First Thursdays

May 2 - Webinar 1: Discover Renode for PolarFire® SoC Design and Debug

June 6 - Webinar 2: How to Get Started with Renode for PolarFire SoC

July 4 - Webinar 3: Learn to Debug a Bare-Metal PolarFire SoC Application with Renode

Aug. 1 - Webinar 4: Tips and Tricks for Even Easier PolarFire SoC Debug with Renode

Sept. 5 - Webinar 5: Add and Debug PolarFire SoC Peripherals with Renode

Oct. 3 - Webinar 6: Add and Debug and Pre-Existing Peripheral in PolarFire SoC

Nov. 7 - Webinar 7: How to write custom models – filters, offloading, acceleration etc

Dec. 5 - Webinar 8: Handling Binaries

Contd.

Second Thursdays

Jan. 9 - Webinar 9: Run Linux on Renode (PolarFire SoC Model as a Quad-core SMP) – this is not a Linux / Buildroot tutorial

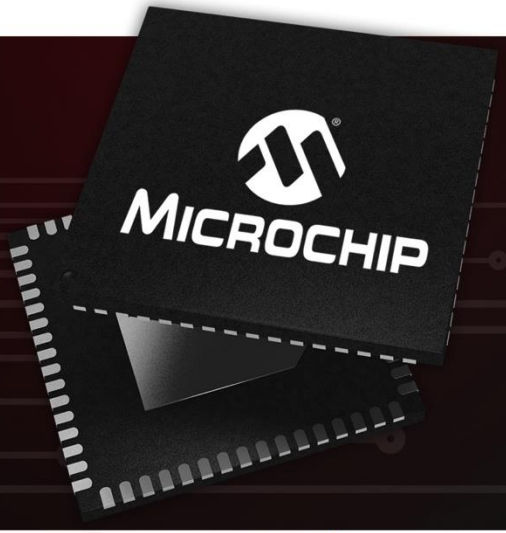
Feb. 13 - Webinar 10: Build applications for Linux on PolarFire SoC

Mar. 12 - Webinar 11: Introduction to PolarFire SoC MSS Configuration and Software Flow

Apr. 9 - Webinar 12: Two baremetal Applications on PolarFire SoC

May 14 - Webinar 13: Linux + Real-Time (AMP Mode) on PolarFire SoC

MICROCHIP



A Leading Provider of Microcontroller, Security, Mixed-Signal, Analog & Flash-IP Solutions



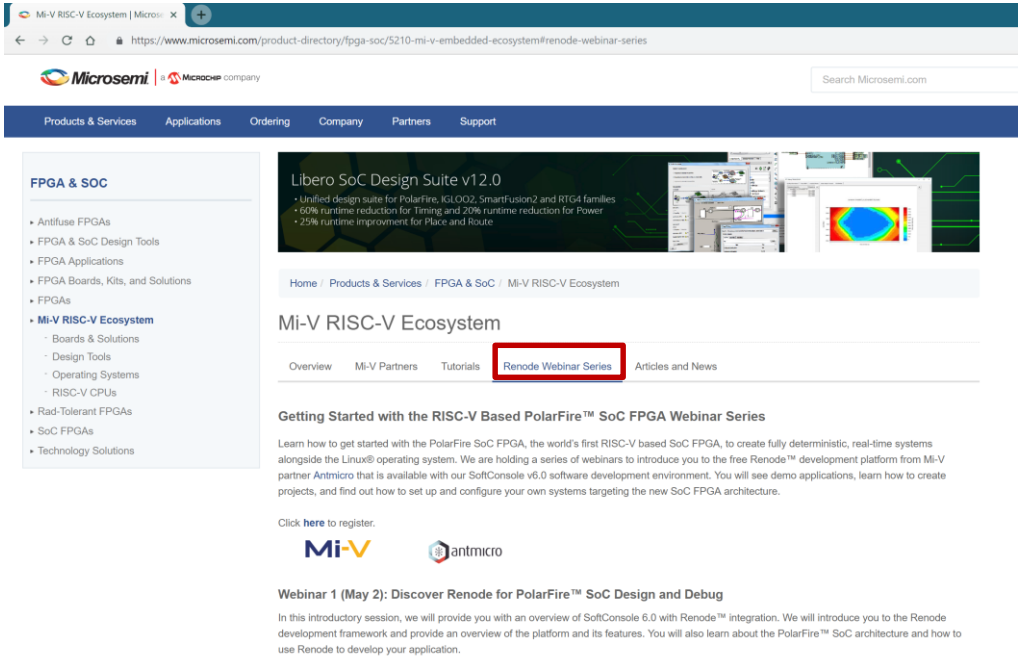

Getting Started with the RISC-V Based PolarFire® SoC FPGA Webinar Series

Session 5: "Add and Debug PolarFire SoC Peripherals with Renode"

Hugh Breslin, Embedded Linux Engineer

Thursday Sep. 5, 2019

Supporting Content



www.microsemi.com/Mi-V "Renode Webinar Series"

Add and Debug PolarFire® SoC Peripherals with Renode

- The files that make a Renode system
- The launch script
- CPU file
- Board file
- Summary
- Debugging a peripheral

The Files That Make a Renode System

The Files That Make a Renode System

- **We have two pre-configured systems:**

  Mi-V Renode emulation platform (Mi-V system):

