

MIV_RV32 Migration Guide

Introduction

This document describes the Libero[®] SoC design migration process for Mi-V Soft Processors. The legacy CoreRISCV_AXI4, MIV_RV32IMA_L1_AHB, MIV_RV32IMA_L1_AXI, and MIV_RV32IMAF_L1_AHB soft processor cores are to be replaced with a single highly-configurable MIV_RV32 soft processor core. The objective of this document is to ease the customer Hardware (HW) and Firmware (FW) migration process to the MIV_RV32 platform.

In this document, the <code>CoreRISCV_AXI4</code>, <code>MIV_RV32IMA_L1_AHB</code>, <code>MIV_RV32IMA_L1_AXI</code>, and <code>MIV_RV32IMAF_L1_AHB</code> soft processor cores are collectively referred to as <code>MIV_Legacy</code> cores. The <code>MIV_RV32IMC</code> v2.1.100 and <code>MIV_RV32</code> v3.0.100 or greater are collectively referred to as <code>MIV_RV32</code>, unless otherwise stated.

Table of Contents

Intr	oductic	on	1
1.	Rease	ons to Migrate	3
	1.1.	MIV RV32 Core	3
	1.2.	MIV_RV32 HAL	3
2.	Migra	ting Hardware Configurations	4
	2.1.	Peripherals Connected to the MEM Interface	4
	2.2.	Peripherals Connected to the MMIO Interface	
	2.3.	Mirrored Master Interfaces	9
	2.4.	Interfaces and Memory Maps	11
	2.5.	System Time	28
	2.6.	Debug	31
	2.7.	ECC	
	2.8.	Interrupts	
	2.9.	RISC-V Extensions	37
3.	Migra	ting Software Projects	
	3.1.	Prerequisites	41
	3.2.	Recommended Migration Process	41
	3.3.	Example of Recommended Migration Process	41
	3.4.	Updating the MIV_RV32 HAL	45
	3.5.	Defining the Core to the HAL	47
	3.6.	Interrupts	
	3.7.	MIV_RV32 Extensions	51
	3.8.	Maintaining Performance in Code Implementations From MIV_Legacy	
4.	Revis	ion History	
The	Micro	chip Website	57
		hange Notification Service	
		ů –	
Cus	stomer	Support	57
Mic	rochip	Devices Code Protection Feature	57
Leg	jal Noti	се	57
Tra	demarl	KS	
		anagement System	
Wo	rldwide	Sales and Service	59

1. Reasons to Migrate

The following sections explain why an existing Libero SoC and accompanying Software design should be migrated to ${\tt MIV_RV32}.$

1.1 MIV_RV32 Core

The <code>CoreRISCV_AXI4</code> core is no longer recommended for new designs. The <code>MIV_RV32 IMA_L1_AHB/MIV_RV32</code> <code>IMA_L1_AHB/MIV_RV32 IMAF_L1_AHB</code> Mi-V cores are minimally configurable, and the unused features are left in place during post Synthesis. If the requirement is low resource and medium performance without the need for cache, the <code>MIV_RV32</code> core should be used. Where cache is required, the <code>MIV_Legacy</code> core should be retained. The <code>MIV_RV32</code> will be enhanced over time to supersede the <code>MIV_Legacy</code> cores.

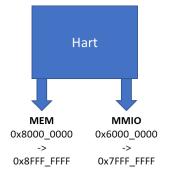
1.2 MIV_RV32 HAL

MIV_RV32 HAL v3.0 or greater contains bug fixes and adds support for the MIV_RV32 core.

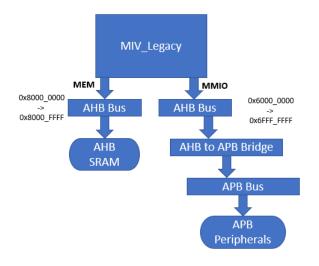
2. Migrating Hardware Configurations

Migrating designs from a MIV_Legacy configuration to a MIV_RV32 configuration is relatively straight forward. The MIV_Legacy cores have a fixed memory map based on their Hardware Architecture. They use the MEM interface for cached instructions and data, and the MMIO interface for peripherals and non-cached memory.

The following figure shows the fixed memory map of the ${\tt MIV_Legacy}$ cores.



The following figure shows a typical system.



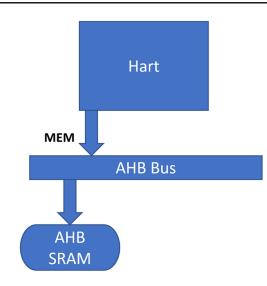
2.1 Peripherals Connected to the MEM Interface

The primary function of the MEM interface in a MIV_Legacy core is to allow cached access to software code and data. It can be connected to SRAM embedded within the FPGA or to discrete DDR memory devices. The MEM interface has a restricted address range on the MIV_Legacy cores from 0x8000_0000 to 0x8FFF_FFF. The MIV_RV32 core does not feature a cache, instead it features a Tightly Coupled Memory (TCM). Note: The TCM must be used in preference to SRAM in systems where the processor requires faster memory

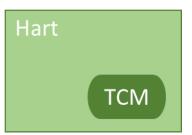
accesses. The address range for the MEM interface on the MIV_RV32 core is much less restrictive and is described in the following sections.

2.1.1 SRAM

The typical use of the MEM interface is interfacing a memory. When using SRAM, the configuration can be an AHB or an AXI as shown in the following figure.



The TCM in the MIV_RV32 core operates in the same way as external SRAM, except with lower latency due to it being internally coupled to the core.



2.1.1.1 Configuring the TCM

Review the maximum size of TCM available in the relevant *MIV_RV32 Handbook*. The TCM must have a start address greater than 0x1000_0000.

Note: The TCM on the MIV_R32 is limited to a maximum size of 256 Kbytes in v3.0.x.

The TCM is enabled from the **Configuration** tab of the **Configuration** window.

Configurator	-		×
Mi-V RV32 Configurator			
Microsemi:MiV:MIV_RV32:3.0.100			
Configuration Memory Map			
Extension Options			
RISC-V Extensions: IMC 💌 🐧 Multipler: Fabric 💌 🐧			
Interface Options			
AHB Master: AHBLite 💌 AHB Mirrored I/F: 🗆 🚯			
APB Master: APB3 APB Mirrored I/F:			
AXI Master: None 💌 AXI Mirrored I/F: Г 🚯			
Reset Vector Address			
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0			
- BootROM Options			
BootROM: 🗆 🚯 Reconfigure BootROM: 🗖 🚯			
Tightly Coupled Memory (TCM) Options			
Interrupt Options			
External System IRQs: 0 🗾 🚯			
Vectored Interrupts:			
System Timer Options			
Internal MTIME: 🔽 🕈 MTIME Prescaler: 100			
Internal MTIME IRQ: 🔽 🚯			
Other Options			
Debug: 🗟 🖬 Register Forwarding: 🗆 🚯			
ECC: 🔽 🚯 GPR Registers: 🔽 🚯			
Help	ок	Can	cel

The depth of the TCM, up to the maximum defined TCM size, is calculated from its accessible range in the **Memory Map** tab of the **Configurator** window.

Configurator		-		×
Mi-V RV32 Configurator				
Microsemi:MiV:MIV_RV32:3.0.100				
Configuration Memory Map AHB Master Address				1
Start Address: Upper 16bits (Hex): 0x7000	Lower 16bits (Hex):			
End Address: Upper 16bits (Hex): 0x700f	Lower 16bits (Hex): 0xffff			
APB Master Address				
Start Address: Upper 16bits (Hex): 0x6000	Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x6fff	Lower 16bits (Hex): 0xffff			
AXI Master Address				
Start Address: Upper 16bits (Hex): 0x6000	Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x6fff	Lower 16bits (Hex): 0xffff			
TCM Address		_		
Start Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x3fff			
TCM APB Slave Address				
Start Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0xffff			
BootROM Address				
Source Start Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x0			
Source End Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x3fff		-	
Destination Address: Upper 16bits (Hex): 0x4000	Lower 16bits (Hex): 0x0			
		or 1	0	
Help		ОК	Can	.ei

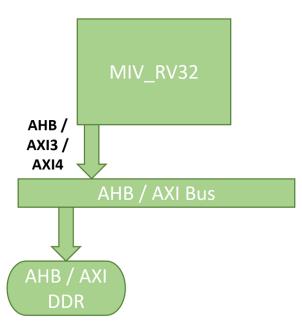
The following table gives examples of TCM address widths and their corresponding memory depth.

Start Address	End Address	Depth	# 32-Bit Words	Kbytes
0x8000_0000	0x8000_03FF	0x400	256	1
0x8000_0000	0x8000_07FF	0x800	512	2
0x8000_0000	0x8000_0FFF	0x1000	1024	4
0x8000_0000	0x8000_1FFF	0x2000	2048	8
0x8000_0000	0x8000_3FFF	0x4000	4096	16
0x8000_0000	0x8000_7FFF	0x8000	8192	32
0x8000_0000	0x8000_FFFF	0x1_0000	16384	64
0x8000_0000	0x8001_FFFF	0x2_0000	32768	128

2.1.2 DDR

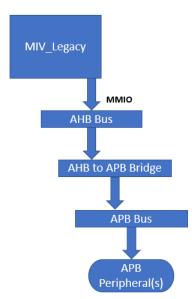
DDR can be used as external memory available to the core. As the core features AHB and AXI3/AXI4 interfaces with no addressing restrictions, except a start address greater than 0x1000_0000, the DDR can be connected to either of these interfaces depending on the slave interface type and the accessible range given in the **Memory** tab.

The MIV_RV32 data interfaces are 32 bits wide. In many instances DDR can require 64-bit data access. In this case, an IP core such as CoreAXI4Interrconnect can be used to provide data width conversion for DDR memory. It should be noted that MIV_RV32 does not feature an L1 cache and as such AXI burst transactions are not available. In this instance, careful consideration should be given before migrating to MIV_RV32 as performance with DDR will be limited.

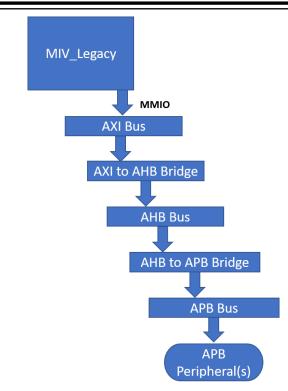


2.2 Peripherals Connected to the MMIO Interface

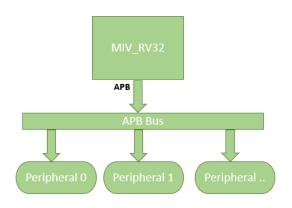
Each peripheral connected to a Mi-V Legacy core has an APB interface and is connected to an APB bus. As the MIV_Legacy core does not have an APB interface, use the APB bus bridges as shown in the following figure.



The following figure shows the bridges required to convert from AXI to APB when using a MIV_Legacy core.

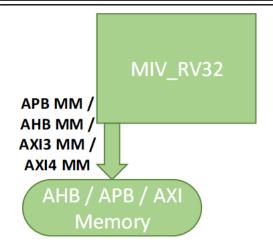


The following figure shows that as $\texttt{MIV}_{\texttt{RV32}}$ features an APB interface, no conversion is required.



2.3 Mirrored Master Interfaces

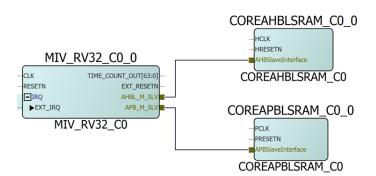
If MIV_RV32 is the only core that is going to access a memory or a peripheral and there are no additional peripherals connected on the interface, the Mirrored Master mode can be selected to allow a direct connection. It improves performance and reduces area as a bus master is not used.



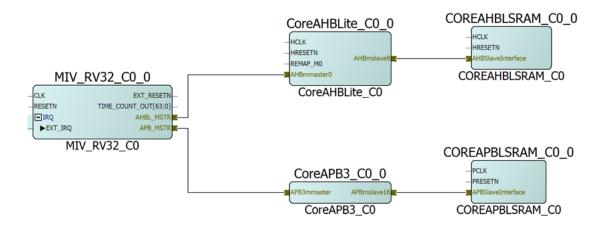
It can be enabled by selecting the Mirrored Master options under the Interface Options in the Configurator window.

Configurator		-		>
Mi-V RV32 Configurator				
icrosemi:MiV:MIV_RV32:3.0.100				
Configuration Memory Map				
Extension Options				_
RISC-V Extensions: IMC 💌 🚯 Multipler: Fabric 💌 🚯				
Interface Options				
AHB Master: AHBLite 💌 AHB Mirrored I/F: 🗹 🚯				
APB Master: APB3 APB Mirrored I/F:				
AXI Master: None 💽 AXI Mirrored I/F: 🔽 🚯				
Reset Vector Address				
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0	0			
BootROM Options				
BootROM: 🗆 🕄 Reconfigure BootROM: 🗖 🚯				
Tightly Coupled Memory (TCM) Options				
TCM: 🗌 🚯 TCM APB Slave (TAS): 🗐 🚯				
External System IRQs: 0 💽 🕄				
System Timer Options				
Internal MTIME: 🔽 🕥 MTIME Prescaler: 100				
Internal MTIME IRQ: 🔽 🚯				
Other Options				
Debug: 🗖 🚯 Register Forwarding: 🗖 🚯				
ECC: 🗆 🚯 GPR Registers: 🗖 🚯				
1				_
Help		ОК	Can	el

For example, the following figure shows APB and AHB SRAMs connected directly to the MIV_RV32 using the Mirrored Master configuration. After place-and-route, the following design has used 4774 logic elements.



