MIV_ESS User Guide

Introduction

The MIV_ESS v2.0 is a Mi-V ecosystem IP core available for the Microchip FPGA and System-on-Chip (SoC) FPGA device families. The core is a multi-featured, highly-configurable, Extended Subsystem (ESS), which supports both bootstrap and base peripherals. It is specifically designed to use with the MIV_RV32 soft processor.

Features

MIV_ESS has the following features.

- Designed for low-power FPGA implementations.
- Highly configurable, compact, and extended subsystem solution for the MIV_RV32 soft processor.
- Optional bootstrap function from the following nonvolatile sources.
  - Serial peripheral interface (SPI) Flash
  - I^2C EEPROM
  - On-chip µPROM (PolarFire® and RTG4™ devices)
- APB interface for bootstrap transfers to TCM on the MIV_RV32 (TAS compatible).
- Optional memory-mapped peripherals.
  - Timer (64-bit with pre-scaler)
  - Watchdog
  - SPI
  - I^2C
  - µDMA with AHB-Lite read port and either AHB-Lite or TAS (APB) write port options
  - Platform-Level Interrupt Controller (PLIC) with up to 31 interrupts
  - GPIO
  - UART
- APB interface to access subsystem memory-mapped peripherals.
- Seven optional external APB interfaces to connect additional peripherals.

Core Versions

This user guide applies to the MIV_ESS v2.0 core. A design guide is also provided as a supplementary document for this core. The following four existing DirectCore IPs are integrated within the MIV_ESS core.

- CoreGPIO v3.2.102
- CoreSPI v5.2.104
- CoreUARTapb v5.7.100
- CoreAPB v4.2.100

For more information about these DirectCore IP cores, see **IP Catalog** → **Peripherals** in the Libero® tool.

Supported Families

- PolarFire®
- PolarFire® SoC
- RT PolarFire®
- RTG4™
- IGLOO®2
- SmartFusion®2

Abbreviations

The following acronyms are used in this document.

Table 1. List of Acronyms

Acronym	Expanded
ECC	Error Correction Code
TCM	Tightly Coupled Memory
TAS	TCM APB Slave
PLIC	Platform Level Interrupt Controller

Table of Contents

1. Resource Utilization and Performance

The Resource Utilization and Performance (RUP) data is listed in Table 1-1 through Table 1-4 for the supported device families. The listed PolarFire information is also applicable to PolarFire SoC and RT PolarFire. This data is indicative only. The overall resource utilization and performance of the core is system-dependent. The RUP data is generated using Libero SoC v2021.2 and Synplify R-2021.03M. The Place-and-Route Logic Element (LE) signifies the number of logic elements used in the synthesized component for benchmarking.

Note: These values are for reference only and vary depending on Place-and-Route runs.

The following tables also list the device resource utilization and performance for selected configurations of the MIV_ESS IP core.

Table 1-1. SPI Boot

Family	Part Number	Synthesis			Place-and-Route LE	Performance (MHz)
		DFF	4LUT	Total		
PolarFire®	MPF500T-1 FCG1152E	379	733	1,112	759	361.7
RTG4™	RTG4150L FCG1657M	380	667	1,047	697	86.8
SmartFusion®2	M2S150T FC1152	380	739	1,119	768	164.1
IGLOO®2	M2GL150 FC1152	380	739	1,119	768	164.1
Configuration Parameters	Bootstrap: Enabled, Bootstrap Source: SPI, UART Enable: Yes, GPIO Enable: Yes					

Table 1-2. I²C Boot

Family	Part Number	Synthesis			Place-and-Route LE	Performance (MHz)
		DFF	4LUT	Total		
PolarFire®	MPF500T-1 FCG1152E	579	901	1,480	950	374.5
RTG4™	RTG4150L FCG1657	578	828	1,406	884	91.4
SmartFusion®2	M2S150T FC1152	579	893	1,472	948	184.8
IGLOO®2	M2GL150 FC1152	579	893	1,472	948	184.8
Configuration Parameters	Bootstrap: Enabled, Bootstrap Source: I²C, I2C Enable: Yes, UART Enable: Yes, GPIO Enable: Yes					

Table 1-3. PolarFire µPROM Boot

Family	Part Number	Synthesis			Place-and-Route LE	Performance (MHz)
		DFF	4LUT	Total		
PolarFire®	MPF500T-1 FCG1152E	438	544	982	655	377.7
RTG4™	RTG4150L FCG1657	—	—	—	—	—
SmartFusion®2	M2S150T FC1152	—	—	—	—	—

.....continued

Family	Part Number	Synthesis			Place-and-Route LE	Performance (MHz)
		DFF	4LUT	Total		
IGLOO®2	M2GL150 FC1152	—	—	—	—	—
Configuration Parameters	Bootstrap: Enabled, Bootstrap Source: µPROM, UART Enable: Yes, GPIO Enable: Yes					

Table 1-4. RTG4 µPROM Boot

Family	Part Number	Synthesis			Place-and-Route LE	Performance (MHz)
		DFF	4LUT	Total		
PolarFire®	MPF500T-1 FCG1152E	—	—	—	—	—
RTG4™	RTG4150L FCG1657	424	527	951	595	88.5
SmartFusion®2	M2S150T FC1152	—	—	—	—	—
IGLOO®2	M2GL150 FC1152	—	—	—	—	—
Configuration Parameters	Bootstrap: Enabled, Bootstrap Source: µPROM, UART Enable: Yes, GPIO Enable: Yes					

1.1 Typical Resource Utilization

The following table lists a breakdown of average resource usage for the MIV_ESS module across the supported families.

Table 1-5. Component Resources

Feature	Parts	Synthesis		
		Average DFF	Average 4LUT	Average Total
Bootstrap SPI	MPF500T-1FCG1152E RTG4150L FCG1657 M2S150T FC1152 M2GL150 FC1152	210	293	503
Bootstrap I^2C		118	110	228
Bootstrap µPROM (PolarFire)		109	97	206
Bootstrap µPROM (RTG4)		86	74	160
uDMA (AHB Write)		217	223	440
uDMA (TAS Write)		414	402	816
GPIO (4 Inouts)		16	6	22
I^2C		162	224	386
PLIC (8 sources)		63	79	142
Timer		192	311	503
Timer (RTC)		195	338	533
UART		114	150	264
Watchdog		133	266	399

Note:
The I^2C module must be enabled while using Bootstrap I2C.

2. MIV_ESS Architecture

The MIV_ESS core has been primarily developed to provide extended subsystem features to the MIV_RV32 soft processor core. It is a compact, highly-configurable, support core which is intended to enhance and simplify the design experience for systems utilizing the MIV_RV32 core.

2.1 Description

The core uses a GUI configurator to generate only the required logic blocks. On Reset, the Bootstrap feature (if enabled) automatically copies the code from SPI, I^2C, or FPGA µPROM memory to the MIV_RV32 TCM. The bootstrap transfer occurs across the TCM APB Slave (TAS) interface and the processor is released from reset on completion of the transfer cycle.

The core supports the following range of optional peripheral modules:

- µDMA with Read/Write to AHB-Lite ports and Write to optional TAS port.
- GPIO interface offering up to 32 inputs and/or outputs.
- I^2C interface to connect to external I^2C compliant devices.
- Platform Level Interrupt Controller (PLIC) configurable with up to 31 interrupts.
- SPI interface to connect to external SPI compliant devices.
- Timer with 64-bit resolution, which can be used as a system timer across multiple processors or as an additional timer resource.
- UART for simple serial communications.
- Watchdog, which performs a system Reset on time-out.

The core has the following APB interfaces:

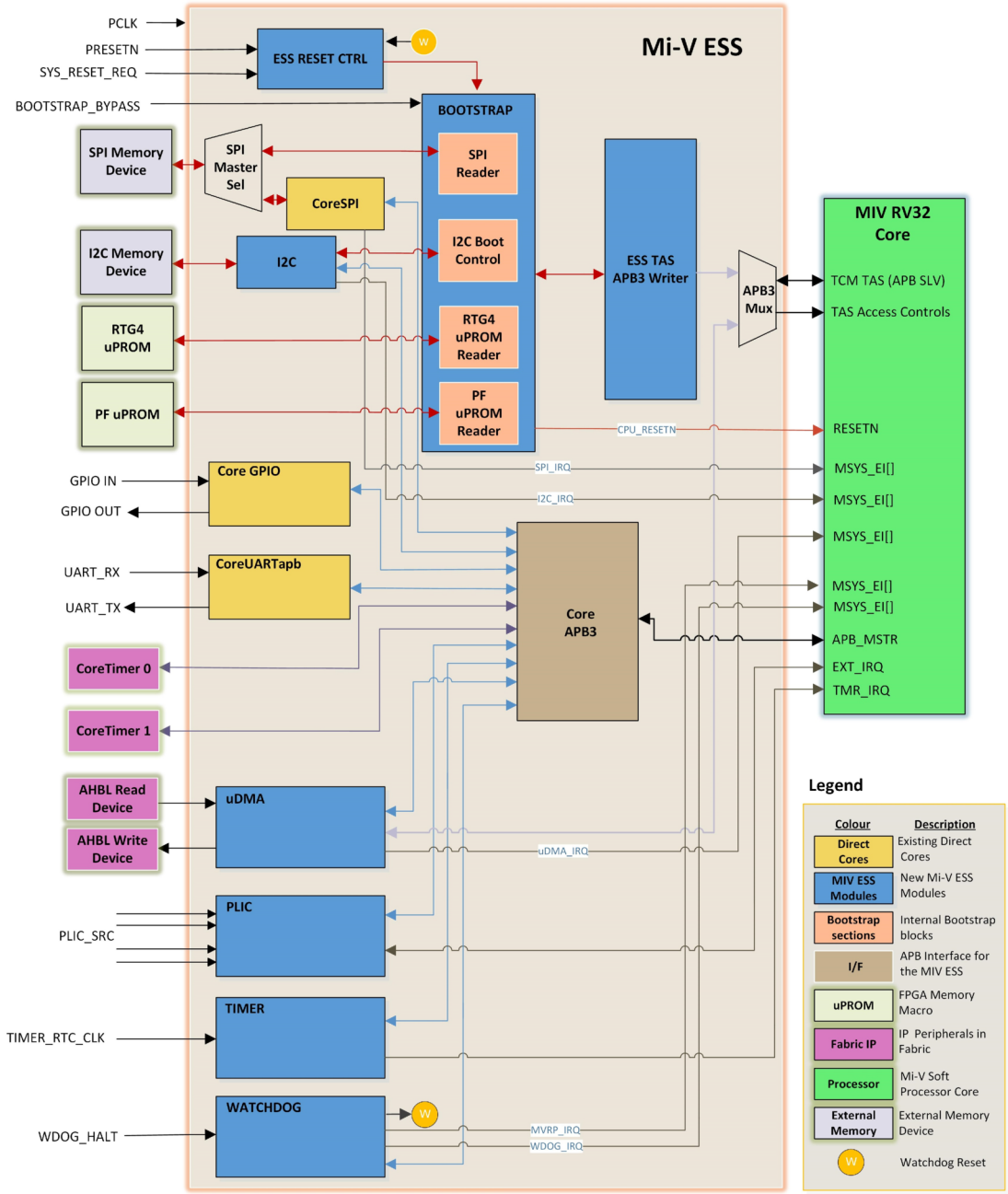
- APB Target interface to access the preceding memory-mapped peripheral modules.
- Optional APB Initiator interface to connect to the MIV_RV32 TAS for bootstrap support and for µDMA Write operations to the TCM.

The GPIO, SPI, and UART are the pre-existing DirectCore IP—CoreGPIO, CoreSPI, and CoreUARTapb. For more information on these cores, see the respective documentation. This document provides information on all other peripherals integrated as MIV_ESS modules.

The Bootstrap feature, once enabled and configured correctly is a hardware boot function which is active following a PRESETN or SYS_RESET_REQ Reset and requires no software intervention. The peripheral modules within the core, once enabled and configured correctly in the Configurator, are accessible through an APB mirrored Initiator/Target interface.

A block diagram of the MIV_ESS is provided in the following figure.

Figure 2-1. MIV_ESS Block Diagram



2.2 Interface

The following table lists the global signal names associated with the core.

Table 2-1. Global Ports

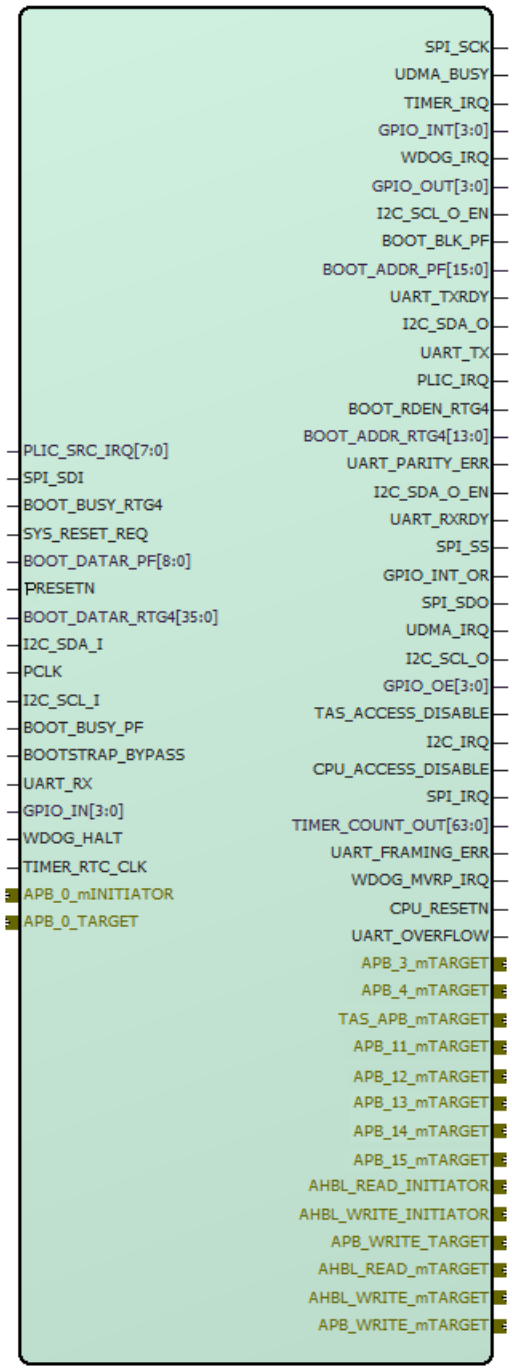
MIV_ESS Ports			
Ports	**Width**	**Direction**	**Description**
PCLK	1	Input	Clock input
PRESETN	1	Input	Active-low reset
SYS_RESET_REQ	1	Input	An active-high reset request from the system. For example, a system controller
APB_0_mINITIATOR			
Ports	**Width**	**Direction**	**Description**
APB_T0_PADDR	32	Input	APB Initiator Interface
APB_T0_PSEL	1	Input	
APB_T0_PENABLE	1	Input	
APB_T0_PWRITE	1	Input	
APB_T0_PRDATA	32	Output	
APB_T0_PWDATA	32	Input	
APB_T0_PREADY	1	Output	
APB_T0_PSLVERR	1	Output	

A synchronous reset architecture is applied when the MIV_ESS core is implemented on RTG4. An asynchronous reset architecture is applied for all other supported FPGA families.

The APB port is a mirrored Initiator configuration by default so it can connect directly to the MIV_RV32 soft processor. Alternatively, the APB port can be configured as a target interface so multiple MIV_ESS IP cores can form part of a larger system.

The core has a number of configuration options that are dependent on the enabled features. Ports are generated to support the configuration as required. The following figure illustrates the complete range of ports available in the core.

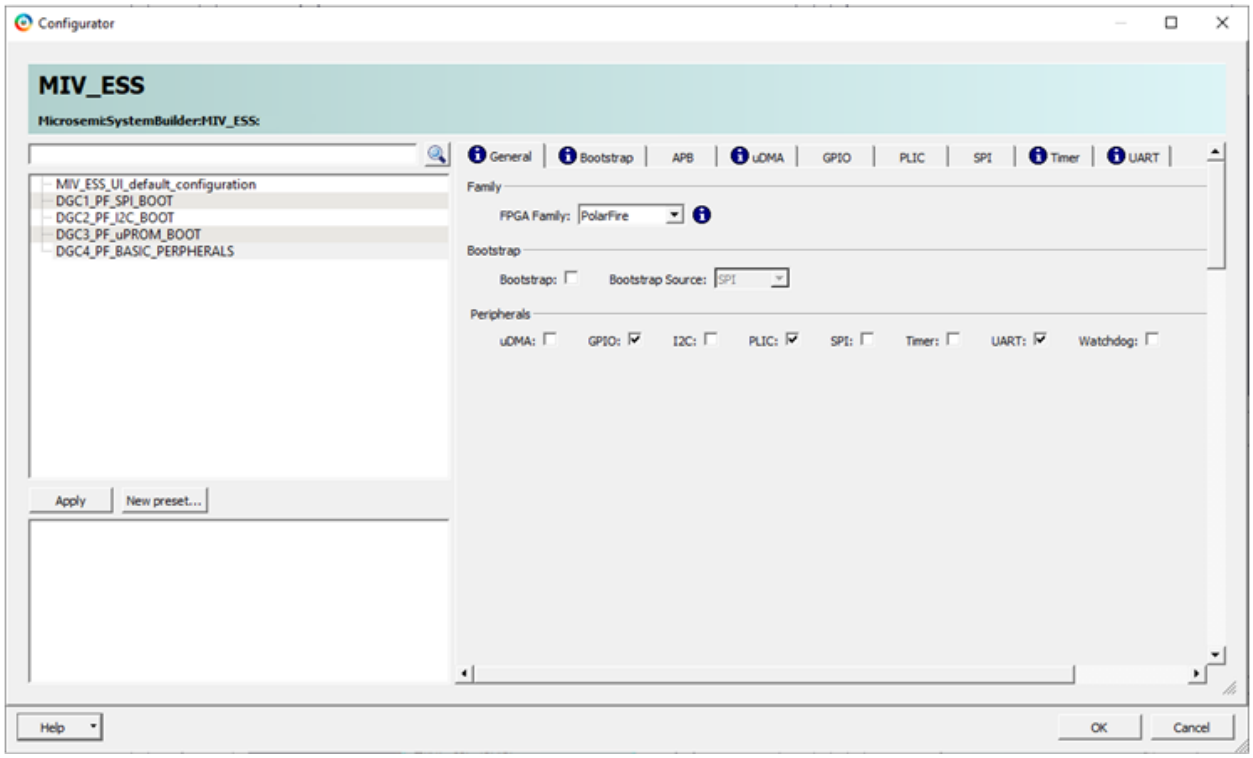
Figure 2-2. MIV_ESS Ports



2.3 Programming

The MIV_ESS core is a highly configurable core, and the configurator provides a top-level general tab which allows the user to enable or disable features.

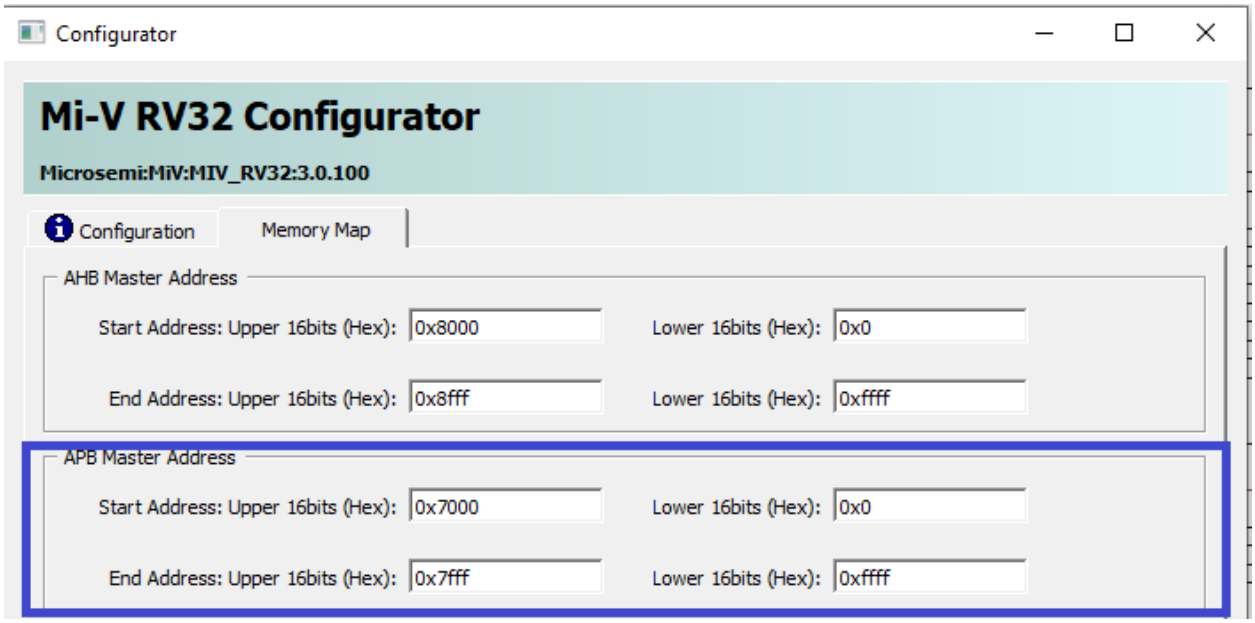
Figure 2-3. General Configuration Tab



The user must select the relevant FPGA family from the dropdown menu. If the Bootstrap feature is enabled, it requires selection of the appropriate Bootstrap Source. For ease of use, there are pre-set Design Guide Configurations DGC 1 – 3 available which can be selected to auto configure the core per the supplementary Design Guide document. Finally, the required peripherals are selected by ticking the appropriate check box.

The MIV_ESS Base Address is determined by the connected APB Initiator address range which must be large enough for the complete memory map of the enabled peripheral modules within the MIV_ESS. The APB address range on MIV_RV32 is set up in the GUI Configurator as shown in the following figure. The default range is 0x7000_0000 to 0x7fff_ffff.

Figure 2-4. Mi-V RV32 APB Address Range



The following table lists the memory-mapped peripheral modules.

Table 2-2. Peripheral Module Address Offsets

Peripheral Modules	Offset from MIV_ESS Base Address	Note
PLIC	0x000_0000	—
UART	0x100_0000	—
TIMER	0x200_0000	—
APB_TARGET	0x300_0000	External APB slot 3
APB_TARGET	0x400_0000	External APB slot 4
GPIO	0x500_0000	—
SPI	0x600_0000	—
RESERVED	0x700_0000	Future use
uDMA	0x800_0000	—
WATCHDOG	0x900_0000	—
I^2C	0xA00_0000	—
APB_TARGET	0xB00_0000	External APB slot 11
APB_TARGET	0xC00_0000	External APB slot 12
APB_TARGET	0xD00_0000	External APB slot 13
APB_TARGET	0xE00_0000	External APB slot 14
APB_TARGET	0xF00_0000	External APB slot 15

3. Bootstrap

This section provides information on the the Bootstrap module used in the MIV_ESS core.

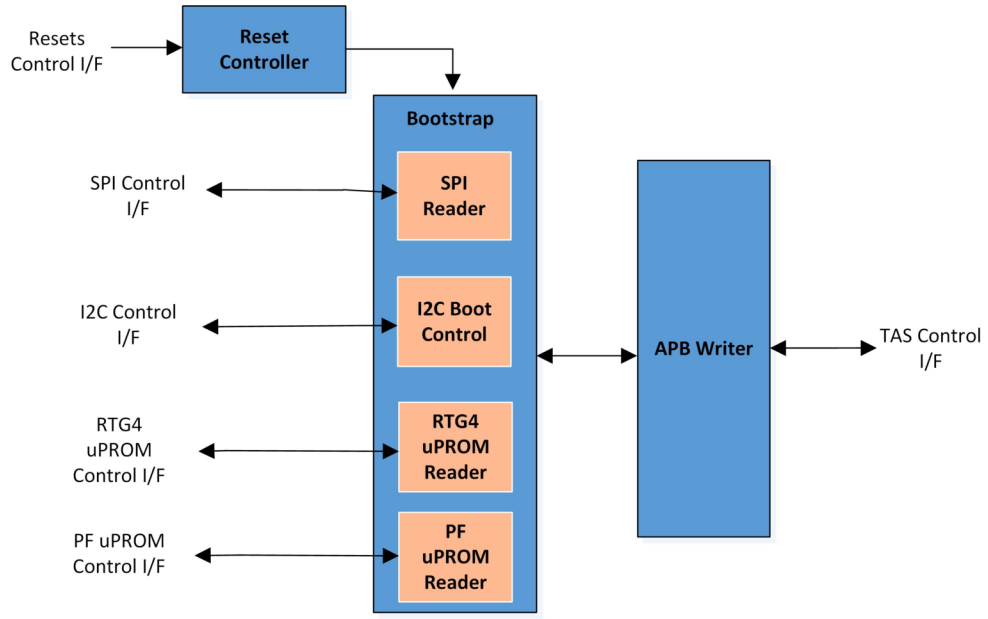
3.1 Description

The Bootstrap module is used for MIV_ESS bootstrap operations and the module operates by integrating the following three components in the MIV_ESS core:

- Bootstrap: A controller unit for accessing external memories such as SPI, I^2C, and µPROM.
- Reset Controller: A controller used for holding the target CPU in Reset so that the boot source data can be transferred into TCM for MIV_RV32 to boot.
- APB Writer unit: It is used to transfer data from target memories SPI/I^2C/µPROM to TCM over the TAS interface.

The following figure describes the bootstrap module.

Figure 3-1. Bootstrap Diagram



The bootstrap controller allows booting the MIV_RV32 soft processor from either a SPI, I^2C, or µPROM device indirectly, by first copying the boot code from the SPI/I^2C/µPROM memory device to the MIV_RV32 internal TCM via the APB TAS interface.

3.1.1 Features

The Bootstrap module has the following features.

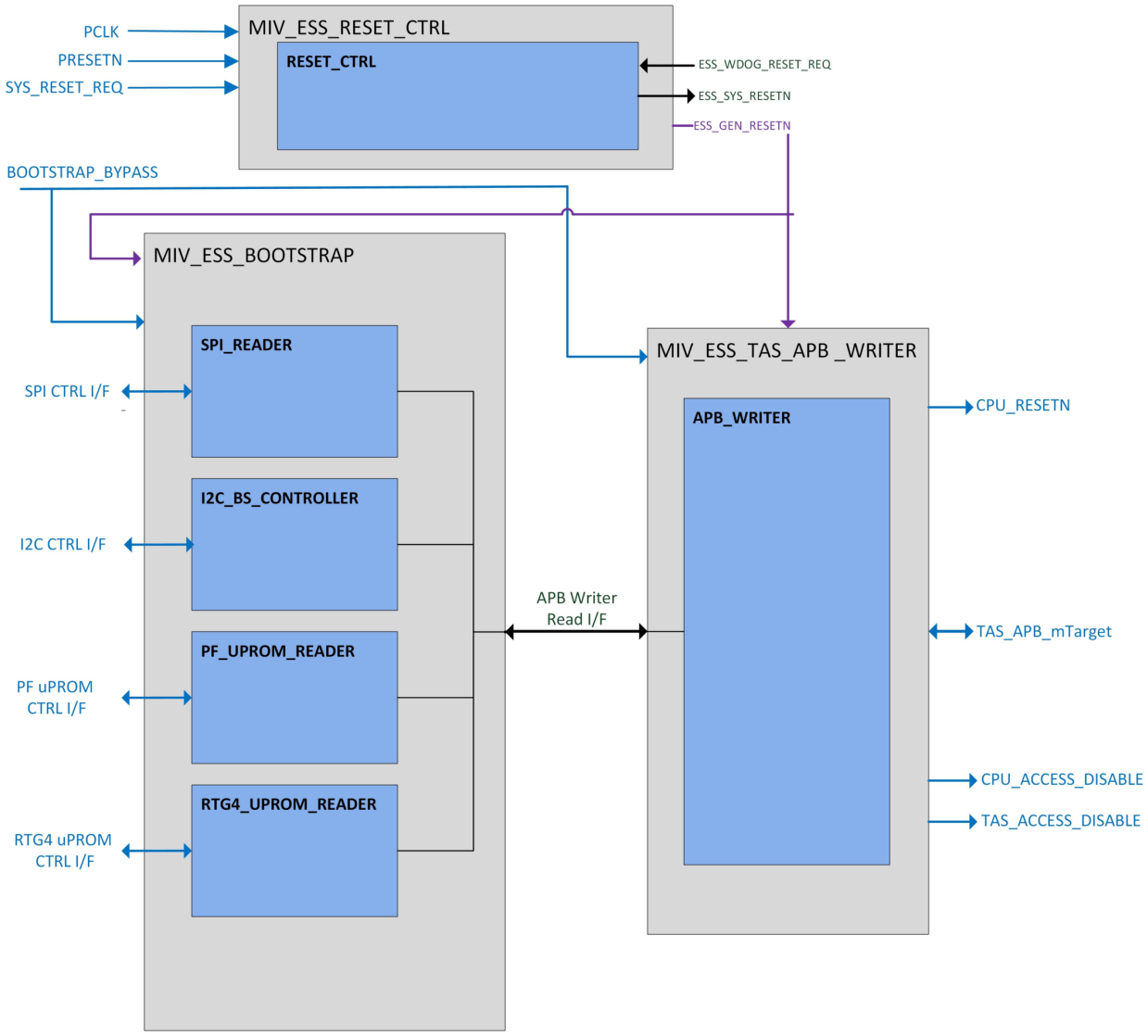
- An APB 3.0 mirrored host interface to use with the MIV_RV32 TCM TAS.
- Support for enabling and disabling MIV_RV32 CPU and TAS I/F access.
- An optional µPROM interface compatible for loading boot code from external PolarFire and RTG4 µPROM memory devices.
- An optional SPI interface for loading boot code from an external SPI memory device.
- An optional I^2C interface for loading data from an external I^2C memory device.
- Supports three Reset sources.
    a. External Reset
    b. System Reset Request
    c. Watchdog Reset Request

- Supports all available SPI Flash chips, through Motorola Mode 0 signaling, and parameterized software Reset command sequences along with various timing parameters to handle differences between SPI chip manufacturers.