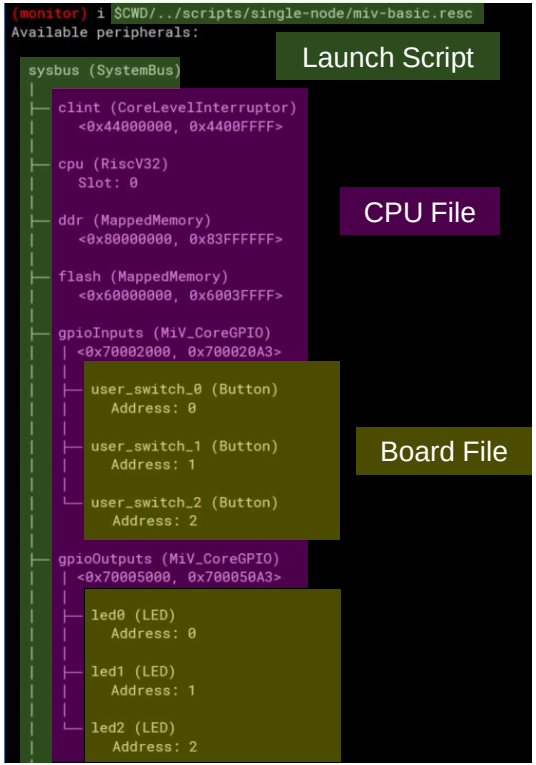
  Configured to emulate the Mi-V soft CPUs (RV32G core with UART, GPIOs and Timers)

  PolarFire SoC Renode emulation platform (PolarFire SoC system):

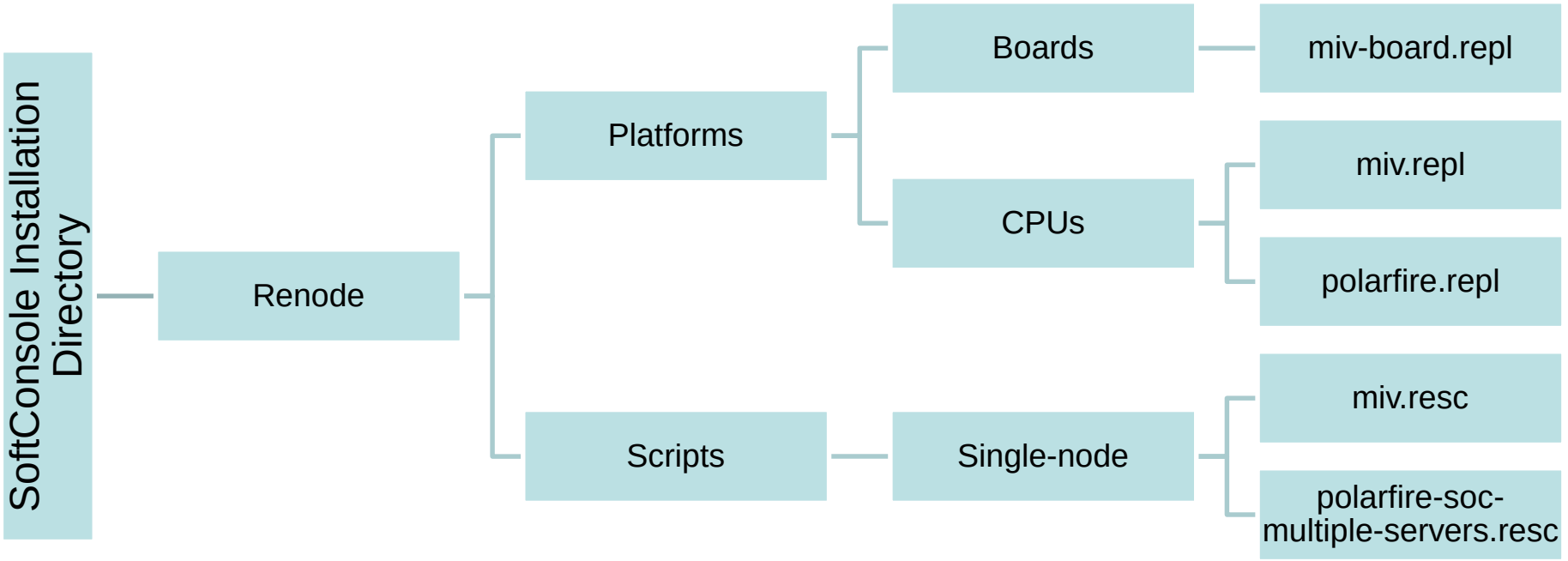
  Configured to emulate PolarFire Soc and its peripherals

The Files That Make a Renode System

- **3 files that can make up a Renode system**
  - Only the script file is passed as an argument when launching Renode
- **Script file: Tells Renode to create a machine, load the board file and run commands**
- **Board file: Tells Renode to load the CPU file and describes the hardware connections for the system (e.g. GPIO connecting to an LED)**
- **CPU file: Describes the system (e.g. CPU and peripherals)**