The following figure is the equivalent design, without the mirrored masters. After place-and-route, the following design has used 4927 logic elements.



2.4 Interfaces and Memory Maps

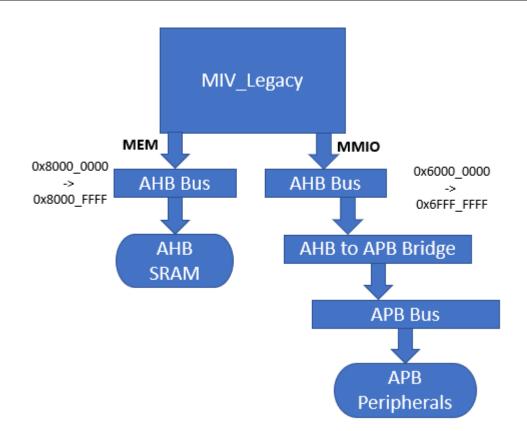
When migrating a design from an MIV_Legacy core to an MIV_RV32 core, there are several ways to configure the updated design to retain the functionality of the original, while taking advantage of the benefits of the MIV RV32 core.

Sample designs are shown in the following sections, featuring an AHB as the primary configuration. The same configurations can be applied to the AXI cores as well.

2.4.1 Sample Design 1 – Base Design

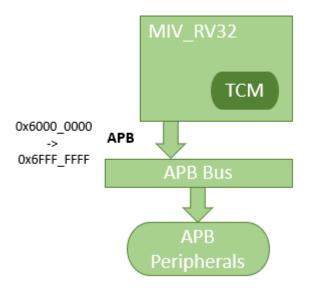
2.4.1.1 MIV_Legacy Configuration

Memory is connected to the MEM interface at 0x8000_0000. Peripherals are connected to the MMIO interface at 0x6000_0000.



2.4.1.2 MIV_RV32 Configuration

TCM is enabled and set to start at 0x8000_0000 and end at 0x8000_FFFF. The APB bus is enabled and configured to start at 0x6000_0000 and end at 0x6FFF_FFF. The configuration settings for this example are shown in the following figures.

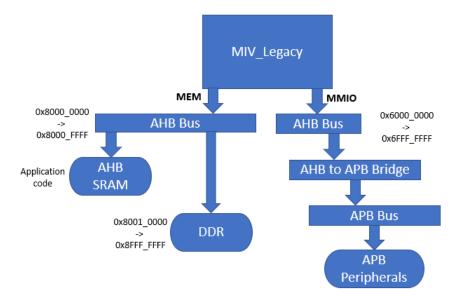


-V RV32 Configurator semtMW:MIV_RV32:3.0.100		
semi:MiV:MIV_RV32:3.0.100		
onfiguration Memory Map ension Options		
RISC-V Extensions: TMC V Multipler: Fabric V		
rface Options		
AHB Master: None 💌 AHB Mirrored I/F: 🗂 🚯		
APB Master: APB3 💽 APB Mirrored I/F: 🗖 🚯		
AXI Master: None 💌 AXI Mirrored I/F: Г		
et Vector Address Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0	7	
tROM Options		
BootROM: BootROM: Coupled Memory (TCM) Options		
TCM: 🗟 🚺 TCM APB Slave (TAS): 🗆 🚯		
rrupt Options		
External System IRQs: 0 C		
Vectored Interrupts:		
Internal MTIME: 🔽 🕈 MTIME Prescaler: 100		
Internal MTIME IRQ: 🔽 🕤		
Internal MTIME IRQ: M 🚺		
Debug: 🗌 🚯 Register Forwarding: 🗖 🚯		
ECC: 🗆 🚯 GPR Registers: 🗆 🚯		
p	OK	Ca
	ОК	Ca
o nfigurator	OK _	_
	- -	_
nfigurator -V RV32 Configurator		_
nfigurator -V RV32 Configurator seemitHWHU RV323.0.100 Configuration Memory Map 6 Master Address	- -	_
nfigurator -V RV32 Configurator semitWVHV/RV3230.100 Configuration Memory Map 8 Master Address Start Address: Upper 18bits (Hex); [0x8000 Lower 18bits (Hex); [0x0		_
nfigurator -V RV32 Configurator seemittWHURV_RV323.0.100 Configuration Memory Map 8 Master Address Start Address: Upper 16bits (Hex); [0x8000 Lower 16bits (Hex); [0x800 End Address: Upper 16bits (Hex); [0x8077 Lower 16bits (Hex); [0x7777		_
nfigurator -V RV32 Configurator asemitHWHV, RV323.30.100 Configuration Memory Map 6 Master Address Start Address Start Address: Upper 16945 (Hex): [0x8000 Lower 16945 (Hex): [0x007 End Address: Upper 16945 (Hex): [0x8777 Lower 16945 (Hex): [0x7777 8 Master Address		_
nfigurator -V RV32 Configurator seemitHWHU RV32.30.100 Configuration Memory Map 8 Master Address Start Address: Upper 18bits (Hex): [0x8000 End Address: Upper 18bits (Hex): [0x8007 End Address: Upper 18bits (Hex): [0x8077 Lower 18bits (Hex): [0x8077 R Matter Address Start Address: Upper 18bits (Hex): [0x6000 Lower 18bits (Hex): [0x6000		_
-V RV32 Configurator -V RV332 Configurator SeemittWittUR RV323.0.100 Configuration Memory Map 40 Master Address Start Address Start Address: Upper 16bits (Hex): [0x8000 End Address: Upper 16bits (Hex): [0x8007 End Address Start Address Start Address Start Address: Upper 16bits (Hex): [0x6000 End Address: Upper 16bits (Hex): [0x600 End Address: Upper 16bits		_
-V RV32 Configurator configuraton SeemitHWHUV_RV323.30.100 Configuraton Memory Map 6 Master Address Start Address Start Address: Upper 16bits (Hex): Dis8fff Lower 16bits (Hex): Start Address Start Address Start Address Start Address: Start Address: Start Address: Dipper 16bits (Hex): Ox6000 Lower 16bits (Hex): Start Address: Dipper 16bits (Hex): Ox6000 Lower 16bits (Hex): Ox6000 <t< td=""><td></td><td>_</td></t<>		_
nfigurator -V RV32 Configurator seemittikkitty RV323.0.100 Configuration Memory Map 6 Master Address Start Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x0 6 Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x7 7 8 Master Address Start Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x0 6 Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x7 7 7 1 Master Address Start Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x7 7 7 1 Master Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x7 7 7 1 Master Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x0 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8		_
nfigurator -V RV32 Configurator seemittikkitty RV323.0.100 Configuration Memory Map 6 Master Address Start Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x777 8 Master Address Start Address Start Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x777 8 Master Address Start Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x777 11 Master Address Start Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x777 11 Master Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x777 11 Master Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x777 11 Master Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x777 11 Master Address: Upper 16bits (Hex); [0x6000 Lower 16bits (Hex); [0x777 11 Master Address: Upper 16bits (Hex); [0x6070 Lower 16bits (Hex); [0x777 11 Master Address: Upper 16bits (Hex); [0x6070 Lower 16bits (Hex); [0x777 12 Lower 16bits (Hex); [0x777 13 Lower 16bits (Hex); [0x777 14 Address: Upper 16bits (Hex); [0x6777 15 Lower 16bits (Hex); [0x7777 1		_
Higurator -V RV32 Configurator ssemiHWHV KV232.3.0.00 Endpartson Memory Map B Matter Address Start Address Start Address Lower 16bits (Hex): [0x000 End Address Lower 16bits (Hex): [0x000 End Address Start Address Start Address Start Address Start Address Start Address Start Address Start Address Start Address Lower 16bits (Hex): [0x000 End Address: Upper 16bits (Hex): [0x6000 Lower 16bits (Hex): [0x000 End Address Start Address Start Address Start Address Start Address Lower 16bits (Hex): [0x000 End Address Lower 16bits (Hex): [0x000 End Address Lower 16bits (Hex): [0x000 End Address: Upper 16bits (Hex): [0x0000 Lower 16bits (Hex): [0x000 End Address: Upper 16bits (Hex): [0x0000 Lower 16bits (Hex): [0x000 End Address: Upper 16bits (Hex): [0x0000 Lower 16bits (Hex): [0x000		_
nfigurator -V RV32 Configurator seemitHWHUR RV323.0.100 Configuration Memory Map 6 Master Address Start Address Start Address: Upper 16bits (Hex): [0x6000 Lower 16bits (Hex): [0x6007 End Address: Upper 16bits (Hex): [0x6077 Lower 16bits (Hex): [0x7777 8 Master Address Start Address: Upper 16bits (Hex): [0x6077 Lower 16bits (Hex): [0x7777 Maddress Start Address: Upper 16bits (Hex): [0x6077 Lower 16bits (Hex): [0x7777 Maddress: Upper 16bits (Hex): [0x6777 Lower 16bits (Hex): [0x7777 Maddress		_
nfigurator -V RV32 Configurator SeemitHWHUX RV323.0.100 Configuration Memory Map 6 Master Address Start Address S		_
nfigurator -V RV32 Configurator SeemitHWHUX RV323.0.100 Configuration Memory Map 6 Master Address Start Address S		_
-V RV32 Configurator osemitMMHUR_RV323.30.100 Configuration Memory Map 60 Master Address		_
Infiguration -V RV32 Configurator osemitHWHVLV, RV323.30.100 Configuration Memory Map 61 Master Address Start Address		_
nfigurator -V RV32 Configurator seemittWittV RV323.30.100 Configuration Memory Map 40 Master Address Start Addres		_
-V RV32 Configurator osemitMMHUR_RV323.30.100 Configuration Memory Map 60 Master Address Start Address		_

2.4.2 Sample Design 2 – Base Design with DDR

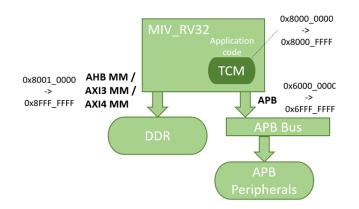
2.4.2.1 MIV_Legacy Configuration

SRAM is connected to the MEM interface at 0x8000_0000 with DDR connected at 0x8001_0000. Peripherals are connected to the MMIO interface at 0x6000_0000.



2.4.2.2 MIV_RV32 Configuration

TCM is enabled and set to a range from 0x8000_0000 to 0x8000_FFFF to run the application code. The AHB or AXI interfaces can be used to access the DDR in mirrored master mode with a range from 0x8001_0000 to 0x8FFF_FFF. The APB interface is enabled with a range from 0x6000_0000 to 0x6FFF_FFF.



By enabling the AHB master, it allows the AHB Master address fields of the memory map tab to be edited. The same applies to the APB and AXI masters along with the TCM.

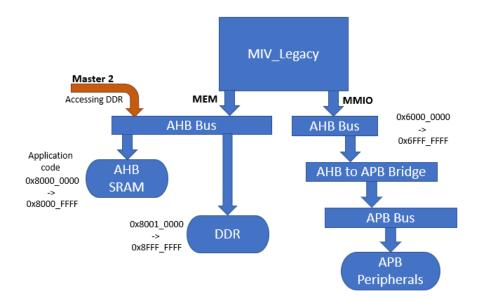
Configurator	-	×
Mi-V RV32 Configurator		
Microsemi:MiV:MIV_RV32:3.0.100		
Configuration Memory Map		
Extension Options		_
RISC-V Extensions: IMC 💌 🚯 Multipler: Fabric 💌 🚯		
Interface Options		
AHB Master: AHBLite 💌 AHB Mirrored I/F: 🔽 🌖		
APB Master: APB3 💌 APB Mirrored I/F: 🗖 🚯		
AXI Master: None 💌 AXI Mirrored I/F: 🔽 🚯		
Reset Vector Address		
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0)	
BootROM Options		
BootROM: 🗆 🚯 Reconfigure BootROM: 🗖 🚯		
Tightly Coupled Memory (TCM) Options		
TCM: 🔽 🚺 TCM APB Slave (TAS): 🗌 🚯		
-Interrupt Options		
External System IRQs: 0 💌 🚯		
Vectored Interrupts: 🗌 🕤		
System Timer Options		_
Internal MTIME: 🔽 🚺 MTIME Prescaler: 100		
Internal MTIME IRQ: 🔽 🏮		
Other Options		_
Debug: 🗖 🚯 Register Forwarding: 🗐 🚯		
ECC: 🗖 🚯 GPR Registers: 🗖 🚯		

Configuration Memory Map	
AHB Master Address	
Start Address: Upper 16bits (Hex): 0x8001	Lower 16bits (Hex): 0x0
End Address: Upper 16bits (Hex): 0x8fff	Lower 16bits (Hex): 0xffff
APB Master Address	
Start Address: Upper 16bits (Hex): 0x6000	Lower 16bits (Hex): 0x0
End Address: Upper 16bits (Hex): 0x6fff	Lower 16bits (Hex): 0xffff
AXI Master Address	
Start Address: Upper 16bits (Hex): 0x6000	Lower 16bits (Hex): 0x0
End Address: Upper 16bits (Hex): 0x6fff	Lower 16bits (Hex): 0xffff
TCM Address	
Start Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x0
End Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0xffff
TCM APB Slave Address	
Start Address: Upper 16bits (Hex): 0x4000	Lower 16bits (Hex): 0x0
End Address: Upper 16bits (Hex): 0x4000	Lower 16bits (Hex): 0x3fff
BootROM Address	
Source Start Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x0
Source End Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x3fff
Destination Address: Upper 16bits (Hex): 0x4000	Lower 16bits (Hex): 0×0

2.4.3 Sample Design 3 – Base Design with DDR and a Second Master

2.4.3.1 MIV_Legacy Configuration

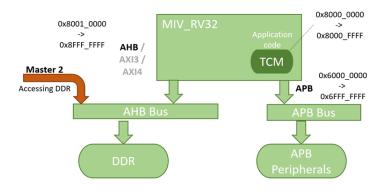
SRAM is connected to the MEM interface at 0x8000_0000 with DDR connected at 0x8001_0000. Peripherals are connected to the MMIO interface at 0x6000_0000. Master 2 is connected to the AHB bus used by the MEM interface accessing DDR.