3.1.2 Block Diagram

The following figure shows the block diagram of Bootstrap.

Figure 3-2. Bootstrap - Block Diagram



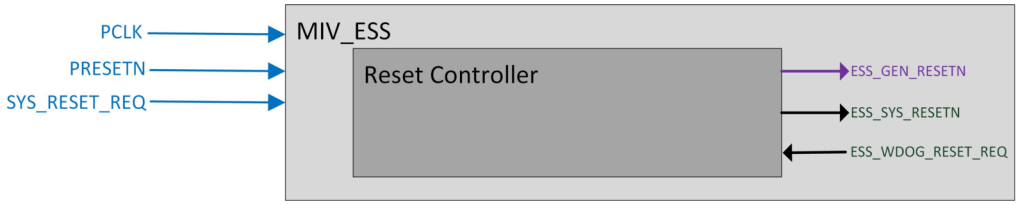
The Bootstrap is responsible for reading the boot code byte-by-byte from the external device (SPI/I^{2}C/µPROM), assembling a 32-bit instruction from the read data, and passing the assembled instructions to the APB Writer.

The APB Writer is responsible for writing 32-bit instructions from the Bootstrap into the MIV_RV32 TCM via the TAS APB interface.

The Reset Controller is responsible for generating the Reset signal used by the Bootstrap and the APB Writer.

The following figure shows the top-level block diagram of the Reset Controller.

Figure 3-3. Reset Controller



3.1.3 Bootstrap Operation

Following the assertion of one of the reset sources in the Reset Controller, the Bootstrap asserts the `CPU_ACCESS_DISABLE` signal for MIV_RV32 and de-asserts the `TAS_ACCESS_DISABLE` signal. This halts the Hart of MIV_RV32 before executing the first instruction, and it allows the APB Writer to access the TCM via the APB TAS interface.

The Bootstrap copies the boot code from SPI/I^{2}C, or µPROM memory device to the MIV_RV32 TCM. After copying with no errors, the Bootstrap de-asserts the `CPU_ACCESS_DISABLE` signal and asserts the `TAS_ACCESS_DISABLE` signal, which allows MIV_RV32 to boot from the code copied into TCM.

The Bootstrap initially takes control over the SPI/I^{2}C/µPROM interface during the booting process, the Bootstrap then finishes by passing control of the SPI/I^{2}C/µPROM interface to the respective modules SPI/I^{2}C/µPROM, allowing for the SPI/I^{2}C/µPROM memory device to be accessed by the MIV_RV32 over an APB interface.

3.1.4 APB Writer Operation

This section describes the APB Writer operation.

1. Following the assertion of the `ESS_GEN_RESETN` signal, the APB Writer.
   - De-asserts the `CPU_RESETN` signal to hold the MIV_RV32 in reset.
   - Asserts the `CPU_ACCESS_DISABLE` signal to block Hart access to TCM.
   - De-asserts the `TAS_ACCESS_DISABLE` signal to allow TAS access to TCM.
2. The APB Writer asserts the `APB_WRITER_RD_READY` signal to indicate that it is ready to receive a 32-bit instruction from the Bootstrap.
3. Once the Bootstrap asserts the `APB_WRITER_RD_DATA_AVAIL` signal, the APB Writer writes the 32-bit instruction from the `APB_WRITER_RD_DATA` line into TCM.
4. If the `PSLVERR` signal on the TAS interface is asserted at any point, the APB Writer will abort the transfer and assert bit [0] of the `APB_ERR` signal.
5. Once the Bootstrap has finished reading data from the source, the `APB_WRITER_RD_ALL_DONE` signal will be asserted. The APB Writer will then read the first instruction back from TCM, and compare it with the first instruction received from the Bootstrap. If a mismatch is detected, bit [1] of the `APB_ERR` signal will be asserted.
6. Once it is checked, the APB Writer waits for the External Processor Reset Duration. After this, the APB Writer:
   - Asserts the `CPU_RESETN` signal to release the MIV_RV32 from reset.
   - De-asserts the `CPU_ACCESS_DISABLE` signal to allow Hart access to TCM.
   - Asserts the `TAS_ACCESS_DISABLE` signal to block TAS access to TCM.

3.1.5 Reset

When the Bootstrap module completes the code transfer into the TCM, the Bootstrap asserts the `CPU_RESETN` signal, which in turn resets the Hart and allows MIV_RV32 boot from TCM.

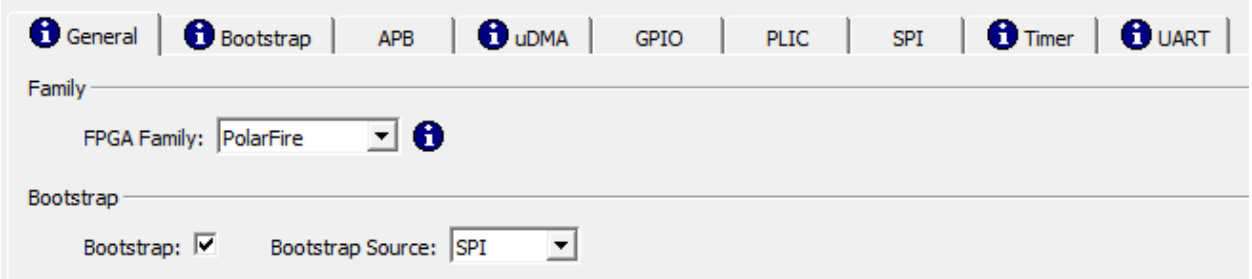
3.2 Interface

This section provides details of the bootstrap interfaces.

3.2.1 General Bootstrap Parameters

The Bootstrap module is enabled from the **General** tab in the MIV_ESS GUI, as shown in the following figure.

Figure 3-4. GUI – General Bootstrap Tab Options



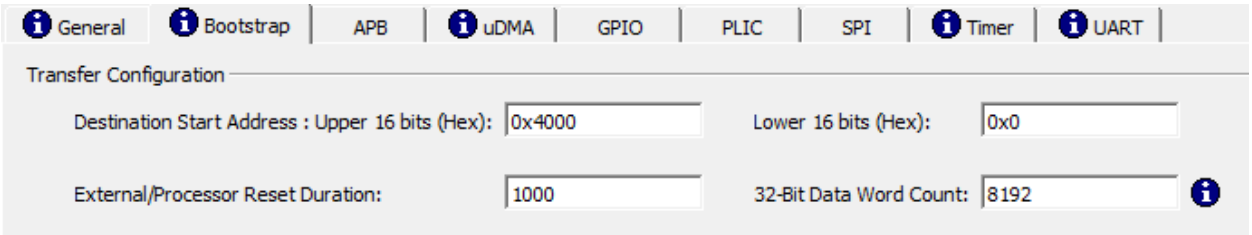
The following table list the parameters that apply to the Bootstrap module.

Table 3-1. Bootstrap General Parameters

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
Bootstrap Enable	BOOTSTRAP_EN	0	0	0 = Disable Bootstrap module 1 = Enable Bootstrap module Option to enable Bootstrap module in the design.
Bootstrap Source	BOOTSTRAP_SOURCE	0 or 1 or 2	0	This option determines the source memory interface to be enabled for Bootstrap transfers. Only a single Bootstrap Source can be selected for the APB Writer to handle transfers from. The inputs and outputs as they appear on the MIV_ESS instance varies depending on the selected memory interface. Available options: 0 = Select 'SPI' as Bootstrap Source memory interface 1 = Select I^2C as Bootstrap Source memory interface 2 = Select 'µPROM' as Bootstrap Source memory interface

After the Bootstrap is enabled, the Bootstrap Source parameters will be available in the Bootstrap tab as shown in the following figure.

Figure 3-5. Bootstrap Transfer Configuration Options



The following table lists the Transfer Configuration parameters apply to all the selected Bootstrap Sources.

Table 3-2. Transfer Configuration Parameters

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
Destination Start Address: Upper 16 bits (Hex)	APB_DST_ADDR_UPPER	0x0 – 0xFFFF	0x4000	Defines the upper 16 bits of target address for the Bootstrap transfers. Together with 'Lower 16 bits (Hex)' it makes up the 32-bit transfer destination address.
Lower 16 bits (Hex)	APB_DST_ADDR_LOWER	0x0 -0xFFFF	0x0000	Defines the lower 16-bits of target address for the Bootstrap transfers. Together with 'Destination Start Address: Upper 16 bits (Hex)' it makes up the 32-bit transfer destination address.
External/Processor Reset Duration	RST_POR_DURATION	4 – 65535	1000	The External Processor Reset Duration. After the completion of the Bootstrap Transfer operation, the Bootstrap module wait for this number of clock cycles before releasing the `CPU_RESETN` signal, allowing the MIV_RV32 Hart to come out of reset.
32-Bit Data Word Count	DATA_WORD_CNT	0 - 262,144	8192	The number of 32-bit words to be read from the source device and transferred. (Source .hex file size in bytes divided by 4) For example, 32 kB = 8192 words

3.2.2 General Bootstrap Ports

The ports that appear on the MIV_ESS core instance in relation to the Bootstrap, vary depending on the selected Bootstrap Source in the design. The following table lists Bootstrap inputs and outputs that are available in the design, if the Bootstrap is enabled, as they are not specific to any Bootstrap Source.

Table 3-3. Bootstrap – General Ports

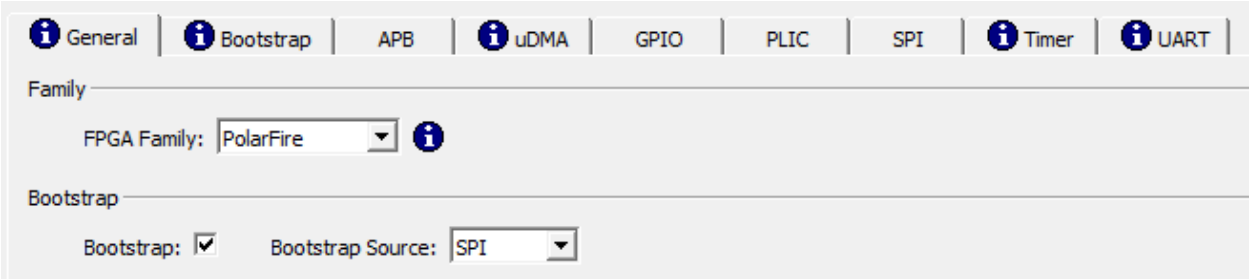
Ports	Width	Direction	Description
BOOTSTRAP_BYPASS	1	Input	An input signal to effectively bypass the Bootstrap if it is enabled in the design. An active High signal that will inhibit the Bootstrap function.
SYS_RESET_REQ	1	Input	Active-high reset request signal.

.....continued			
Ports	Width	Direction	Description
CPU_RESETN	1	Output	Active-low signal for CPU reset. This signal is used to hold the target processor in the reset mode during the Bootstrap transfer operation.
CPU_ACCESS_DISABLE	1	Output	When asserted, CPU's access to the TCM is disabled.
TAS_ACCESS_DISABLE	1	Output	When asserted, TAS access to the TCM is disabled.

3.2.3 SPI Bootstrap Parameters

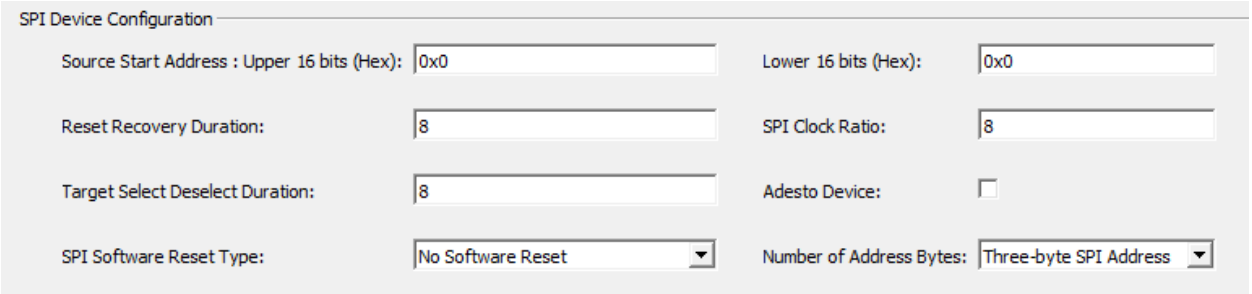
To configure the Bootstrap in SPI mode, enable the Bootstrap from the **General** tab and select SPI as the Bootstrap Source as shown in the following figure.

Figure 3-6. SPI Bootstrap Enable



To configure Bootstrap's SPI transfer parameters, navigate to the **Bootstrap** tab and update parameters under **SPI Device Configuration** as shown in the following figure.

Figure 3-7. SPI Device Configuration



The following table lists the SPI device configuration parameters.

Table 3-4. SPI Device Configuration Parameters

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
Source Start Address : Upper 16 bits (Hex)	SPI_SRC_ADDR_UPPER	0x0 – 0xFFFF	0x0	Upper 16-bits of the SPI Source Start Address: location of the first 32-bit word of boot code in the SPI-Flash memory device.

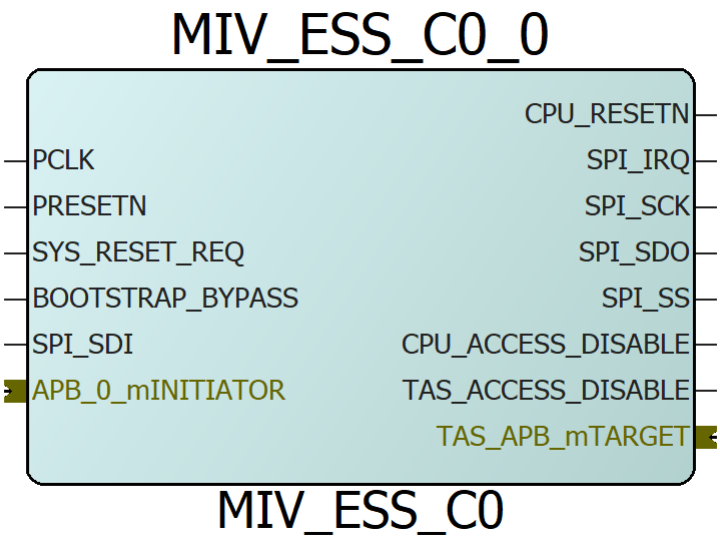
.....continued

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
Lower 16 bits (Hex)	SPI_SRC_ADDR_LOWER	0x0 – 0xFFFF	0x0	Lower 16-bits of the SPI Source Start Address: location of the first 32-bit word of boot code in the SPI-Flash memory device.
Reset Recovery Duration	RST_RECOVERY_DURATION	4 - 65535	8	Number of PCLK cycles following a hardware or software reset before enabling polling the SPI chip. This ranges from under 50 ns to over 100 µs.
SPI Clock Ratio	SPI_CLK_RATIO	1 - 32768	4	The SPI clock prescaler/divider. Indicates the number of PCLK cycles in a SPI_CLK period.
Target Select Deselect Duration	SS_DESELECT_DURATION	1 - 65535	8	The deselect duration in PCLK cycles for the SPI chip's SS (Slave Select) pin between commands.
Adesto Device	READ_STATUS_TYPE	0 or 1	0	Indicates if a Flash memory device made by Adesto Technologies is used.
SPI Software Reset Type	SW_RESET_TYPE	0 - 3	0	SPI Software Reset type: 0 = No software reset 1 = Command sequence 66h, 99h (covers most devices) 2 = 4-byte command "f0,00,00,00" (Adesto devices) 3 = 1-byte "f0" command (Cypress/Spansion devices)
Number of Address Bytes	READ_4BYTE_ADDR	0 or 1	0	0 = 3 byte SPI addressing 1 = 4 byte SPI addressing for SPI chips ≥ 128 Mbit

3.2.4 SPI Bootstrap Ports

The following figure shows the ports as they appear on the MIV_ESS instance, if Bootstrap is enabled and configured in the SPI mode.

Figure 3-8. SPI Ports



The following table lists the Bootstrap SPI Boot Source configuration ports.

Table 3-5. SPI Port Signals

Ports	Width	Direction	Description
SPI_SDI	1	Input	Bootstrap SPI reader Serial Data In.
SPI_SCK	1	Output	Bootstrap SPI reader Serial Clock (out).
SPI_SDO	1	Output	Bootstrap SPI reader Serial Data Out.
SPI_SS	1	Output	Bootstrap SPI reader Chip Select.

3.2.5 I²C Bootstrap Parameters

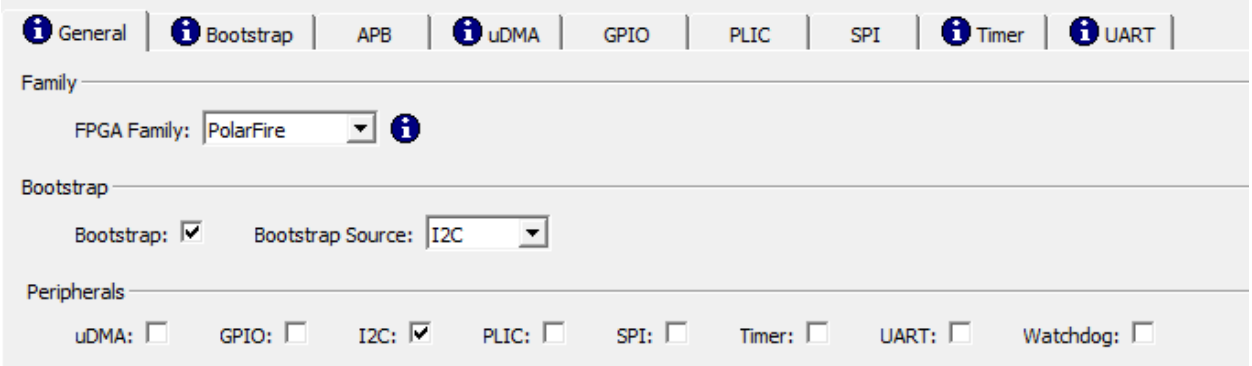
The Bootstrap module has configurable parameters under the **General** and **Bootstrap** tabs in the MIV_ESS core.

To enable the Bootstrap module in MIV_ESS, select the **Bootstrap** check box under the **General** tab in the Bootstrap section. To configure the Bootstrap module to boot from an I²C memory source, the Bootstrap Source parameter value must be **I2C**.

To enable I²C module in MIV_ESS, select **I2C** check box under **Peripherals** → **General**. The I²C module supports Initiator read and write accesses to peripheral I²C devices and can be configured by the Bootstrap Controller to copy I²C boot memory to the TCM of the MIV_RV32 soft processor. See I2C for more details about the I²C module.

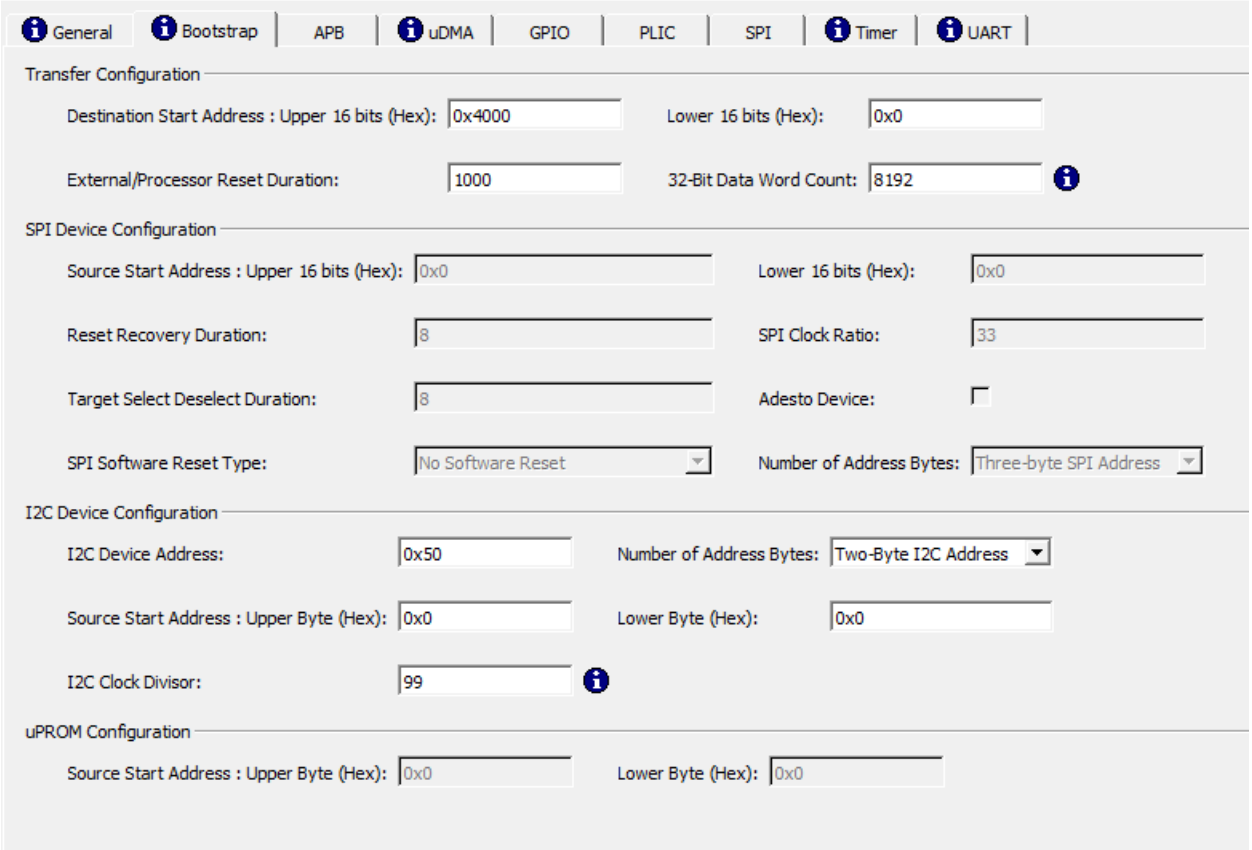
The following figure shows the parameters that need to be configured to use the Bootstrap in the I²C mode.

Figure 3-9. Bootstrap I2C Mode Parameters Under the General Tab



The **I2C Device Configuration** section enables you to configure the I^{2}C settings, as shown in the following figure.
Note: Under the Bootstrap tab, all sections are disabled except **I2C Device Configuration** as the Bootstrap Source is set to **I2C** in the **General** tab.

Figure 3-10. Bootstrap I2C Mode Parameters Under the Bootstrap Tab



The following table lists description of each I^{2}C configurable parameters.

Table 3-6. I^{2}C Bootstrap Configuration Options

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
I2C Enable	I2C_EN	0 or 1	1	If this parameter is set as 1, the I^{2}C module is enabled in the MIV_ESS core.

.....continued

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
I2C Device Address	I2C_SLV_ADDR	0x0 – 0xFF	0x50	The unique I^{2}C device address to begin booting from.
Number of Address Bytes	I2C_MULTI_ADDR_BYTES	1 or 2	2	The number of bytes used to represent the I^{2}C address. 1: 1 Byte 2: 2 Bytes
Source Start Address: Upper Byte (Hex)	I2C_START_ADDR_UPPER	0x0 – 0xFF	0x0	Upper byte of the address in the I^{2}C memory from which the boot-code needs to be copied. Only valid when 2 byte addressing is used. .
Lower Byte (Hex)	I2C_START_ADDR_LOWER	0x0 – 0xFF	0x0	Lower byte of the address in the I^{2}C memory from which the boot-code needs to be copied.
I2C Clock Divisor	I2C_CLK_DIVISOR	0 – 255 (Decimal)	99	The Serial Clock (SCLK) prescaler used to generate the SCLK frequency from the System Clock, see Prescaler Register Description.

3.2.6 I^{2}C Bootstrap Ports

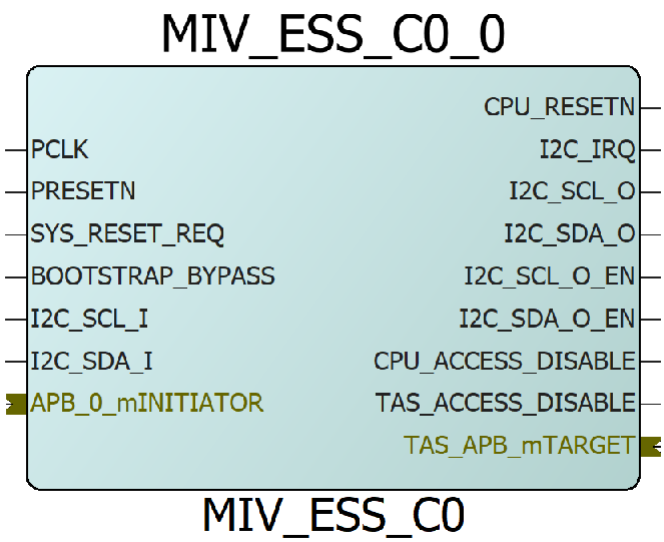
The following table lists the I^{2}C Bootstrap ports available on the MIV_ESS module.

Table 3-7. I^{2}C Port Signals

Ports	Width	Direction	Description
BOOTSTRAP_BYPASS	1	Input	The I^{2}C Module mode select: • When BOOTSTRAP_BYPASS = 0 on Reset, the I^{2}C module enters the Bootstrap mode • When BOOTSTRAP_BYPASS = 1, the I^{2}C module is always in the Peripheral mode
SCL_I	1	Input	I^{2}C Clock Line Input
SCL_O	1	Output	I^{2}C Clock Line Output
SCL_O_EN	1	Output	I^{2}C Clock Line Output Enable
SDA_I	1	Input	I^{2}C Data Line Input
SDA_O	1	Output	I^{2}C Data Line Output
SDA_O_EN	1	Output	I^{2}C Data Line Output Enable
I2C_IRQ	1	Output	I^{2}C Interrupt

The following figure shows the MIV_ESS SmartDesign, configured for Bootstrap using I^{2}C.

Figure 3-11. MIV_ESS Instance for Bootstrap Using I2C

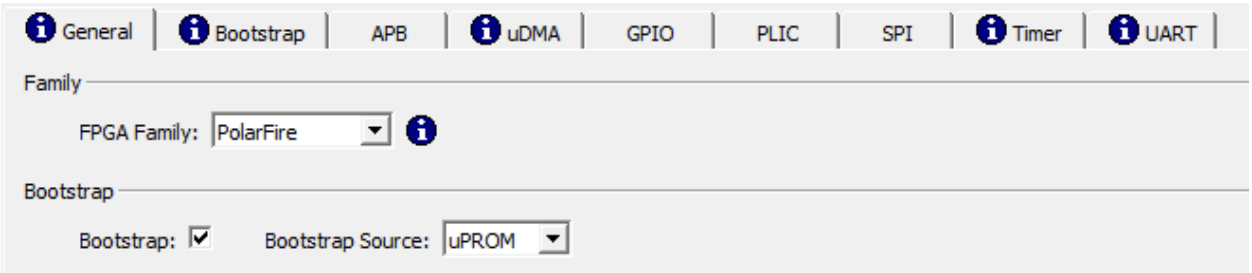


3.2.7 µPROM Bootstrap Parameters

The Bootstrap µPROM configuration is FPGA family specific. This module supports both PolarFire and RTG4 family devices. When instantiating the MIV_ESS core, the correct family must be selected.

The following figure shows how to enable the µPROM Bootstrap in the **General** tab.

Figure 3-12. µPROM Bootstrap General Setup



Select the correct FPGA family and µPROM as the Bootstrap Source, and navigate to the **Bootstrap** → **µPROM Configuration** for further configuration. The **µPROM Configuration** section enables you to configure the µPROM settings, as shown in the following figure.

Figure 3-13. µPROM Source Address Configuration

The following table lists description of each µPROM configurable parameters.

Table 3-8. µPROM Configuration Parameters

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
Source Start Address: Upper Byte (Hex)	uPROM_SRC_ADDR_UPPER	0x0 – 0xFF	0x0	Upper 8-bits of the memory address of the boot code within the µPROM device.

.....continued				
Configurator Parameter	**Parameter Name**	**Valid Values**	**Default Value**	**Description**
Lower Byte (Hex)	UPROM_SRC_ADDR_LOWER	0x0 – 0xFF	0x0	Lower 8-bits of the memory address of the boot code within the µPROM device.