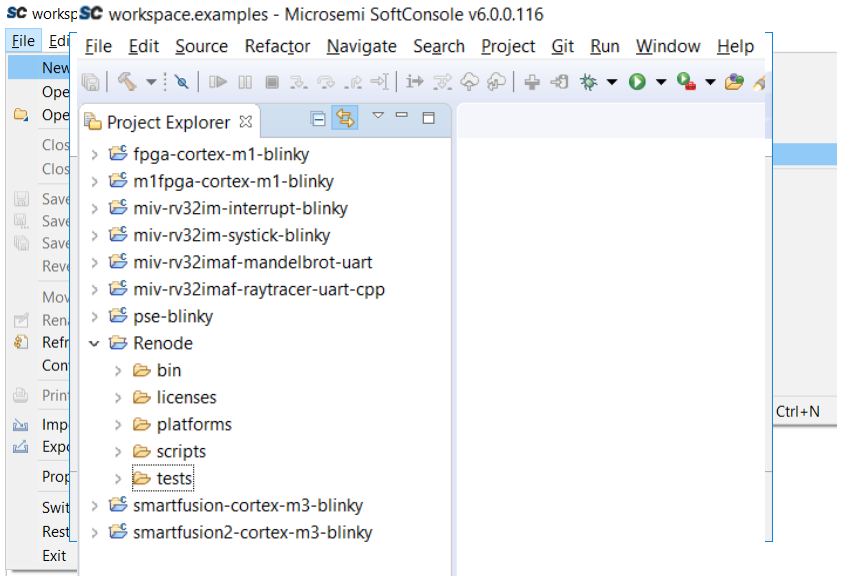


The Files That Make a Renode System



The Files That Make a Renode System

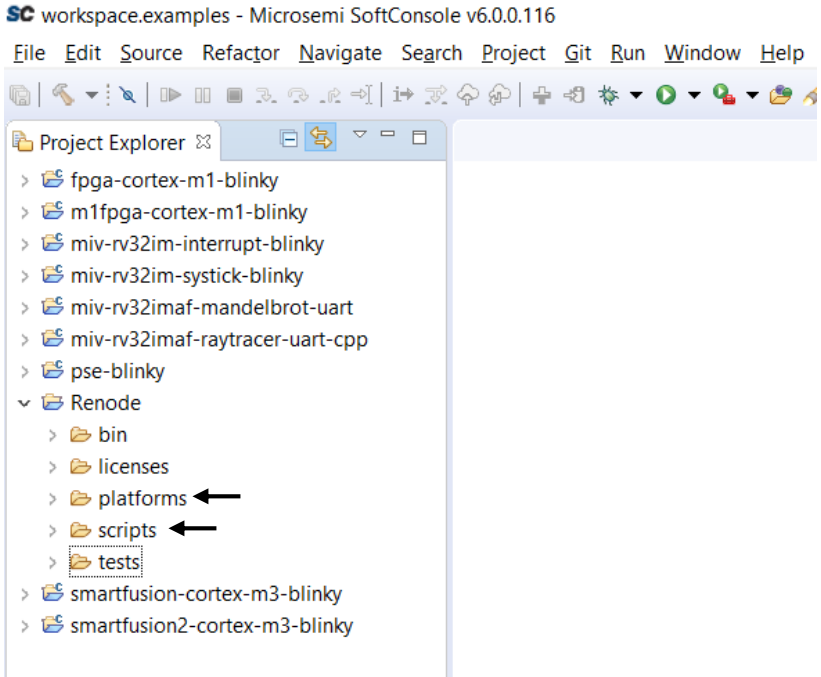
- **Easiest way to deal with the files and folder structure is to create a "Renode" project in SoftConsole**



The Files That Make a Renode System

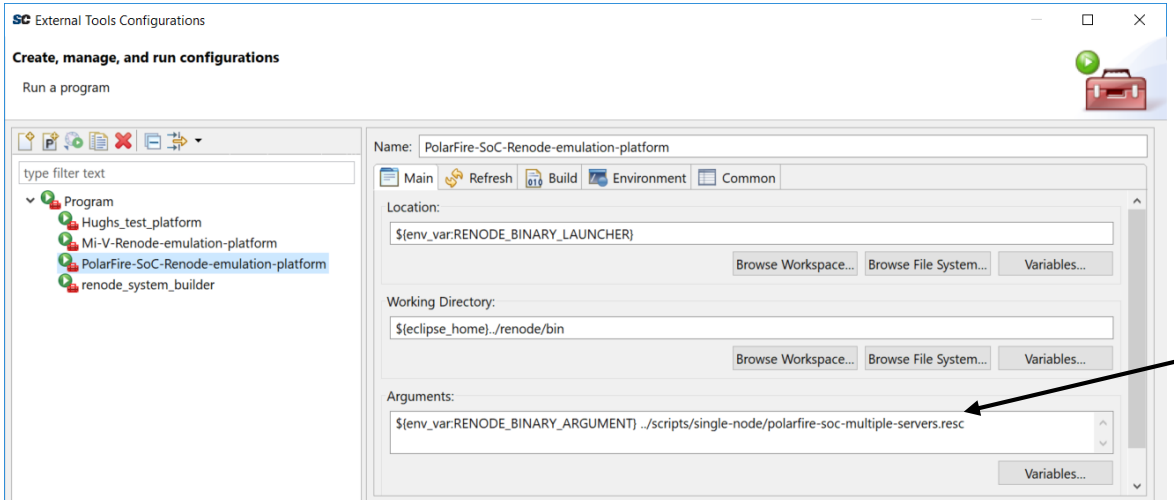
SoftConsole Installation Directory — Renode

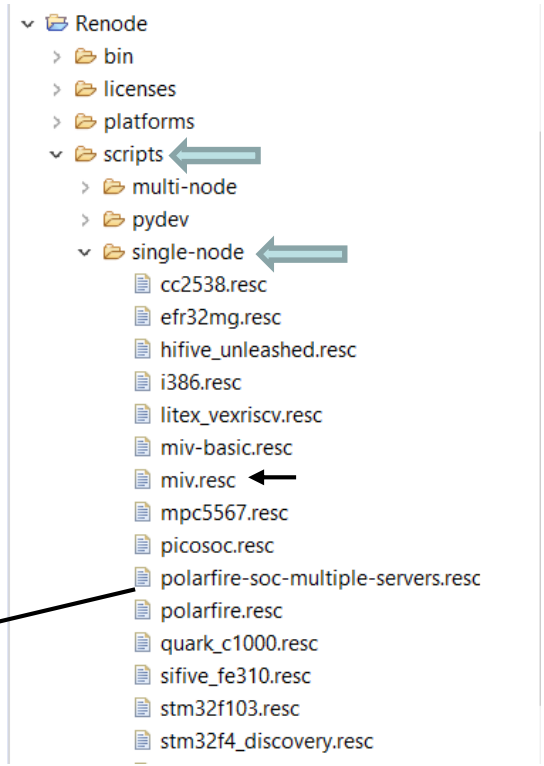
- **Platforms: Contains configuration files for platforms (i.e. CPU and board files)**
- **Scripts: Contains launch files for different platforms (i.e. launch scripts)**