2.4.3.2 MIV_RV32 Configuration

TCM is enabled and set to a range from 0x8000_0000 to 0x8000_FFFF to run the application code. The AHB or AXI interfaces can be used to access the AHB bus and DDR with a range from 0x8001_0000 to 0x8FFF_FFFF. The APB interface is enabled with a range from 0x6000_0000 to 0x6FFF_FFFF. Master 2 can access DDR through the AHB bus.

The block diagram and configuration windows show how to enable this setup.



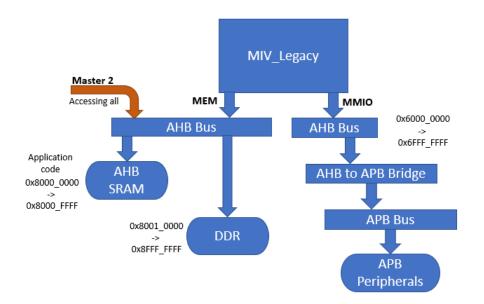
Configurator	-		×
Mi-V RV32 Configurator			
Microsemi:MiV:MIV_RV32:3.0.100			
Configuration Memory Map			
Extension Options RISC-V Extensions: IMC Multipler: Fabric			
Interface Options			
AHB Master: AHBLite 💌 AHB Mirrored I/F: 「 🚯			
APB Master: APB3 💽 APB Mirrored I/F: 🔽 🚯			
AXI Master: None 🗹 AXI Mirrored I/F: Г 🚯			
Reset Vector Address			
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0	•		
BootROM Options			
BootROM: 🗆 🚯 Reconfigure BootROM: 🦵 🚯			
Tightly Coupled Memory (TCM) Options			
_ Interrupt Options			
External System IRQs: 0 🔽 🚯			
Vectored Interrupts: 🗖 🚯			
System Timer Options			
Internal MTIME: 🔽 🚺 MTIME Prescaler: 100			
Internal MTIME IRQ: 🔽 🚯			
Other Options			
Debug: 🗖 🚯 Register Forwarding: 🗖 🚯			
ECC: 🗆 🕇 GPR Registers: 🗖 🚯			
Help	ОК	Can	cel

Configurator	-		×
Mi-V RV32 Configurator			
Configuration Memory Map AHB Master Address			[
Start Address: Upper 16bits (Hex): 0x8001 Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x8fff Lower 16bits (Hex): 0xffff			
APB Master Address			
Start Address: Upper 16bits (Hex): 0x6000 Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x6fff Lower 16bits (Hex): 0xffff			
AXI Master Address			
Start Address: Upper 16bits (Hex): 0x6000 Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x6fff Lower 16bits (Hex): 0xffff			
TCM Address			
Start Address: Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0xffff			
TCM APB Slave Address			
Start Address: Upper 16bits (Hex): 0x4000 Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x4000 Lower 16bits (Hex): 0x3fff			
BootROM Address			
Source Start Address: Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0		_	
Source End Address: Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x3fi	ff	-	
Destination Address: Upper 16bits (Hex): 0x4000 Lower 16bits (Hex): 0x0		-	
Help	ОК	Can	cel

2.4.4 Sample Design 4 - Base Design with DDR and Second Master

2.4.4.1 MIV_Legacy Configuration

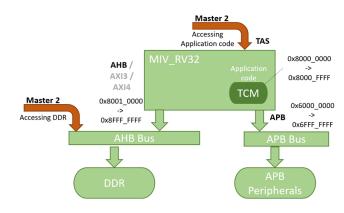
SRAM is connected to the MEM interface at 0x8000_0000 with DDR connected at 0x8001_0000. Peripherals are connected to the MMIO interface at 0x6000_0000. Master 2 is connected to the cached MEM AHB bus accessing the application code and DDR.



2.4.4.2 MIV_RV32 Configuration

TCM is enabled and set to a range from 0x8000_0000 to 0x8000_FFFF to run the application code. The AHB or AXI interfaces can be used to access the AHB bus and the DDR with a range from 0x8001_0000 to 0x8FFF_FFFF. The APB interface is enabled with a range from 0x6000_0000 to 0x6FFF_FFFF. Master 2 can access the DDR through the AHB bus and can access the application code in TCM using the TAS interface; making this change requires an APB master interface on Master 2.

The following block diagram and configuration windows show how to enable this setup.



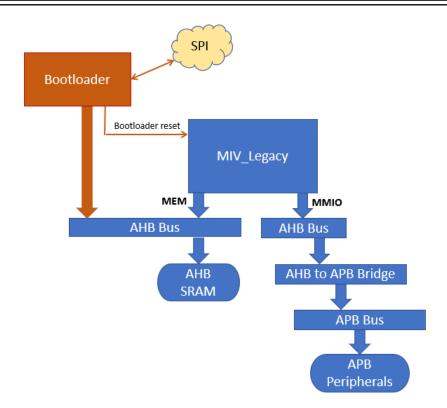
Configurator	-	
Mi-V RV32 Configurator		
Configuration Memory Map		
Extension Options		
RISC-V Extensions: DMC 丈 🐧 Multipler: Fabric 💌 🐧		
Interface Options		
AHB Master: AHBLIte AHB Mirrored I/F:		
APB Master: APB3 APB Mirrored I/F:		
AXI Master: None 💌 AXI Mirrored I/F: Г 🚯		
Reset Vector Address		
Upper 16bits (Hex): 0x3000 Lower 16bits (Hex): 0x0	0	
BootROM Options		
BootROM: 🗖 🚯 Reconfigure BootROM: 🦵 🚯		
Tightly Coupled Memory (TCM) Options TCM: 🔽 🌒 TCM APB Slave (TAS): 🔽 🏮		
Interrupt Options		
External System IRQs: 0 🗾 🚯		
Vectored Interrupts: 🔽 🚯		
System Timer Options		
Internal MTIME: 🔽 🚺 MTIME Prescaler: 100		
Internal MTIME IRQ: 🔽 👔		
Other Options		
Debug: 🗖 🚯 Register Forwarding: 🗖 🚯		
ECC: 🔽 🔁 GPR Registers: 🗖 🔁		
		-

Configuration Memory Map AHB Master Address					
Start Address: Upper 16bits (Hex):	0x8001	Lower 16bits (Hex):	0x0		
End Address: Upper 16bits (Hex):	0x8fff	Lower 16bits (Hex):	Dxffff		
APB Master Address					
Start Address: Upper 16bits (Hex):	0x6000	Lower 16bits (Hex):	0×0		
End Address: Upper 16bits (Hex):	0x6fff	Lower 16bits (Hex):	xffff	_	
AXI Master Address					
Start Address: Upper 16bits (Hex):	0x6000	Lower 16bits (Hex):	0x0		
End Address: Upper 16bits (Hex):	0x6fff	Lower 16bits (Hex):	Dxffff		
TCM Address					
Start Address: Upper 16bits (Hex):	0x8000	Lower 16bits (Hex):	0x0		
End Address: Upper 16bits (Hex):	0x8000	Lower 16bits (Hex):	Dxffff		
TCM APB Slave Address					
Start Address: Upper 16bits (Hex):	0x8000	Lower 16bits (Hex):	0x0		
End Address: Upper 16bits (Hex):	0x8000	Lower 16bits (Hex):	Dxffff		
BootROM Address					
Source Start Address: Upper 16bits	(Hex): 0x8000	Lower 16bits (†	lex): 0x0		1
Source End Address: Upper 16bits ()	Hex): 0x8000	Lower 16bits (F	lex): 0x3	fff	-
Destination Address: Upper 16bits (Hex): 0x4000	Lower 16bits (F	lex): 0x0		-

2.4.5 Sample Design 5 - Base Design Bootloader from SPI Flash

2.4.5.1 MIV_Legacy Configuration

Memory is connected to the MEM interface at $0x8000_0000$. Peripherals are connected to the MMIO interface at $0x6000_0000$. The bootloader is configured to pull data from a SPI flash. The bootloader reset holds the MIV_Legacy in reset while the memory is initialized.

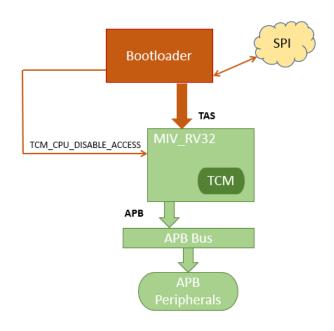


2.4.5.2 MIV_RV32 Configuration

TCM is enabled and set to start at 0x8000_0000 and end at 0x8000_FFFF. The APB master is enabled and configured to start at 0x6000_0000 and end at 0x6FFF_FFF. The TCM APB Slave (TAS) interface is enabled to allow the bootloader to write data to the TCM. The bootloader holds the TCM_CPU_DISABLE_ACCESS input high to prevent the core reading from the TCM, this input becomes available when the TAS is enabled. It means that the core is not held in reset. If the core is held in reset, the interface logic for the TCM and the TAS will also be reset, causing the write operation to the TCM to fail.

Note: It maybe the case that the core requires a reset after the initialization has completed.

The following block diagram and configuration windows show how to enable this setup.



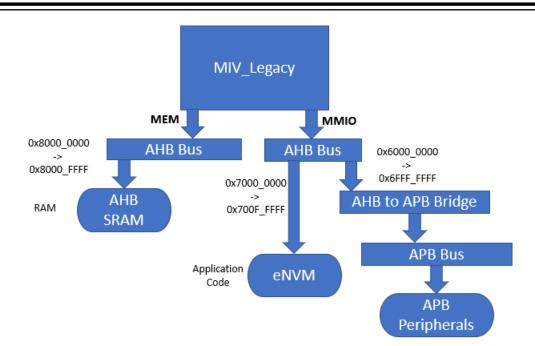
Configurator	31 44 1	
Mi-V RV32 Configurator		
licrosemi:MiV:MIV_RV32:3.0.100		
Configuration Memory Map		
Extension Options		
RISC-V Extensions: IMC 💌 🚯 Multipler: Fabric 💌 🚯		
Interface Options		
AHB Master: AHBLite 🗾 AHB Mirrored I,/F: 🗖 🚯		
APB Master: APB3 💽 APB Mirrored I/F: 🔽 🚯		
AXI Master: None 💌 AXI Mirrored I/F: Г		
Reset Vector Address		
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0	0	
BootROM Options		
BootROM: 🔽 🚯 Reconfigure BootROM: 🖵 🚯		
Tightly Coupled Memory (TCM) Options		
TCM: 🔽 🚯 TCM APB Slave (TAS): 🔽 🚯		
External System IRQs: 0 🗹 🕄		
System Timer Options		
Internal MTIME: 🔽 🐧 MTIME Prescaler: 100		
Internal MTIME IRQ: 🔽 🏮		
Other Options		
Debug: 🗖 🚯 Register Forwarding: 🧮 🚯		
ECC: 「 🕤 GPR Registers: 🔽 🕤		

Mi-V RV32 Configurator				
Microsemi:MIV:MIV_RV32:3.0.100				
Configuration Memory Map				
AHB Master Address Start Address: Upper 16bits (Hex): 0x8001	Lower 16bits (Hex): 0x0	-1		
End Address: Upper 16bits (Hex): 0x8fff	Lower 16bits (Hex): 0xffff			
APB Master Address				
Start Address: Upper 16bits (Hex): 0x6000	Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x6fff	Lower 16bits (Hex): 0xffff	_		
AXI Master Address				
Start Address: Upper 16bits (Hex): 0x6000	Lower 16bits (Hex): 0x0	_		
End Address: Upper 16bits (Hex): 0x6fff	Lower 16bits (Hex): 0xffff	_		
TCM Address				
Start Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x0			
End Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0xffff	-		
TCM APB Slave Address				
Start Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x0		\mathbf{r}	
End Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0xffff	_		
BootROM Address				
Source Start Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x0		1	
Source End Address: Upper 16bits (Hex): 0x8000	Lower 16bits (Hex): 0x3fff		-	
Destination Address: Upper 16bits (Hex): 0x4000	Lower 16bits (Hex): 0x0		-	

2.4.6 Sample Design 6 – Base Design Booting from eNVM and SRAM Used as RAM

2.4.6.1 MIV_Legacy Configuration

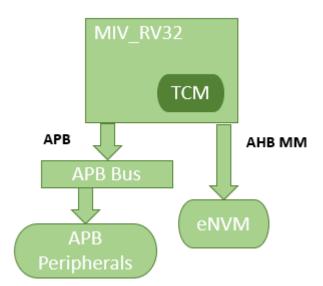
SRAM is connected to the MEM interface at 0x8000_0000 and used as RAM. eNVM is connected to the MMIO interface at 0x7000_0000 and used to store the read only application code. Peripherals are connected to the MMIO interface at 0x6000_0000. The core resets and boots from eNVM and uses the SRAM to hold the application data. As eNVM is read only, there is no code corruption, if an error occurs during execution.