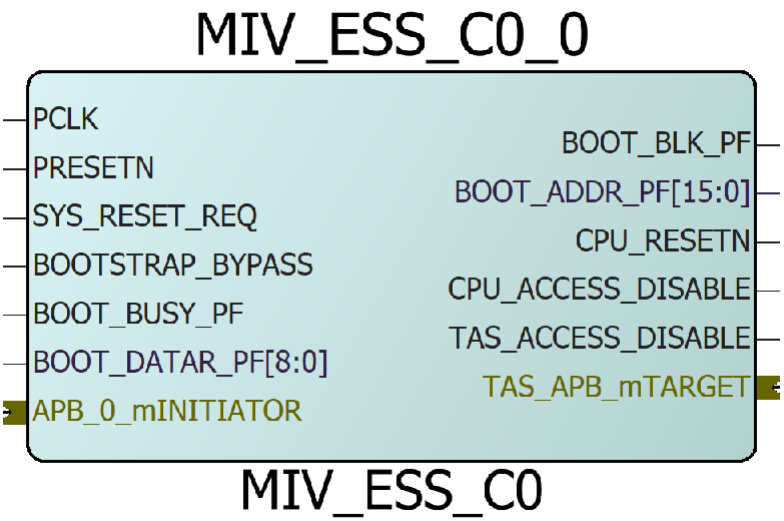
3.2.8 µPROM Bootstrap Ports

Depending on the selected FPGA family, respective ports will appear on the MIV_ESS instance.

PolarFire µPROM

The following figure shows Bootstrap µPROM ports as they appear on the MIV_ESS instance, if PolarFire is selected.

Figure 3-14. PolarFire µPROM Ports



The following table lists Bootstrap µPROM ports signals for the PolarFire family.

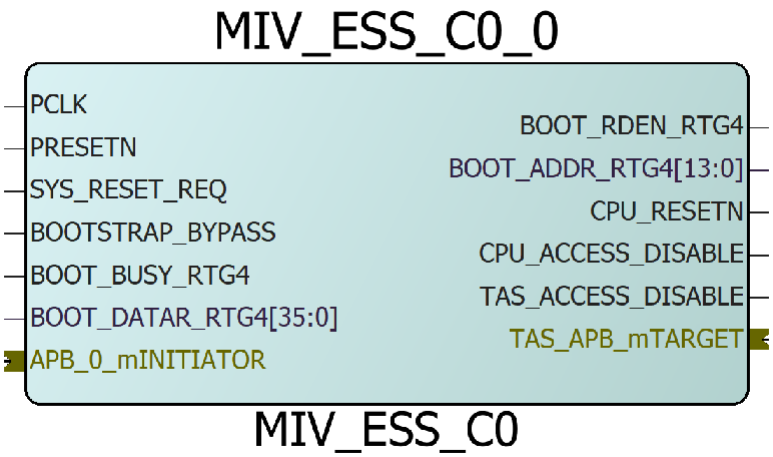
Table 3-9. PolarFire µPROM Port Signals

Ports	Width	Direction	Description
BOOT_BUSY_PF	1	Input	PolarFire µPROM busy
BOOT_DATAR_PF	[8:0]	Input	PolarFire µPROM read data
BOOT_BLK_PF	1	Output	PolarFire µPROM block select
BOOT_ADDR_PF	[15:0]	Output	PolarFire µPROM read address

RTG4 µPROM

The following figure shows the Bootstrap µPROM ports as they appear on the MIV_ESS instance, if RTG4 is selected.

Figure 3-15. RTG4 µPROM Ports



The following table lists the Bootstrap µPROM port signals for the RTG4 family.

Table 3-10. RTG4 µPROM Port Signals

Ports	Width	Direction	Description
BOOT_BUSY_RTG4	1	Input	RTG4 µPROM busy
BOOT_DATAR_RTG4	[35:0]	Input	RTG4 µPROM read data
BOOT_RDEN_RTG4	1	Output	RTG4 µPROM read enable
BOOT_ADDR_RTG4	[13:0]	Output	RTG4 µPROM read address

3.3 SPI Mode - Programming and Operation

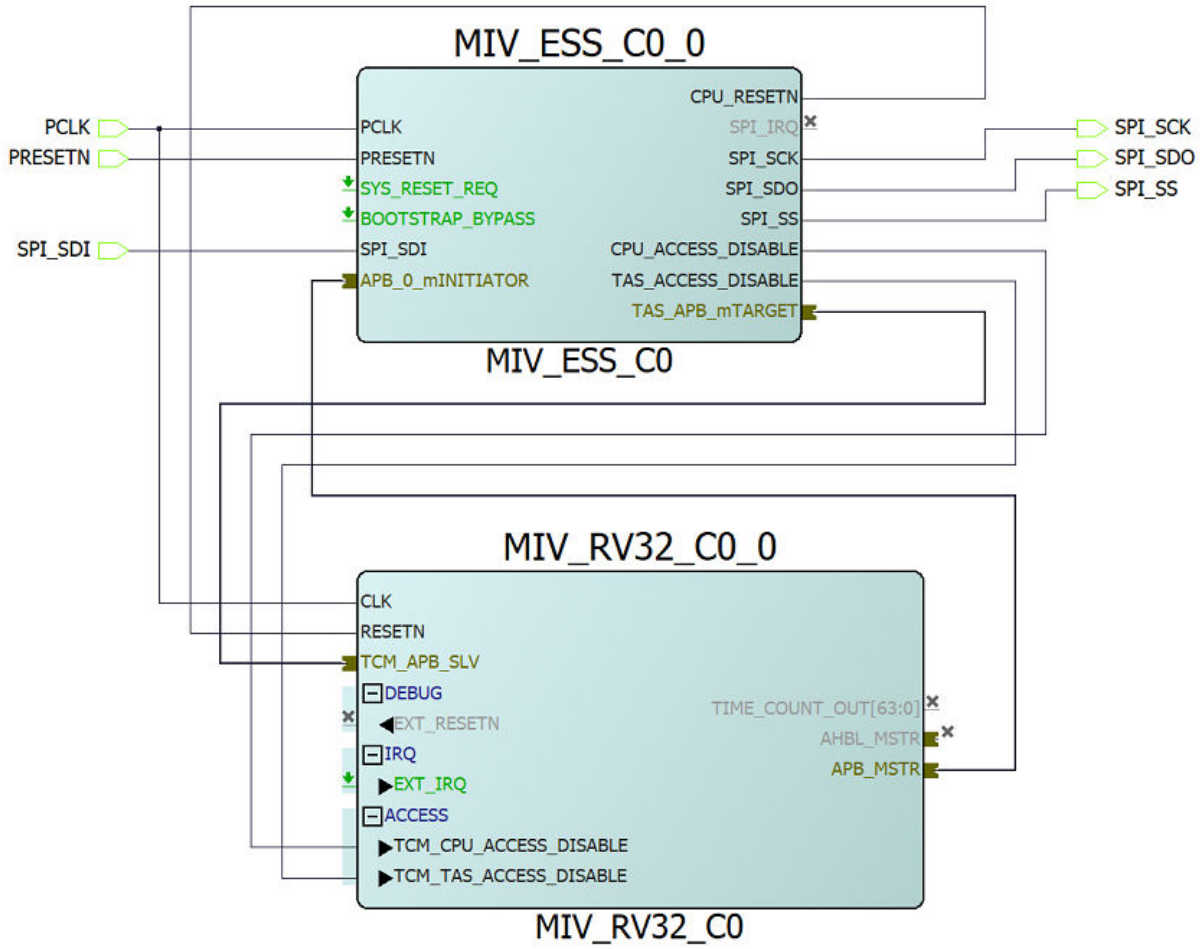
This section describes the programming and operation procedures of the SPI Bootstrap configuration.

3.3.1 How to Use the Bootstrap SPI

After the Bootstrap option is enabled in the **General** tab, it gets integrated into the MIV_ESS instance.

The Bootstrap can be connected to MIV_RV32 directly and the Bootstrap SPI pins can be promoted to the top level, so they can be interfaced with the external SPI memory device via constraints. The following figure shows an example of SmartDesign setup for Bootstrap SPI.

Figure 3-16. Bootstrap Setup in SPI Mode



The Bootstrap operation is a hardware function that does not require a software. It is completely configurable using the MIV_ESS GUI, as shown in the Interface section.

3.3.2 Operation

The SPI memory must be pre-programmed with the boot code to enable Bootstrap to copy boot code from an external SPI memory. For more information on pre-programming the SPI memory, see *MIV_ESS Design Guide*. The Bootstrap SPI operations are only responsible for copying pre-programmed boot code from the SPI Bootstrap Source.

3.3.2.1 SPI Control

The Bootstrap SPI state machine gets started after `PRESETN` or `SYS_RESET_REQ` is released, and the following operations are performed.

1. Hardware Reset Recovery: After releasing `PRESETN`, the state machine waits a period of `RST_RECOVERY_DURATION` PCLK cycles to ensure the completion of internal reset actions in the SPI Flash chip.
   - Micron - 40 ns
   - Spansion/Cypress - 200 ns
   - ISSI - 100 us
   - GigaDevice - 60 us
   - Macronix - 10 us
   - Adesto - 1 us

  - Winbound - NA
2. Check for Flash Busy: This polls the SPI Flash's `STATUS` Register Busy bit. This handles reset during an SPI program, erase, or write to certain registers, which takes time to complete.
3. Apply a Software Reset: Apply a software Reset, if the SPI Flash chip supports it, to clear any volatile registers which may have changed modes of operation such as addressing modes. The parameter `SW_RESET_TYPE` indicates, which reset type applies to the SPI Flash chip.
4. For Micron, ISSI, Winbound, Macronix, and GigaDevice a software Reset is done with a 66H 8-bit command, for SS_DESELECT_DURATION, followed by a 99H 8-bit command.
   - Micron SS_DESELECT_DURATION - 40 ns
   - ISSI - 7 ns (t_{CEH})
   - Winbond - 50 ns (t_{CSH})
   - Macronix - 30 ns
   - GigaDevice - 20 ns
5. For Adesto, a software reset is done by a 32-bit command with the code `f0_00_00_00`, while for Cypress/Spansion it is an 8-bit code `f0`.
6. Software Reset Recovery: After applying software Reset, the state machine waits a period of `RST_RECOVERT_DURATION` PCLK cycles to ensure that internal reset actions in the SPI Flash chip are completed.
7. Read data and pass to APB Writer: The Flash is now ready for reads. For Flash chips of 128 Mbit and above, 4 byte addressing is used via the 13th command, otherwise it uses three-byte addressing via the 13th command. This is setup via the `READ_4BYTE_ADDR` parameter.

An inner loop fetches 32 bits of data one bit at a time, after which it removes 'SPI_SS' for `SS_DESELECT_DURATION`. In addition, for Adesto this is 30 ns and for Spansion/Cypress it's 10 ns. The 32-bit 'rd_data' value is then passed to the APB Writer controller with the indication `rd_data_valid`.

An outer loop increments a word counter and its SPI address, repeating step 5 until DATA_WORD_CNT words are transferred to APB Writer, and completes the copy process by asserting `rd_all_done` to APB. It also indicates to the CKSUM_CTRL block that it can perform a data check.

During the transfer, the current SPI address is compared with CKSUM_SPI_ADDR, and the data at this address is latched. CKSUM_SPI_ADDR must reside at some location within the code being copied.

When the Bootstrap function completes, it indicates that the check is complete internally. It also indicates if there is an error. In the initial release, data checking is not provided and this block assigns `CKSUM_ERR` to 0, and `cksum_done` to 1.

3.3.2.2 Bootstrap Transaction

Following the assertion of `PRESETN`, the Bootstrap module initiates copying boot code from source SPI memory to TCM. The `CPU_RESETN` signal is LOW, holding `MIV_RV32` in Reset. The `CPU_ACCESS_DISABLE` is HIGH to block Hart access to TCM. The `TAS_ACCESS_DISABLE` is LOW to allow the Bootstrap module access TAS interface access into TCM.

The following steps describe how Bootstrap performs the operation.

1. The bytes are assembled into 32-bit words and written into MIV_RV32 TCM. This operation continues until the number of 32-bit words transferred matches the number specified by `DATA_WORD_CNT`.
2. After it is transferred, the Bootstrap module's APB Writer reads the first instruction back from the TCM and compare it to the first instruction received from the Bootstrap.
3. The Bootstrap module then waits for External Processor Reset Duration, then the `CPU_RESETN` is set to HIGH to release MIV_RV32 from reset.
4. The `CPU_ACCESS_DISABLE` is set LOW to allow Hart access to the TCM and TAS_ACCESS_DISABLE is set to HIGH to block TAS access to TCM.
5. The Bootstrap completes the operation.

3.4 I²C Mode – Programming and Operation

Only the Bootstrap configuration with I²C as the Bootstrap Source is covered in this programming and operation section.

3.4.1 How to use the Bootstrap I²C

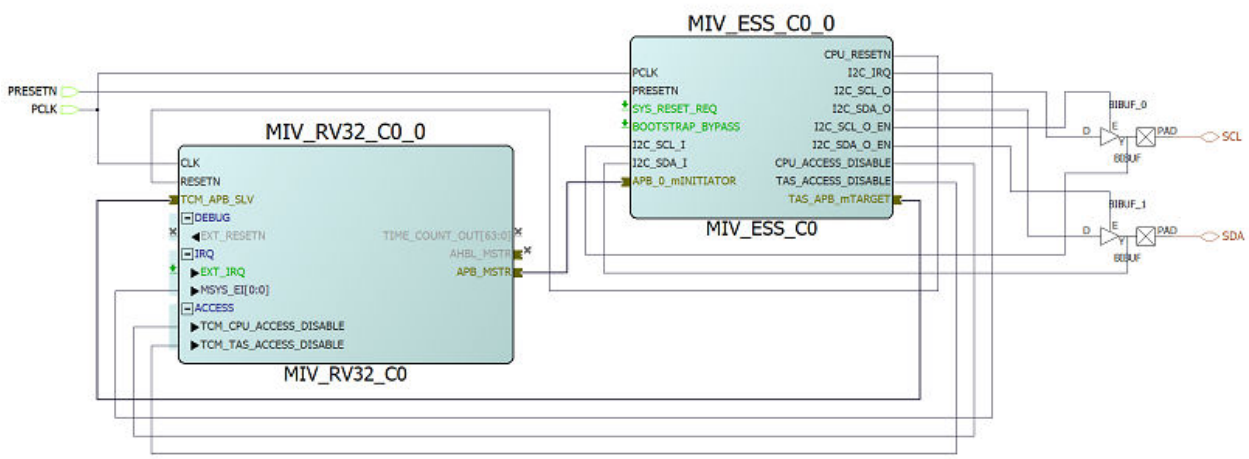
The Bootstrap operation is a hardware function that does not require a software. It is completely configurable via the MIV_ESS GUI, as shown in the Interface section.

3.4.2 Operation

The circuit shown in the following figure describes how the MIV_ESS instance can be connected to MIV_RV32 for booting from an I²C memory source. The MIV_ESS I2C communication pins must be connected with bidirectional buffer lines. For more details on how to connect the I²C lines, see Programming.

1. When the `BOOTSTRAP_BYPASS` pin is logic LOW, then on reset, the Bootstrap module begins the booting process.
2. The control signals, `CPU_ACCESS_DISABLE` and `TAS_ACCESS_DISABLE`, first halt the Hart of the MIV_RV32 and allow the TCM to be accessed via the TAS interface.
3. The bootstrap begins to read data byte-by-byte from the specified I²C device and location.
4. Each time a 32-bit word is assembled, it is written to the internal TCM of the MIV_RV32.
5. Repeat this until the number of 32-bit words transferred matches that was specified.
6. The Hart is resumed, TCM access is disabled, and the MIV_RV32 CPU is reset by the MIV_ESS via `CPU_RESETN`, which connects to the reset of the MIV_RV32.
7. The MIV_RV32 core executes instructions at its Reset Vector Address.

Figure 3-17. MIV_ESS Instance Connected to MIV_RV32 for I2C Booting



3.5 µPROM Mode – Programming and Operation

This section describes the Bootstrap operation with µPROM as the Bootstrap Source. The specific µPROM reading operation used is dependent on the selected FPGA device (PolarFire or RTG4).

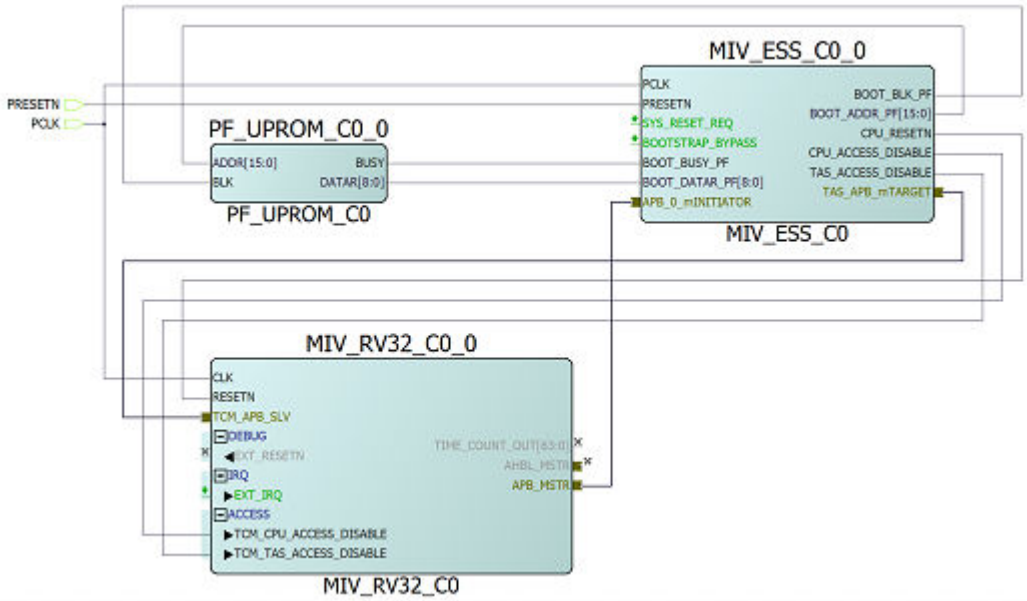
3.5.1 How to Use the Bootstrap µPROM

The Bootstrap operation is a hardware function that does not require any software. It is completely configurable via the MIV_ESS GUI, as shown in the Interface section. Ensure that the 'FPGA Family' parameter selected reflects the intended board (PolarFire or RTG4).

PolarFire µPROM Setup

The MIV_ESS must be connected to the MIV_RV32 and µPROM device, as shown in the following figure. See Details of Operation for operation details.

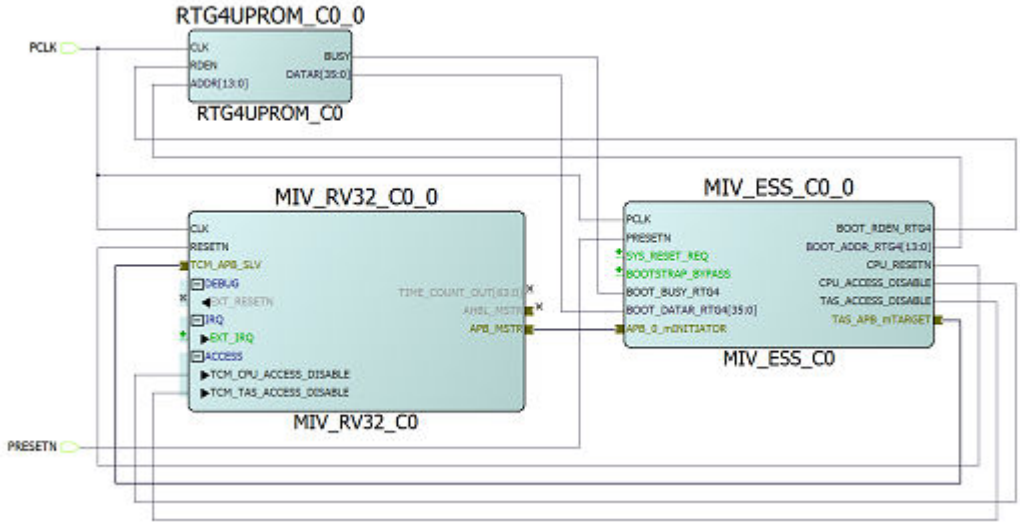
Figure 3-18. Bootstrap PolarFire µPROM Setup



RTG4 µPROM Setup

MIV_ESS must be connected to MIV_RV32 and µPROM device as shown in the following figure when using an RTG4 device. See Details of Operation for operation details.

Figure 3-19. Bootstrap RTG4 µPROM Setup



3.5.2 Details of Operation

PolarFire µPROM Setup

When booting from µPROM on a PolarFire device, the Bootstrap module operates as follows:

1. The Bootstrap initiates a read transfer by asserting the `BOOT_BLK_PF` signal.
2. The Bootstrap drives the start address of the read data onto the µPROM BOOT_ADDR_PF line.

3. The Bootstrap then receives a 9-bit word (MSB = ECC bit) from the µPROM BOOT_DATAR_PF line. The Bootstrap then increments the address pointer to read the next 9-bit word.
4. After the Bootstrap has received four 9-bit words, it assembles a 32-bit instruction (discarding the ECC bit on each 9-bit word) and passes the 32-bit instruction to the APB Writer (logic responsible for writing data to a MIV_RV32's TCM over the TAS interface)
5. After all boot code has been read from the µPROM, the Bootstrap de-asserts the `BOOT_ADDR_PF` signal to indicate the end of the read transfer.

RTG4 µPROM Setup

When booting from a µPROM on a RTG4 device, the Bootstrap module operates as follows:

1. The Bootstrap initiates a read transfer by asserting the `BOOT_RDEN_RTG4` signal.
2. The Bootstrap drives the start address of the read data onto the µPROM BOOT_ADDR_RTG4 line.
3. The Bootstrap then receives a 36-bit word (Bits [35:32] = ECC bits) from the µPROM BOOT_DATAR_RTG4 line.
4. The Bootstrap passes the 32-bit instruction (discarding the ECC bits) to the APB Writer and increments the address pointer to read the next instruction.
5. After all boot code is read from the µPROM, the Bootstrap de-asserts the `BOOT_BLK_PF` signal to indicate the end of the read transfer.

4. APB

This section provides information on the APB module used in the MIV_ESS.

4.1 Description

The APB module is a bus component that provides an advanced microcontroller bus architecture (AMBA®) advanced peripheral bus (APB) fabric to interconnect between an APB Initiator and up to 15 APB targets. The targets may be AMBA 2 or AMBA 3 compatible. Unlike AMBA 2 APB targets, AMBA 3 APB targets provide ready and error signals.

The APB has the following key features:

- Supports up to 15 APB targets (seven external and eight internal).
- Supports Initiator data bus width of 32 bits
- Supports Initiator address bus width of 32 bits

For more information about this IP, see *CoreAPB3 v4.2 Handbook* in Libero Catalog.

4.2 Interface

This section describes the configuration parameters and interface ports of the APB module.

4.2.1 Configuration Parameters

The following table lists the parameters (Verilog) for configuring the RTL code of the core.

Table 4-1. APB Configuration Parameters

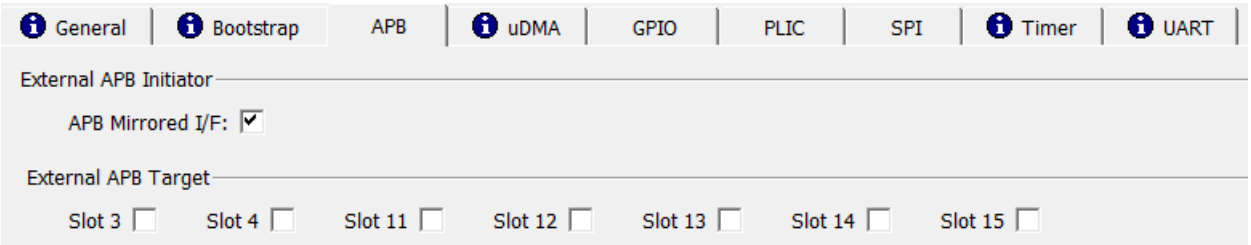
Configurator Name	Parameter Name	Valid Values	Default	Description
APB Mirror I/F	APB_INITIATOR_0_MIRROR	0 or 1	1	APB External Interface option 0: APB Target Interface 1: APB Mirrored Initiator Interface
PLIC	PLIC_EN	0 or 1	1	0: Disables PLIC on target 0 1: Enables PLIC on target 0
UART	UART_EN	0 or 1	1	0: Disables UART on target 1 1: Enables UART on target 1
TIMER	SYS_TIMER_EN	0 or 1	1	0: Disables TIMER on target 2 1: Enables TIMER on target 2
Slot 3	APBSLOT3ENABLE	0 or 1	1	0: Disables target 3 1: Enables target 3
Slot 4	APBSLOT4ENABLE	0 or 1	1	0: Disables target 4 1: Enables target 4
SPI	SPI_EN	0 or 1	1	0: Disables SPI on target 5 1: Enables SPI on target 5

.....continued

Configurator Name	Parameter Name	Valid Values	Default	Description
GPIO	GPIO_EN	0 or 1	1	0: Disables GPIO on target 6 1: Enables GPIO on target 6
uDMA	uDMA_EN	0 or 1	1	0: Disables uDMA on target 8 1: Enables uDMA on target 8
Watchdog	WDT_EN	0 or 1	1	0: Disables Watchdog on target 9 1: Enables Watchdog on target 9
I^2C	I2C_EN	0 or 1	1	0: Disables I^2C on target 10 1: Enables I^2C on target 10
Slot 11	APBSLOT11ENABLE	0 or 1	1	0: Disables target 11 1: Enables target 11
Slot 12	APBSLOT12ENABLE	0 or 1	1	0: Disables target 12 1: Enables target 12
Slot 13	APBSLOT13ENABLE	0 or 1	1	0: Disables target 13 1: Enables target 13
Slot 14	APBSLOT14ENABLE	0 or 1	1	0: Disables target 14 1: Enables target 14
Slot 15	APBSLOT15ENABLE	0 or 1	1	0: Disables target 15 1: Enables target 15

The following figure shows the APB configuration window.

Figure 4-1. APB Configuration Window



4.2.2 I/O Signals

The following table lists the APB I/O signal description.

Table 4-2. APB3 Ports

Port Name	Width	Direction	Description
PRESETN	1	Input	APB Reset, active-low asynchronous reset.
PCLK	1	Input	APB clock signal.
PSEL	1	Input	APB select from Initiator.

.....continued

Port Name	Width	Direction	Description
PENABLE	1	Input	APB enable from Initiator.
PWRITE	1	Input	APB write indication from Initiator.
PADDR	32	Input	APB address bus from Initiator.
PWDATA	32	Input	APB write data from Initiator. Depending on the data bus width configuration, it is possible that only the lower 8 or 16 bits of this bus are in use.
PRDATA	32	Output	APB read data output to Initiator. Depending on the data bus width configuration, it's possible that only the lower 8 or 16 bits of this bus are in use.
PREADY	1	Output	APB ready indication output to Initiator.
PSLVERR	1	Output	APB target error indication to Initiator.
PENABLES	1	Output	APB enable to all targets.
PWRITES	1	Output	APB write indication to all targets.
PADDRS	32	Output	APB address bus to all targets.
PWDATAS	32	Output	APB write data to all targets.
PSELS[n]	1	Output	APB select signal to targets (n = 3, 4, 11, 12, 13, 14, or 15).
PRDATAS[n]	32	Input	APB read data from targets (n = 3, 4, 11, 12, 13, 14, or 15).
PREADYS[n]	1	Input	APB ready signal from targets (n = 3, 4, 11, 12, 13, 14, or 15).
PSLVERRS[n]	1	Input	APB error indication signal from targets (n = 3, 4, 11, 12, 13, 14, or 15).

4.3 Programming

This section describes the APB memory map.

Memory Map

Table 4-3. MIV_ESS APB Address Allocation Map

Target Slot	28-bit Initiator Address	Resource
0	0x0000000 – 0x0FFFFFF	PLIC
1	0x1000000 – 0x1FFFFFF	UART
2	0x2000000 – 0x2FFFFFF	Timer
3	0x3000000 – 0x3FFFFFF	External Target 3
4	0x4000000 – 0x4FFFFFF	External Target 4
5	0x5000000 – 0x5FFFFFF	GPIO

.....continued

Target Slot	28-bit Initiator Address	Resource
6	0x6000000 – 0x6FFFFFF	SPI
7	0x7000000 – 0x7FFFFFF	RESERVED
8	0x8000000 – 0x8FFFFFF	uDMA
9	0x9000000 – 0x9FFFFFF	Watchdog
10	0xA000000 – 0xAFFFFFF	I^2C
11	0xB000000 – 0xBFFFFFF	External Target 11
12	0xC000000 – 0xCFFFFFF	External Target 12
13	0xD000000 – 0xDFFFFFF	External Target 13
14	0xE000000 – 0xEFFFFFF	External Target 14
15	0xF000000 – 0xFFFFFFF	External Target 15

5. µDMA

This section provides information on the Micro Direct Memory Access (µDMA) module used in the MIV_ESS core.

5.1 Description

The µDMA module allows peripherals with AHB interfaces to transfer data independent of the MIV_RV32 processor. This includes the capability of a peripheral to write to the MIV_RV32's internal TCM via the TAS (TCM APB Slave) interface. The term µDMA is used to describe a basic DMA engine/feature set.