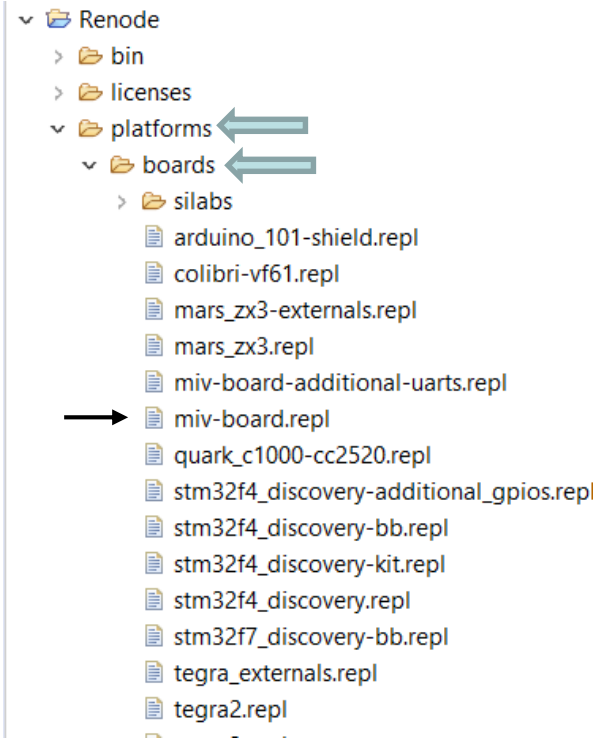
The Files That Make a Renode System

- **miv.resc: Launches the Mi-V system**
- **polarfire-soc-multiple-servers.resc: Launches the PolarFire SoC system**