2.4.6.2 MIV_RV32 Configuration

TCM is enabled and set to start at 0x8000_0000 and end at 0x8000_FFFF and is used in place of SRAM shown in the preceding figure. The AHB interface is enabled to start at 0x7000_0000 and end at 0x700F_FFFF to interface the eNVM. The APB master is enabled and configured to start at 0x6000_0000 and end at 0x6FFF_FFF.

The following block diagram and configuration windows show how to enable this setup.



V RV32 Configurator semicHV/HIV_RV32:3.0.100 onfpuration Memory Map RISC-V Extensions INC	
onfiguration Memory Map	
ension Options	
aser exertiserite inter in the product france in the second interval in the second interval in the second interval in the second i	
face Options	
HB Master: AHBLite 🗾 AHB Mirrored I/F: 🔽 🏮	
PB Master: APB3 ▼ APB Mirrored I/F: □	
AXI Master: None 💌 AXI Mirrored I/F: 🗖 🚯	
et Vector Address	
Jpper 16bits (Hex): 0x7000 Lower 16bits (Hex): 0x0	
ROM Options	
SootROM: 🔽 🚯 Reconfigure BootROM: 🗖 🚯	
rcM: 🔽 🚹 TCM APB Slave (TAS): □ 🚯	
upt Options	
External System IRQs: 0 💌 🚯	
lectored Interrupts: 🗌 🚯	
nternal MTIME: 🔽 🕄 MTIME Prescaler: 100	
nternal MTIME IRQ: 🔽 🚯	
r options Debug: 🗖 🚯 Register Forwarding: 🗖 🚯	
CC: 🗆 🕤 GPR Registers: 🗖 🚯	
	- 0
nfigurator	
infigurator I-V RV32 Configurator rosemitHWHIV_RV323.0.100	
nfigurator	
nfigurator -V RV32 Configurator osemicHWAIV_RV32.3.0.100 Configuraton Memory Map	
Infigurator IV RV32 Configurator SeemetHWHU RV323.0.100 Configuration Memory Map	
nfigurator -V RV32 Configurator seemkHWHVIV_RV323.0.100 Configuration Memory Map 61 Master Address Start Address Start Address: Upper 158/ts (Hex): [0x7000 Lower 188/ts (Hex): [0x7007 Lower 188/ts (Hex): [0x7007 16 Master Address	
nfigurator -V RV32 Configurator complexite data Configuration Memory Map 8 Master Address Start Address: Upper J8bits (Hex): [0x7000 End Address: Upper J8bits (Hex): [0x7007 End Address: Upper J8bits (Hex): [0x707 End Address: Upper J8bits (Hex): [0x	
The Address Upper 18bits (Hex): [0x5fff Lower 18bits (Hex): [0x0]	
nfigurator -V RV32 Configurator seemitHWHUT, RV323.0.100 Configuration Memory Map 60 Master Address Start Address Start Address Start Address Start Address Start Address Start Address Upper 189its (Hex): [0x7007 Lower 189its (Hex): [0x077 Reater Address Start Address Start Address Start Address Upper 189its (Hex): [0x6070 Lower 189its (Hex): [0x077 Reater Address Start Address Start Address Start Address Upper 189its (Hex): [0x6077 Lower 189its (Hex): [0x6777 Lower 189its (Hex): [0x7777 Lower 189its	
nfigurator -V RV32 Configurator seemithWHUV_RV32.30.100 Configuraton Memory Map 6 Master Address Start Address: Upper J8bits (Hex): [0x7000 Lower J8bits (Hex): [0x00 End Address: Upper J8bits (Hex): [0x7007 Lower J8bits (Hex): [0x600 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x6000 Lower J8bits (Hex): [0x0 End Address: Upper J8bits (Hex): [0x0 End A	
nfigurator -V RV32 Configurator seemithWHU, RV32.13.100 Configurato Memory Map 6 Master Address Start Address: Upper 1881s (Hex): [0x7000 Lower 1881s (Hex): [0x0 Configuration Memory Map 7 Master Address: Upper 1881s (Hex): [0x7007 Lower 1881s (Hex): [0x0 7 Master Address Start Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x6000 Lower 1881s (Hex): [0x0 7 Master Address: Upper 1881s (Hex): [0x600 7 Master Address: Upper	
Higurator -V RV32 Configurator ssemitHWHUR RV32.30.100 Emfguraton Master Address Start Address Lower 18bits (Hex): End Address: Upper 18bits (Hex): End Address: End Address: Upper 18bits (Hex): End Address: Upper 18bits (Hex): Start Address: Upper 18bits (Hex): End Address: Upper 18bits (Hex): End Address: Upper 18bits (Hex	
Higurator -V RV32 Configurator seemäHWHVIV_RV32.10.100 Configurato B Master Address Start Address: Upper 189ts (Hex): [0x7000 End Address: Upper 189ts (Hex): [0x7007 Lower 189ts (Hex): [0x7007 End Address Start Address Start Address Start Address: Upper 189ts (Hex): [0x6000 Lower 189ts (Hex): [0x600 Lower 189ts (
Higurator -V RV32 Configurator ssemitHNHVRU RV32.30.100 Endguraton Memory Map B Master Address Start Address:	
Infigurator -V RV32 Configurator comparison Memory Map Endiparation Memory Map 61 Master Address Start Address Start Address: Lower 188ts (Hex): End Address: Lower 188ts (Hex): Start Address: Lower 188ts (Hex): M Address: Lower 188ts (Hex): Start Address: Lower 188ts (Hex): M Address: Lower 188ts (Hex): Start Address: Lower 188ts (Hex): Lower 188ts (Hex): Lo	
nfigurator -V RV32 Configurator asemitMMHUR_RV323.0.100 Configuraton Memory Map 6Master Address Start Address Sta	
nfigurator -V RV32 Configurator seemtHWHVLV,RV323.0.100 Configuraton Memory Map 6 Master Address Start Address Start Address Start Address Start Address Start Address Start Address Upper 189ts (Hex): [0x7007 Lower 189ts (Hex): [0x7007 Lower 189ts (Hex): [0x707 Haster Address Start Address Start Address Start Address Start Address Start Address Upper 189ts (Hex): [0x5000 Lower 189ts (Hex): [0x700 Lower 189ts (Hex): [0x70 Lower 189ts (Hex): [0x7 Lower 18	
nfigurator -V RV32 Configurator seemittWHVTV_RV323.0.100 Configurator Start Address Start Address Start Address: Upper 189ts (Hex): [0x7000 Lower 189ts (Hex): [0x00 End Address: Upper 189ts (Hex): [0x7007 Lower 189ts (Hex): [0x077 Watter Address Start Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x077 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x077 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x077 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x077 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x077 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter Address: Upper 189ts (Hex): [0x6070 Lower 189ts (Hex): [0x777 Watter	
nfigurator	

2.5 System Time

The MIV_RV32 core features an internal 64-bit internal counter. The internal counter has the same function as the counter found in the PRCI module of the MIV Legacy cores. It can be used:

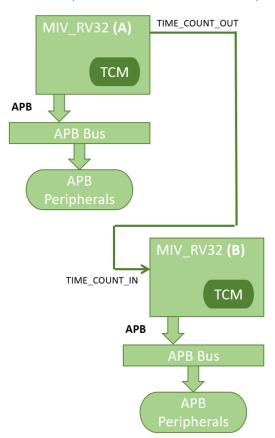
- To generate a time value for the processor.
- To generate a time value for the system.

This counter is disabled by default and must be enabled for use. Once enabled, a 64-bit top-level output (TIME_COUNT_OUT) is exposed to provide a time value to the system. In the default mode (counter disabled), a 64 bit top-level input is available (TIME_COUNT_IN) to provide a time value directly to the processor.

The processor also features a 64-bit compare register, which can be used to generate interrupts to the processor's timer interrupt. This can be enabled if needed, and the processors timer interrupt input is connected to the time count compare register. If it is not needed, the disabled top-level TMR_IRQ input is available on the core.

2.5.1 Sample Design 7—Internal MTIME and Internal MTIME IRQ

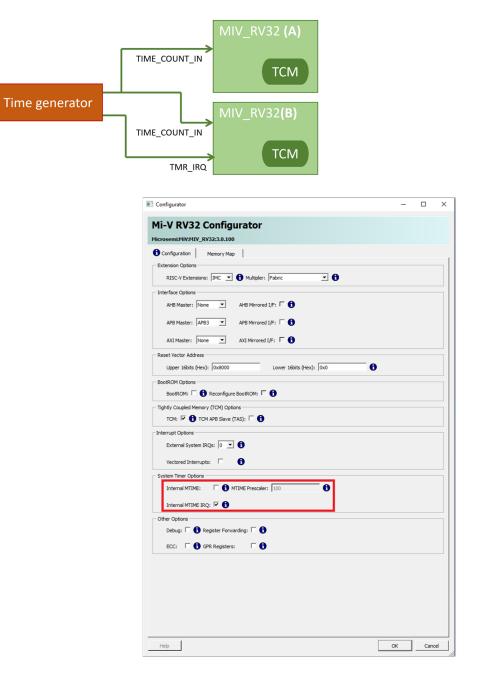
In this design as shown in the following figure, MIV_RV32 (A) has its internal counter enabled and MIV_RV32 (B) has its counter disabled. The MIV_RV32 (B) receives a time value from the "TIME_COUNT_OUT" of the MIV_RV32 (A). Both the processors have their internal compare registers enabled to generate independent periodic interrupts.