The µDMA module provides an APB target interface for interfacing with MIV_RV32, an AHB-Lite (AHBL) source (read) Initiator interface for reading from a source memory, an AHBL destination (write) Initiator interface for writing to a destination memory, and a TAS destination (write) Initiator interface for writing to the MIV_RV32's TCM. The µDMA can operate in the following two possible transfer configurations.

- **AHBL Read –> AHBL Write**: In this configuration, the µDMA reads data from the source memory over an AHBL (Mirrored Main/Initiator) read interface and writes data to the destination memory over an AHBL (Mirrored Main/Initiator) write interface.
- **AHBL Read –> TAS Write**: In this configuration, the µDMA reads data from the source memory over an AHBL (Mirrored Main/Initiator) read interface and writes data to the destination memory over the TAS (Mirrored Main/Initiator) write interface.

The CPU controls µDMA over the APB target interface. It prepares the µDMA for operation by writing a source start address, destination start address, and the block size of the transfer data to the respective µDMA configuration registers.

The CPU then enables the µDMA to begin a transfer without any additional intervention from the CPU. There is a configurable interrupt that allows the detection of errors and the completion of transfers. If this interrupt is configured to trigger at the end of a successful transfer, this can be detected by connecting the interrupt to an interrupt pin on the MIV_RV32 core.

5.2 Interface

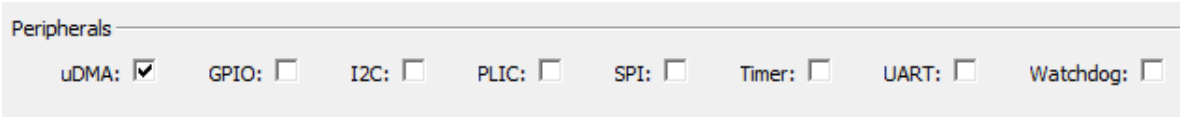
This section describes the configuration parameters and interface ports of the µDMA module.

Configuration Parameters

The µDMA has configurable parameters under two tabs in the MIV_ESS, as shown in the following figure.

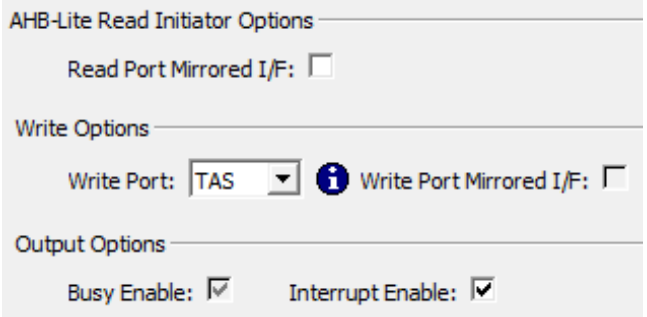
Under the **General > Peripherals**, select the µDMA (Enable) check box to enable the µDMA module in the MIV_ESS core.

Figure 5-1. µDMA Enable



If the µDMA is enabled, then it can be configured in the µDMA tab.

Figure 5-2. uDMA Configuration



The following table lists the µDMA parameters.

Table 5-1. µDMA Parameters

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
uDMA (Enable)	uDMA_EN	0 or 1	1	If checked (1), enables the µDMA in MIV_ESS.
Read Port Mirrored I/F	READ_MIRROR	0 or 1	0	If checked (1), enables the Read Port mirror.
Write Port	WRITE_PORT	0 or 1	1	Allows you to specify AHBL (1) or TAS (0) as the Write Port.
Write Port Mirrored I/F	WRITE_MIRROR	0 or 1	0	If checked (1), enables the Write Port mirror.
Busy Enable	BUSY_SIGNAL	0 or 1	1	If checked (1), brings the busy signal to the top-level of the module.
Interrupt Enable	IRQ_EN_SIGNAL	0 or 1	1	If checked (1), enables and brings the µDMA interrupt signal to the top-level of the module.

Ports

The following table lists µDMA ports available on the MIV_ESS.

Table 5-2. µDMA Port Signals

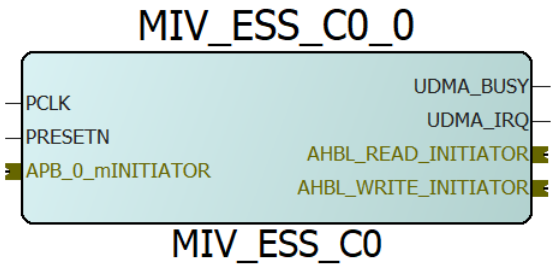
Ports	Width	Direction	Description
uDMA_BUSY	1	Output	Indicates if the µDMA is busy transferring data, currently.
uDMA_IRQ	1	Output	Interrupt, which indicates that the data transfer is completed.
AHBL_READ_INITIATOR	32-bit address and data bus	Port contains both Input and Output	This AHBL Initiator Port facilitates reading data from peripherals that have an AHBL Target port.
AHBL_WRITE_INITIATOR	32-bit address and data bus	Port contains both Input and Output	This AHBL Initiator Port facilitates writing data to peripherals that have an AHBL Target port.

.....continued			
Ports	**Width**	**Direction**	**Description**
TAS_MUX_APB_M_TARGET	32-bit address and data bus	Port contains both Input and Output	This APB Initiator Port facilitates writing data to the TCM of the MIV_RV32 via the TAS port.

The µDMA is controlled over the APB target interface called APB_TARGET, or APB_mINITIATOR if APB Initiator mirroring is enabled in the MIV_ESS GUI.

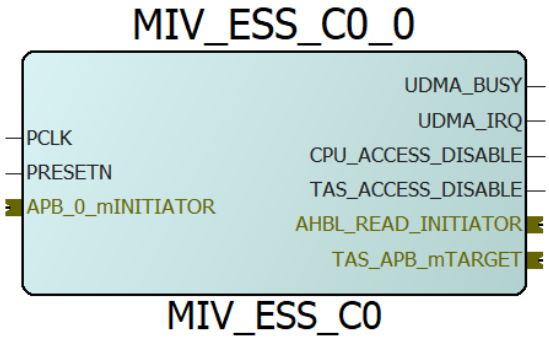
The µDMA inputs and outputs, as seen on the MIV_ESS SmartDesign instance, for the AHBL Read to AHBL Write configuration is shown in the following figure.

Figure 5-3. AHBL Read to AHBL Write



The following figure shows the SmartDesign instance for the AHBL Read to TAS Write configuration.

Figure 5-4. AHBL Read to TAS Write



5.3 Programming

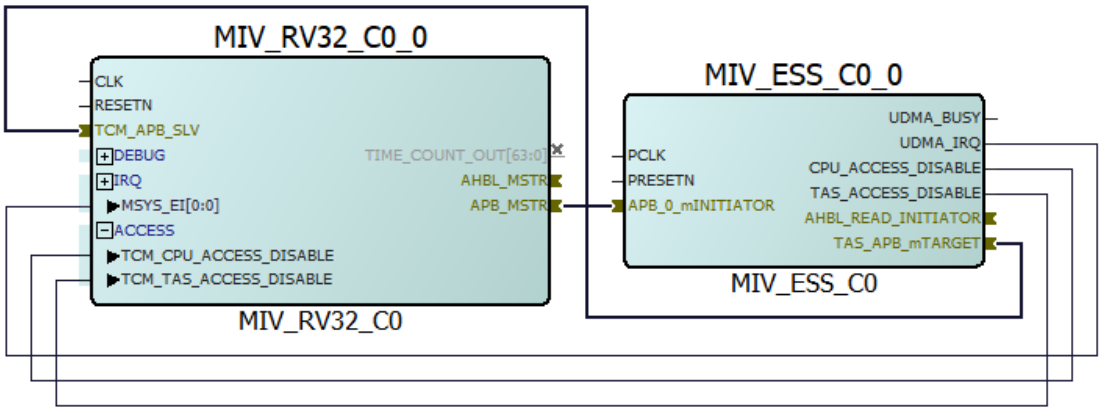
This section describes the programmer's model of the uDMA module.

How to use the µDMA

The software is required to set up a data transfer between two peripherals and handle the interrupt that can be asserted at the end of the transfer. The device driver is available from the GitHub page: https://github.com/Mi-V-Soft-RISC-V/platform.

When operating the µDMA in the AHBL Read to TAS Write configuration, the circuit must be connected to the MIV_RV32, as shown in the following figure. The wire connecting uDMA_IRQ to MIV_RV32 interrupt pin is optional. The `CPU_ACCESS_DISABLE` and `TAS_ACCESS_DISABLE` are necessary control signals to halt the Hart of the MIV_RV32 core and allow the TCM to be accessed via the TAS interface.

Figure 5-5. AHBL Read to APB Write to TCM with MIV_RV32 Connections



µDMA Memory Map

The µDMA module's register addresses are determined with respect to the MIV_ESS base address. The MIV_ESS base address [MIV_ESS_BASE] is configurable, see *Programming*. The µDMA module base address is [MIV_ESS_BASE + 0x800_0000]. The address offset of each µDMA register is given in the following table.

```
Register address = MIV_ESS_BASE + 0x800_0000 + Register Address Offset
```

Table 5-3. µDMA Module Register Map

Register Name	Address Offset	Read/Write	Reset Value	Description
CONTROL	0x00_0000	R/W	0	Control Start/Reset Register
IRQ CONFIG	0x00_0004	R/W	0	Control IRQ configuration Register
STATUS	0x00_0008	R	0	Transfer STATUS Register
SOURCE ADDR	0x00_000C	R/W	0	Source Memory Start Address Register
DESTINATION ADDR	0x00_0010	R/W	0	Destination Memory Start Address Register
TRANSFER SIZE	0x00_0014	R/W	0	Data Transfer Size Register

Each register in the preceded table is described in the following tables.

Control Register

The Control Register determines when a µDMA transfer is started and reset.

Table 5-4. CONTROL

Bit Number	Name	R/W	Reset Value	Description
1	uDMA_reset	R/W	0	Reset Transfer. When set, the µDMA transfer is reset.
0	uDMA_start	R/W	0	Start transfer. When set, the µDMA transfer is started.

Control IRQ Configuration Register

The IRQ Configuration Register determines the behavior of the µDMA interrupt.

Table 5-5. IRQ CONFIG

Bit Number	Name	R/W	Reset Value	Description
1	Reserved	R/W	0	Reserved
0	IRQ_cfg	R/W	0	Configures the µDMA interrupt behaviour. When set, the IRQ is asserted when an error occurs during a µDMA transfer or on the completion of a µDMA transfer. When clear, the IRQ is only asserted when an error occurs during a µDMA transfer.

Transfer STATUS Register
The Transfer STATUS Register contains two read-only bits, which indicate the status of the current µDMA transfer.

Table 5-6. STATUS

Bit Number	Name	R/W	Reset Value	Description
1	Error	R	0	When set, it indicates that the last µDMA transfer caused an error.
0	Busy	R	0	When set, it indicates that a µDMA transfer is in progress. When clear, indicates that a µDMA transfer is completed, cancelled, or not yet started.

Source Memory Start Address Register
The SRC ADDR register specifies the start address of the source memory from where the µDMA will read the data to be copied to the destination memory.

Table 5-7. SOURCE ADDR

Bit Number	Name	R/W	Reset Value	Description
31:0	SOURCE ADDR	R/W	0	Source start address

Destination Memory Start Address Register
The DESTINATION ADDR register specifies the start address of the destination memory to which the data will be copied by the µDMA.

Table 5-8. DESTINATION ADDR

Bit Number	Name	R/W	Reset Value	Description
31:0	DESTINATION ADDR	R/W	0	Destination start address

Data Transfer Size Register
The Data Transfer Size Register specifies the number of 32-bit words to be transferred from source to destination memory.

Table 5-9. TRANSFER SIZE

Bit Number	Name	R/W	Reset Value	Description
31:0	TRANSFER SIZE	R/W	0	Number of 32-bit words to transfer.

6. GPIO

This section provides information on the GPIO module used in the MIV_ESS core.

6.1 Description

The general purpose inputs output (GPIO) module provides an APB register-based interface to up to 32 general purpose inputs and 32 general purpose outputs. The input logic contains a simple three-stage synchronization circuit, and the output is set synchronously. Each bit can be set to either fixed configuration or register-based configuration via top-level parameters, including input type, interrupt type/enable, and output enable.

The GPIO has the following key features:

- AMBA 2 APB support, forward compatibility with AMBA 3 APB
- 8-, 16-, or 32-bit APB data width
- 1 to 32 bits of I/O, for all APB-width configurations
- Fixed or configurable interrupt generation
  - Negative edge
  - Positive edge
  - Both edges
  - Level High
  - Level Low
- Parameter-configurable for single-interrupt signal or up to 32-bit-wide interrupt bus.
- Fixed or configurable I/O type (input, output, or both).
- Configurable output enable (internal or external implementation).

For more information about this IP, see *CoreGPIO v3.2 Handbook* in the Libero Catalog.

6.2 Interface

This section describes the configuration parameters and interface pots of the GPIO module.

6.2.1 Configuration Parameters

The following table lists the parameters (Verilog) for configuring the RTL code of the core.

Table 6-1. GPIO Parameters and Generics Descriptions

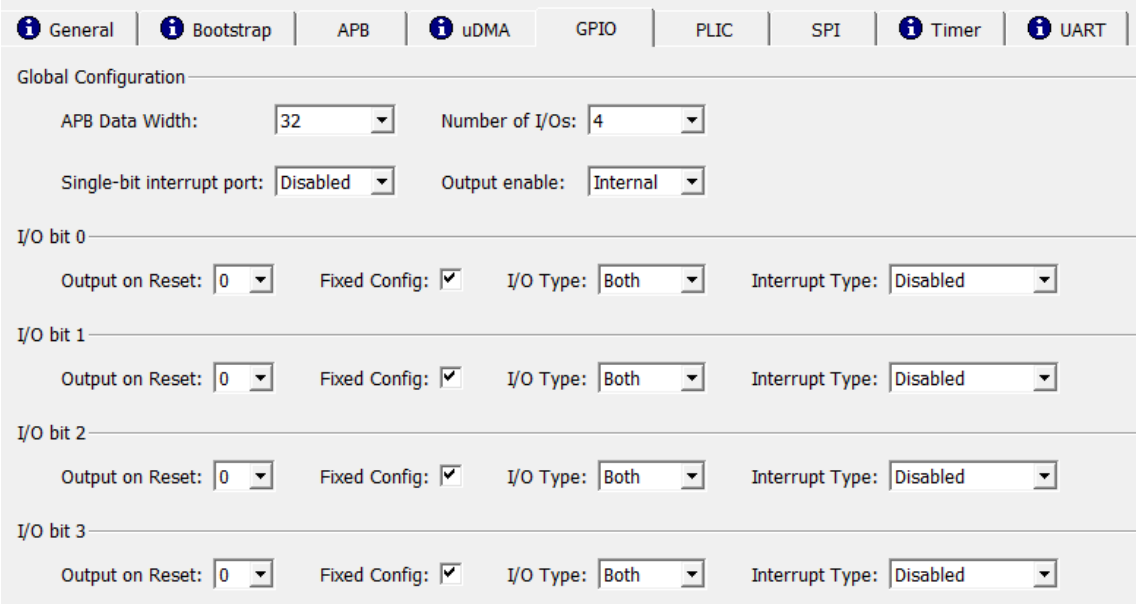
Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
GPIO	GPIO_EN	0 or 1	1	GPIO Enable 0: Disabled 1: Enabled
APB Data Width	APB_WIDTH	8, 16, 32	32	APB data width
Number of I/Os	IO_NUM	1–32	32	Number of GPIOs
Output Enable	OE_TYPE	0 or 1	0	If 0, output buffering is implemented outside GPIO. The user is responsible for instantiating tri-state buffers outside of the core. If 1, output buffering (if enabled) is implemented inside the core. When GPIO_OE[i] is 0, GPIO_OUT is high impedance (Z).

.....continued

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
Fixed Config	FIXED_CONFIG_x	0 or 1	0	If 0, configuration for bit x (0-31) is set via APB-accessible register CONFIG_x (see the Register Map section). If 1, configuration for bit x (0-31) is set via IO_INT_TYPE_x and IO_TYPE_x.
Interrupt Type	IO_INT_TYPE_x	0-5	0	Interrupt types selected according to the following scheme: 0 – Level High 1 – Level Low 2 – Edge Positive 3 – Edge Negative 4 – Edge Both 7 – Disabled **Note:** Selecting one type will synthesize out logic for other types. For example, Level High will remove AND/OR gates for edge detect.
I/O Type	IO_TYPE_x	0-2	0	If 0, bit x is of type input-only. Output logic will be synthesized out. If 1, bit x is of type output only. Input logic will be synthesized out. If 2, bit x is of type input and output (both).
Output on Reset	IO_VAL_x	0 or 1	0	Sets the output at reset for GPIO bit x.
Single-bit interrupt port	INT_BUS	0 or 1	0	If 0, the GPIO_INT_OR output is fixed at 0 (unused). If 1, the GPIO_INT_OR output is set when any of the GPIO_INT signals are set (OR operation).

The following figure shows the GPIO configuration window and cross-references to the corresponding top-level parameters.

Figure 6-1. GPIO Configuration Window



6.2.2 I/O Signals

The following table lists the GPIO I/O signal description.

Table 6-2. GPIO I/O Signal Description

Port Name	Width	Direction	Description
APB Signals			
PCLK	1	Input	APB system clock – Reference clock for all internal logic.
PRESETN	1	Input	APB active-low asynchronous reset
GPIO Signals			
GPIO_IN	IO_NUM	Input	GPIO input
GPIO_OUT	IO_NUM	Output	GPIO output
GPIO_OE	IO_NUM	Output	GPIO output enable
GPIO_INT	IO_NUM	Output	Interrupt mask; can be connected directly to processor.
GPIO_INT_OR	1	Output	Bitwise OR version (single wire) of the interrupt mask values provided on INT[(IO_NUM-1) : 0]

6.3 Programming

6.3.1 Register Map

The following tables describe GPIO register map.

Table 6-3. GPIO Register Address Map (APB_WIDTH = 8)

PADDR[7:0]	R/W	Reset Value	Description
0x00-0x7C (0x00, 0x04, 0x08, …, 0x7C)	R/W	0x00	8-bit configuration registers for all 32 bits; One register per bit.
0x80	W	0x00	Interrupt clear register 1 (bits 7:0)
0x84	W	0x00	Interrupt clear register 2 (bits 15:8)
0x88	W	0x00	Interrupt clear register 3 (bits 23:16)
0x8C	W	0x00	Interrupt clear register 4 (bits 31:24)
0x90	R	0x00	Input register 1 (bits 7:0)
0x94	R	0x00	Input register 2 (bits 15:8)
0x98	R	0x00	Input register 3 (bits 23:16)
0x9C	R	0x00	Input register 4 (bits 31:24)
0xA0	R/W	0x00	Output register 1 (bits 7:0)
0xA4	R/W	0x00	Output register 2 (bits 15:8)
0xA8	R/W	0x00	Output register 3 (bits 23:16)
0xAC	R/W	0x00	Output register 4 (bits 31:24)

Notes:
1. Values shown in hexadecimal format; type designations: R = read-only; R/W = read/write.
2. Lower 2 bits of PADDR are unconnected inside GPIO.

Table 6-4. GPIO Register Address Map (APB_WIDTH = 16)

PADDR[7:0]	R/W	Reset Value	Brief Description
0x00-0x7C (0x00, 0x04, 0x08, …, 0x7C)	R/W	0x00	8-bit configuration registers for all 32 bits; One register per bit.
0x80	W	0x00	Interrupt clear register 1 (bits 15:0)
0x84	W	0x00	Interrupt clear register 2 (1bits 31:16)
0x90	R	0x00	Input register 1 (bits 15:0)
0x94	R	0x00	Input register 2 (bits 31:16)
0xA0	R/W	0x00	Output register 1 (bits 15:0)

Notes:
1. Values shown in hexadecimal format; type designations: R = read-only; R/W = read/write.
2. Lower 2 bits of PADDR are unconnected inside GPIO.

Table 6-5. GPIO Register Address Map (APB_WIDTH = 32)

PADDR[7:0]	R/W	Reset Value	Brief Description
0x00-0x7C	—	—	—
(0x00, 0x04, 0x08, …, 0x7C)	R/W	0x00	(0x00, 0x04, 0x08, …, 0x7C)
0x80	W	0x00	—
0x90	R	0x00	—
0xA0	R/W	0x00	—

Notes:
1. Values shown in hexadecimal format; type designations: R = read-only; R/W = read/write.
2. Lower 2 bits of PADDR are unconnected inside GPIO.

6.3.2 Configuration Registers

GPIO has up to 32 8-bit configuration registers, depending on the IO_NUM parameter. The following table lists operations of the GPIO Configuration register.

Table 6-6. Per-bit Configuration Register

Bits	Name	Function
7:5	INTTYPE	Sets the interrupt type for this particular bit: 000 – Level High 001 – Level Low 010 – Edge Positive 011 – Edge Negative 100 – Edge Both 101 to 111 – Invalid
4	Reserved	UNUSED
3	INTENABLE	Interrupt enable for this particular bit 1 – Enable interrupt generation 0 – Disable interrupt generation
2	OUTBUFF	Sets the output enable for this particular bit, whether through the GPIO_OE signal or implemented internally (see parameter "OE_TYPE"). 1 – Enables output 0 – Disables output
1	INREG	Input register enable 1 – Enables input register for this particular bit 0 – Disables input register for this particular bit

.....continued		
Bits	Name	Function
0	OUTREG	Output register enable 1 – Enables output functionality for this particular bit 0 – Disables output functionality for this particular bit

6.3.3 Interrupt Registers

These are per-bit interrupt clear registers. Writing a one to any bit clears the interrupt bit register of the corresponding GPIO bit.

- In 32-bit mode, all 32 interrupt bits are in a single 32-bit register located at address 0x80.
- In 16-bit mode, 32 interrupt bits are split into two 16-bit registers located at addresses 0x80 and 0x84.
- In 8-bit mode, 32 interrupt bits are split into four 8-bit registers located at addresses 0x80, 0x84, 0x88, and 0x8C.

6.3.4 Input Registers

These are read-only for input configured ports. Disabling a bit in this register with the CONFIG_X[1] (INREG) bit forces the bit to 0 through a MUX, while storing the incoming current value in the register.

- In 32-bit mode, all 32 input bits are in a single 32-bit register located at address 0x90.
- In 16-bit mode, 32 input bits are split into two 16-bit registers located at addresses 0x90 and 0x94.
- In 8-bit mode, 32 input bits are split into four 8-bit registers located at addresses 0x90, 0x94, 0x98, and 0x9C.

6.3.5 Output Registers

The output registers are writeable/readable for output configured ports, and are logical "don't cares" for input configured ports. Disabling a bit in this register with the CONFIG_X[0] (OUTREG) bit forces the bit to 0 through a MUX, while keeping the previously written value in the output register.

- In 32-bit mode, all 32 output bits are in a single 32-bit register located at address 0xA0.
- In 16-bit mode, 32 output bits are split into two 16-bit registers located at addresses 0xA0 and 0xA4.
- In 8-bit mode, 32 output bits are split into four 8-bit registers located at addresses 0xA0, 0xA4, 0xA8, and 0xAC.

7. I²C

This section provides information on the I²C module used in the MIV_ESS core.

7.1 Description

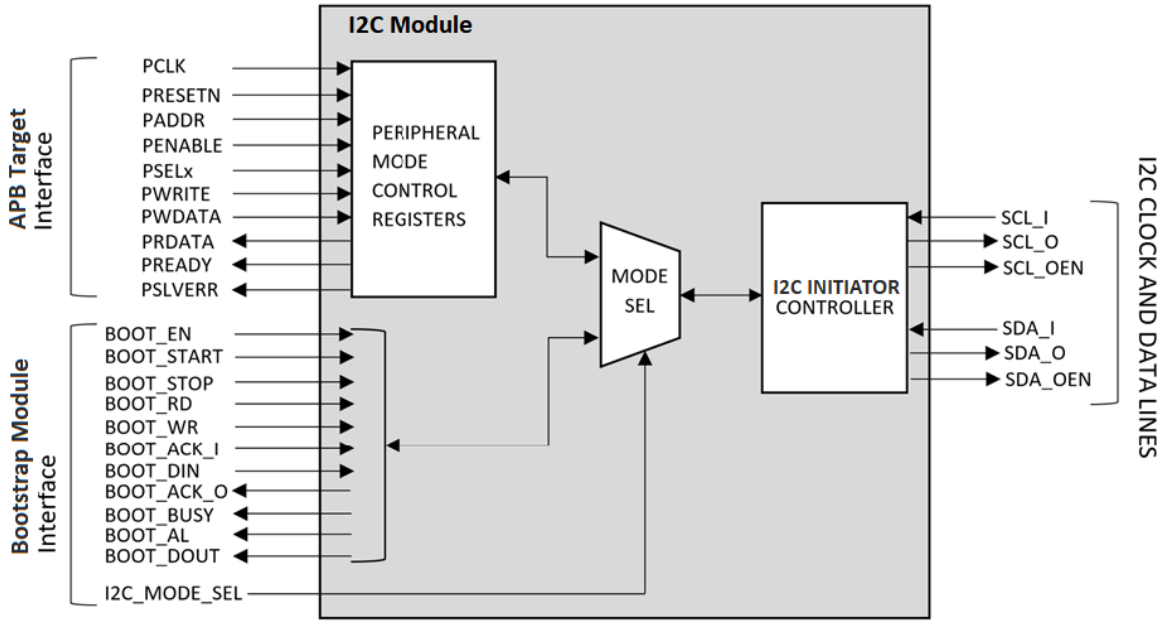
The I²C module is an optional integral component of the MIV_ESS core and is based on an open-source I²C core designed by Richard Herveille. The purpose of the I²C is to provide a minimal APB-driven I²C interface, supporting Initiator read and write accesses to peripheral I²C devices. The I²C module can also be configured by the Bootstrap controller to copy I²C boot memory to the TCM of the MIV_RV32 soft processor.

The I²C module has the following features:

- AMBA APB 3.0 Target interface.
- Compatibility with the Phillips I²C bus standard.
- Bootstrap and peripheral I²C Initiator modes.
- Support for I²C "Normal" 100 kbps, "Fast" 400 kbps and "Fast-Plus" 1 Mbps transmission speeds.
- Support for 7-, 8-, 10-, and 18-bit addressed I²C devices.
- Interrupt driven and byte-by-byte data transfers.
- I²C (Repeated) Start/Stop signal generation/detection.
- Clock stretching and wait-state generation.
- Multi-Initiator operation.
- I²C bus busy detection.

A block diagram of the I²C module is provided in the following figure, illustrating the use of a MUX to control whether the I²C module is acting as a peripheral device to the MIV_RV32 (Peripheral Mode) or configured to read data from I²C boot memory (Bootstrap Mode).

Figure 7-1. The Block-level Architecture of the I²C Module



7.2 Interface

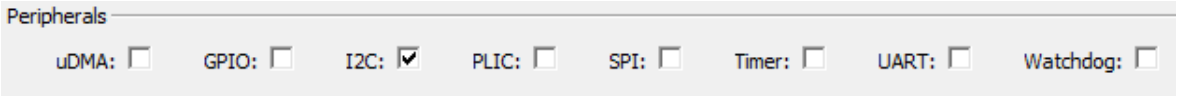
This section describes I²C configuration parameters and ports.

Configuration Parameters

The I²C module has configurable parameters under two tabs in MIV_ESS, as shown in the following figure.

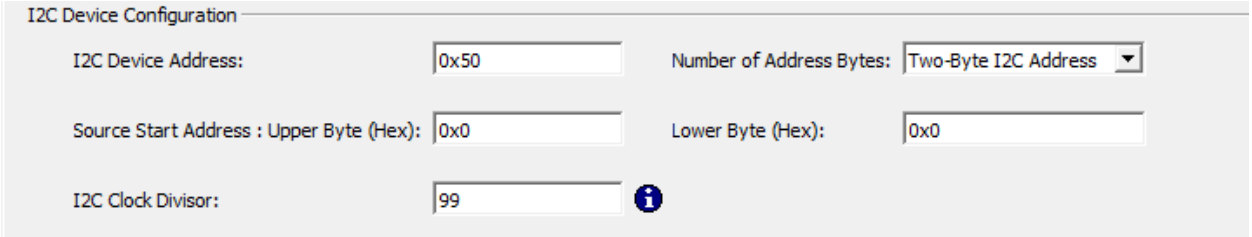
Under the **General –> Peripherals** tab, select the **I²C** check box to enable the I²C module in the MIV_ESS.

Figure 7-2. I²C Peripheral Enable



If the I²C module is intended for use in the Peripheral mode, no further configuration in the MIV_ESS GUI is required. However, if the I²C module is intended for use in conjunction with the Bootstrap, the I²C memory device used for booting purposes can be configured under the **Bootstrap** tab in the **I2C Device Configuration** section.

Figure 7-3. I2C Device Configuration



The following table describes each parameter in the I²C module.