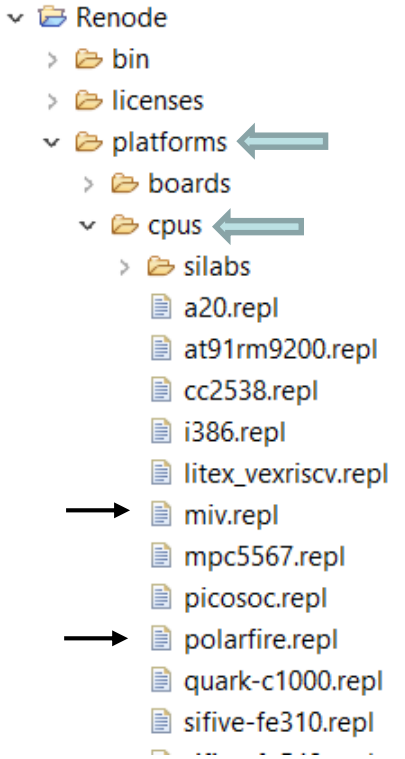
The Files That Make a Renode System



Board files

- **PF SoC system: n/a**
- **Mi-V system: miv-board.repl**

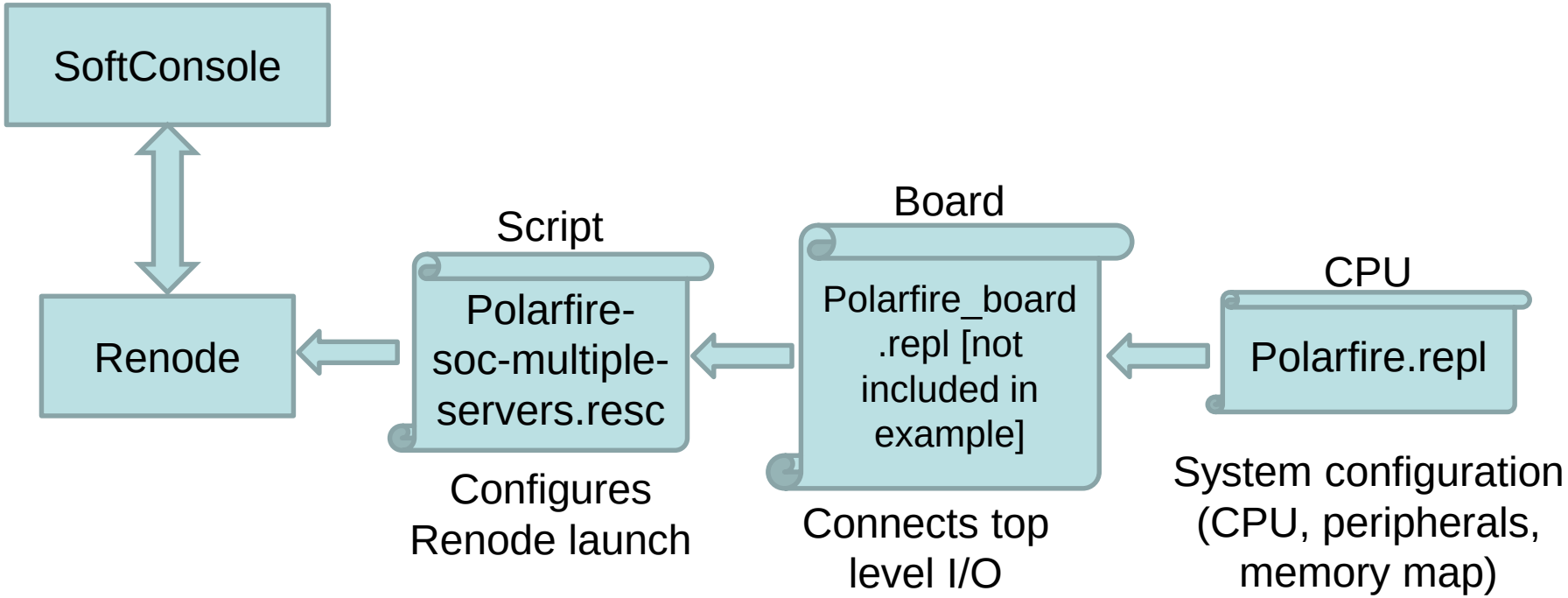
The Files That Make a Renode System



CPU files

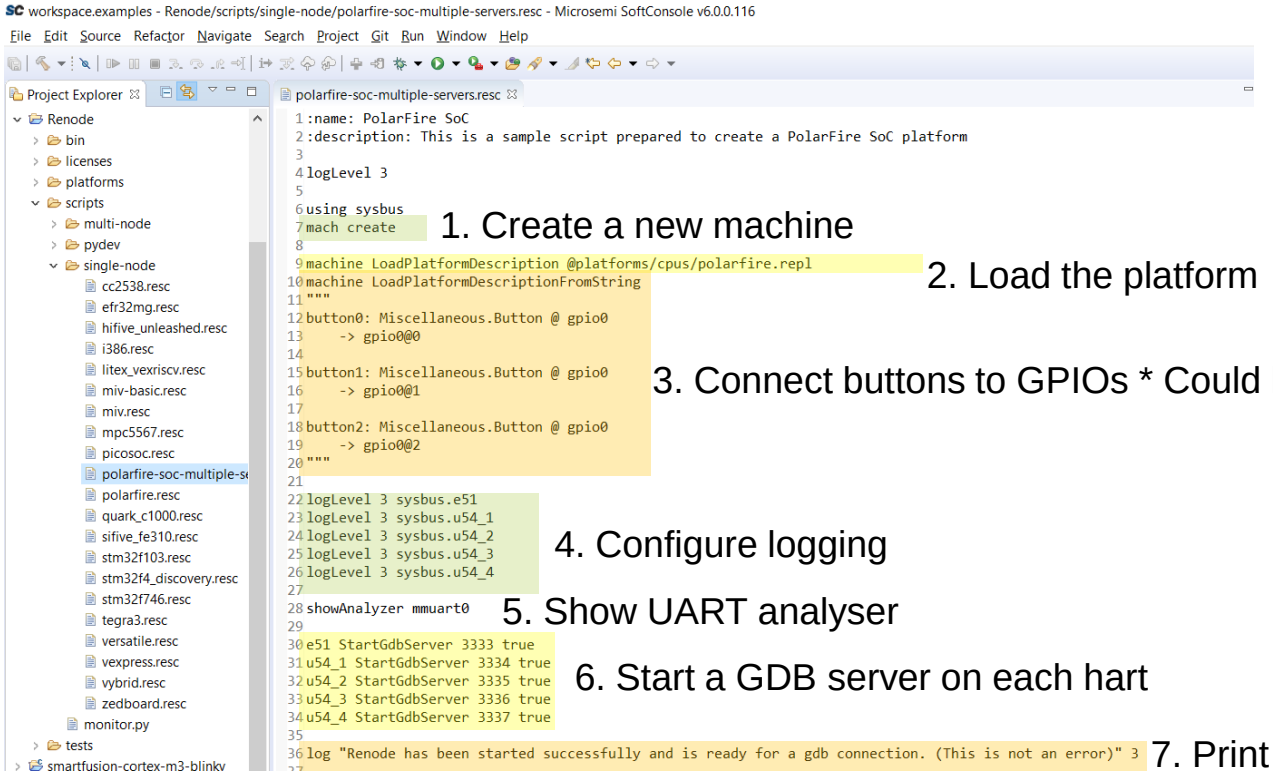
- **PolarFire SoC system: polarfire.repl**
- **Mi-V system: miv.repl**