Configurator			
Ii-V RV32 Configurator			
icrosemi:MiV:MIV_RV32:3.0.100			
Configuration Memory Map Extension Options			
RISC-V Extensions: IMC 🛨 🐧 Multipler: Fabric 💽 🐧			
Interface Options			
AHB Master: None 💌 AHB Mirrored I/F: 🔽 🚯			
APB Master: APB3 APB Mirrored I/F:			
AXI Master: None 💌 AXI Mirrored I/F: 🔽 🚯			
Reset Vector Address			
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0	0		
BootROM Options			
BooROM: BootROM: BootRO			
TCM: 🖓 🕇 TCM APB Slave (TAS): 🗆 🕇			
Interrupt Options			
External System IRQs: 0 💌 🚯			
Vectored Interrupts:			
System Timer Options			
Internal MTIME: 🔽 🚯 MTIME Prescaler: 100			
Internal MTIME IRQ: 🔽 🚯			
Other Options Debug: T 1 Register Forwarding: T 1			
ECC:			
Help		ок	Can
			Can
] Configurator			
Configurator Mi-V RV32 Configurator			
Configurator Mi-V RV32 Configurator MicrosemiHWMTV_RV323.0.100 Configuration Memory Map Extension Options			
Configurator MicrosemicHWHIV_RV32.2.0.100 Configuration Configuration Extension Options RISC-V'Extensions: IMC Multipler: Fabric C			
Configurator MicrosemitHWHTV_EV32:3.0.100 Configuration Memory Map Configuration Memory Map Extension Options RISC-VEXtensions: [MC] Interface Options			
Configurator Mi-V RV32 Configurator MicrosemiHWHIV_RV323.0.100 Configuration Memory Map Extension Options RISC-V Extensions: [MC Multiple:: Fabric Interface Options AHB Marter: None AHB Merrored Lift:			
Configurator MicrosemicHWHIV_RV323.0.00 Configuration Configuration Extension Options RISC-V Extensions: MC Multipler: Fabric AHB Marrored I/F: AHB Marrored I/F: AHB Marrored I/F: AHB Marrored I/F:			
Configurator MicrosemicHWHIV_RV323.0.00 Configuration Configuration Memory Map Extension Options RISC-V Extensions: IMC Multipler: Pabric Interface Options AHB Marter: None AHB Merrored I,F: AHB Master: None AHB Merrored I,F: AIX Master: None AXX Merrored I,F: AXX Merrored I,F:			
Configurator MicrosemicHWHIV_RV323.0.00 Configuration Configuration Extension Options RISC-V Extensions: MC Multipler: Fabric AHB Marrored I/F: AHB Marrored I/F: AHB Marrored I/F: AHB Marrored I/F:	6		
Configurator Mi-V RV32 Configurator MicrosemidWMW_RV323.0.100 Configuration Configuration Memory Map Configuration RISC-V Extensions: MMC AHB Memored I/F: AHB Memored I/F: AHB Master: AHB Memored I/F: C AHB Master: Inter AHB Memored I/F: C C Reset Vector Addess Upper 168hts (hex): Data BootROM Options			
Configurator Mi-V RV32 Configurator MicrosemitHWHV_RV323.0.00 Configuration Configuration Memory Map Configuration Co			
Configurator Mi-V RV32 Configurator MicrosemidWMW_RV323.0.100 Configuration Configuration Memory Map Configuration RISC-V Extensions: MMC AHB Memored I/F: AHB Memored I/F: AHB Master: AHB Memored I/F: C AHB Master: Inter AHB Memored I/F: C C Reset Vector Addess Upper 168hts (hex): Data BootROM Options			
Configurator Mi-V RV32 Configurator MicrosemicHWHIV_RV32.3.0.00 Configuration Configuration Configuration Memory Map Extension Options RISC-V Extensions: MC Multipler: Fabric A+B Memored I,F: A+B Memored I,F: A+B Memored I,F: A+B Meter: A+B Memored I,F: A+B			
Configurator Mi-V RV32 Configurator HicrosemicHWHIV_RV323.0.00 Configuration Memory Map Extension Options RISC-V Extensions: [MC ♥ ♠ Multipler: Pabric ♥ ♠ Phterface Options AFB Master: None ♥ AFB Minored I,F: 「 ♠ AFB Master: None ♥ AFB Minored I,F: 「 ♠ ATI Master: None ♥ AFB Minored I,F: 「 ♠ ATI Master: None ♥ AFB Minored I,F: 「 ♠ ATI Master: None ♥ AFB Minored I,F: 「 ♠ Reset Vector Address Upper 188ts (Nex): [0x8000 Lower 188ts (Nex): [0x0 BooRIXM Options BooRIXM (Options BooRIXM (Copied Memory (TCM) Options TCM: [F ♠ TCM APB Save (TAS): 「 ♠			
Configurator Mi-V RV32 Configurator MicrosemitHWHIV_RV32.3.0.00 Configuration Memory Map Extension Options RISC-V Extensions: MC ♥ ♥ Multipler: Pabric ♥ ♥ Dinterface Options AHB Master: None ♥ AHB Memored I,F: F ♥ AHB Master: None ♥ AHB Memored I,F: F ♥ AHB Master: None ♥ AHB Memored I,F: F ♥ Reset Vector Addess Upper 168ts (Hex): [0x6000 Lower 168ts (Hex): [0x0 BootBCOM: F ♥ Reconfigure BootBCOM: F ♥ Toffully Coupled Memory (TOM) Options TOM: F ♥ TOM AHB Slave (TAS): F ♥ Interrupt Options External System IRQs: 0 ♥ ♥ Vectored Interrupts: F ♥			
Configurator Mi-V RV32 Configurator MicrosemidWMIT_RV32.2.0.00 Configuration Configuration Memory Map Configuration Memory Map Extension Options RISC-V Extensions: MC Multipler: Fabric A+B Memored I,F: F A+B Memored I,F: F AXI Memored I,F: F AXI Memory I,F: F Reset Vector Address Upper Stabils (Hex): Doo BooROM: F C BooROM: F C C C D D D D D D D D D D			
Configurator Mi-V RV32 Configurator Microsensit/WHIV_RV32.3.0.00 Configuration Configuration Memory Map Extension Options RISC-V Extensions: MC Multipler: Pabric A+B Marrored I,F: BootROM: BootROM: BootROM: BootROM: Boot			
Configurator Mi-V RV32 Configurator MicrosemidWMIT_RV32.2.0.00 Configuration Configuration Memory Map Configuration Memory Map Extension Options RISC-V Extensions: MC Multipler: Fabric A+B Memored I,F: F A+B Memored I,F: F AXI Memored I,F: F AXI Memory I,F: F Reset Vector Address Upper Stabils (Hex): Doo BooROM: F C BooROM: F C C C D D D D D D D D D D			
Configurator Mi-V RV32 Configurator MicrosemicHWHIV_RV32.3.0.00 Configuration Configuration Memory Map Extension Options RISC-V Extensions: MC Multipler: Fabric AHB Marored I,F: B AHB Marored I,F: C AHB Marored I,F: AHB Marored I,F: C AHB Marored I,F: AHB M			
Configurator Mi-V RV32 Configurator Hicrosemi-HWHTU_RV32.3.0.00 Configuration Configuration Configuration Memory Map Extension Options RISC-V Extensions: MC Memory Map Extension Reset: File APB			
Configurator Mi-V RV32 Configurator Microsenti/WMT_RV32.2.0.00 Configuration Configuration Memory Map Configuration Memory Map Extension Options RISC-V Extensions: MC Multipler: Fabric A+B Memored I,F: F A+B Memored I,F: F AXI Memored I,F: F AXI Memory I,F: F AXI			
Configurator Mi-V RV32 Configurator Microsenti/WMT_RV32.2.0.00 Configuration Configuration Memory Map Configuration Memory Map Extension Options RISC-V Extensions: MC Multipler: Fabric A+B Memored I,F: F A+B Memored I,F: F AXI Memored I,F: F AXI Memory I,F: F AXI			
Configurator Mi-V RV32 Configurator Microsenti/WMT_RV32.2.0.00 Configuration Configuration Memory Map Configuration Memory Map Extension Options RISC-V Extensions: MC Multipler: Fabric A+B Memored I,F: F A+B Memored I,F: F AXI Memored I,F: F AXI Memory I,F: F AXI			
Configurator Mi-V RV32 Configurator Microsenti/WMT_RV32.2.0.00 Configuration Configuration Memory Map Configuration Memory Map Extension Options RISC-V Extensions: MC Multipler: Fabric A+B Memored I,F: F A+B Memored I,F: F AXI Memored I,F: F AXI Memory I,F: F AXI			

2.5.2 Sample Design 8—External MTIME and External MTIME IRQ

In this sample design, MIV_RV32 (A) receives time from a system time generator and internally generates an interrupt. MIV_RV32 (B) receives time and a timer interrupt from the time generator.



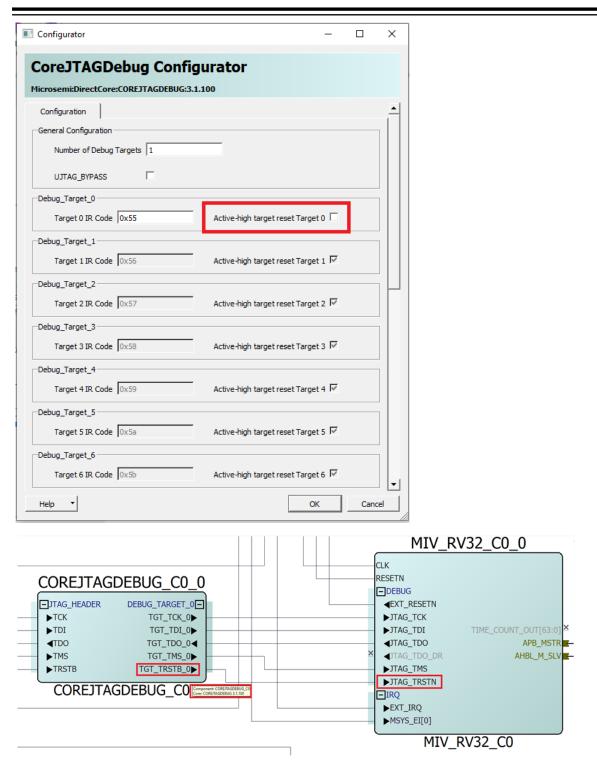
Configurator	-	
Mi-V RV32 Configurator		
4icrosemi:MiV:MIV_RV32:3.0.100		
Configuration Memory Map		
Extension Options		
RISC-V Extensions: IMC 💌 🚯 Multipler: Fabric 💌 🚯		
Interface Options		
AHB Master: None 💌 AHB Mirrored I/F: 🗆 🚯		
APB Master: APB3 APB Mirrored I/F:		
AXI Master: None 💌 AXI Mirrored I/F: 🗖 🚯		
Reset Vector Address		
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0	0	
BootROM Options		
BootROM: 🗖 🚯 Reconfigure BootROM: 🗖 🚯		
Interrupt Options External System IRQs: 0 🔽 🚯		
Vectored Interrupts:		
System Timer Options Internal MTIME:		
Internal MTIME IRQ: 🗆 🚯		
Other Options		
Debug: 🗆 🔂 Register Forwarding: 🗖 🚯		
ECC: 🗆 🚯 GPR Registers: 🗖 🚯		

2.6 Debug

MIV_RV32 features a JTAG compliant debug unit. A key difference between this debugger and the MIV_Legacy cores debugger is that the debugger is optional in the MIV_RV32. If the debug is not needed in a design, the feature can be disabled in the **Configurator** window.

Configurator			-		×
Mi-V RV32 Configurator					
Microsemi:MiV:MIV_RV32:3.0.100					
Configuration Memory Map					
Extension Options					
RISC-V Extensions: IMC 💌 🚯 Multipler: Fabric 💌 🚯					
Interface Options					
AHB Master: None 💌 AHB Mirrored I/F: 🔽 🚯					
APB Master: APB3 💽 APB Mirrored I/F: 🗆 🚯					
AXI Master: None 💌 AXI Mirrored I/F: 🗆 🕤					
Reset Vector Address	_				
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0	0				
BootROM Options					
BootROM: 🗌 🕦 Reconfigure BootROM: 🗖 🚯					
Tightly Coupled Memory (TCM) Options					
TCM: 🔽 🚯 TCM APB Slave (TAS): 🗌 🚯					
Interrupt Options					
External System IRQs: 0 💌 🚯					
Vectored Interrupts:					
System Timer Options					
Internal MTIME: MTIME Prescaler: 100					
Internal MTIME IRQ: 🗌 🚹					
Other Options					_
Debug: 🔽 🚯 Register Forwarding: 🗖 🚯					
ECC: 🗖 🚯 GPR Registers: 🗖 🚯					
			_		
Help	L	OK		Can	cel

A critical debug difference relates to the JTAG TRST polarity. The MIV_Legacy cores are active high JTAG_TRST, whereas the MIV_RV32 from v3.0.100 onwards uses an active low JTAG_TRSTN. A typical Libero MIV_RV32 design with debug features uses CoreJTAGDebug IP. Therefore, the user needs to ensure the correct polarity is used for the MIV_RV32. The following figures illustrate a typical design and the configuration of JTAG_TRST polarity on CoreJTAGDebug.



2.7 ECC

Some of the MIV_Legacy cores have support for ECC on their caches. As the MIV_RV32 does not have a cache, it does not need this protection, but there are SRAM implementations within the core that can be protected from errors.

1. In its standard configuration, the MIV_RV32 core uses RAM-based General Purpose Resources (GPRs). These are susceptible to errors.

- 1.1. By enabling the **GPR Registers** option for the core, generates GPRs as registers, which are not susceptible to the same errors.
- 1.2. By enabling the **ECC** option for the core, generates a fabric EDAC wrapper around the RAM-based GPRs and any single bit errors are corrected and cause an interrupt to be generated to the hart. Double bit errors cause a soft reset.
- 2. If the TCM is enabled, it may also need error protection.
 - 2.1. By enabling the **ECC** option for the core, generates a fabric EDAC wrapper around the RAM-based GPRs and any errors cause interrupts to be generated to the hart.