Table 7-1. I²C Parameters

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
I2C Enable	I2C_EN	0 or 1	1	If this parameter is 1, the I²C module is enabled in the MIV_ESS.
I2C Device Address	I2C_SLV_ADDR	0x0 – 0xFF	0x50	The unique I²C device address to begin booting from.
Number of Address Bytes	I2C_MULTI_ADDR_BYTES	1 or 2	2	The number of bytes used to represent the I²C Address. 1: One-Byte 2: Two-Byte
Source Start Address: Upper Byte (Hex)	I2C_START_ADDR_UPPER	0x0 – 0xFFFF	0x0	The upper four hex digits of the address in the I²C device where reading/writing must begin.
Lower Byte (Hex)	I2C_START_ADDR_LOWER	0x0 – 0xFFFF	0x0	Specifies the lower four hex digits of the address in the I²C device where reading/writing must begin.
I2C Clock Divisor	I2C_CLK_DIVISOR	0 – 255	99	The Serial Clock (SCLK) prescaler is used to generate the SCLK frequency from the System Clock, see Prescaler Register Description.

Ports

The following table lists the ports available on the MIV_ESS core directly pertaining to the I²C module.

Table 7-2. I²C Port Signals

Ports	Width	Direction	Description
BOOTSTRAP_BYPASS	1	Input	The I²C module mode select: • When BOOTSTRAP_BYPASS = 0, on reset the I²C module enters Bootstrap mode. • When BOOTSTRAP_BYPASS = 1, the I²C module is always in Peripheral mode.
SCL_I	1	Input	I²C Clock Line Input
SCL_O	1	Output	I²C Clock Line Output
SCL_O_EN	1	Output	I²C Clock Line Output Enable
SDA_I	1	Input	I²C Data Line Input
SDA_O	1	Output	I²C Data Line Output
SDA_O_EN	1	Output	I²C Data Line Output Enable
I2C_IRQ	1	Output	I²C Interrupt

The I²C module is controlled over the APB target interface in the Peripheral mode, called APB_TARGET, or APB_mINITIATOR if APB Initiator mirroring is enabled in the MIV_ESS GUI.

The I²C module's inputs and outputs, as seen in the MIV_ESS SmartDesign instance, configured for Bootstrap mode and Peripheral mode, is shown in the following figure. The following figure also shows the SmartDesign instance for the configuration in which only the Peripheral mode is enabled.

Figure 7-4. MIV_ESS Instance for Bootstrap and Peripheral Mode

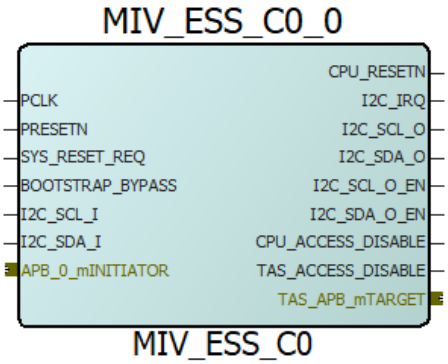
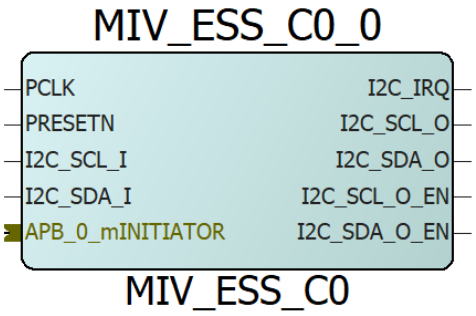


Figure 7-5. MIV_ESS Instance for Peripheral Mode Only



7.3 Programming

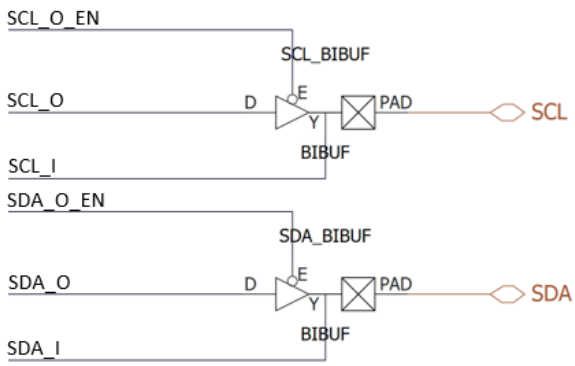
Software

The I²C Bootstrap operation (Bootstrap mode) is a hardware function that does not require any software. However, software is required to control the I²C module using the MIV_RV32 (when in Peripheral mode). The device driver is available from the GitHub page linked here.

Hardware

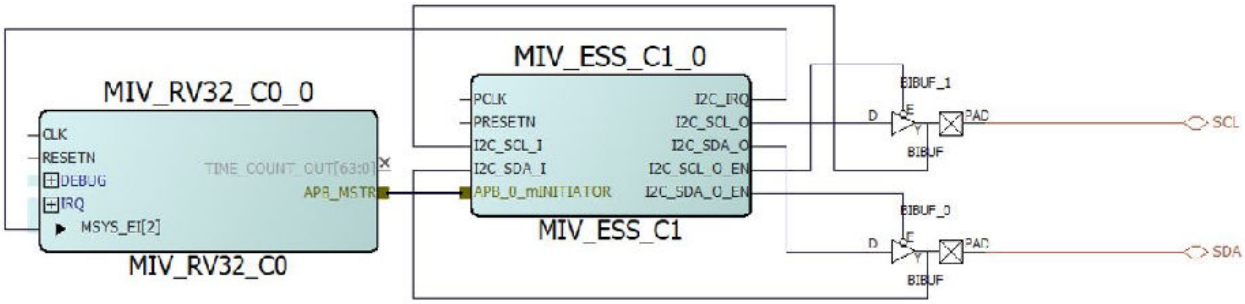
The I²C module uses individual input, output and output enable signals for the I²C serial clock and data lines (SCL and SDA). In a hardware design, bidirectional buffers are required to pull these input, output and output enable signals into open-drain, bidirectional lines. These must be connected, as shown in the following figure:

Figure 7-6. Bidirectional Buffers I²C Connections



The I²C interrupt (I2C_IRQ), when enabled, is asserted when a byte read/write transfer has been completed or arbitration was lost. This interrupt is connected to an interrupt pin on the MIV_RV32. The MIV_ESS with I²C bidirectional buffer lines and connections to the MIV_RV32 is shown in the following figure.

Figure 7-7. MIV_ESS with I2C Bidirectional Buffer Lines and Connections to MIV_RV32



I²C Module Memory Map

The I²C module's register addresses are determined with respect to the MIV_ESS base address. The MIV_ESS base address [MIV_ESS_BASE] is configurable, see section *Programming*. The I²C module base address is [MIV_ESS_BASE + 0xA00_0000]. The address offset of each I²C register is given in the following table.

```
Register address = MIV_ESS_BASE + 0xA00_0000 + Register Address Offset
```

Table 7-3. I²C Module Register Map

Register Name	Address Offset	R/W	Reset Value	Description
I2C_PRESCALER	0x00_0000	R/W	0xfff	Serial Clock Prescaler Register

.....continued

Register Name	Address Offset	R/W	Reset Value	Description
I2C_CTR	0x00_0004	R/W	0	Control Register
I2C_TXR	0x00_0008	R/W	0	Transmit Register
I2C_RXR	0x00_000C	R	X	Receive Register
I2C_CR	0x00_0010	R/W	0	Command Register
I2C_SR	0x00_0014	R	0	Status Register

Prescaler Register Description

The Prescaler Register is used to set the frequency of the I^2C serial clock (SCLK) generated by the I^2C module. The prescaler value required to set a particular I^2C clock frequency can be calculated using the following formula.

$$prescaler = \frac{SystemClockFrequency(PCLK)}{5*(DesiredI2CClockFrequency)} - 1$$

For example, for a System Clock Frequency of 50 MHz and a Desired I^2C Clock Frequency of 100 kHz:

$$prescaler = \frac{50MHz}{5*(100kHz)} - 1 = 99(dec)$$

For convenience, precalculated values for Normal, Fast, and Fast-Plus transmission speeds for a range of System Clock Frequencies are listed in the following table.

Table 7-4. I^2C Prescaler Register Calculated Values

System Clock Frequency (MHz)	Normal Speed (100 kHz)	Fast Speed (400 kHz)	Fast-Plus Speed (1 MHz)
100	199	49	19
90	179	44	17
80	159	39	15
70	139	34	13
60	119	29	11
50	99	24	9
40	79	19	7
30	59	14	5
20	39	9	3
10	19	4	1

Control Register Description

Table 7-5. I²C Control Register (I2C_CTR)

Bit Number	Bit Name	Reset Value	Description
7	core_en	0	I²C Module Enable Bit. • When set to 1, the I²C module is enabled. • When set to 0, the I²C module is disabled. • Default/Reset Value: 0
6	ien	0	I²C module Interrupt Enable Bit. • When set to 1, the I²C module interrupt is enabled. • When set to 0, the I²C module interrupt is disabled. • Default/Reset Value: 0
5:0	Reserved	0	Reserved

Additional operational information:

- The I²C module will only respond to new commands when the ‘core_en’ bit is set.
- The ‘core_en’ bit must only be cleared when there is no I²C operation in progress.
- Before changing the ‘ien’ bit or the value in the Prescaler Register, the ‘core_en’ bit must be set to 0.

Transmit Register Description

The following two tables describe the fields in the Transmit Register. This register has two interpretations depending on the operation being executed by the I²C module.

While transmitting an I²C control byte, the Transmit Register is interpreted as shown in the following table.

Table 7-6. I²C Transmit Register (I2C_TXR) – Control

Bit Number	Bit Name	Reset Value	Description
7:1	I2C Target Address	0	The 7-bit hardware address of the I²C target device.
0	Direction	0	Indicates the direction of the transfer. • When set to 1, reading data from target device. • When set to 0, writing data to target device.

When transmitting an I²C data byte, the Transmit Register is interpreted as shown in the following table.

Table 7-7. I²C Transmit Register (I2C_TXR) – Data

Bit Number	Bit Name	Reset Value	Description
7:0	Transmit Data	0	The data byte to be transmitted to the I²C target device.

Receive Register Description

The following table describes the field in the Receive Register.

Table 7-8. I²C Receive Register (I2C_RXR)

Bit Number	Bit Name	Reset Value	Description
7:0	Receive Data	X	The last byte received from the I²C target device.

Command Register Description
The following table describes the field in the Command Register.

Table 7-9. I²C Command Register (I2C_CR)

Bit Number	Bit Name	Reset Value	Description
7	STA	0	Generate I²C (Repeated) Start Condition. When set to 1, the I²C Module will transmit a (repeated) start condition with the next write transmission.
6	STO	0	Generate I²C Stop Condition. When set to 1, the I²C Module will transmit a Stop condition.
5	RD	0	Receive data from target device. When set to 1, the I²C Module will receive a data byte from the target device.
4	WR	0	Transmit data to target device. When set to 1, the I²C Module will transmit a data byte to the target device.
3	ACK	0	Read Mode Acknowledge. • When set to 0, the I²C Module will transmit an ACK to the target after receiving the next data byte. • When set to 1, the I²C Module will transmit a NACK to the target after receiving the next data byte.
2:1	Reserved	0	Reserved
0	IACK	0	Interrupt Acknowledge. • When set to 1, the I²C Module interrupt flag will be cleared. • When set to 0, the I²C Module will be able to transmit a new interrupt request.

Additional operational Information:

- Setting the ‘STA’ bit does not instantly transmit a start condition. After setting the ‘STA’ bit, the I²C start condition will not be transmitted until the next I²C write operation (that is, when the ‘WR’ bit is set).
- Setting the ‘ACK’ bit does not instantly transmit an ACK/NACK. After setting the ‘ACK’ bit, the ACK/NACK will not be transmitted until the next I²C read operation (that is, when the RD bit is set).
- When an interrupt is claimed by the processor, the ‘IACK’ bit must be toggled (set to ‘1’, then set to ‘0’) as the ‘IACK’ but must be set back to ‘0’ after the interrupt is claimed to allow for another interrupt to be generated.
- All bits in the command register are cleared automatically on the completion of an I²C operation.

Status Register Description
The following table describes the fields in the Status Register.

Table 7-10. I²C Status Register (I2C_SR)

Bit Number	Bit Name	Reset Value	Description
7	Received Acknowledgment	0	Acknowledge received ACK from the addressed target. • 1: indicates a NACK was received from target. • 0: indicates an ACK was received from target.
6	Busy	0	I²C Bus Busy. • Set to 1 after a START condition is detected on the I2C bus. • Set to 0 after a STOP condition is detected on the I2C bus.
5	Arbitration Lost	0	I²C Bus Arbitration Lost. Arbitration is lost (and this bit is set to 1) when: • A Stop condition is detected, but not requested by the I²C module. • The I²C module drives the SDA line HIGH, but another I²C device is driving the SDA line LOW.
4:2	Reserved	0	Reserved.
1	Transfer in Progress	0	Transfer in Progress. • 1: when the I²C module is currently performing a read/write transmission. • 0: when the I²C module has completed a read/write transmission.
0	Interrupt Flag	0	Interrupt Flag. The interrupt flag is set when: • A byte read/write transfer is completed. • Arbitration was lost. When the interrupt flag is set, the I2C_IRQ will be asserted if the Interrupt Enable bit in the Control Register has also been set. The processor must toggle the IACK bit in the Command Register to claim the interrupt.

8. PLIC

This section provides information on the Platform Level Interrupt Controller (PLIC) module used in the MIV_ESS core.

8.1 Description

The PLIC multiplexes multiple external interrupt signals into a single interrupt signal that can be connected to an external interrupt input pin of a processor.

The PLIC has the following four main blocks:

1. PLIC Gateway: The PLIC gateway is used to capture the interrupt before it is registered by the interrupt pending register. The PLIC gateway is asserted until the interrupt is cleared by a write to the claim complete register or by a system reset. Each of the interrupts enabled in the PLIC has its own gateway, which is connected to the interrupt pending register.
2. Interrupt Enable: When written to this register, it can enable or disable a specific interrupt. For example, if there are six PLIC interrupts in the design but only interrupt 1 and 3 are enabled, then these are the only interrupt that must be serviced.
3. Interrupt Pending: When the PLIC gateway asserts, the value is captured in the pending register until the interrupt is cleared or the system is reset.
4. Interrupt Claim Complete: It is responsible for generating the single external interrupt when a valid interrupt has occurred. A valid interrupt is when the interrupt is enabled and there is a pending interrupt. The PLIC identification is generated. The ID is read by the firmware and the correct interrupt handler is selected from the ID. When the register is written to it clears the PLIC gateway and interrupt pending.

When an interrupt occurs on an enabled interrupt, the PLIC gateway captures the interrupt and asserts the corresponding interrupt pending bit. Once the enable bit and the pending bit are asserted, then the `PLIC_IRQ` signal asserts until the interrupt is claimed by the software interrupt handler, or the system is reset.

When multiple interrupts assert, then the lowest interrupt number will be serviced first. For example, if interrupt 1 and 6 assert at the same time, interrupt 1 will be serviced first, followed by interrupt 6.

8.2 Interface

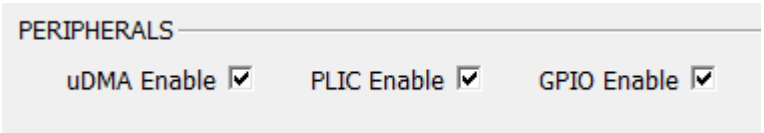
The following table lists the two PLIC parameters in the MIV_ESS.

Table 8-1. PLIC Parameters

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
Available Interrupt when PLIC Enabled	NUM_OF_INTS	1 - 31	8	The number of interrupts available in the design.
PLIC Enable	PLIC_EN	0 or 1	0	Enables or disables the PLIC interrupts in the design.

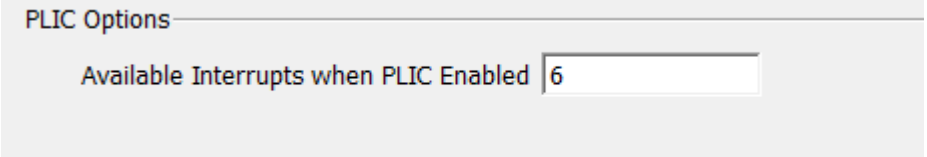
To enable the PLIC, the PLIC Enable parameter must be set in the **General** tab of the MIV_ESS configurator.

Figure 8-1. PLIC Enable



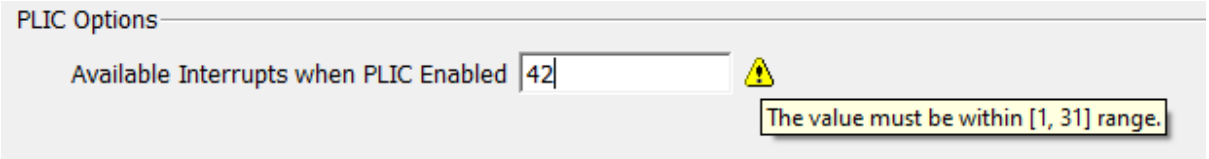
When the PLIC is enabled, the number of interrupts can be set in the **PLIC** tab of **MIV_ESS Configurator** → **Available Interrupts**, when PLIC is enabled. This allows you to select between 1 and 31 source interrupts.

Figure 8-2. PLIC Interrupts - Valid



Note: A warning message is displayed, if the value is beyond the specified range.

Figure 8-3. PLIC Interrupts - Invalid



The following table lists the ports available in MIV_ESS for the PLIC:

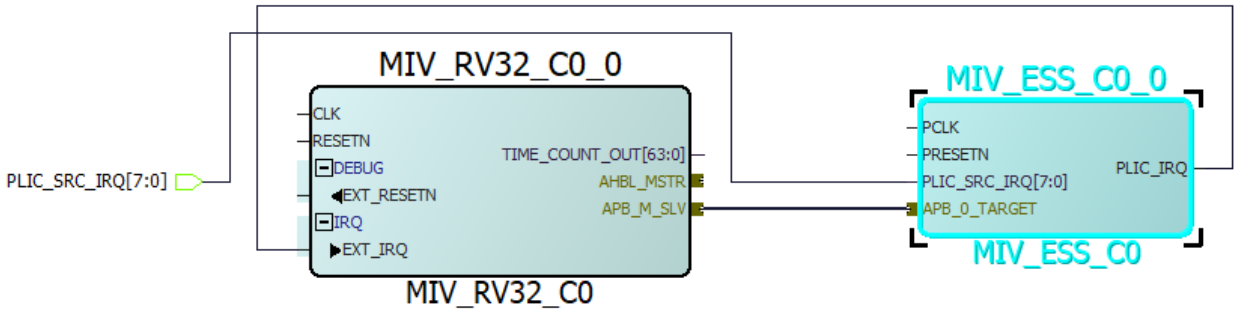
Table 8-2. PLIC Port Signals

Ports	Width	Direction	Description
PLIC_SCR_IRQ	NUM_OF_INTS:0	Input	Interrupt sources are active-high.
PLIC_IRQ	1	Output	PLIC interrupt, which connects to the external interrupt of a processor.

8.3 Programming

Software is required to enable, disable, and handle an asserted interrupt. This is available from https://github.com/Mi-V-Soft-RISC-V/platform. The following figure shows how the PLIC_IRQ signal is connected to the MIV_RV32 processor's external interrupt signal (`EXT_IRQ`).

Figure 8-4. PLIC Connection Diagram



The PLICs register addresses are determined with respect to the MIV_ESS base address. The MIV_ESS base address (MIV_ESS_BASE) is configurable. The PLICs base address is MIV_ESS_BASE + 0x0000000. The address offset of each register is as follows.

```
Register address = MIV_ESS_BASE + 0x000_0000 + Register Address Offset.
```

Table 8-3. PLIC Registers

Register	Address Offset	Reset Value	R/W	Description
Interrupt Pending	0x1000	0	R	Interrupt Pending Register

.....continued				
Register	**Address Offset**	**Reset Value**	**R/W**	**Description**
Interrupt Enable	0x2000	0	R/W	Interrupt Enable Register
Interrupt Claim Complete	0x20_0004	0	R/W	Interrupt Claim Complete Register

Interrupt Pending Register

The following table lists the interrupt pending status of each of the interrupt sources.

Table 8-4. Interrupt Pending Register

Bit Number	Bit Name	Reset Value	R/W	Description
0	Interrupt_pending 0	0	R	Reserved
1	Interrupt_pending 1	0	R	Interrupt Pending register for source interrupt 1.
…	…	…	…	…
31	Interrupt_pending 31	0	R	Interrupt Pending register for source interrupt 31.

Interrupt Enable Register

The Interrupt Enable register allows enabling each of the global interrupts corresponding to the bit in the register.

Table 8-5. Interrupt Enable Register

Bit Number	Bit Name	Reset Value	R/W	Description
0	Interrupt_enable 0	0	R/W	Reserved
1	Interrupt_enable 1	0	R/W	Interrupt Enable register for source interrupt 1.
…	…	…	…	…
31	Interrupt_enable 31	0	R/W	Interrupt Enable register for source interrupt 31.

Interrupt Claim/Complete Register

The register generates the interrupt source ID of each interrupt. It also sends the interrupt completion message to the associated gateway, which then clears the interrupt on the gateway and on the pending register.

Table 8-6. Interrupt Claim/Complete Register

Bit Number	Bit Name	Reset Value	R/W	Description
0	Interrupt_claim_complete 0	0	R/W	Reserved
1	Interrupt_claim_complete 1	0	R/W	Interrupt Claim Complete register for source interrupt 1.
…	…	…	…	…
31	Interrupt_claim_complete 31	0	R/W	Interrupt Claim Complete register for source interrupt 31.

9. SPI

This section provides information on the Serial Peripheral Interface (SPI) module used in the MIV_ESS core.

9.1 Description

The SPI is a controller core designed for synchronous serial communication using a Motorola, TI, or NSC mode of operations. The core is parameterized to allow user specification of the Operating mode, FIFO depth, and frame width. Operation is fully synchronous and operates on the system clock, as well as the external SPI clock for the Target mode.

The SPI consists of an APB interface designed to connect to an APB bus. Its registers, including transmit and receive FIFOs can be accessed by an APB Initiator.

The SPI controller has the following key features:

- SPI clock rate is configurable through parameter:
  - From PCLK/512 to PCLK/2 in two steps
  - Maximum data-rate of PCLK/2 in Initiator mode and PCLK/8 in Target mode.
- SPI protocol is configurable:
  - Initiator and target operation
  - As an initiator, supports up to eight target devices
  - Motorola SPI support
  - TI SPI support
  - NSC SPI support
  - Target select behavior configurable during IDLE cycles
  - Supports broadcast operation
  - Configurable frame size (4 to 32 bits)
- FIFO:
  - Width set to frame size for optimal core size
  - Depth configurable through the parameter
- Interrupt generation:
  - Receive/transmit data interrupts
  - FIFO overflow and under run
  - Command transmitted interrupt
- APB3 compliant

For more information about this IP, see *CoreSPI v5.2 Handbook* in the Libero Catalog.

9.2 Interface

9.2.1 Configuration Parameters

The following table lists the parameters (Verilog) for configuring the RTL code of the core.

Table 9-1. SPI Parameters

Configurator Name	Parameter Name	Value Values	Default Value	Description
SPI	SPI_EN	0 or 1	1	SPI Enable 0: Disabled 1: Enabled

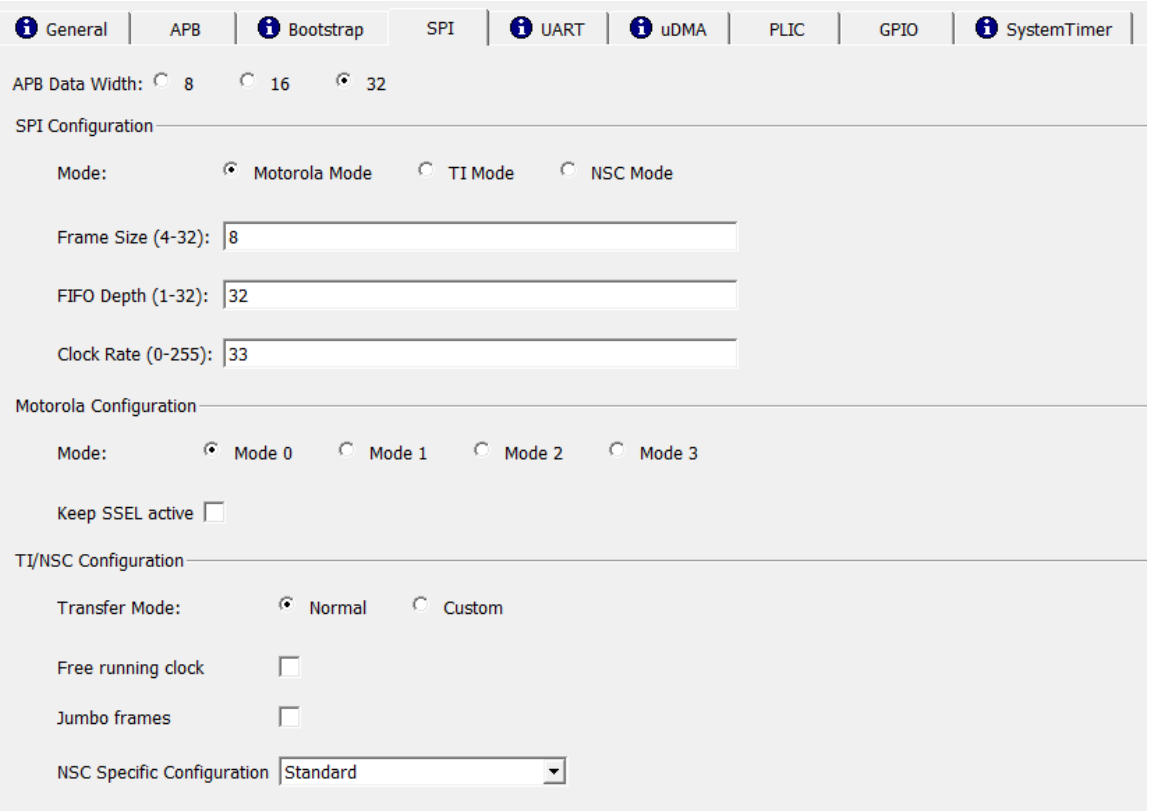
.....continued

Configurator Name	Parameter Name	Value Values	Default Value	Description
APB Data Width	APB_DWIDTH	8, 16, 32	8	APB data width can be 8, 16, or 32 bits. Operation in NSC mode is only possible with an APB data width of 32.
Frame Size (4-32)	CFG_FRAME_SIZE	4 to 32	4	SPI frame size, in bits. For Motorola and TI modes, this is the actual required frame size. For NSC mode, this is set to 9 + the required data frame size.
FIFO Depth (1-32)	CFG_FIFO_DEPTH	1 to 32	4	Number of frames that can be stored in the FIFO at any given time (both TX and RX FIFOs).
Clock Rate (0-255)	CFG_CLK	0 to 255	7	Clock rate parameter, which determines the generated SPI Initiator clock by: `SPICLK = PCLK/(2*(CFG_CLK+1))`
Mode	CFG_MODE	0 – 2	0	Determines Operating mode: 0: Motorola mode 1: TI mode 2: NSC mode
Mode	CFG_MOT_MODE	0 – 3	0	Motorola mode selection: 0: Mode 0 1: Mode 1 2: Mode 2 3: Mode 3
Keep SSEL active	CFG_MOT_SSEL	0 – 1	0	Target select active between back-to-back transfers in Motorola mode. 0: Target select behavior varies depending on the Motorola mode selected. 1: Active – Remains active between back-to-back transfers.
Transfer Mode	CFG_TI_NSC_CUSTOM	0 – 1	0	Enable custom transfer configuration in TI/NSC mode. 0: Normal transfer 1: Custom transfer
Free running clock	CFG_TI_NSC_FRC	0 – 1	0	Free running clock in TI/NSC mode. 0: Clock in-active between transfers 1: Clock remains active
Jumbo frames	CFG_TI_JMB_FRAMES	0-1	0	Concatenate frames in a TI mode back-to-back transfer: 0: Standard TI transfers 1: Jumbo frame transfers

.....continued				
Configurator Name	**Parameter Name**	**Value Values**	**Default Value**	**Description**
NSC Specific Configuration	CFG_NSC_OPERATION	0 -2	0	NSC specific transfer settings: 0: Standard NSC transfers 1: Idle cycles inserted between frames in back-to-back transfers. 2: Large response frames. Response frames stored in TX_FIFO are concatenated to form a single large response frame.

The following figure shows the **SPI** Configuration tab.

Figure 9-1. SPI Configuration Window



9.2.2 I/O Signals

The following table lists the SPI I/O signals.

Table 9-2. SPI I/O Signal Descriptions

Name	Direction	Description
PCLK	Input	APB System Clock – Reference clock for all internal logic.
PRESETN	Input	APB active-low asynchronous reset.

.....continued		
Name	**Direction**	**Description**
SPIINT	Output	Interrupt pending: This active-high output signal is the interrupt output signal from SPI. It can be programmed to become active on certain events to inform the CPU that such an event has occurred. The CPU can then take appropriate action.

9.3 Programming

This section describes SPI registers.

9.3.1 Register Summary

The following tables list the values in hexadecimal format; type designations: R = read-only; W = write-only; R/W = read/write.

