178	–	7.129
SETUP	SWCLKTCK	TDI	10.315	–	6.13	–	9.715	–	5.547
SETUP	SWCLKTCK	nTRST	6.282	–	2.862	–	4.673	–	3.472
CLOCK2OUT	SWCLKTCK	JTAGTOP	7.543	–	5.064	–	7.424	–	5.624
CLOCK2OUT	SWCLKTCK	TDO	2.658	–	1.682	–	3.187	–	2.116
CLOCK2OUT	SWCLKTCK	nTDOEN	2.664	–	1.695	–	3.187	–	2.116

9 – Ordering Information

Ordering Codes

Cortex-M1 can be ordered through your local Microsemi sales representative. It should be ordered using the following number scheme: CortexM1-XX, where XX is listed in Table 9-1.

***Table 9-1* • Ordering Codes**

XX	Description
OM	RTL for Obfuscated RTL – multiple-use license

Note: CortexM1-OM is included free with a Libero IDE license

10 – List of Changes

The following table lists critical changes that were made in each revision of the document.

Date	Changes	Page
Revision 13 (September 2012)	Added "JTAGTOP" section (SAR 34336).	38
Revision 12 (September 2010)	The core version has been revised to v3.1.	N/A
	Two rows were added to Table 1 • Cortex-M1 Utilization and Performance Data for the M1A3PE1500 device, for configurations 134820 and 134821.	4
	The "Cortex-M1 Processor" section was modified to add M1A3PE1500 to the sentence, "Currently OS extensions and TCMs are not supported on the majority of Microsemi M1 devices but are available when using an M1AFS1500 or M1A3PE1500 device." Mention of the M1A3PE1500 device was similarly added to the following sections of the document:	7
	"Cortex-M1 Configurations" section	8
	Table 1-1 • Cortex-M1 Configurations	8
	"Tightly Coupled Memory (TCM) Interface" section	14
	"Main Stack and Process Stack Access" section	15
	"Nested Vectored Interrupt Controller" section	22
	Table 8-2 • AC Parameters for Cortex-M1 Black Box on ProASIC3E Devices is new.	41
Revision 11 (May 2010)	The core version has been revised to v3.0.	N/A
	Table 1 • Cortex-M1 Utilization and Performance Data was updated.	4
	The "Utilization of Global Nets" section was revised.	5
	The "Cortex-M1 Processor" section was revised. Figure 1-1 • Cortex-M1 Block Diagram was replaced.	7
	References to CoreConsole were removed and information regarding SmartDesign was added as appropriate.	N/A
	The "Cortex-M1 Configurations" section and Figure 1-2 • M1 Configuration Window are new.	8
	The "Delivery and Deployment" chapter was revised, abbreviated, and incorporated into the "Cortex-M1 Overview" chapter.	7
	The Cortex-M1 Signals section was renamed to "Cortex-M1 I/O Ports" and Table 1-2 • Cortex-M1 Port Descriptions was updated.	10
	The "Supported Interfaces" chapter was renamed to "Interfaces", revised, and reorganized.	13
	Several sections in the "Cortex-M1 Features" chapter were revised, clarifying OS extensions, stack pointers, clocks, and resets.	15
	The "Cortex-M1 Design Entry Flow", "Bus Functional Model (BFM)", and "AC Parameters" chapters were revised extensively.	27, 29, 39

A – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group 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 North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

Customer Technical Support Center

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

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

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

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Index

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

50200127-13/9.12