The Files That Make a Renode System



The Launch Script

The Launch Script



1. Create a new machine

2. Load the platform

3. Connect buttons to GPIOs * Could be done in the board file *

4. Configure logging

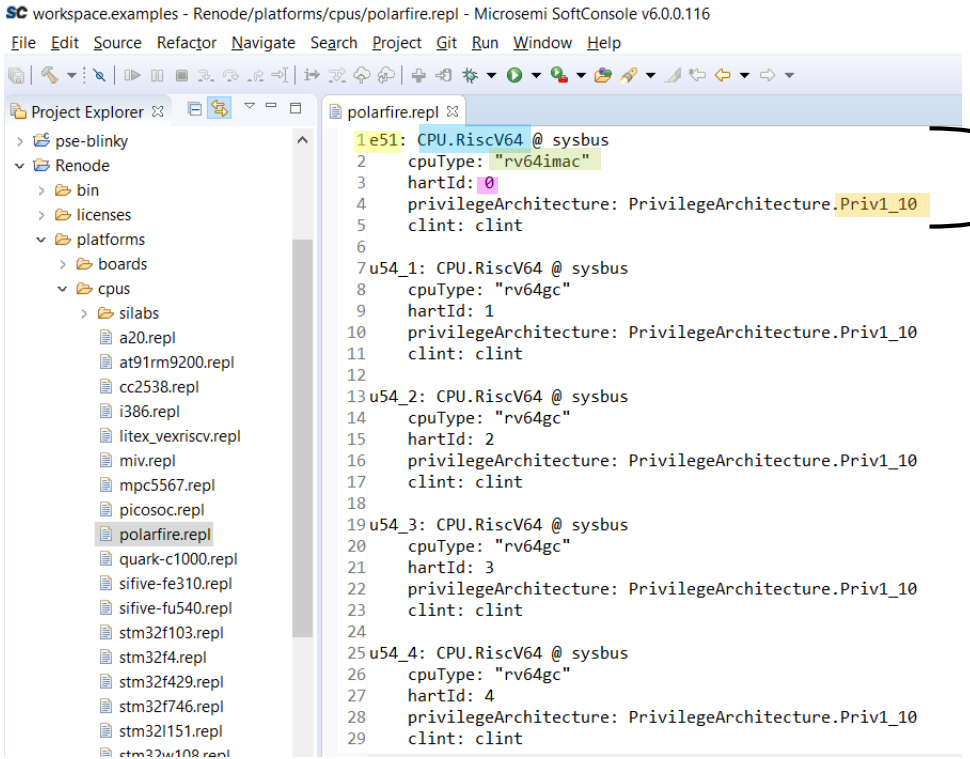
5. Show UART analyser

6. Start a GDB server on each hart

7. Print message to allow SC to start GDB

The CPU File

The CPU File

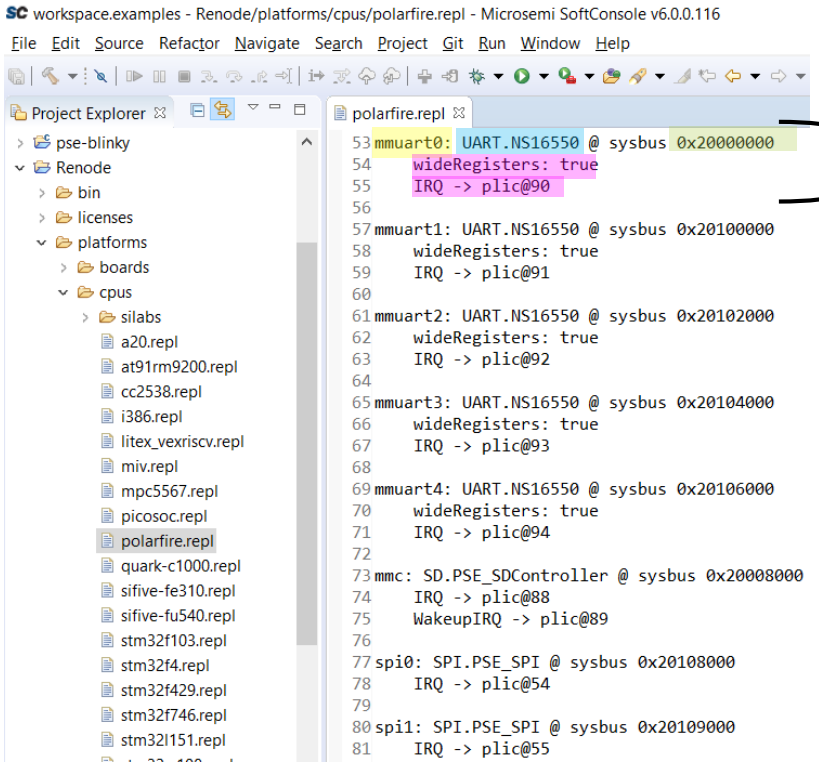


ADD CPU to the system

Connecting a CPU:

[name]: [path_to_cpu] @ sysbus
cpu_Type: "[cpu_type]"
hartId: [ID]
privilegeArchitecture: PrivilegeArchitecture.[version]
clint: clint

The CPU File



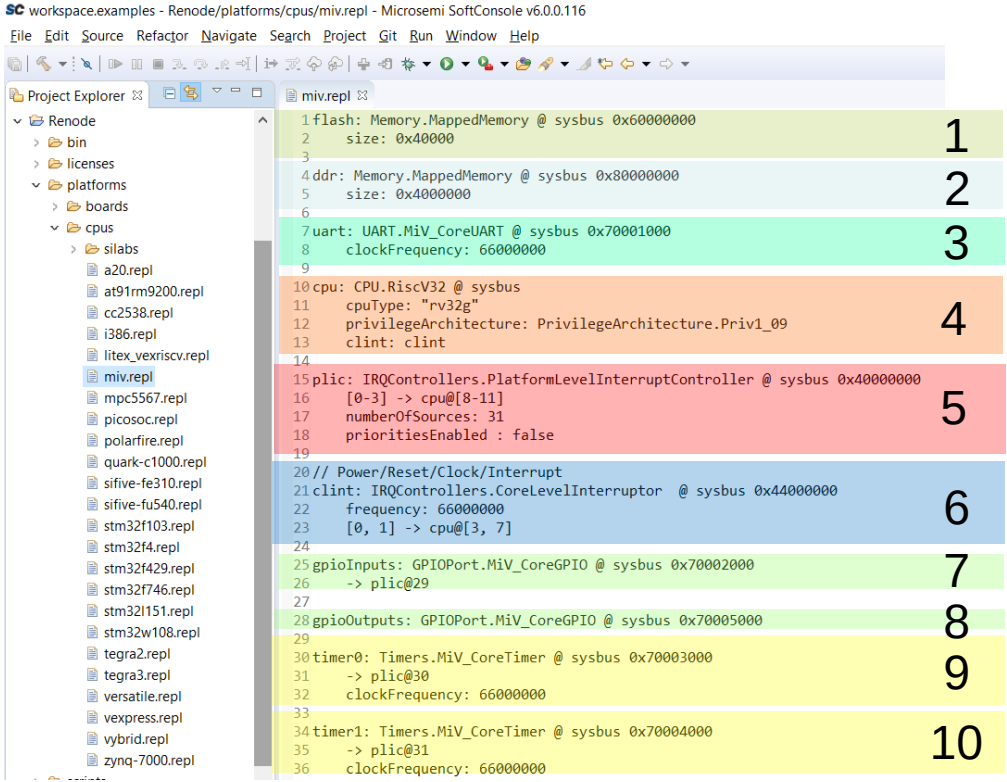
ADD UART to the system

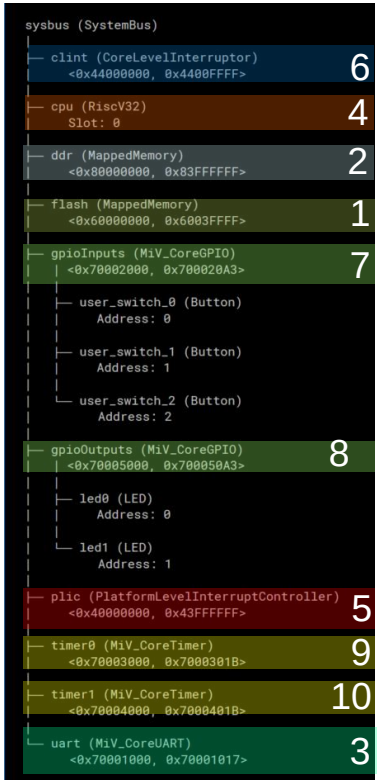
Connecting a peripheral:

It depends on the peripheral!