Table 9-3. SPI Internal Register Address Map

Register Name	Address Offset	R/W	Width	Reset Value	Description
CONTROL	0x00	R/W	8	0x00	Control Register 1
INTCLEAR	0x04	W	8	0x00	Interrupt Clear Register
RXDATA	0x08	R	32	0x00	Receive Data Register Reading from this register reads one frame from the RX FIFO.
TXDATA	0x0C	W	32	0x00	Transmit Data Register Writing to this register writes one frame to the TX FIFO.
INTMASK	0x10	R	8	0x00	Masked interrupt status These bits indicate the masked interrupt status by ANDing the interrupt enables in the CONTROL and CONTROL2 registers with the raw interrupt register. When any of these bits are set, the interrupt output will be active. Bits are cleared by writing to the Interrupt clear register.
INTRAW	0x14	R	8	0x00	Raw interrupt status
CONTROL2	0x18	R/W	8	0x80	Control Register 2
COMMAND	0x1C	W	8	0x00	Command Register
STAT	0x20	R	8	0x00	Status Register

.....continued

Register Name	Address Offset	R/W	Width	Reset Value	Description
SSEL	0x24	R/W	8	0x44	Target Select Register Specifies the targets selected Default 0 (nothing selected). Write 1 to each bit to select one or more targets. Target select output pin is active Low. In TI mode, the target select outputs are inverted to become active High.
TXDATA_LAST	0x28	W	32	0x00	Transmit Data Register Writing to this register writes one frame to the TX FIFO. Also indicates to SPI that this is the last frame in this packet before SSEL is supposed to go inactive, effectively allowing for the specification of the number of transmitted frames.
CLK_DIV	0x2C	R/W	8	CFG_CLK	Clock rate register. Writing to this register will update clock division factor of SPI generated clock (SPICLKO) in the Initiator mode.

9.3.2 Control Register 1

The following tables list the Control register 1 and bit definitions.

Table 9-4. Control Register 1

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x00	CONTROL	R/W	8	0x00	Control Register 1

Table 9-5. Control Register 1 Bit Definition

Bits	Name	R/W	Description
7	OENOFF	R/W	0: SPI output enable active as required 1: The core will not assert the SPI output enable. This allows multiple targets to be connected to a single Initiator sharing a single target select and software protocol implemented that can enable the targets transmit data when a certain broadcast address SPI command is received.

.....continued

Bits	Name	R/W	Description
6	FRAMEURUN	R/W	W 0: Under runs are generated whenever a read is attempted from an empty transmit FIFO 1: Under run condition will be ignored for the complete frame if the first data frame read resulted in a potential overflow, that is, the target was not ready to transmit any data. If the first data frame is read from the FIFO and transmitted then an under run will be generated if the FIFO becomes empty for any of the remaining packet frames, that is, while SSEL is active. Initiator operation will never create a transmit FIFO under run condition.
5	INTTXURUN	R/W	Interrupt on transmit under run 0: Interrupt disabled 1: Interrupt enabled.
4	INTRXOVFLOW	R/W	Interrupt on receive overflow 0: Interrupt disabled 1: Interrupt enabled.
3	INTTXDONE	R/W	Interrupt on transmit data of data which has been placed in TX FIFO through the TXDATA_LAST register. 0: Interrupt disabled 1: Interrupt enabled.
2	—	—	Reserved
1	INITIATOR	R/W	0: Run SPI in Target mode 1: Run SPI in Initiator mode
0	ENABLE	R/W	0: Core does not respond to external signals until this bit is enabled. SPISCLKO driven to zero and SPIOEN, SPISS (target select) driven inactive. 1: Core is active

9.3.3 Interrupt Clear Register

The following tables list the Interrupt Clear Register and bit definitions.

Table 9-6. Interrupt Clear Register

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x04	INTCLEAR	W	8	0x00	Interrupt Clear Register

Table 9-7. Interrupt Clear Register Bit Definition

Bits	Name	R/W	Description
7	TXRFM	W	Writing 1 clears the TXRFM interrupt.
6	DATA_RX	W	Writing 1 clears the DATA_RX interrupt.
5	SSEND	W	Writing 1 clears the SSEND interrupt.

.....continued			
Bits	**Name**	**R/W**	**Description**
4	CMDINT	W	Writing 1 clears the CMDINT interrupt.
3	TXUNDERRUN	W	Writing 1 clears the TXUNDERRUN interrupt.
2	RXOVERFLOW	W	Writing 1 clears the RXOVERFLOW interrupt.
1	—	—	Reserved
0	TXDONE	W	Writing 1 clears the TXDONE interrupt.

9.3.4 RX Data Register

The following tables list the Rx and Tx Data Registers.

Table 9-8. RX Data Register

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x08	RXDATA	R	32	0x00	Receive Data Register Reading from this register reads one frame from the RX FIFO.

Table 9-9. TX Data Register

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x0C	TXDATA	W	32	0x00	Transmit Data Register Writing to this register writes one frame to the TX FIFO.

9.3.5 Interrupt Masked Register

The following table lists the Interrupt Masked Register.

Table 9-10. Interrupt Masked Register

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x10	INTMASK	R	8	0x00	Masked interrupt status These bits indicate the masked interrupt status by ANDING the interrupt enables in the CONTROL registers with the raw interrupt register. When any of these bits are set, the INTERRUPT output will be Active. The bits are cleared by writing to the Interrupt clear register.

9.3.6 Interrupt Raw Register

The following tables list Interrupt Raw Register and bit definitions.

Table 9-11. Interrupt Raw Register

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x14	INTRAW	R	8	0x80	Raw interrupt status

Table 9-12. Interrupt Raw Register Bit Definition[1]

Bits	Name	R/W	Description
7	TXRFM	R	Indicates that there is at least one frame free in the transmit FIFO for writing.
6	DATA_RX	R	Indicates that at least one byte is received. Check the RXEMPTY bit in the Status Register to determine if there is more Rx data available in the Rx FIFO. Writing a 1 to the corresponding bit in the Interrupt Clear register clears this bit, provided that the RX FIFO is empty.
5	SSEND	R	Indicates that SSEL is Inactive.
4	CMDINT	R	Indicates that the number of frames set by the CMDSIZE register are received as a single packet of frames (SSEL held active).
3	TXUNDERRUN	R	Indicates that in Target mode that the data was not available when required in the transmit FIFO.
2	RXOVERFLOW	R	Indicates that in Initiator and Target mode, the receive FIFO is overflowed.
1	—	—	Reserved
0	TXDONE	R	Indicates that all frames, including last frame (see Aliased TX Data Register), are transmitted.

1. Writing a 1 to the corresponding bit in the Interrupt Clear register clears the associated Interrupt Raw register bit, provided that the hardware condition that triggered the interrupt in the first instance is resolved.

9.3.7 Control Register 2

The following tables list the Control register 2 and bit definitions.

Table 9-13. Control Register 2

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x18	CONTROL2	R/W	8	0x00	Control Register 2

Table 9-14. Control Register 2 Bit Definition

Bits	Name	R/W	Description
7	INTEN_TXRFM	R/W	0: No effect 1: Enables the interrupt when there is room in the Tx FIFO.
6	INTEN_DATA_RX	R/W	0: No effect 1: Enables the interrupt when at least one byte is received.
5	INTEN_SSEND	R/W	0: No effect 1: Enables the interrupt as SSEL goes High. SPI Initiator and target modes.

.....continued

Bits	Name	R/W	Description
4	INTEN_CMD	R/W	0: No effect 1: Enables Interrupt after the number of frames set by CMDSIZE (above) is received as a single packet of frames (SSEL held active).
3	—	R/W	Reserved
2:0	CMDSIZE	R/W	Number of frames sent before interrupt is generated when INTEN_CMD is set (see above).

9.3.8 Command Register

The following tables list the Command Register and bit definitions.

Table 9-15. Command Register

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x1C	COMMAND	W	8	0x00	Command Register (write-only)

Table 9-16. Command Register Bit Definition

Bits	Name	R/W	Description
7:2	—	—	Reserved
1	TXFIFORST	W	Writing 1 will reset the TX FIFO. This bit always reads as zero.
0	RXFIFORST	W	Writing 1 will reset the RX FIFO. This bit always reads as zero.

9.3.9 Status Register

The following tables list the Status Register and bit definitions.

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x20	STAT	R	8	0x44	Status Register (read-only)

Table 9-17. Status Register Bit Definition[1]

Bits	Name	R/W	Description
7	ACTIVE	R	Core is still transmitting data.
6	SSEL	R	Current state of SSEL.
5	TXUNDERRUN	R	Transmit FIFO under flowed.
4	RXOVFLOW	R	Receive FIFO overflowed.
3	TXFULL	R	Transmit FIFO is full, that is, no space for more data.
2	RXEMPTY	R	Receive FIFO is empty, that is, no data available to read.
1	DONE	—	No of requested frames have been transmitted and received.

.....continued			
Bits	**Name**	**R/W**	**Description**
0	FIRSTFRAME	R	Next frame in Receive FIFO was first received after SSEL went active (Command Frame).

9.3.10 Target Select Register

The following table lists the Target Select Register.

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x24	SSEL	R/W	8	0x00	Target Select Register Specifies the targets selected. Default 0 (nothing selected). Write 1 to each bit to select one or more targets. Target select outputs are active-low in Motorola and NSC modes. In TI mode, the target select outputs are inverted to become active High. For example, to select Target 0 and Target 5, write the value 0x00100001 to this register.

9.3.11 Aliased TX Data Register

The following table lists the Aliased TX Data Register.

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x28	TXDATA_LAST	W	32	0x00	Transmit Data Register Writing to this register writes one frame to the TX FIFO. Also indicates to SPI that this is the last frame in this packet before SSEL is supposed to go inactive, effectively allowing for the specification of the number of transmitted frames.

9.3.12 Clock Rate Register

PADDR[5:0]	Register Name	R/W	Width	Reset Value	Description
0x2C	CLK_DIV	R/W	8	CFG_CLK	Clock rate register. Writing to this register will update clock division factor of SPI generated clock (SPICLKO) in the Initiator mode. Clock rate of generated SPI Initiator clock is determined by the formula:SPICLK = PCLK / (2 * (CLK_DIV + 1)) The register value overrides the parameter value (CFG_CLK) set in the core configuration window. At the power ON, the CLK_DIV register will have the value of configurable parameter `CFG_CLK` and this value will be used to determine the clock rate of the generated SPI Initiator clock. Whenever the CLK_DIV register is updated, the new value is used to determine the clock rate of the generated SPI Initiator clock. **Note:** It is recommended that the user updates this register when the core is not performing any SPI transactions.

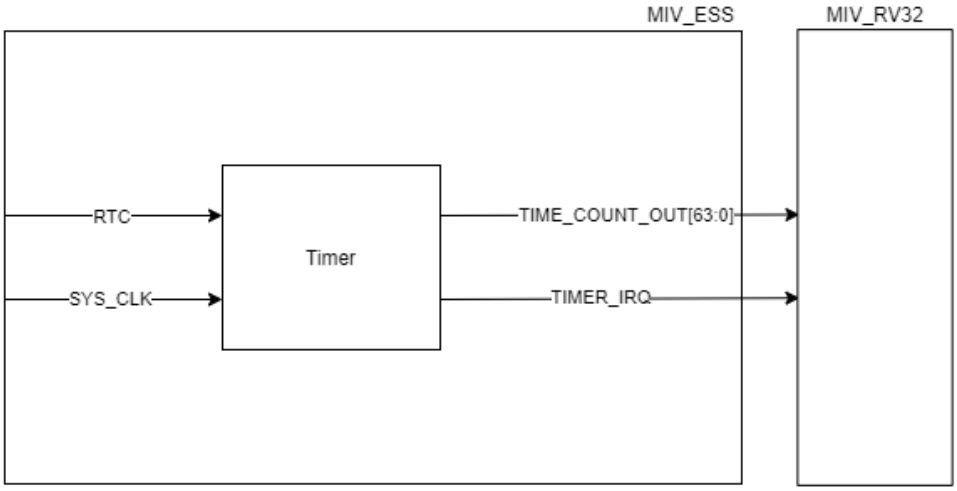
10. TIMER

This section provides information on the Timer module used in the MIV_ESS core.

10.1 Description

Timer with 64-bit resolution, which can be used as a system machine timer or as a general timer resource.

Figure 10-1. Timer – Top-Level Diagram



10.1.1 Features

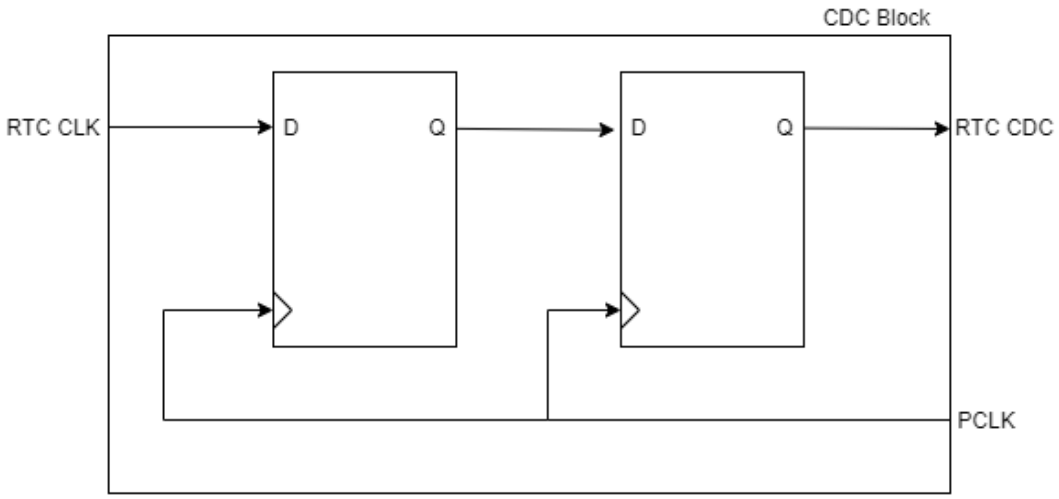
The following features are implemented in the Timer module:

- An APB interface for accessing the timer count (MTIME) register, the timer compare (MTIMECMP) register, and the configuration registers.
- A 64-bit Timer register ‘MTIME’.
- A 64-bit Timer Compare register ‘MTIMECMP’.
- A 16-bit Prescaler that divides the input clock source by a predefined integer value to derive an MTIME timebase.
- A Timer interrupt signal (TIMER_IRQ), if enabled, generates an interrupt when the count register (MTIME) exceeds the value found in the Timer Compare register (MTIMECMP).
- Two input clock options are available:
  - External Real Time Clock (RTC) source.
  - The System Clock (PCLK) source.

10.1.2 RTC CDC – Clock Domain Crossing

Only the Real-Time Clock (RTC) source clock uses the Clock Domain Crossing (CDC) feature. The CDC is used to synchronize the RTC with the system clock. Therefore, no timing errors are created due to two separate clock domains for the interrupt generation. The clock pulses are synchronized using two flip-flops. These two flip-flops create an area for any metastability that might occur while synchronizing the RTC. The following figure shows the flip-flop synchronization.

Figure 10-2. Timer - CDC Block

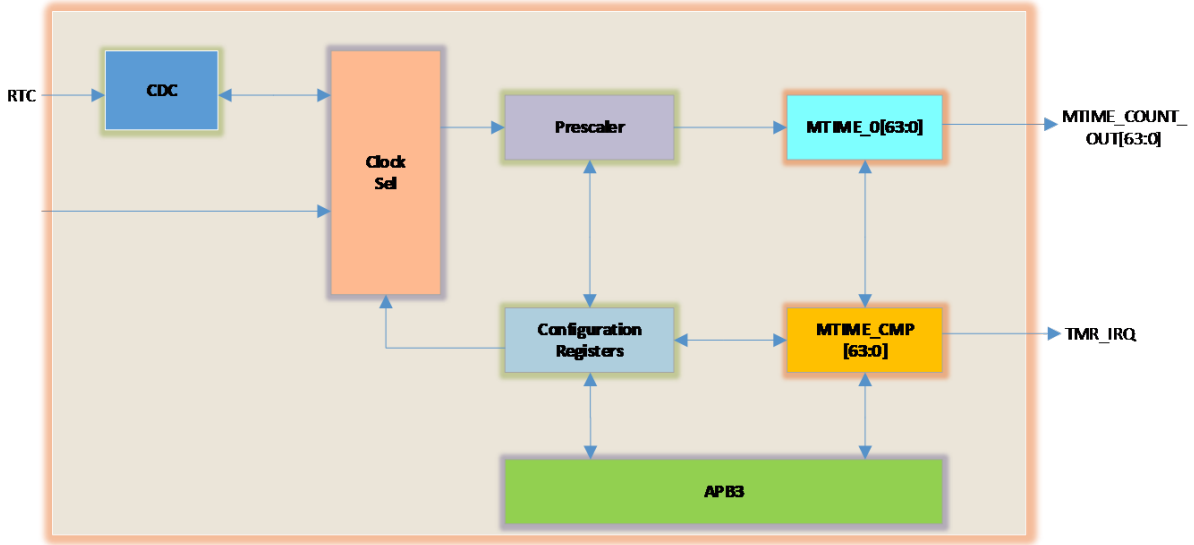


The source RTC_CLK is flopped twice before it is used by destination logic. The two flip-flops aid in correct sampling of the signal without Metastable States. The source RTC_CLK must run at strictly less than half the rate of the System Clock (PCLK).

10.1.3 Block Diagram

The following figure shows timer's block diagram with a brief description.

Figure 10-3. Timer Block Diagram



CDC and Clock Sel

The CDC block takes the input RTC and performs Clock Domain Crossing on it to bring it into the system clock domain. The new post CDC RTC is fed into the Clock Sel block. Depending on user configuration, the RTC or system clock are either used as the target clock for timing intervals in the system.

Prescaler

The Prescaler component is used to divide the selected clock source by a defined integer value, which is then fed to the input of the MTIME counter register. The Prescaler has Read Only access over the APB I/F.

MTIME Register

The MTIME register contains the 64-bit value of the timer count. The count increments by 1 every time the Prescaler ticks (reaches the end of its own count). The MTIME register also outputs an `MTIME_COUNT_OUT[63:0]` signal that contains the current value of the timer count. The register has Read/Write access over the APB I/F.

MTIMECMP

MTIMECMP is the 64-bit timer compare register, it pre-sets the threshold which needs to be reached by the MTIME register. When the MTIME register value is greater than or equal to the value found in register MTIMECMP, a timer interrupt signal (TIMER_IRQ) is generated. The register has Read/Write access over the APB I/F.

TIMER_IRQ

The `TIMER_IRQ` is the timer generated interrupt signal output from the Timer module and appears as an output on MIV_ESS. The signal can be fed directly into the MIV_RV32's TMR_IRQ input if the soft-processor's internal MTIMER module has been internally disabled. Alternatively, it can be connected to the MIV_RV32 processor core as a regular external interrupt.

The `TIMER_IRQ` is asserted High only if the value of MTIME register is greater than or equal to the value of the MTIMECMP register. The interrupt is Low for all other conditions.

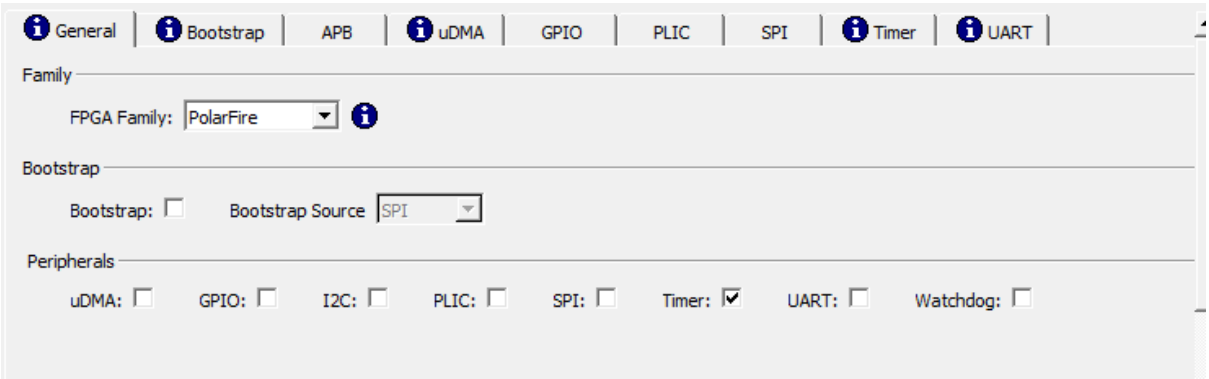
10.2 Interface

This section describes parameters and port lists of the Timer module.

10.2.1 Parameters

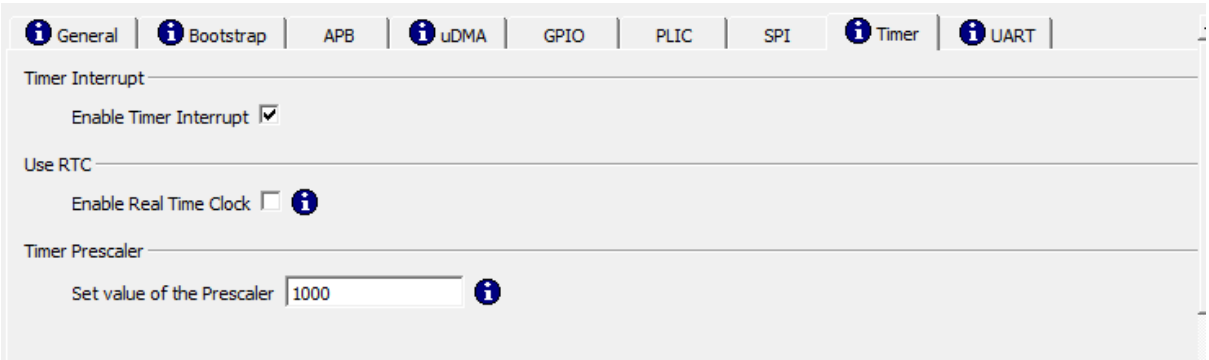
To enable the timer, select the **Timer** check box in the **Peripherals** section.

Figure 10-4. Selecting Timer Option



After the Timer option is enabled, the 'Timer' tab will be available for configuring, and it can be used to specify the Timer configuration as shown in the following figure.

Figure 10-5. Timer Settings



The following table lists the Timer parameters.

Note: Checked boxes for parameters hold the value ‘1’ and un-checked boxes hold the value ‘0’.

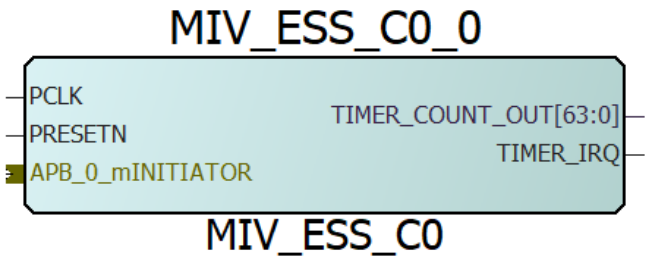
Table 10-1. Timer Parameters

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
Timer (Enable)	SYS_TIMER_EN	0 or 1	1	0: The Timer is not enabled in MIV_ESS 1: The Timer is enabled in MIV_ESS
Enable Timer Interrupt	INTERNAL_MTIME_IRQ	0 or 1	1	0: Disable Timer-generated interrupt 1: Enable Timer-generated interrupt If enabled, the timer interrupt (TIMER_IRQ) output signal is enabled and it is asserted High if the timer based interrupt occurs. If disabled, the timer interrupt signal is not present in the design.
Enable Real Time Clock	MTIME_RTC_CLOCK	0 or 1	0	0: Disable Real-Time Clock (RTC) 1: Enable Real-Time Clock (RTC) If the RTC is enabled, an RTC input appears on the MIV_ESS core and the timer uses that RTC as the reference clock for generating the timer interrupt for the system. If the RTC is disabled, the timer uses the input system clock (PCLK) as the reference clock for generating the timer interrupt for the system.
Timer Prescaler	MTIME_PRESCALER	1 – 65,335	1000	This parameter pre-sets the Prescaler value in decimal number. It determines, the amount of clock cycles the target system clock or RTC need to make for MTIME register to increment by a value of 1. The minimum Prescaler value is 0, where no scaling of the target clock occurs. The maximum Prescaler value is 65,535, where the target clock needs to go through 65,535 cycles before a single MTIME register value increment.

10.2.2 Port List

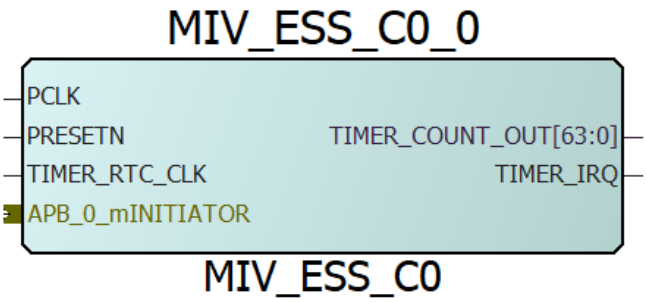
The timer inputs and outputs as seen on the MIV_ESS instance without the Real-Time Clock is shown in the following figure.

Figure 10-6. MIV_ESS: Timer Module



The Timer inputs and outputs of the MIV_ESS instance with the Real-Time Clock enabled is shown in the following figure.

Figure 10-7. MIV_ESS: Timer Module



The following table shows only the unique inputs and outputs to the timer.

Table 10-2. Timer – I/Os

Ports	Width	Direction	Description
TIMER_COUNT_OUT	[63:0]	Output	The current timer count value of the Timer module.
TIMER_IRQ	1	Output	The timer interrupt output signal generated around the timer's special clock intervals.
TIMER_RTC_CLK	1	Input	The input for the RTC. Only present if enabled in the MIV_ESS GUI.

Note: The RTC input is optional, and it does not appear in the top-level design unless the 'Enable Real Time Clock' option is checked in the GUI. The same applies for the timer generated interrupt signal, it is only available if not disabled in the GUI.

10.3 Programming

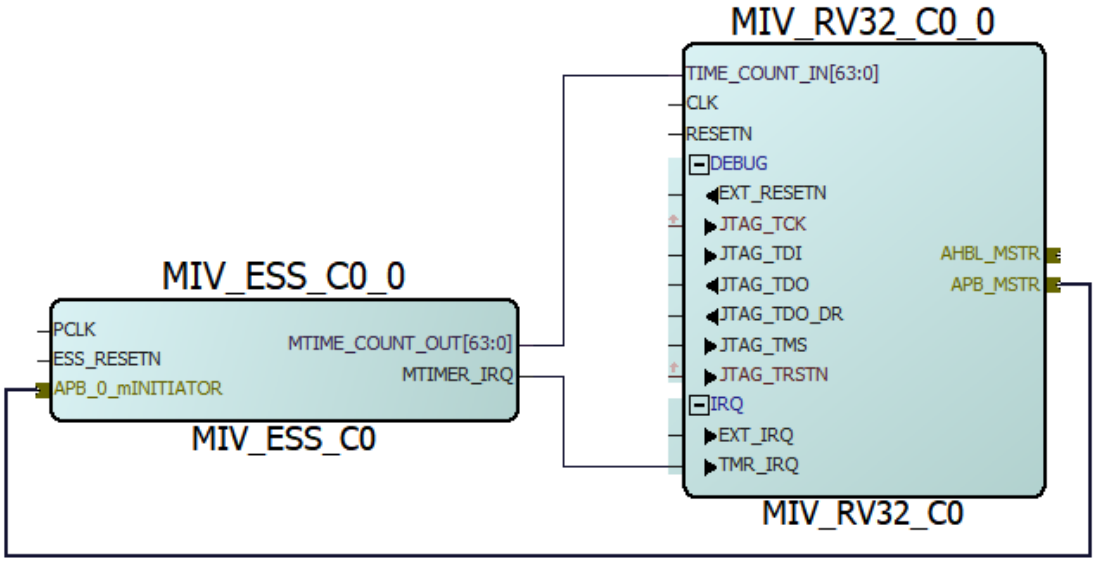
This section describes how to use the timer and timer register maps.

10.3.1 How to Use the Timer

After the Timer is enabled, it gets configured from the individual **Timer** tab and gets integrated into the MIV_ESS instance.

The timer can be connected to MIV_RV32 directly through the `TMR_IRQ` input on the core, provided the internal MTIMER on the MIV_RV32 core is disabled.

Figure 10-8. MIV_ESS: Timer Module - Connections



Alternatively, the timer can be wired up as an external interrupt for the target soft-processor core.

10.3.2 Details of Operation

The initial operation of the timer is determined in the GUI from the following:

- Enable the timer interrupt
- Determine the timer count clock
- Pre-set the value by which the timer count must be prescaled.

When reset, the value of MTIME register is 0, Prescaler is 0 and the value of MTIMECMP register is 64'FFFF_FFFF_FFFF_FFFF.

On next positive clock edge of the selected timer clock, the Prescaler counter increments by 1 and continues to increment until it reaches the value pre-set in the GUI. When the counter reaches that value, the MTIME register increments by 1 and the Prescaler counter goes back to zero.

Both the MTIME and MTIMECMP register values can be written through software over the APB I/F.

The MTIME register is typically read and the MTIMECMP value updated with a value of MTIME plus the required interrupt period.

The MTIME register continues to increment until the count reaches a value that is greater than or equal to the value of the MTIMECMP register. When this occurs, the timer interrupt is asserted High. The timer interrupt is asserted Low when the MTIMECMP register is updated with a value greater than MTIME. This is typically the current MTIME plus the next required interrupt period value.