Configurator –	- (×	
Mi-V RV32 Configurator				
Microsemi:MiV:MIV_RV32:3.0.100				
Configuration Memory Map				
Extension Options				
RISC-V Extensions: IMC 💌 🚺 Multipler: Fabric 💌 🚯				
Interface Options				
AHB Master: None 💌 AHB Mirrored I/F: 🔽 🚯				
APB Master: APB3 S APB Mirrored I/F:				
AXI Master: None 🔄 AXI Mirrored I,F: 🗖 🚯				
Reset Vector Address				
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0				
BootROM Options				
BootROM: 🗆 🚯 Reconfigure BootROM: 🧮 🊯				
Tightly Coupled Memory (TCM) Options				
TCM: 🔽 🚯 TCM APB Slave (TAS): 🗌 🚯				
- Interrupt Options				
External System IRQs: 0 💌 🚯				
Vectored Interrupts:				
System Timer Options			-	
Internal MTIME: 🗌 🔂 MTIME Prescaler: 100				
Internal MTIME IRQ: 🗖 🚯				
Other Options				
Debug: 🔽 🚯 Register Forwarding: 🔽 🚯				
ECC: 🔽 🚯 GPR Registers: 🔽 🚯				
Enabling this option replaces RAM based registers with fabrie	c based	registe	rs.	
This improves throughput performance but will increase reso	ources u	sed an	a reduc	e the max operating frequency
Help		Cancel		
			11	

2.8 Interrupts

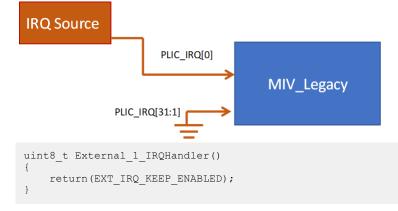
MIV_RV32 does not feature a PLIC like the MIV_Legacy cores. It has support for the three standard interrupts defined in the RISC-V Spec (Soft, Timer, and External) and also has the option to generate up to six additional external interrupts. An option to use Vectored Interrupts is also provided on the MIV_RV32 configuration GUI as shown in the following figure.

Configurator		-		×
li-V RV32 Configurator				
crosemi:MiV:MIV_RV32:3.0.100				- 1
Configuration Memory Map				
Extension Options				
RISC-V Extensions: IMC 🔽 🐧 Multipler: Fabric 🔽 🐧				_11
Interface Options				-11
AHB Master: AHBLite 💌 AHB Mirrored I/F: 🗌 🚯				
APB Master: APB3 💽 APB Mirrored I/F: 🔽 🚯				
AXI Master: None 💌 AXI Mirrored I/F: 🗖 🚯				
Reset Vector Address				-11
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0	•			
BootROM Options				
BootROM: 🗖 🚯 Reconfigure BootROM: 🦵 🌖				
Tightly Coupled Memory (TCM) Options				511
TCM: 🗌 🔂 TCM APB Slave (TAS): 🗖 🚯				
nterrupt Options				511
External System IRQs: 6 💌 🚯				
Vectored Interrupts:				
System Timer Options Enabling vectored interrupts will create an individual int	errupt vecto	or for e	each inte	rrupt in the
Internal MTIME: 🔽 🚺 MTIME Prescaler: 100				
Internal MTIME IRQ: 🔽 🚯				
Other Options				
Debug: 🗆 🚯 Register Forwarding: 🗖 🚯				
ECC: 🔽 🕤 GPR Registers: 🔽 🕤				
				- 11
				- 11
				- 11
Help	ОК		Can	rel

2.8.1 Sample Design 9 – Single Interrupt Source

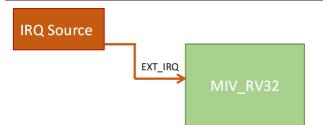
2.8.1.1 MIV_Legacy Configuration

In this sample design, MIV_Legacy has one interrupt source with the remaining 31 PLIC interrupts tied low.



2.8.1.2 MIV_RV32 Configuration

The interrupt source is connected to EXT_IRQ input of MIV_RV32.



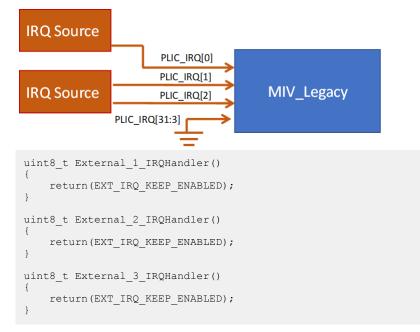
There are no configuration options that need to be selected to use EXT_IRQ . If required, you can enable the Vectored mode.

```
uint8_t External_IRQHandler()
{
    return(EXT_IRQ_KEEP_ENABLED);
}
int main(int argc, char **argv)
{
    HAL_enable_interrupts();
    asm volatile("wfi");
}
```

2.8.2 Sample Design 10 – Multiple Interrupt Sources

2.8.2.1 MIV_Legacy Configuration

In this sample design, MIV_Legacy has an interrupt source generating an interrupt for PLIC_IRQ[0], and a second source generating interrupts for PLIC_IRQ[1] and PLIC_IRQ[2] with the remaining PLIC interrupts tied low.



2.8.2.2 MIV_RV32 Configuration

The interrupt source generating a single interrupt is connected to the EXT_IRQ, and the source generating the two second interrupts is connected to two of the custom external interrupts.

IRQ Source		
IRQ Source	EXT_IRQ CUSTOM_IRQ_0 CUSTOM_IRQ_1	MIV_RV32
void Externa {	l_IRQHandler()	
}		
void MSYS_E1 {	0_IRQHandler(voic	L)
} void MSYS_E1 {	1_IRQHandler(voic	1)
}		

2.9 RISC-V Extensions

The MIV_RV32 core can use the base RISC-V Integer extension along with the Multiply and/or Compressed extensions as shown in the following figure. The multiply extension can be used with several versions of multipliers, depending on the processor frequency required and processor performance needed; multiplication can be completed in 1 cycle, 2 cycles or 32 cycles. The MIV_Legacy cores featured the Integer, Multiplication and Atomic extensions. The I and M extensions can be enabled in the MIV_RV32 core and the Atomic extension is used for multi-core systems, if atomics are required, an MIV_Legacy core must be used.

	Configurator			_	×
	Mi-V RV32 Configu	irator			
	Microsemi:MiV:MIV_RV32:3.0.100				
1	Configuration Memory Map				
	Extension Options				
	RISC-V Extensions: I	Multipler: Fabric	D		
	- Interface Options				
	AHB Master: AHBL IMC	AHB Mirrored I/F: 🗌 🚺			
	APB Master: APB3	APB Mirrored I/F: 🔲 🚹			
	AXI Master: None	AXI Mirrored I/F: 「 🚺			

2.9.1 RISC-V I Extension

This is the base RISC-V extension and is required in all cores.

2.9.2 RISC-V M Extension

The M extension adds multiply and divide instructions to the core. These can be used in place of software equivalents to improve code performance while increasing the area of MIV_RV32.

A benefit will only be seen from the RISC-V Multiply extension, if multiply operations are used frequently by software.

The multiplier in MIV_RV32 can be one of several types: MACC, MACC Pipelined, and Fabric, as seen in the following figure.

Configurator	-		\times							
Mi-V RV32 Configurator										
Microsemi:MiV:MIV_RV32:3.0.100	Microsemi:MiV:MIV_RV32:3.0.100									
Configuration Memory Map Extension Options Extensions: IM I Multipler: Fabric RISC-V Extensions: IM I Multipler: Fabric Interface Options MACC (Non-Pipelined) AHB Master: AHBLite AHB Mirrored I/F:										
APB Master: APB3 APB Mirrored I/F: 🗌 🕄										
AXI Master: None 💌 AXI Mirrored I/F: 🗆 🕄										

The MACC options use the math blocks included in the FPGA fabric to carry out the multiplication operations, while the fabric option instantiates a fabric multiplier.

Using the non-pipelined multiplier option, operations complete in one cycle.

Using the pipelined multiplier option, operations complete in two cycles.

Using the fabric multiplier option, operations complete in 32 cycles.

Using the 32-cycle multiplier can still be very beneficial, depending on the values being multiplied. Software multiplication (that is, only using the RV32I extension) can take many multiples of 32-cycle to complete and will not take the same number of cycles for different values. The fabric multiplier is still faster than this and completes multiplication in 32-cycle regardless of values.

For application that rely heavily on multiplication operations, a MACC option is recommended. For those applications that require less or none at all, a fabric multiplier can be used or the M extension can be excluded respectively.

2.9.2.1 Using Software Multiplication

Using the M extension with the following C code:

```
uint32_t val0 = 5;
uint32_t val1 = 7;
val0 = val0 * val1;
```

Compiles to the following RISC-V assembly:

lw a4,-28(s0) lw a5,-24(s0) mul a5,a4,a5 sw a5,-28(s0)

With the highlighted "mul" instruction, taking a fixed number of cycles to complete depending on the multiplier type chosen.

Using software multiplication, the same C code complies to the following RISC-V assembly:

```
a5,-28(s0)
1 w
lw
      a4,-24(s0)
mv
      al,a4
      a0,a5
mv
      ra,80001330 < mulsi3>
jal
mv
      a5,a0
      a5,-28(s0)
sw
 mulsi3():
      a2,a0
mν
li
      a0,0
andi
      a3,a1,1
        a3,80001344 <__mulsi3+0x14>
beqz
add
       a0,a0,a2
srli
      a1,a1,0x1
slli
       a2,a2,0x1
      a1,80001338 < mulsi3+0x8>
bnez
ret
```

The C code* for the loop being executed by the __mulsi3(): function is as follows:

```
unsigned int
__mulsi3 (unsigned int a, unsigned int b)
{
    unsigned int r = 0;
    while (a)
        {
            if (a & 1)
            r += b;
            a >>= 1;
            b <<= 1;
        }
        return r;
}
```

This loop executes until the multiplication operation has completed as opposed to the "mul" instruction available with the M extension.

Note: This function is included in the standard C library, included by GCC automatically when building your code, if the M extension is not selected.

2.9.3 RISC-V C Extension

Twenty-five of the base RV32I instructions have a compressed variant, which can be used in place of the base instruction. The compressed variant is only 16 bits instead of 32. This allows for a 20%–30% reduction in overall code size for a given application.

The following figure is an example chunk of RISC-V instructions, each cell is a 32-bit memory location.

ADD	LW	SW	SUB	SLL	LBU	SUB	SW	
LB	LH	ADD	CSRS	JAL	BNE	ADD	MUL	
RET	LH	ADD	BEQ	ADD	JAL	SUB	ADDI	

The following figure is the same chunk of instructions, but this time the C extension is included and the 16-bit instructions are mixed with the 32-bit instructions.

C.ADD C.LW	C.SW C.SUB	SLL	C.LBU C.SUB	sw	C.LB C.LH	ADD	CSRS
C.JAL C.BNE	ADD	MUL	C.RET C.LH	ADD	C.BEQ C.ADD	C.JAL C.SUB	ADDI

Using the C extension, it allows for a reduction in code size with a small increase in core area. The reduced code size allows for a smaller TCM and reduced RAM usage, which outweighs the increase in area from adding the extension. The C extension is recommended in most circumstances to reduce the code size.

3. Migrating Software Projects

MIV_RV32 HAL v3.0.100 or greater is required to use MIV_RV32.

3.1 Prerequisites

- Download and install latest SoftConsole at https://www.microsemi.com/product-directory/design-tools/4879softconsole#downloads.
- Download and install latest Firmware Catalog at https://www.microsemi.com/product-directory/design-tools/ 4880-firmware-catalog#downloads.

Note: If you have Libero[®] SoC Software installed, you need not install the Firmware Catalog as it is included in the Libero SoC Software.

3.2 Recommended Migration Process

The recommended way to migrate is to use the default Mi-V RV32IMA application from the SoftConsole workspace.

The migration process involves the following steps:

- 1. Generate the SoftConsole example projects from MIV_RV32 HAL v3.0, or greater, package in the firmware catalog.
- 2. Import the miv-rv32i-systick-blinky example project into workspace.
- 3. Copy your application specific files (main.c and other application specific files including driver) into the mivrv32i-systick-blinky example project.
- 4. Replicate your application project properties like pre-processor, include paths, optimization levels, and so on in the miv-rv32i-systick-blinky example project.
 - The readme.txt document located in the root directory of miv-rv32i-systick-blinky example project describes the linker script and macro combinations required for conditional compilation. If you have any application specific modifications in the linker script, then those should also be ported to the new linker script you are going to use for miv-rv32i-systick-blinky project.
 - The default debug and release build configurations are provided with the miv-rv32i-systick-blinky example project.
- 5. Build the Debug or Release target. Fix any build errors, if they occur.
- 6. Debug the application using debug or release launch configuration.

3.3 Example of Recommended Migration Process

The following steps describe migration to an MIV_RV32 core SoftConsole application.

1. In the Firmware catalog, search for the latest MIV_RV32 HAL v3.0.x, or greater. Right-click **MIV_RV32 Hardware Abstraction Layer (HAL)** to generate a sample project, as shown in following figure.