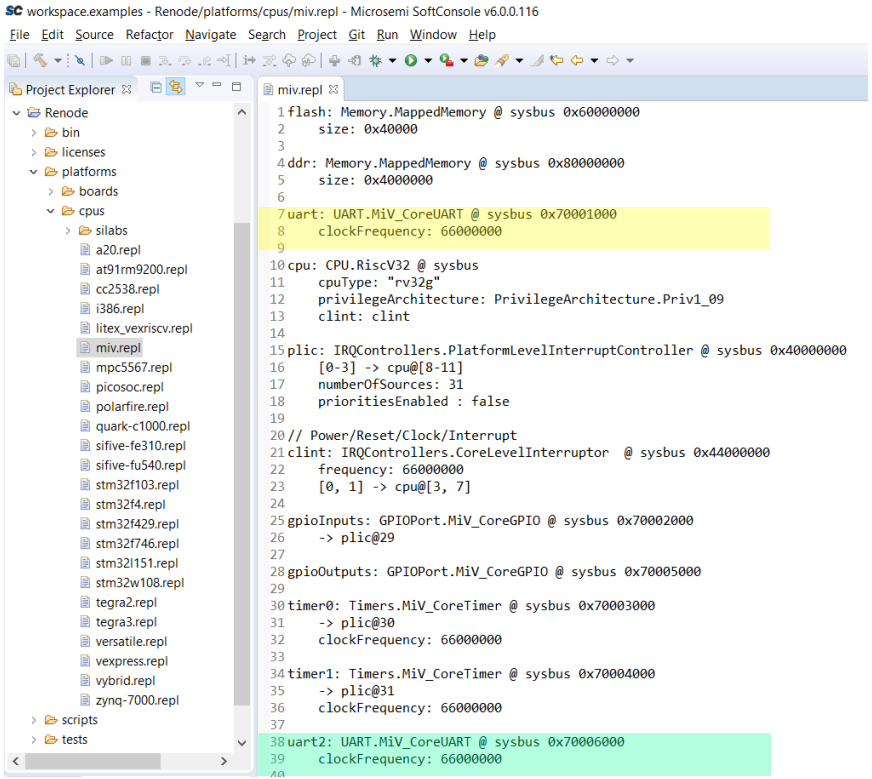
[name]: [path_to_peripheral] @ sysbus [address]
[parameters0]
[parameters1]
etc.

The CPU File


workspace.examples - Renode/platforms/cpus/miv.repl - Microsemi SoftConsole v6.0.0.116
File Edit Source Refactor Navigate Search Project Git Run Window Help
Project Explorer
Renode
bin
licenses
platforms
boards
cpus
silabs
a20.repl
at91rm9200.repl
cc2538.repl
i386.repl
litex_vexriscv.repl
miv.repl
mpc5567.repl
picosoc.repl
polarfire.repl
quark-c1000.repl
sifive-fe310.repl
sifive-fu540.repl
stm32f103.repl
stm32f4.repl
stm32f429.repl
stm32f746.repl
stm32l151.repl
stm32w108.repl
tegra2.repl
tegra3.repl
versatile.repl
vexpress.repl
vybrid.repl
zynq-7000.repl
miv.repl
1 flash: Memory.MappedMemory @ sysbus 0x60000000
2 size: 0x40000
3
4 ddr: Memory.MappedMemory @ sysbus 0x80000000
5 size: 0x4000000
6
7 uart: UART.MiV_CoreUART @ sysbus 0x70001000
8 clockFrequency: 66000000
9
10 cpu: CPU.RiscV32 @ sysbus
11 cpuType: "rv32g"
12 privilegeArchitecture: PrivilegeArchitecture.Priv1_09
13 clint: clint
14
15 plic: IRQControllers.PlatformLevelInterruptController @ sysbus 0x40000000
16 [0-3] -> cpu@[8-11]
17 numberOfSources: 31
18 prioritiesEnabled : false
19
20 // Power/Reset/Clock/Interrupt
21 clint: IRQControllers.CoreLevelInterruptor @ sysbus 0x44000000
22 frequency: 66000000
23 [0, 1] -> cpu@[3, 7]
24
25 gpioInputs: GPIOPort.MiV_CoreGPIO @ sysbus 0x70002000
26 -> plic@29
27
28 gpioOutputs: GPIOPort.MiV_CoreGPIO @ sysbus 0x70005000
29
30 timer0: Timers.MiV_CoreTimer @ sysbus 0x70003000
31 -> plic@30
32 clockFrequency: 66000000
33
34 timer1: Timers.MiV_CoreTimer @ sysbus 0x70004000
35 -> plic@31
36 clockFrequency: 66000000
1
2
3
4
5
6
7
8
9
10


sysbus (SystemBus)
clint (CoreLevelInterruptor)
<0x44000000, 0x4400FFFF>
6
cpu (RiscV32)
Slot: 0
4
ddr (MappedMemory)
<0x80000000, 0x83FFFFFF>
2
flash (MappedMemory)
<0x60000000, 0x6003FFFF>
1
gpioInputs (MiV_CoreGPIO)
<0x70002000, 0x700020A3>
7
user_switch_0 (Button)
Address: 0
user_switch_1 (Button)
Address: 1
user_switch_2 (Button)
Address: 2
gpioOutputs (MiV_CoreGPIO)
<0x70005000, 0x700050A3>
8
led0 (LED)
Address: 0
led1 (LED)
Address: 1
plic (PlatformLevelInterruptController)
<0x40000000, 0x43FFFFFF>
5
timer0 (MiV_CoreTimer)
<0x70003000, 0x7000301B>
9
timer1 (MiV_CoreTimer)
<0x70004000, 0x7000401B>
10
uart (MiV_CoreUART)
<0x70001000, 0x70001017>
3