10.3.3 Timer Register Map

The Timer module's register addresses are determined with respect to the MIV_ESS base address. The MIV_ESS base address [MIV_ESS_BASE] is configurable. The Timer module base address is [MIV_ESS_BASE + 0x200_0000]. The address offset of each timer register is given in the following table.

```
Register address = MIV_ESS_BASE + 0x200_0000 + Register Address Offset
```

Table 10-3. Timer Register Map

Register Name	Address Offset	R/W	Reset Value	Description
MTIMECMP_L	0x4000	R/W	0xFFFF_FFFF	The lower 32-bits of the compare time register Write (PWDATA[31:0] and Read (PRDATA[31:0]).
MTIMECMP_U	0x4004	R/W	0xFFFF_FFFF	The upper 32-bits of the compare time register Write (PWDATA[31:0] and Read (PRDATA[31:0]).
MTIME_L	0xBFF8	R/W	0x0	The lower 32-bits of the time count register Write (PWDATA[31:0] and Read (PRDATA[31:0]).
MTIME_U	0xBFFC	R/W	0x0	The upper 32-bits of the time count register Write (PWDATA[31:0] and Read (PRDATA[31:0]).
Prescaler	0x5000	R	—	The pre-set Prescaler value can be read by Read (PRDATA[31:0]).

11. UART

This section provides information on the UART module used in the MIV_ESS core.

11.1 Description

The UART is a serial communication controller with a flexible, serial data interface that is intended primarily for embedded systems. The UART can be used to interface directly to industry standard UARTs. The UART is intentionally a subset of full UART capability to make the function cost-effective in a programmable device.

The UART has the following key features:

- Asynchronous mode to interface with industry standard UART
- Optional transmit and receive FIFOs
- Advanced Peripheral Bus (APB) interface
- Fixed and Programmable modes of operation

For more information about this IP, see *CoreUARTapb v5.7 Handbook* in the Libero Catalog.

11.2 Interface

11.2.1 Configuration Parameters

This section describes configuration parameters of UART.

The following table lists the parameters (Verilog) for configuring the RTL code of the core.

Table 11-1. UART Configurable Options

Configurator Name	Parameter Name	Valid Values	Default Value	Description
UART	UART_EN	0 or 1	1	UART Enable 0: Disabled 1: Enabled
TX FIFO	TX_FIFO	0 or 1	0	Transmit FIFO 0: Disabled 1: Enabled
RX FIFO	RX_FIFO	0 or 1	0	Receive FIFO 0: Disabled 1: Enabled
Configuration	FIXEDMODE	0 or 1	1	0- Programmable 1- Fixed Fixed or Programmable mode. In Fixed mode, the parameters BAUD_VALUE, Character Size, and Parity are hardwired. In Programmable mode, they are programmed by the control registers.
Baud Value[1]	BAUD_VALUE	1 to 8191	1	Baud value is set only when configuration is set to Fixed mode.

.....continued

Configurator Name	Parameter Name	Valid Values	Default Value	Description
Fractional Part of Baud Value	BAUD_VAL_FRCTN	0 to 7	0	This parameter is only relevant when the parameter FIXEDMODE is set to Fixed and parameter BAUD_VAL_FRCTN_EN has been enabled. The value chosen here is added to the baud value to give a precise baud value.
Enable Extra Precision	BAUD_VAL_FRCTN_EN	0 or 1	0	When parameter FIXEDMODE is set to Programmable, enabling this parameter enables an additional control register (Control Register 3) that can be used to set a fractional part for the baud value. The baud value can be set with a precision of 0.125. When parameter FIXEDMODE is set to Fixed, enabling this parameter allows you to set a fixed fractional part for the baud value. The size of the fractional part is specified by the BAUD_VAL_FRCTN parameter.
Status Flags	UART_STATUS_FLAGS	0 or 1	0	When enabled the error, `READY` and `OVERFLOW` status signals are available as outputs.
Character Size	PRG_BIT8		0	This option can only be set when Configuration mode is set to Fixed mode. This option defines the number of valid data bits in the serial bitstream. Character size can be 8 bits or 7 bits.
Parity	PRG_PARITY		0	This option can only be set when Configuration mode is set to Fixed mode. The options for parity are as follows: Parity Disable, Even Parity, or Odd Parity.
RX Legacy Mode	RX_LEGACY_MODE		0	When disabled, the `UART_RXRDY` signal is synchronized with the UART_FRAMING_ERR output, which occurs after the STOP bit. When enabled (Legacy mode), the `UART_RXRDY` signal is asserted after all data bits have been received, but before the STOP bit.
FIFO Implementation	USE_SOFT_FIFO		0	When disabled, the FIFO is implemented using a device-specific hard macro. When enabled, a 16-byte FIFO is implemented in the FPGA logic instead. 54SXA and RTSX-S devices use this soft-FIFO by default.

Note:
1. BAUD_VALUE = 0 is not supported.

The following table lists the baud value fractions and precisions.

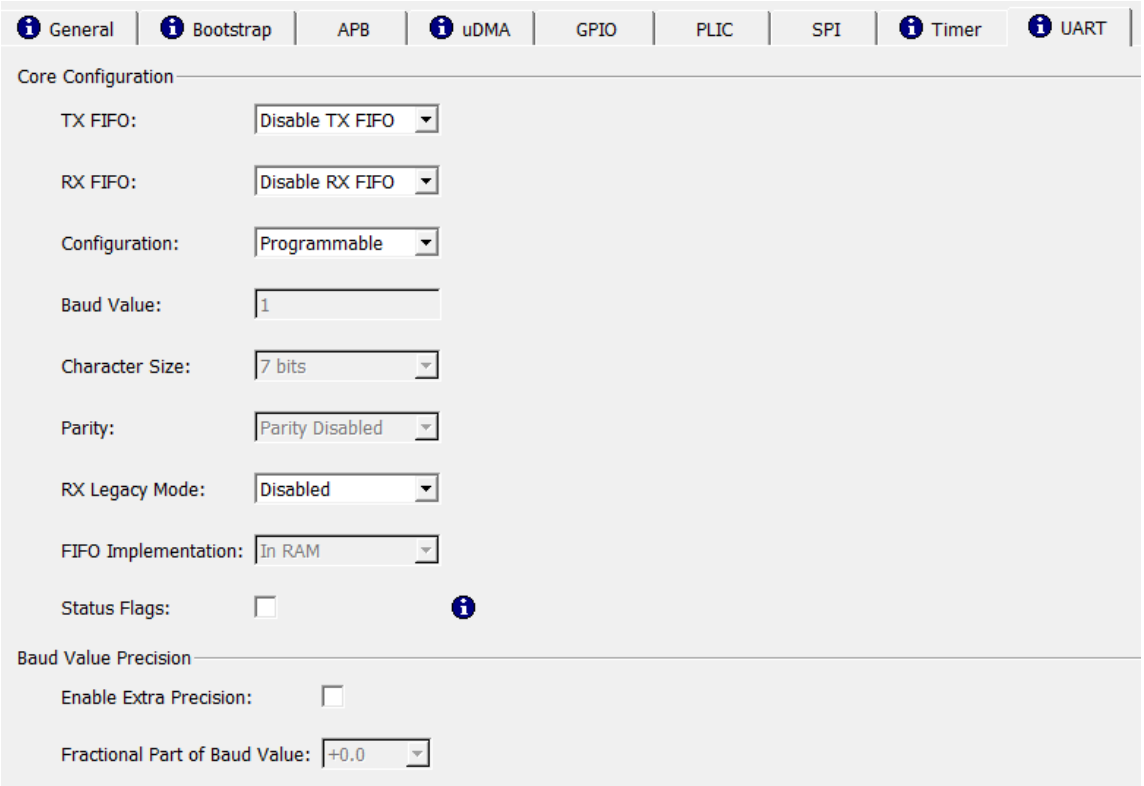
Table 11-2. Baud Value Fraction

BAUD_VAL_FRCTN	Precision
0	+0.0

.....continued	
BAUD_VAL_FRCTN	**Precision**
1	+0.125
2	+0.25
3	+0.375
4	+0.5
5	+0.625
6	+0.75
7	+0.875

The following figure shows the **UART** tab, as well as cross-references to the corresponding top-level parameters.

Figure 11-1. UART Configuration Window



11.2.2 I/O Signals

The following table lists the UART I/O signal description.

Table 11-3. UART Signals

Name	Direction	Description
PCLK	Input	APB System Clock – Reference clock for all internal logic.
PRESETN	Input	APB active-low asynchronous reset.

.....continued		
Name	**Direction**	**Description**
UART_TXRDY	Output	Status bit; when set to logic 0, indicates that the transmit data buffer/FIFO is not available for additional transmit data.
UART_RXRDY	Input	Status bit; when set to logic 1, indicates that data are available in the receive data buffer/FIFO to be read by the system logic. The data buffer must be read through APB via the Receive Data Register (0x04) to prevent an overflow condition from occurring.
UART_PARITY_ERR	Output	Status bit; when set to logic 1, indicates a parity error during a receive transaction. When RX FIFO is enabled, this bit is self-clearing between bytes. Otherwise, this bit is synchronously cleared by performing a Read operation on the Receive Data register through the APB target interface.
UART_FRAMING_ERR	Output	Status bit; when set to logic 1, indicates a framing error (that is, a missing STOP bit) during the last received transaction. When RX FIFO is enabled, this bit is self-clearing between bytes. Otherwise, this bit is synchronously cleared by performing a read operation on the Receive Data register through the APB target interface.
UART_OVERFLOW	Output	Status bit; when set to logic 1, indicates that a receive overflow has occurred. This bit is synchronously cleared by performing a read operation on the Receive Data register through the APB target interface.
UART_RX	Input	Serial receive data
UART_TX	Output	Serial transmit data

11.3 Programming

This section describes UART Programmer's models.

Table 11-4. UART Registers

Register Name	Address Offset	R/W	Reset Value	Description
TxData	0x000	W	0x00	Transmit Data register
RxData	0x004	R	0x00	Receive Data register
Ctrl1	0x008	R/W	0x00	Control register 1
Ctrl2	0x00C	R/W	0x00	Control register 2
Status	0x010	R	0x01	Status Register
Ctrl3	0x014	R/W	0x00	Control register 3

11.3.1 Transmit Data Register

The Transmit Data Register contains the 7- or 8-bit transmit data.

11.3.2 Receive Data Register

The Receive Data Register contains the 7- or 8-bit receive data.

11.3.3 Control Register 1

Control Register 1 contains a single-field, baud value, used to set the baud rate for UART.

The baud value must be set according to the following equation:

$$\text{baud val} = \frac{clk}{16 \times \text{baud rate}} - 1$$

Where, clk is the system clock frequency in Hertz.

The result of this calculation must be rounded to the nearest integer and converted to hexadecimal to obtain the value that must be written to Control register 1 and Control register 2, as shown in the following table. For example, when the clock frequency is 10 MHz and a baud rate of 9,600 is desired, 0x40 must be written to Control register 1 and 0x00 must be written to Control register 2. When the clock frequency is 50 MHz and a baud rate of 381 is desired, 0xFF must be written to Control register 1, and 0x1F must be written to the top 5 bits of Control register 2.

Table 11-5. Control Register 1

Bit(s)	Name	R/W	Function
7:0	Baud value	R/W	Bits 7:0 of 13-bit baud value

11.3.4 Control Register 2

The following table shows Control Register 2, which is used to assign values to the configuration inputs available on UART.

Table 11-6. Control Register 2

Bit(s)	Name	R/W	Function
0	BIT8	R/W	Data width setting: BIT8 = 0: 7-bit data BIT8 = 1: 8-bit data
1	PARITY_EN	R/W	Parity is enabled when this bit is set to 1.
2	ODD_N_EVEN	R/W	Parity is set as follows: ODD_N_EVEN = 0: even ODD_N_EVEN = 1: odd
7:3	BAUD_VALUE	R/W	Bits 12:8 of 13-bit baud value.

11.3.5 Control Register 3

The following table shows Control Register 3, which is used to assign values to the configuration inputs available on UART.

Table 11-7. Control Register 3

Bit(s)	Name	R/W	Function
2:0	BAUD_VAL_FRACTION	R/W	When configuration is set to Programmable, this register can be used to set a fractional part of the baud value. The baud value can be set with a precision of 0.125.

The following table lists the fractional part of the baud value.

Table 11-8. Fractional Baud Value Settings

Bit(s)	Extra Precision
000	+0.0
001	+0.125
010	+0.25
011	+0.375
100	+0.5
101	+0.625
110	+0.75
111	+0.875

11.3.6 Status Register

The following table shows Control register 3, which is used to assign values to the configuration inputs available on UART.

Table 11-9. Status Register

Bit(s)	Name	R/W	Function
0	TXRDY	R	When Low, the transmit data buffer/FIFO is not available for additional transmit data.
1	RXRDY	R	When High, data are available in the receive data buffer/FIFO. This bit is cleared by reading the Receive Data register.
2	PARITY_ERR	R	When High, a parity error has occurred during a receive transaction. This bit is cleared by reading the Receive Data register.
3	OVERFLOW	R	When High, a receive overflow occurs. This bit is cleared by reading the Receive Data register.
4	FRAMING_ERR	R	When High, a framing error occurrs during a receive transaction. This bit is cleared by reading the Receive Data register.
7:5	—	—	Unused

Note: When RX_FIFO is enabled, `PARITY_ERR` is asserted when a parity error occurs, but de-asserted before UART receives the next byte. You need to monitor the `UART_PARITY_ERR` signal (for example, treat it as an interrupt signal), as it is non-persistent when `RX_FIFO = 1`. Similarly, when RX_FIFO is enabled, `UART_FRAMING_ERR` is asserted when a framing error occurs, but de-asserted before UART receives the next byte. It must be treated in the same way as an interrupt signal.

12. Watchdog

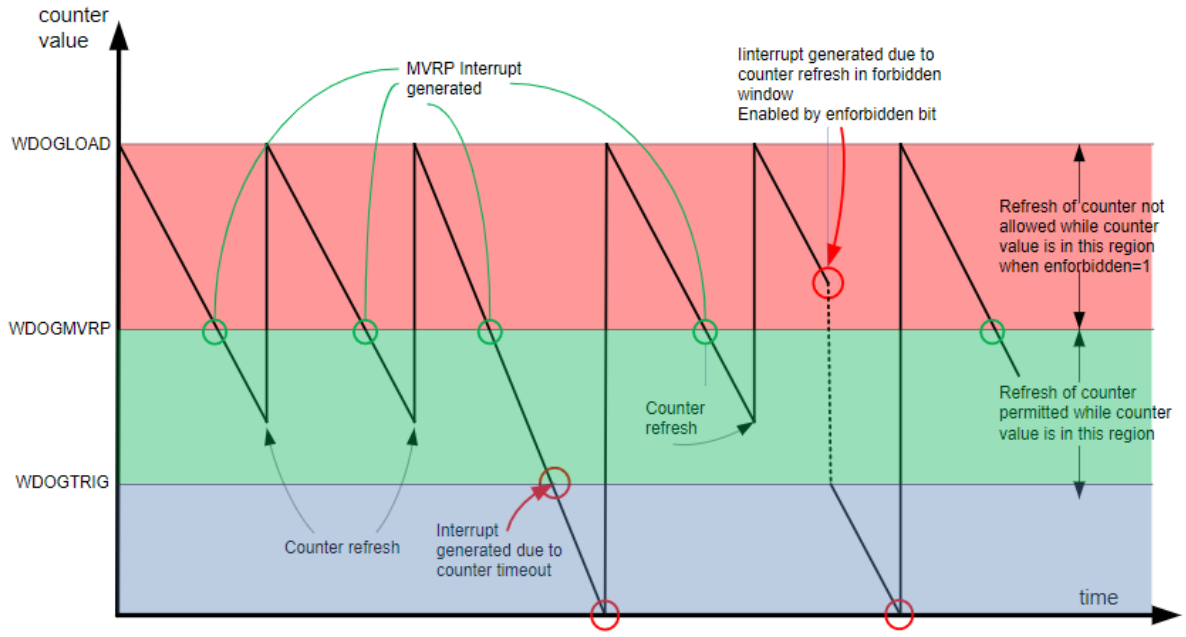
This section provides information on the Watchdog module used in the MIV_ESS core.

12.1 Description

The Watchdog is a hardware timer that generates a reset for the system automatically if the software does not periodically update or refresh the Timer Countdown register. The timer is not allowed to be updated within the forbidden window or if it has already triggered and timed out.

The following sawtooth timing diagram represents the Watchdog's behavior. The red area represents the 'forbidden window', the green area represents 'refresh legal' window, and the blue area represents 'time-out' window.

Figure 12-1. Watchdog – Sawtooth Waveform



The timer countdown begins from a pre-set value. The timer decrements by 1 every time the 8-bit prescaler value/ register reaches the end of its count. With the default configuration, the timer decrements by 1 every 256 system clock cycles. The timer initially decrements through the forbidden window, within this window the timer cannot be legally refreshed. As the timer decrements, it reaches a threshold called the 'Maximum Value up to which Refresh is Permitted' (MVRP). When it is below this threshold, the MVRP Interrupt is asserted, flagging the system that the timer is now within a 'window' where refreshes to the timer are legal and permitted.

The Watchdog firmware must update the timer in this window. If not updated, the timer continues to decrement until it reaches a trigger threshold where refreshes are no longer allowed, and the timer has timed out. This causes the WDOG interrupt to assert, and the timer counts down to a reset. The WDOG Interrupt is asserted, and the timer continues counting down to reset. This window is much smaller than the other two and it is meant to allow the Hart to prepare for a reset recovery before it is reset by the Watchdog.

12.1.1 Features

The Watchdog has the following features implemented.

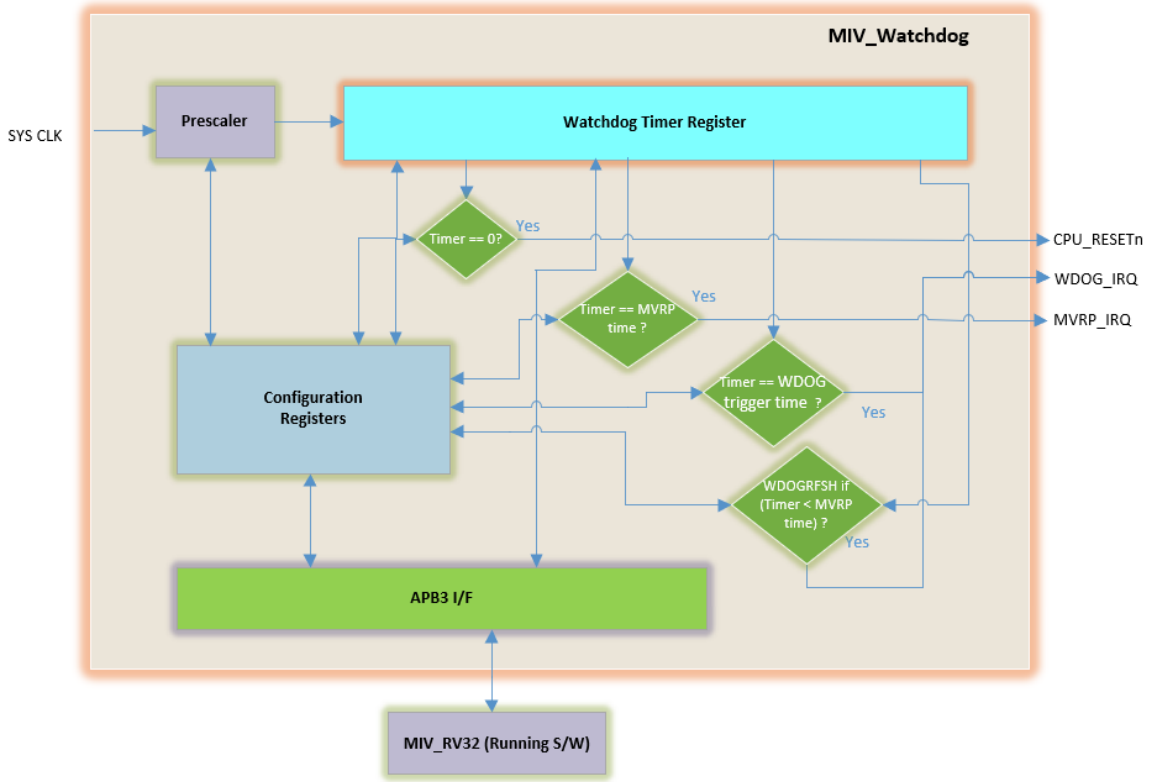
- An APB3 interface for accessing the Watchdog registers.
- A 23-bit register with initial values for Watchdog runtime.
- A 23-bit register with initial values for MVRP runtime threshold.
- A 12-bit register with initial values for the Watchdog's trigger time.

- An 8-bit Prescaler to divide the Watchdog countdown.
- A Control register, which allows enabling the interrupts and the forbidden window.
- The Watchdog generates the following two interrupt signals.
  - An MVRP interrupt that gets asserted when Watchdog countdown leaves the forbidden window. Refresh must be permitted after this interrupt.
  - A WDOG interrupt is a time-out interrupt that gets asserted when the Watchdog count value falls below the trigger time value. Watchdog initiates a countdown to reset, refresh is no longer permitted once, this interrupt is asserted.
- When refreshed, the Watchdog's registers must be updated with pre-set or default values. Watchdog refresh conditions must be as follows:
  - The refresh is only allowed if timer count leaves the forbidden window, otherwise the time-out interrupt is asserted.
  - The refresh must occur before the countdown reaches zero.
  - The refresh must occur before the Watchdog has triggered.
  - Once a refresh occurs, the Watchdog must be stopped from triggering or tripping.
- When the timer countdown reaches zero, the timer reaches expiration, and a reset request must be sent to the Hart.

12.1.2 Block Diagram

The following figure shows the block diagram of the Watchdog.

Figure 12-2. Watchdog - Block Diagram



The Watchdog operates within a set of parameters that determine the thresholds for the forbidden window, legal refresh window, and time-out window. These values are determined by key registers, such as the Watchdog runtime for timer start value, the MVRP runtime (threshold) for marking the forbidden window boundary, and the Watchdog trigger time value for timer time-out. These registers are initialized with default values and can be configured through the software.

Prescaler

The Prescaler value is the amount of clock cycles that must occur before a single Prescaler tick.

- The Prescaler is fixed and counts 256 clock ticks before the prescaler tick occurs.
- The prescaler tick is used for the Watchdog Timer register.

Watchdog Timer Register

This component performs the Watchdog countdown to zero. Depending on the count, you can instruct the Watchdog to refresh the count if the countdown value is not in the forbidden window or already below the Watchdog trigger runtime value.

Counter Operation

- The counter decrements the count by 1 on each prescale tick (every 256 clock cycles).
- The Watchdog count down operates whilst the WDOG_HALT input is deasserted.

Watchdog Interrupts and Resets

There are two output interrupt signals that are generated by the Watchdog.

- WDOG_MVRP_IRQ: This interrupt is asserted when the 'Timer' countdown register leaves the MVRP window. The assertion of this interrupt must signal software to update the 'Timer' register to prevent the countdown from decrementing further.
- WDOG_IRQ: This interrupt is asserted when the 'Timer' times out, it is no longer in the region where refreshes are permitted. The Watchdog Triggers and timer begins counting down towards a reset. This interrupt forewarns of a pending reset and allows the Hart some time to save data and prepare for a reset.

There is a single output reset signal generated by the Watchdog

- `CPU_RESETN`: This is the reset request signal for the Hart. This asserts when the Watchdog has reached the reset condition.

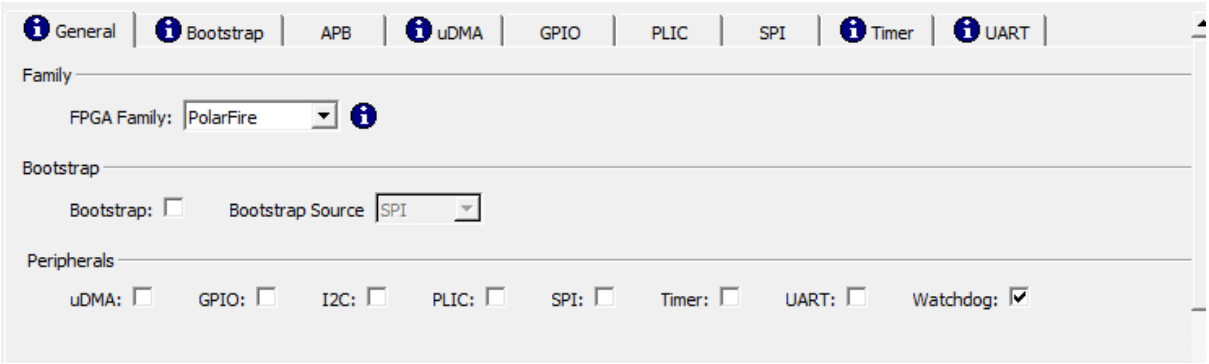
12.2 Interface

This section describes the Watchdog parameters and port lists.

12.2.1 Parameters

You must enable the Watchdog from **MIV_ESS GUI > General** tab. The Watchdog has no further parameters in the GUI. The initial values for registers are fixed and can only be configured in the software. The following figure shows that the Watchdog is enabled.

Figure 12-3. Watchdog Enable



The following table lists the Watchdog parameters available in MIV_ESS.

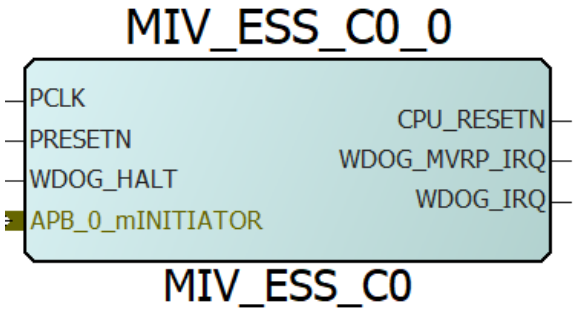
Table 12-1. Watchdog Enable Parameter

Configurator Parameter	Parameter Name	Valid Values	Default Value	Description
Watchdog (Enable)	WDT_EN	0 or 1	1	0: Disable the Watchdog in the MIV_ESS. 1: Enable the Watchdog in the MIV_ESS.

12.2.2 Port List

The following figure shows the Watchdog inputs and outputs in the MIV_ESS instance.

Figure 12-4. MIV_ESS: Watchdog Module



The following table shows only the unique inputs and outputs of the Watchdog.

Table 12-2. Watchdog Port Signals

Ports	Width	Direction	Description
WDOG_HALT	1	Input	An input, which halts the Watchdog countdown. During processor debugging, this input must be asserted (active High) to prevent the Watchdog resetting the whilst the processor is halted by the debugger. For normal Watchdog operation, this input must be deasserted. Versions later than MIV_RV32 v3.0 have a dedicated debug mode output, which can be connected directly to this input so the Watchdog is automatically halted and resumed by the debugger.
CPU_RESETN	1	Output	The Hart reset request signal output by the Watchdog.
WDOG_MVRP_IRQ	1	Output	Maximum value up to which a refresh is permitted, Interrupt.
WDOG_IRQ	1	Output	This Watchdog interrupt is asserted when the Watchdog times out.

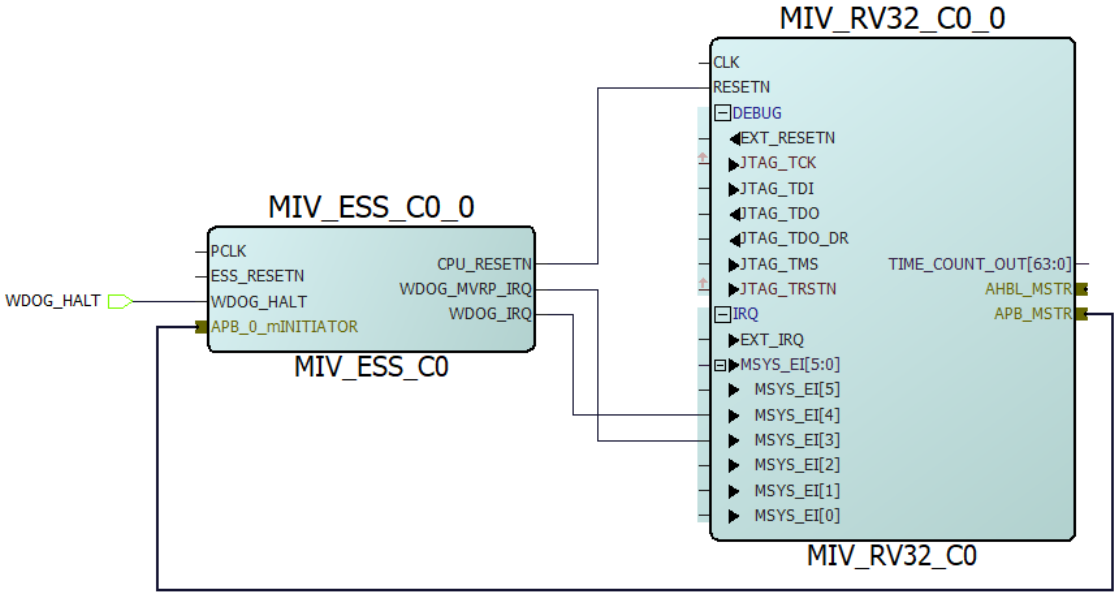
12.3 Programming

This section describes Watchdog registers.