Firmware Catalog e View Tools Help					
View (<u>53/269</u>):	Search by all fields (53	53):			
🔏 All 🛛 🧏 Vault 🔍 Web repositories					
display only the latest version of a core)c	▼			
Vame	△ Version	<u> </u>			
CoreLPC Driver	2.1.101				
CoreMACFilter Driver	2.1.100				
CoreMMC Driver	3.0.101				
CoreMMC Driver	2.0.100				
CorePWM Driver	2.4.100				
CoreSDLC Driver	2.1.100				
CoreSPI Driver	3.3.100				
CoreSysServices_PF Driver	2.0.102				
CoreTSE Driver	2.5.100				
CoreTimer Driver	2.4.100				
CoreUARTapb Driver	3.3.101				
CoreWatchdog Driver	2.2.100				
Cortex-M1 CMSIS Hardware Abstraction Laver	2.1.101				
Hardware Abstraction Layer (HAL)	2.3.102				
/IV_RV32 Hardware Abstraction Layer (HAL)	3.0.105		L.		
PolarFire PCIe Driver	2.1.100 🔛 Generate	·			
PolarFire Transceiver Driver	2.0.106 🎇 Remove	rom voult			
PolarFire User Crypto Driver	2.2.102 Kemove	rom vault			
martFusion CMSIS-PAL	2.4.102				
martFusion MSS ACE Driver	2.3.102 😚 Show det	ails			
martFusion MSS Ethernet MAC Driver	3.1.102 Open do	umentation 🔹 🕨			
martFusion MSS GPIO Driver	2.1.100	sample project 🔸			4
martFusion MSS I2C Driver	3.1.101 Generate	sample project	Mi-V ►	SoftConsole 🕨	Systick Timer Exar
Documentation:					
IIV RV32 HAL UG.pdf					
<u>IIV RV32 HAL RN.pdf</u>					
Description: Hardware Abstraction Layer for the Mi-V soft process	sors.				
itart-up code for Mi-V soft processors Supports MIV RV32, MIV RV32IMC and legacy RV32IMA core varia					
supports MIV_RV32, MIV_RV32IMC and legacy RV32IMA core varia Support for the Integer multiplication and division(M), Floating point		(c) •			
support for the integer multiplication and division(*), Floating point	to y and Atomic instruction (Ay and Compressed				
		🔛 Generate			

2. In the **Generate Sample Options** dialog box, enter a folder location in which the project must be generated, as shown in the following figure.

🞯 Generate Sample Options	?	×	
Samples folder: D:/Systick_Timer			
Files will be generated in: D:\\$ystick_Timer\MIVRV32I_GNU_SC_Systick_Timer			
Show generation report			
Help	ОК	Cance	9

3. Open SoftConsole workspace and import the generated project using the option, as shown in the following figure.

SC W	vorkspa	ce.examp	oles - SoftCo	onsole v6.5.	0.442								
File	Edit	Source	Refactor	Navigate	Search	Project	Git	Run	Window	Help			
	New Open Open		rom File Sy	stem	Alt+Sh	iift+N >	ף קר שיין איין איין איין איין איין איין איין		-48 * - □	0 -	Q . •	- 🥭	S?
	Close Close	All			C Ctrl+Sh	trl+W ift+W							
	Save Save A	ls			(Ctrl+S							
¢.	Save A Revert				Ctrl+SI	hift+S							
	Move.												
2 8	Renan Refres					F2 F5							
V			elimiters To			>							
4	Print				C	Ctrl+P							
è	Impor	t											
4	Export	t											
	Prope	rties			Alt+	Enter							
	Exit												

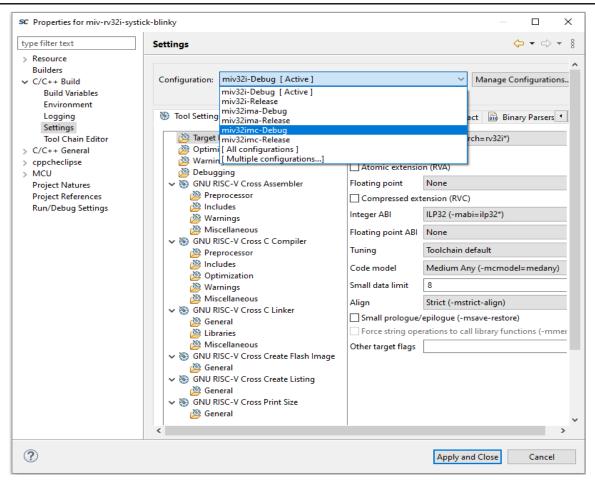
- 3.1. Select General > Existing Projects into workspace and click Next.
- 3.2. Copy the root directory (the generated project path) or use **Browse** to navigate to the root directory.
- 3.3. Select the application in the directory to import and click **Finish**.

<i>sc</i> Import				-			×			
Import Projects Select a directory to sear			7							
Select root directory:	D:\Svstick Time	r\MIVRV32I_GNU	_SC_Systick_Timer		F	rowse				
O Select archive file:			/ -	\ \		Browse				
Projects:										
miv-rv32i-systick-blinky (D:\Systick_Timer\MIVRV32I_GNU_SC_Systick_Timer)										
					De	select A	dl			
						Refresh				
<				2	•					
Options Search for nested pro	iects									
Copy projects into w										
Close newly imported										
Hide projects that alr	eady exist in the	workspace								
Working sets										
Add project to work	ing sets				N	ew				
Working sets:				\sim	Sel	ect				
?		< Back	Next >	Finish		Cancel				

The following	ig ligure	SHOWS			sole proje		ie worksp	ace.				
SC workspace	e.examp	les - SoftC	onsole v6.5.	0.442								
File Edit S	Source	Refactor	Navigate	Search Pr	roject Git	Run	Window	Help				
💼 🐔 📲	X D	>	J. G	2 ⇒[i⇒	\$ \$	-	-8 🎋 -	0 -	Q . •	· 😕 6	8 🕶 🎿	1 🏷 🄇
陷 Project Ex	xplorer 🛛	2			🗆 🔁	7 8						
📋 fpga-o	cortex-m	1-blinky										
📋 m1fpg	ga-corte	x-m1-blin	ky									
∽ 🔗 miv-n	v32i-syst	ick-blinky										
> 🔊 Inc	ludes											
> 🔁 drivers												
> 🔁 hal												
> 🗁 miv_rv32_hal												
> <u>h</u> hw	_platfor	m.h										
> 💼 ma	ain.c											
	v-rv32-e											
	📄 miv-rv32-ram-ima.ld											
		am-imc.ld										
	📄 miv-rv32i-systick-blinky Debug.launch											
		·	1 C C C C C C C C C C C C C C C C C C C	Debug.launc								
		-	nky Renode	Start-platfor	m-and-debi	ug.laun	ch					
> 📄 RE/	ADME.tx	t										

The following figure shows the imported SoftConsole project in the workspace.

- 4. As described in the Recommended migration process section, replace your application specific files in the example.
- 5. Open the hw_platform.h file and configure,
 - 5.1. The peripheral base addresses as per the memory map generated by Libero SoC Software design.
 - 5.2. The system clock frequency based on the Libero SoC Software design.
- Right-click the project name and open the properties menu (last option in menu). The project settings offer six types of configurations like debug and release configurations for Mi-V I, IMA, and IMC cores.
 Note: The selected configuration must match with the processor core in the design.



- 7. Select the configuration that matches your processor design. Make any application specific changes like pre-processor, include paths and so on.
- 8. Click **Apply and Close**. The same process must be followed to build the release target.
- 9. Build the debug or release target. Fix any build errors that arise in the process.
- 10. Use the default build configurations and look for any application specific settings.
- 11. Launch the application in debug mode to test the functionality.

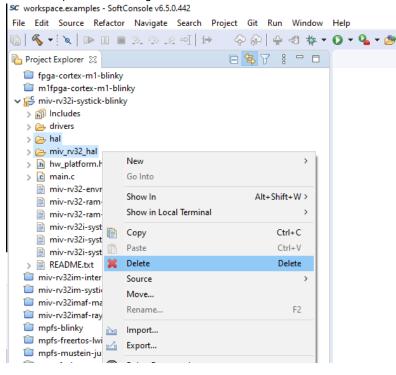
3.4 Updating the MIV_RV32 HAL

The MIV_RV32 HAL can be updated to the latest version. The source files are generated from the Firmware Catalog, which is installed with Libero SoC Software.

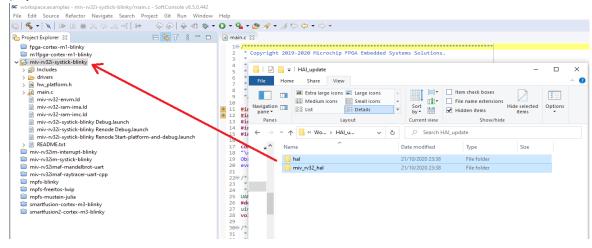
- 1. Open the Firmware Catalog and search for hal.
- 2. Right-click **MIV_RV32 Hardware Abstraction Layer (HAL)** and click **Generate**. Select a location for the HAL update files.

Firmware Catalog Ie View Tools Help	- 0
View (<u>53/269</u>): Xall Vault Web repositories ✓ display only the latest version of a core	Search by all fields (4/53):
Name	△ Version
Cortex-M1 CMSIS Hardware Abstraction Layer Hardware Abstraction Layer (HAL)	2.1.101 2.3.102 3.0
MIV_RV32 Hardware Abstraction Layer (HAL) SmartFusion2 CMSIS Hardware Abstraction Layer	2.3 Generate Remove from vault Show details Open documentation Generate sample project
Documentation: MIV_RV32_HAL_UG.pdf <u>MIV_RV32_HAL_RN.pdf</u> Description: Hardware Abstraction Layer for the Mi-V soft processors.	
Start-up code for Mi-V soft processors Supports MIV_RV32, MIV_RV32IMC and legacy RV32IMA core variants	Atomic instruction (A) and Compressed (C)

3. After generating the HAL update, the **hal** and **miv_rv32_hal** folders from the SoftConsole project must be updated. Note that the existing project specific linker script in the **miv_rv32_hal** folder will be over written. If required, it must be backed up before deleting the folder.

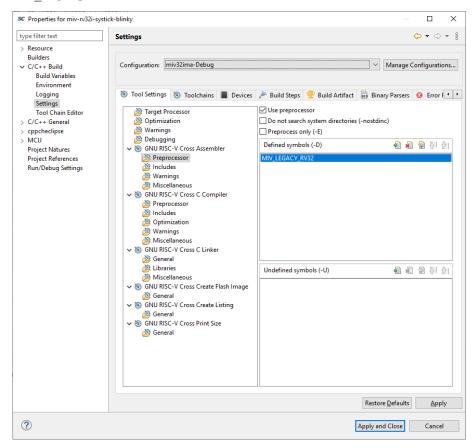


4. Copy the generated **hal** and **miv_rv32_hal** folders, generated by the Firmware Catalog into the SoftConsole project.

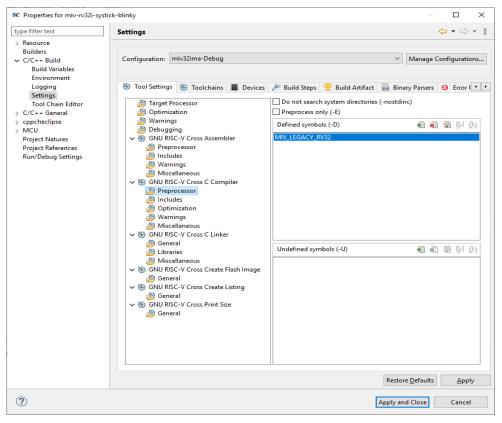


3.5 Defining the Core to the HAL

The MIV_RV32 and MIV_Legacy cores handle traps and interrupts differently. They also have different methods of causing internal interrupts. Each one includes unique interrupts, for example, a PLIC or ECC errors. Due to this, the HAL must be configured for the core that is being used, by defining a symbol in the SoftConsole project properties. This symbol must be defined in the following preprocessor settings. In C/C++ Build > Settings > Tool Settings > GNU RISC-V Cross Assembler > Preprocessor, the symbol MIV_LEGACY_RV32 must be defined, if an MIV_Legacy core is used.



In C/C++ Build > Settings > Tool Settings > GNU RISC-V Cross Compiler > Preprocessor, the symbol MIV_LEGACY_RV32 must be defined, if an MIV_Legacy core is used. If MIV_RV32 or MIV_RV32IMC core is used, then no symbol needs to be defined.



Two additional defines can be included, depending on the configuration of the MIV_RV32 core to define the system timer configuration of the core:

- MIV_RV32_EXT_TIMER
- MIV_RV32_EXT_TIMECMP

These symbols must be defined only when the internal MTIME and internal MTIMECMP options are not selected in the IP Core configurations. These symbols must be defined in the following project settings path.

C/C++ Build > Settings > Tool Settings > GNU RISC-V Cross Compiler > Preprocessor.

 > Resource Builders > C/C++ Build Build Variables Environment Logging > Settings > Tool Chain Editor > C/C++ General > cpchecipse > MCU > project Natures > Project References Run/Debug Settings > Miloudes > Winnigs > Miscellaneous > % GNU RISC-V Cross C Compiler > Miscellaneous > % GNU RISC-V Cross C Linker > % GNU RISC-V Cross C Linker > % GNU RISC-V Cross C Linker 	sc Properties for miv-rv32i-systic	k-blinky	— 🗆 X
Builders Configuration: miv32imc-Debug Manage Configuration Build Variables Environment Logging Jogging Tool Chain Editor Build Steps Build Attifact Binary Parsers Error O Configuration: Target Processor Do not search system directories (-nostdinc) Preprocesson hyl (-E) Cycheclipse Warnings Stoll Steps Preprocessor Defined symbols (-D) Tool (-D) Tool (-D) Tool (-D) Warnings Stocellaneous Stocell	type filter text	Settings	
	Builders V C/C++ Build Build Variables Environment Logging Settings Tool Chain Editor > C/C++ General > cpcheclipse > MCU Project Natures Project References	 Tool Settings Toolchains Devices Target Processor Optimization Warnings Debugging GNU RISC-V Cross Assembler Preprocessor Includes Warnings Miscellaneous Optimization Miscellaneous GOU RISC-V Cross C Compiler Preprocessor Includes Optimization Warnings Gouranta Construction Warnings Miscellaneous GOU RISC-V Cross C Linker General Libraries Miscellaneous Goura Cross Create Flash Image General GOU RISC-V Cross Create Listing GOU RISC-V Cross Create Listing GOUR RISC-V Cross Print Size 	Do not search system directories (-nostdinc) Preprocess only (-E) Defined symbols (-D)
Restore <u>D</u> efaults <u>App</u>			Restore <u>D</u> efaults <u>Apply</u>

When the "Internal MTIME" and "Internal MTIMECMP" are enabled in the core (default configuration). The SoftCosole project uses the same default settings and it works without adding any symbols to the project settings.

Configurator	-		×
Mi-V RV32 Configurator			
Microsemi:MiV:MIV_RV32:3.0.100			
APB Master: APB3 🔽 APB Mirrored I/F: 🗆 🚯			
AXI Master: None 💌 AXI Mirrored I/F: 🗖 🚯			
Reset Vector Address			
Upper 16bits (Hex): 0x8000 Lower 16bits (Hex): 0x0)	
BootROM Options BootROM: BootROM: BootROM:			
Tightly Coupled Memory (TCM) Options TCM: □ ① TCM APB Slave (TAS): □ ①			
Interrupt Options			
External System IRQs: 0 💽 🕄			
Vectored Interrupts: 🗖 🚯			
System Timer Options			
Internal MTIME: T			
Internal MTIME IRQ:			
Other Options			
Debug: 🗹 🚯 Register Forwarding: 🗔 🚯			
ECC: 🗆 🖨 GPR Registers: 🗆 🕤			
Help	ОК	Can	

3.6 Interrupts

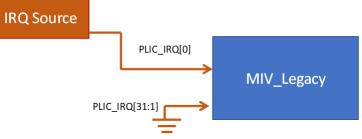
In MIV_RV32 HAL v3.0.x or greater handler, names for the standard RISC-V interrupts have not been changed, that is, external interrupt, software interrupt, and timer interrupt. If the MIV_LEGACY_RV32 symbol is defined in the GCC pre-processor, the software is built to support the MIV_Legacy cores. If the symbol is not defined, then the software will be built to support the MIV_RV32 core (as well as the MIV_RV32IMC core).

In the MIV_Legacy cores, PLIC interrupts cause the external interrupt to assert, and the core determines which interrupt in the PLIC has occurred and jump to its handler. In the MIV_RV32 core, each interrupt has its own encoding and a PLIC does not need to be polled to determine which interrupt has occurred.

3.6.1 Sample Design 11 – Single Interrupt Source

3.6.1.1 MIV_Legacy Configuration

In this sample design, MIV Legacy has one interrupt source with the remaining 31 PLIC interrupts tied low.



The following code block is a software implementation of the interrupt handling.

```
uint8_t External_1_IRQHandler()
{
    return(EXT_IRQ_KEEP_ENABLED);
}
```

3.6.1.2 MIV_RV32 Configuration

The interrupt source is connected to the EXT_IRQ input of MIV_RV32.



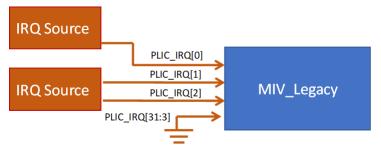
The following code block is a software implementation of the interrupt handling.

```
uint8_t External_IRQHandler()
{
    return(EXT_IRQ_KEEP_ENABLED);
}
```

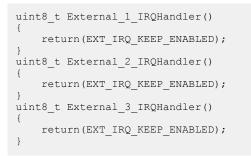
3.6.2 Sample Design 12 – Multiple Interrupt Sources

3.6.2.1 MIV_Legacy Configuration

In this sample design, MIV_Legacy has an interrupt source generating an interrupt for PLIC_IRQ[0] and a second source generating interrupts for PLIC_IRQ[1] and PLIC_IRQ[2] with the remaining PLIC interrupts tied low.

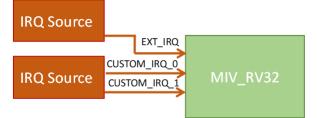


The following code block is a software implementation of the interrupt handling.



3.6.2.2 MIV_RV32 Configuration

The interrupt source generating a single interrupt is connected to the EXT_IRQ core; the source generating the two second interrupts is connected to two of the custom external interrupts.



The core configuration to enable CUSTOM IRQ 0 and CUSTOM IRQ 1 is as follows:

The following code block is a software implementation of the interrupt handling.

```
void External_IRQHandler()
{
    void MSYS_E10_IRQHandler(void)
{
    void MSYS_E11_IRQHandler(void)
    {
}
```

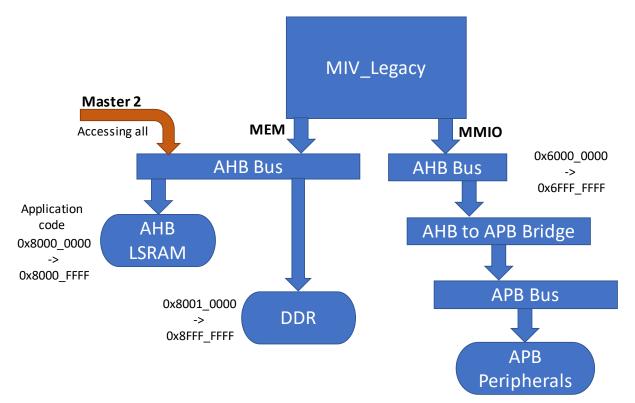
3.7 MIV_RV32 Extensions

As MIV_RV32 supports any configuration of RV32I, RV32IM, RV32IC, or RV32IMC, the SoftConsole projects need to be configured appropriately. In the **Project Properties** > C/C, select the check box for the **Multiply extension** (RVM), if the M Extension is included in the core, select the check box for the **Compressed extension** (RVC), if the C Extension is included in the core.

ource	Settings			$\diamond \bullet \bullet \bullet$	•
ders ++ Build Build Variables	Configuration: miv32i-Debug [Active]		~ Ma	anage Configurations	^
nun vanables nivironment Logging Settings Tool Chain Editor + General checlipse U Liect Natures iect References //Debug Settings	 Tool Settings Toolchains Devices Target Processor Optimization Warnings Debugging SGNU RISC-V Cross Assembler Preprocessor Includes Warnings Miscellaneous SGNU RISC-V Cross C Compiler Preprocessor Includes 	Build Steps 👻 Build J Architecture Multiply extensi Atomic extensic Floating point Compressed ext Integer ABI Floating point ABI Tuning Code model Small data limit	RV32I (-march=rv32i*) on (RVM) on (RVA) None tension (RVC) ILP32 (-mabi=ilp32*)	Error Parsers V	
>	 Optimization Warnings Miscellaneous S GNU RISC-V Cross C Linker General Libraries Miscellaneous S GNU RISC-V Cross Create Flash Image General S GNU RISC-V Cross Create Listing General S GNU RISC-V Cross Print Size General 	Align	Strict (-mstrict-align) /epilogue (-msave-restore) erations to call library functio	v	~
			Apply and Cl	ose Cancel	
onfigurator			Apply and Cl		
	2 Configurator		Apply and Cl		
i-V RV32	2 Configurator /_RV32:3.0.100		Apply and Cl		
i-V RV3 crosemi:MiV:MIN	-		Apply and Cl		
i-V RV32	Memory Map	- 0	Apply and C		
i-V RV32 rosemi:MiV:MIV Configuration	Memory Map	• •	Apply and C		
i-V RV32 crosemi:MiV:MIN Configuration Configuration Extension Options RISC-V Extens nterface Options	/_RV32:3.0.100 Memory Map ions: IMC ▼ Multipler: Fabric I IC IM A+BLIMC A+B Mirrored I/F: ▼ 1	- ()	Apply and C		
i-V RV32	/_RV32:3.0.100 Memory Map ions: IMC I I IC AHB Mirrored I/F: AHB3<	• 6	Apply and C		
i-V RV32	/_RV32:3.0.100 Memory Map ions: IMC I I IC AHB Mirrored I/F: AHB3<		Apply and C		
i-V RV32 rosemi:MiV:MIN Configuration Extension Options RISC-V Extens AIB Master: AVB Master: AXI Master: Exset Vector Addr Upper 16bits (tootROM Options	/_RV32:3.0.100 Memory Map ions: IMC I IC IM A+BL IMC A+B Mirrored I/F: I APB3 APB Mirrored I/F: I AXI MIRROV AXI MIRROV				
i-V RV32	/_RV32:3.0.100 Memory Map ions: IMC I I IC Multipler: JHB AHB Mirrored I/F: APB3<				

3.8 Maintaining Performance in Code Implementations From MIV_Legacy

In some use cases, certain code requirements are needed in software running on MIV_Legacy. The main requirement is that if memory is needed to appear consistent to another master accessing it, as shown in the following use case.



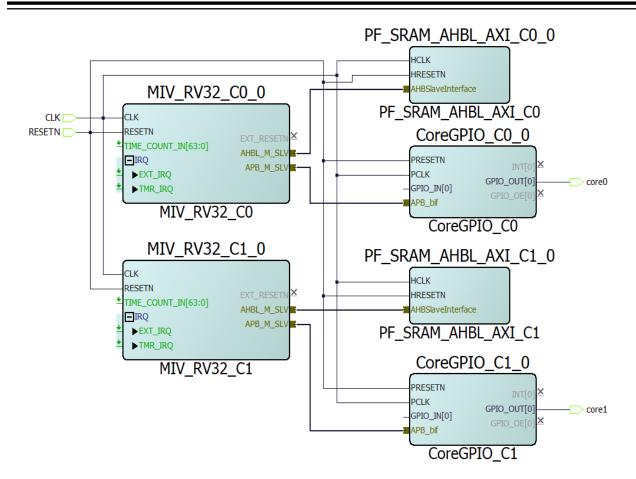
Due to the cache of MIV_Legacy, fence and fence.i instructions are required to make the memories in question appear consistently to the other masters. As there is no cache on MIV_RV32, there is no requirement to execute fence or fence.i instructions. Using the TAS to access, the TCM also appears consistently without fence or fence.i instructions.

Any ported code with fence or fence.i instructions still executes on the MIV_RV32. The instructions themselves have no effect when executed, but still need to be decoded, incurring a five-cycle delay.

Note: Any code that is ported from a MIV_Legacy configuration to a MIV_RV32 configuration, should have any fence and fence.i instructions removed.

3.8.1 Latency of fence and fence.i Instructions

The following figure is an example system to test the delay added by fence and fence.i instructions.



Core 0 boots and executes the following code in main.

```
int main(int argc, char **argv)
{
    GPIO_init(&g_gpio0, COREGPIO_IN_BASE_ADDR, GPIO_APB_32_BITS_BUS);
    GPIO_set_outputs(&g_gpio0, 1);
}
```

Core 1 boots and executes the following code in main.

```
int main(int argc, char **argv)
{
    asm volatile("fence.i");
    GPIO_init(&g_gpio0, COREGPIO_IN_BASE_ADDR, GPIO_APB_32_BITS_BUS);
    GPIO_set_outputs(&g_gpio0, 1);
}
```

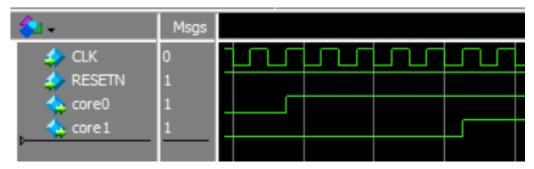
The time taken by each core to set its GPIO output indicates how long it takes to execute the code. The only difference between the code being executed on both cores is that the core 1 executes a fence.i instruction before initializing and setting its GPIO. The following figure shows that this result in core 1 settings is GPIO 5 cycles after core 0.

<u></u>				
💫 🗸	Msgs			
stand de la clik	0			
A RESETN	1			
👍 core0	0			
🔶 core 1	0			
r				

The code for core 1 is modified as in the following:

```
int main(int argc, char **argv)
{
    asm volatile("fence");
    GPIO_init(&g_gpio0, COREGPIO_IN_BASE_ADDR, GPIO_APB_32_BITS_BUS);
    GPIO_set_outputs(&g_gpio0, 1);
}
```

In this case, a fence instruction is executed instead of a fence.i instruction.



The same delay as seen with fence.i can been seen here. This delay is compounded every time and the instruction is executed.

4. Revision History

Revision	Date	Description
А	October 2020	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 devices:

- · Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- · Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like 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. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2020, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-7048-9

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