12.3.1 How to Use the Watchdog

The enabled Watchdog can be connected to MIV_RV32 directly. The following figure shows the sample SmartDesign connections of the Watchdog module to MIV_ESS.

Figure 12-5. MIV_ESS: Watchdog Module - Connections



The Watchdog's interrupts are fed into the MIV_RV32 core as external interrupts WDOG_IRQ as MSYS_EI[4] and WDOG_MVRP_IRQ as MSYS_EI[3].

12.3.2 Details of Operation

The Watchdog module can be added to the MIV_ESS instance, but the timer does not initiate the countdown until the following conditions are true.

- The WDOG_HALT input is deasserted.
- The system is not reset.

The Watchdog has a basic default mode of operation. It relies on software for control, so the timer gets updated and refreshed through the Interrupt Service Routines (ISRs).

Hardware Function

The Watchdog operates on default values unless the registers are updated in the software. Assuming Watchdog is enabled, it operates as follows in default conditions.

1. If the Watchdog is enabled, the Prescaler increments by 1 on every SYS CLK tick, 256 times.
2. When the prescaler reaches the end of its count. The Watchdog countdown register (WDOGTIME) decrements by 1 from its initial value of 16,777,200.
3. Assuming the forbidden window is enabled. When timer value falls to the value stored in the MVRP threshold register (WDOGMSVP), which holds the value of 10,000,000 initially, it leaves the forbidden window and `WDOG_MVRP_IRQ` is asserted.
4. The Watchdog is then currently found in the time window where refreshes to the timer are permitted. This is when software must write to the 'WDOGRFSH' register to update the timer count.
5. If not, refresh service routine is established, the timer register continues to decrement, when timer is equal to the value of Watchdog Trigger Time register (WDOGTRIG), the Watchdog times out and `WDOG_IRQ` is

asserted. Refreshes are no longer permitted at this point. If there is still no software response, the 'Timer' continues the countdown.

6. If the timer continues the countdown and reaches the value 0, the Hart reset request is generated on the CPU_RESETN output. The timer resets and the CPU is given a chance to reload the Watchdog registers.

Software Function

The Watchdog relies on configuration through software. The operation relies on how the internal registers are configured in the software. Software support allows Watchdog to have:

- A 'service' or a 'refresh' routine to be setup in software that periodically updates the Watchdog's Timer register. The Timer register is updated with the default pre-set values on reset, the timer continuously decremented by -1 until it reaches a time-out. To prevents this, update the timer register.
- Control over Watchdog features that can either be enabled or disabled. The software allows for enabling and disabling of the following:
  - Forbidden window: Refreshes are not permitted within this region.
  - WDOG_MVRP_IRQ: Asserts when timer count leaves forbidden window threshold.
  - WDOG_IRQ: Asserts when timer count reaches time-out value.
- Force reset
- A status check of the Watchdog to see if it is in the forbidden window or not
- A way to clear the interrupts
- For reads and writes to registers used to update the values of — WDOGMSVP, WDOGTRIG, and WDOGFORCE.

12.3.3 Watchdog Register Map

The Watchdog contains read/write registers, which can be accessed over APB I/F.

You can access the seven control registers to read or write to. Reading these registers provide information about the status of the Watchdog or timer value. Writing to these registers allows Watchdog to perform operations, such as 'refresh' of timer to prevent a time-out, or manually force a reset.

The Watchdog module's register addresses are determined with respect to the MIV_ESS base address. The MIV_ESS base address [MIV_ESS_BASE] is configurable. The Watchdog module base address is [MIV_ESS_BASE + 0x900_0000]. The address offset of each Watchdog register is given in the following table.

```
Register address = MIV_ESS_BASE + 0x900_0000 + Register Address Offset
```

Table 12-3. Watchdog Register Map

Register Name	Address Offset	R/W	Reset Value	Description
WDOGRFSH	0x00	R/W	0x0	**Refresh register** This register is used for refreshing the Watchdog counter value when a 'Refresh' is allowed.
WDOGCNTL	0x04	R/W	0x2	**Control Register** This register is used for enabling and disabling the Watchdog interrupts and the forbidden window. By default, the `WDOG_IRQ` is enabled, and the forbidden window and `WDOG_MVRP_IRQ` are disabled.
WDOGSTAT	0x08	R/W	0x0	**Status Register** This register is used to check, if the interrupts are tripped or clearing the interrupts.

.....continued

Register Name	Address Offset	R/W	Reset Value	Description
WDOGTIME	0x0C	R/W	16,777,200	**Watchdog Runtime Register** The register stores the value from which the Watchdog initiates its countdown from. Register cannot be written to if the Watchdog is locked.
WDOGMSVP	0x10	R/W	10,000,000	**Watchdog MVRP Runtime Threshold Register** The Maximum Value up to which a Refresh is permitted. If the forbidden window is enabled and Watchdog count falls below the value determined by this register, `WDOG_MVRP_IRQ` is asserted and Watchdog is legally allowed to 'Refresh'. The register cannot be written to if the Watchdog is locked.
WDOGTRIG	0x14	R/W	1,0000	**Watchdog trigger time value Register** If the Watchdog countdown falls below the value determined by this register, the Watchdog times out and `WDOG_IRQ` is asserted. The register cannot be 'written to' if the Watchdog is locked.
WDOGFORCE	0x18	W	0x0	**Watchdog Force Reset Register** This register can be used to force a Watchdog reset.

12.3.3.1 Watchdog Refresh Register

Table 12-4. Watchdog Refresh Register

Bit Number	Bit Name	R/W	Reset Value	Description
31:0	WDOGRFSH	R/W	0x0	To refresh the Watchdog, special value needs to be written: 0xdeadc0de. The refresh can only be performed if the Watchdog has not yet triggered and if the timer is not within the forbidden window. The refresh in forbidden window is only allowed, if the CPU is Debug mode where format is HIGH. The Watchdog becomes locked when writing to this register. Reading the register shows the current 24-bit value of Timer Countdown register.

12.3.3.2 Watchdog Control Register

Table 12-5. Watchdog Control Register

Bit Number	Bit Name	R/W	Reset Value	Description
31:5	Reserved	—	0x0	Reserved

.....continued

Bit Number	Bit Name	R/W	Reset Value	Description
4	next_enforbidden	R/W	0x0	Write a value to bit to enable/disable the forbidden window: 1 = Enable the forbidden window 0 = Disable the forbidden window Reading the bit indicates whether the forbidden window is enabled: 1 = Forbidden window is enabled 0 = Forbidden window is disabled
3:2	Reserved	—	0x0	Reserved
1	next_intent_wdog	R/W	0x1	Write a value to bit to enable/disable `WDOG_IRQ` 1 = Enable `WDOG_IRQ` 0 = Disable `WDOG_IRQ` Reading the bit indicates whether `WDOG_IRQ` is enabled 1 = `WDOG_IRQ` is enabled 0 = `WDOG_IRQ` is disabled
0	next_intent_msvp	R/W	0x0	Write a value to bit to enable/disable `MVRP_IRQ`: 1 = Enable `MVRP_IRQ` 0 = Disable `MVRP_IRQ` Reading the bit indicates whether `MVRP_IRQ` is enabled 1 = `MVRP_IRQ` is enabled 0 = `MVRP_IRQ` is disabled

12.3.3.3 Watchdog Status Register

The following table list the Watch Status registers.

Table 12-6. Watchdog Status Register

Bit Number	Bit Name	R/W	Reset Value	Description
31:5	Reserved	—	0x0	Reserved
4	wdoglocked	R	0x0	Reading the bit indicates whether the Watchdog is locked.
3	triggered	R	0x0	Reading the bit indicates whether the Watchdog has timed out.
2	forbidden	R	0x0	Reading the bit indicates whether the timer is currently within the forbidden window.

.....continued

Bit Number	Bit Name	R/W	Reset Value	Description
1	wdog_tripped	R/W	0x0	Write a value to this bit to clear the time-out interrupt 1 = Clear format Reading the bit indicates whether the Watchdog time-out interrupt is currently asserted 1 = The `WDOG_IRQ` is currently asserted 0 = The `WDOG_IRQ` is not currently asserted
0	msvp_tripped	R/W	0x0	Write a value to bit to clear the maximum value up to which a Refresh is Permitted interrupt. 1 = Clear `WDOG_MVRP_IRQ` Reading the bit indicates whether the Maximum value up to which a Refresh is Permitted interrupt is asserted 1 = The `WDOG_MVRP_IRQ` is currently asserted 0 = The `WDOG_MVRP_IRQ` is not currently asserted

12.3.3.4 Watchdog Runtime Register

The following table lists the Watchdog Runtime registers.

Table 12-7. Watchdog Runtime Register

Bit Number	Bit Name	R/W	Reset Value	Description
31:24	Reserved	—	0x0	Reserved
23:0	wdogvalue	R/W	16,777,200	If the Watchdog is not locked, write a 24-bit value to the register to serve as the next Watchdog runtime value. Read the 24-bit value of the register that is used as the Watchdog runtime value.

12.3.3.5 Watchdog MVRP Runtime Register

The following table lists the MVRP runtime registers.

Table 12-8. Watchdog MVRP Register

Bit Number	Bit Name	R/W	Reset Value	Description
31:24	Reserved	—	0x0	Reserved
23:0	wdogmsvp	R/W	10,000,000	If the Watchdog is not locked, write a 24-bit value to the register to serve as the next MVRP runtime value. Read the 24-bit value of the register that is used as the MVRP runtime threshold value.

12.3.3.6 Watchdog Trigger Timeout Register

The following table lists the Trigger Time-out registers.

Table 12-9. Watchdog Trigger Timeout Register

Bit Number	Bit Name	R/W	Reset Value	Description
31:24	Reserved	—	0x0	Reserved
23:0	wdogrst	R/W	1,000	If the Watchdog is not locked, write a 24-bit value to the register to serve as the next Watchdog Trigger Timeout value. Read the 24-bit value of the register that is used as the Watchdog Trigger Timeout value.

12.3.3.7 Watchdog Force Reset Register

The following table lists the Watchdog Force Reset registers.

Table 12-10. Watchdog Force Reset Register

Bit Number	Bit Name	R/W	Reset Value	Description
31:0	WDOGFORCE	W	0x0	Writing any value to register when the Watchdog has not timed out will result in Watchdog time-out. The time-out interrupt `WDOG_IRQ` will be set to HIGH and Watchdog Timer countdown register updated with Watchdog Trigger Timeout register value. If the Watchdog has timed out, a special 16-bit value must be written to the register to force a reset on format. 0xDEAD Then the Watchdog countdown is reset/updated with the top Watchdog Runtime register value.

13. Tool Flow

13.1 License

No license required.

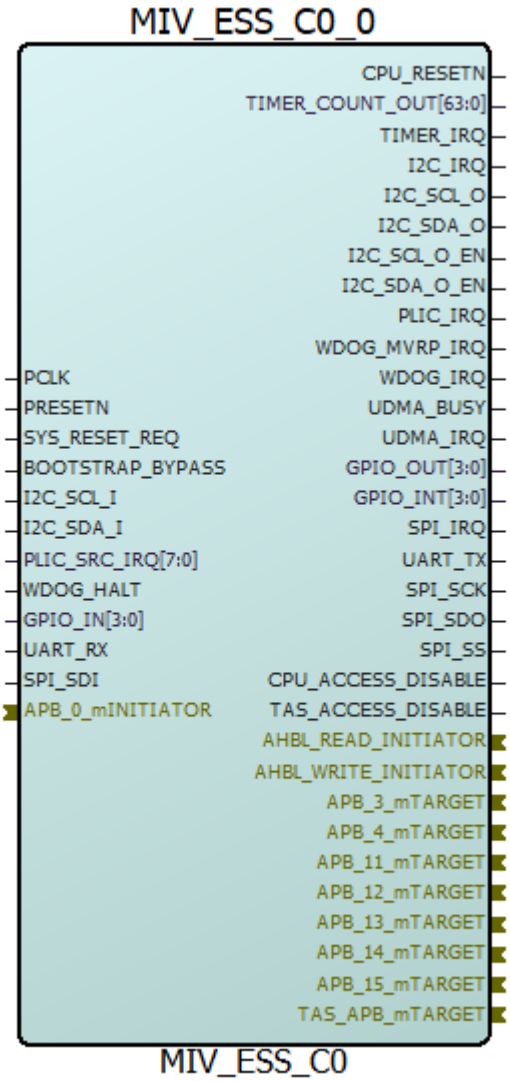
13.2 RTL

Complete RTL source code is provided.

13.3 SmartDesign

MIV_ESS is available in the *Processor* section of the IP Catalog in the Libero SoC design environment. The core can be used with the Libero v12.1 onwards. The following figure illustrates the available ports on the core with SPI Bootstrap and all the peripheral modules enabled.

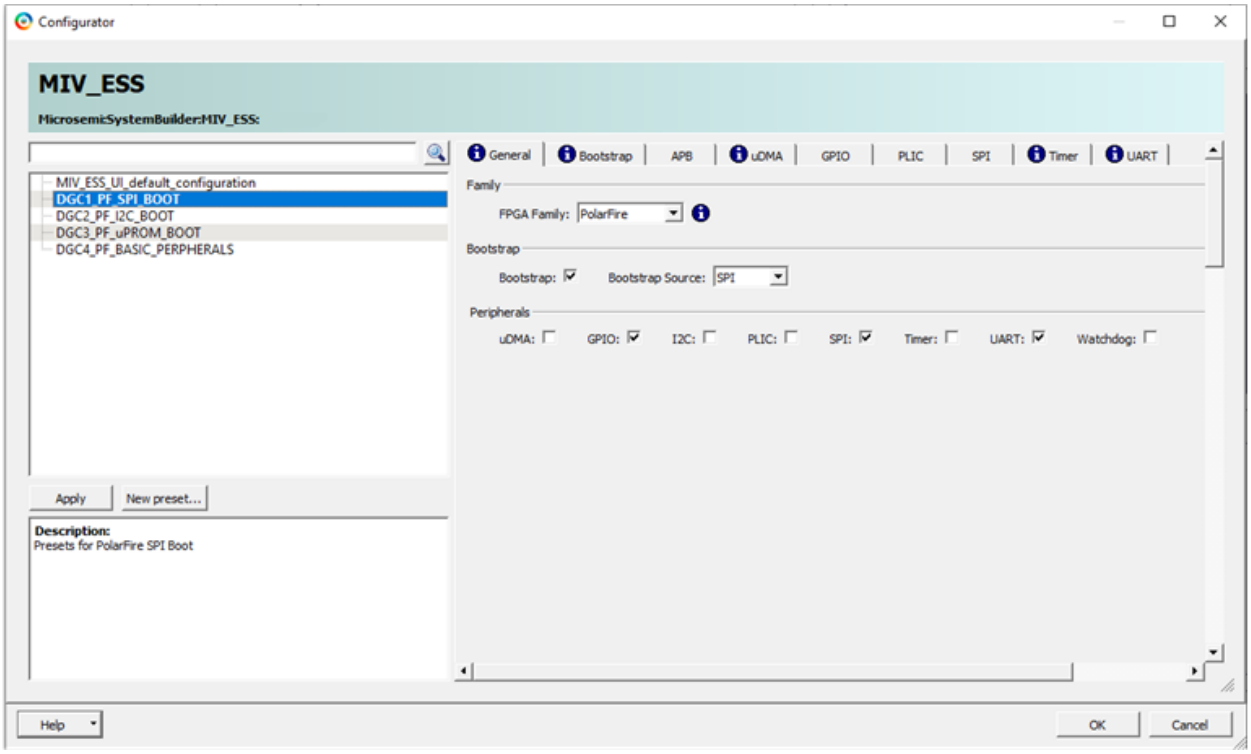
Figure 13-1. MIV_ESS Instance



13.4 Configuring the MIV_ESS

The top-level **General** tab allows you to select FPGA Family and enable Bootstrap and peripheral modules, as required. It also has three pre-set configurations, which follow the Bootstrap configurations provided in the accompanying *MIV_ESS Design Guide*. A fourth pre-set configuration provides the MIV_ESS basic peripheral settings for a design outlined in the *MIV_RV32 Quick Start Design Guide*.

Figure 13-2. Configurator General Tab



For information on configurator settings, see 2. MIV_ESS Architecture section.

13.5 Simulation

This code does not provide any testbench. MIV_ESS RTL can be simulated as part of a standard MIV_RV32 system with a Libero generated HDL testbench. A hex file generated within SoftConsole, with the first line removed, can be imported into an LSRAM and MIV_RV32 booted from this code. Core transactions can be simulated as part of the MIV_RV32 processor system.

13.6 Synthesis in Libero

To run synthesis on the core, set the SmartDesign sheet as the design root and click **Synthesize** in the Libero SoC.

13.7 Place-and-Route in Libero

After the design is synthesized, run the compilation and the Place-and-Route tools. Click the **Layout** icon in Libero SoC to invoke the designer.

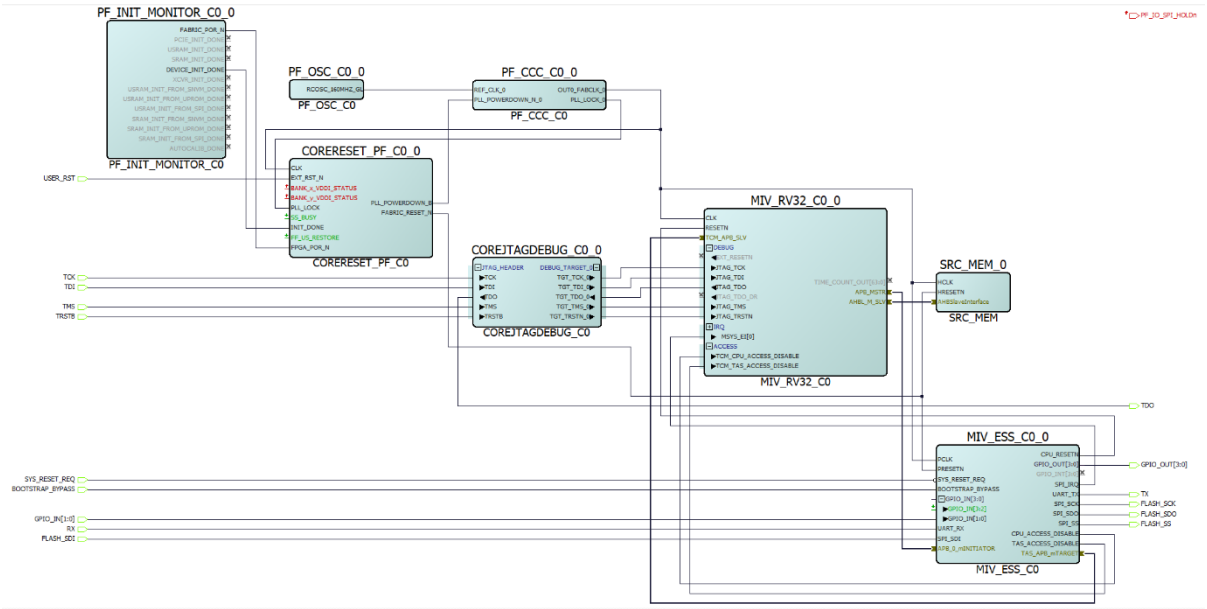
14. System Integration

This section describes system integration examples of MIV_ESS.

14.1 MIV_ESS Bootstrap Example

The following figure shows MIV_ESS in the SPI Bootstrap configuration.

Figure 14-1. DGC1 - SPI Bootstrap Design

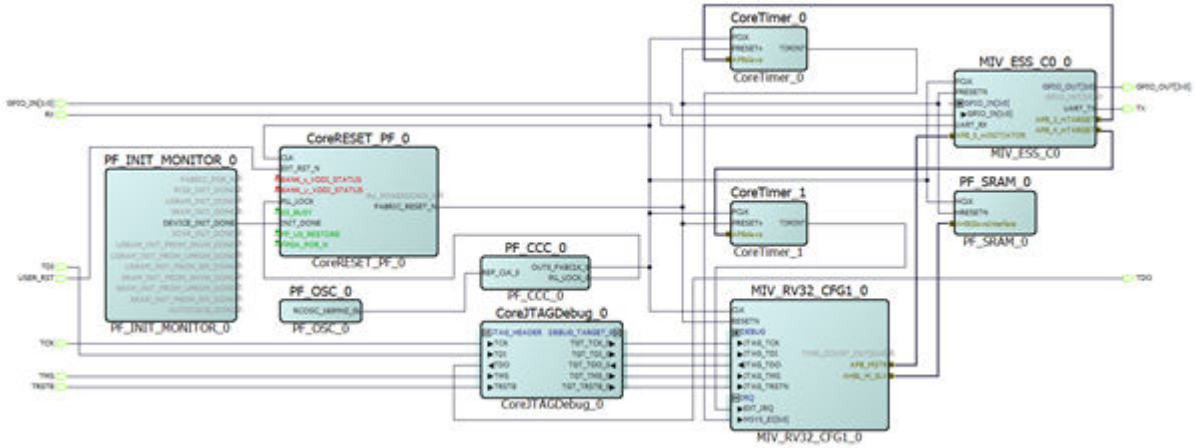


For more information on design configuration, see the *MIV_ESS Design Guide*.

14.2 MIV_ESS Peripheral Example

The following figure illustrates the standard reference design from the *MIV_RV32 Quick Start Design Guide* implemented with MIV_ESS.

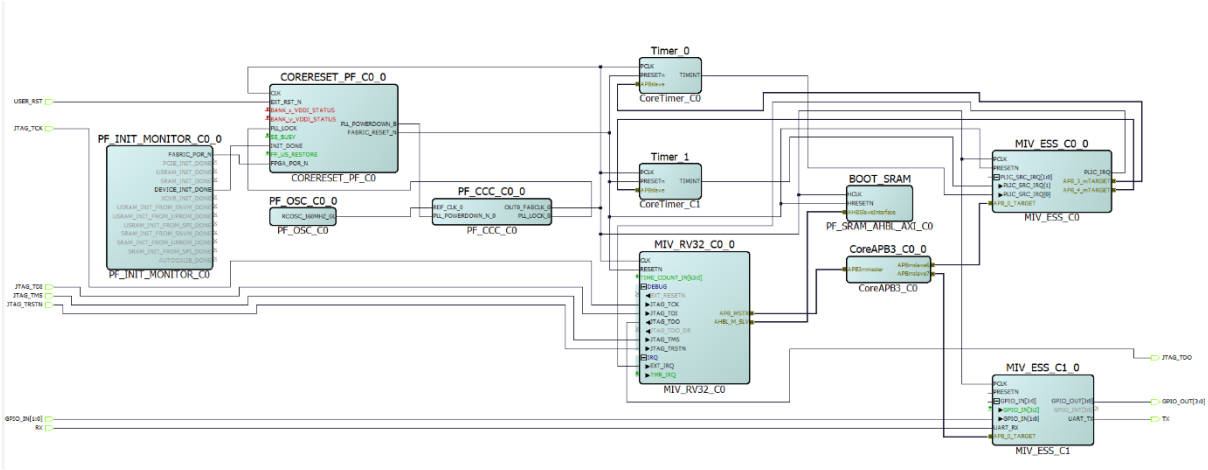
Figure 14-2. MIV_RV32 Design using MIV_ESS



14.3 Multiple MIV_ESS Example

Multiple MIV_ESS cores can be used in a design. CoreAPB3_C0 is configured, so the 32 bits are driven by the MIV_RV32 APB mapping MIV_ESS_C0 to an address at 0x6000_0000 and MIV_ESS_C1 to 0x7000_0000 as shown in the following figure.

Figure 14-3. Dual MIV_ESS Design



15. SoftConsole

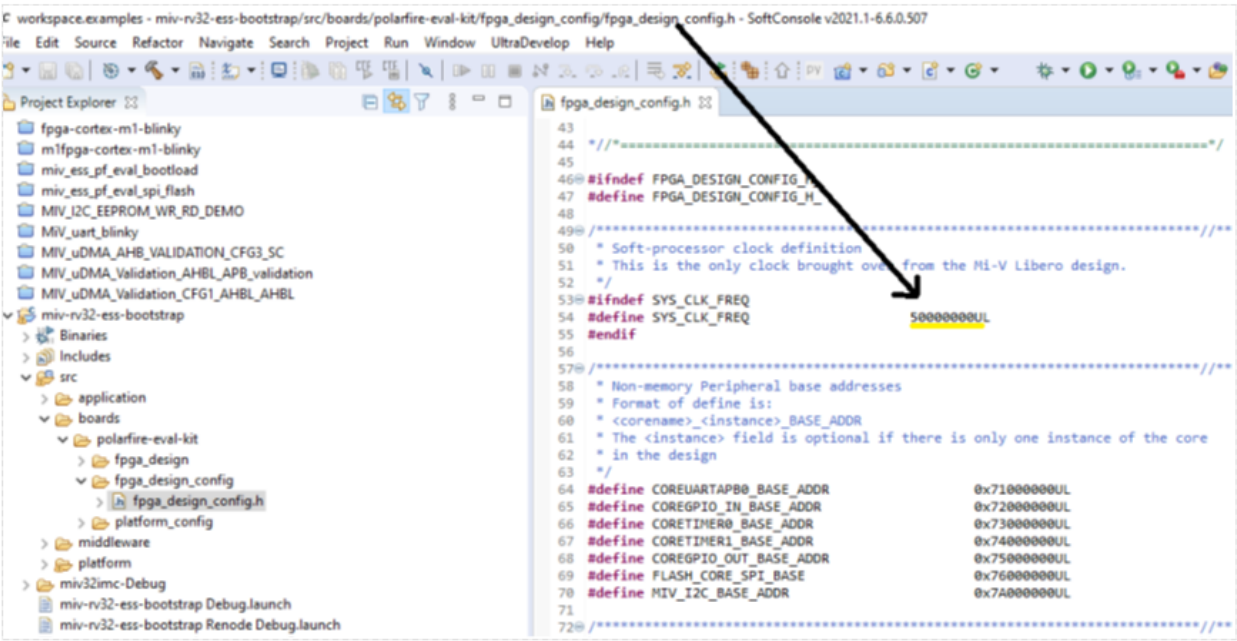
For information on setting up SoftConsole for the MIV_ESS core, see *MIV_ESS Design Guide* from the Help menu in the configurator, or if the core is selected in the Libero catalog.

15.1 Setting the System Clock Frequency and Peripheral Base Addresses

If the UART is used, the system clock frequency is provided to the software and this is done in the `fpga_design_config.h` file by changing the `#define SYS_CLK_FREQ` to the clock frequency.

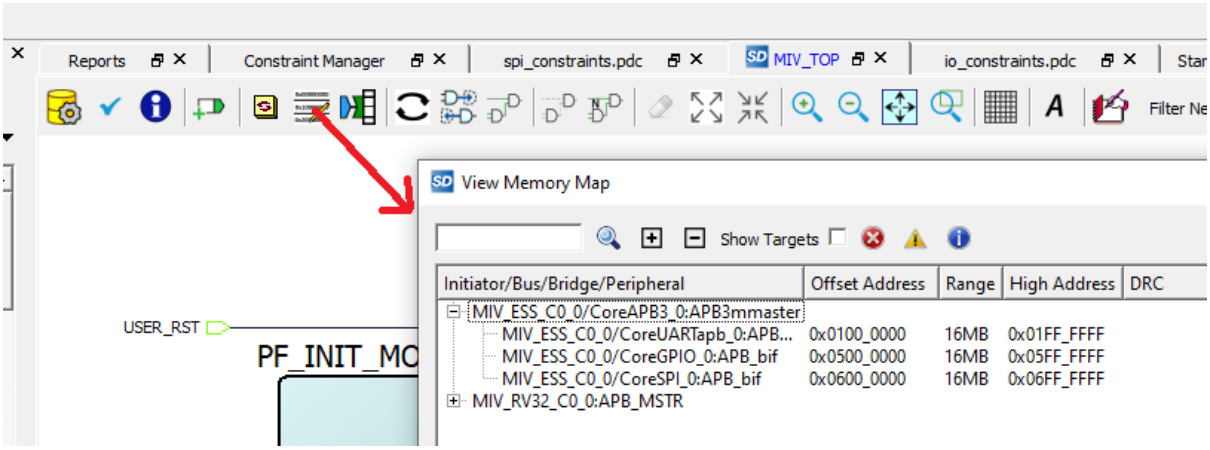
Note: This value must be in Hertz.

Figure 15-1. Setting System Clock Frequency and Peripheral Addresses



The `fpga_design_config.h` file sets the base address for peripherals. The base address of a peripheral can be found in the project memory map generated by Libero.

Figure 15-2. Libero Memory Map Window



Note: Some of the Libero versions do not support the memory map feature as shown in the preceding figure.

The peripheral module address [27:0] in the `fpga_design_config.h` file must match the address in Libero for the peripheral to function correctly. These upper nibble [31:28] is determined by the APB Initiator connected to MIV_ESS.

In the MIV_RV32, the default APB Initiator address is 0x7000_0000, which corresponds to the address shown in the preceding figure.

16. Revision History

Revision	Date	Description
A	01/2022	Initial Revision

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

Trademarks

ISBN: 978-1-5224-9694-6

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820