The CPU File



Add a second UART

Current UART:

uart: UART.MiV_CoreUART @ sysbus 0x70001000

clockFrequency: 66000000

Second UART:

uart2: UART.MiV_CoreUART @ sysbus 0x70006000

clockFrequency: 66000000

The CPU File

workspace.examples - Renode/platforms/cpus/miv.repl - Microsemi SoftConsole v6.0.0.116

File Edit Source Refactor Navigate Search Project Git Run Window Help

Project Explorer

- Renode
  - bin
  - licenses
  - platforms
    - boards
    - cpus
      - silabs
      - a20.repl
      - at91rm9200.repl
      - cc2538.repl
      - i386.repl
      - litex_vexriscv.repl
      - miv.repl
      - mpc5567.repl
      - picosoc.repl
      - polarfire.repl
      - quark-c1000.repl
      - sifive-fe310.repl
      - sifive-fu540.repl
      - stm32f103.repl
      - stm32f4.repl
      - stm32f429.repl
      - stm32f746.repl
      - stm32l151.repl
      - stm32w108.repl
      - tegra2.repl
      - tegra3.repl
      - versatile.repl
      - vexpress.repl
      - vybrid.repl
      - zynq-7000.repl
  - scripts
  - tests

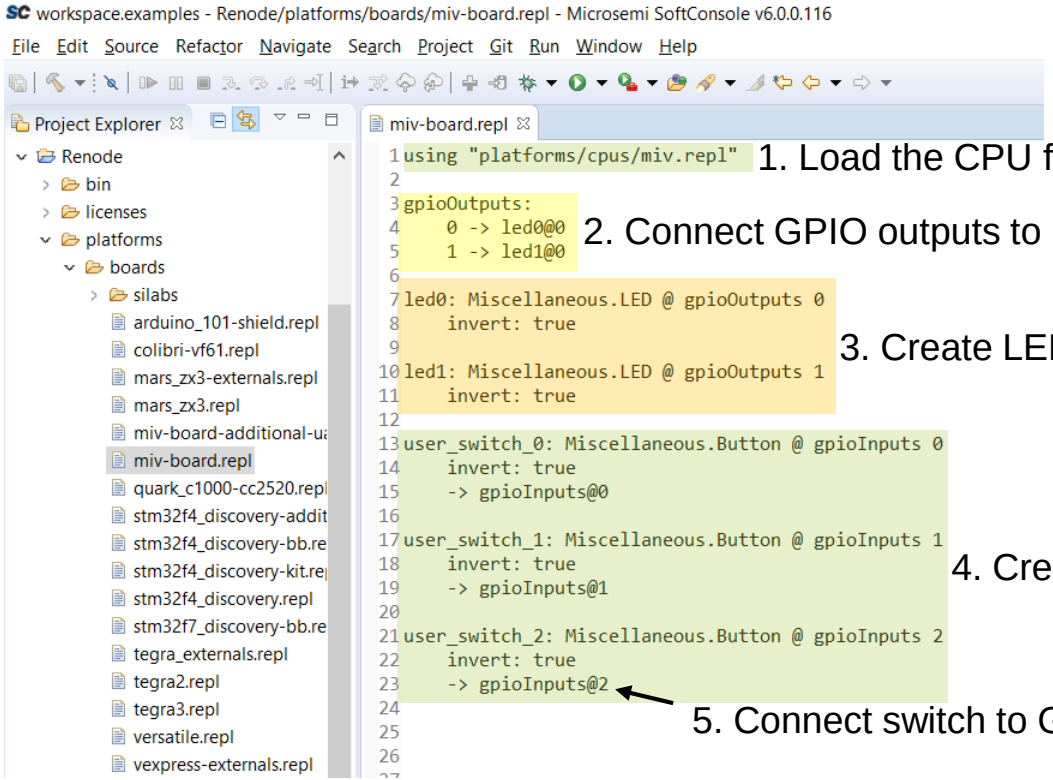
miv.repl

```
1  flash: Memory.MappedMemory @ sysbus 0x60000000                          1
2      size: 0x40000
3
4  ddr: Memory.MappedMemory @ sysbus 0x80000000                            2
5      size: 0x4000000
6
7  uart: UART.MiV_CoreUART @ sysbus 0x70001000                             3
8      clockFrequency: 66000000
9
10 cpu: CPU.RiscV32 @ sysbus                                               4
11     cpuType: "rv32g"
12     privilegeArchitecture: PrivilegeArchitecture.Priv1_09
13     clint: clint
14
15 plic: IRQControllers.PlatformLevelInterruptController @ sysbus 0x40000000  5
16     [0-3] -> cpu@[8-11]
17     numberOfSources: 31
18     prioritiesEnabled : false
19
20 // Power/Reset/Clock/Interrupt                                            6
21 clint: IRQControllers.CoreLevelInterruptor  @ sysbus 0x44000000
22     frequency: 66000000
23     [0, 1] -> cpu@[3, 7]
24
25 gpioInputs: GPIOPort.MiV_CoreGPIO @ sysbus 0x70002000                     7
26     -> plic@29
27
28 gpioOutputs: GPIOPort.MiV_CoreGPIO @ sysbus 0x70005000                   8
29
30 timer0: Timers.MiV_CoreTimer @ sysbus 0x70003000                         9
31     -> plic@30
32     clockFrequency: 66000000
33
34 timer1: Timers.MiV_CoreTimer @ sysbus 0x70004000                         10
35     -> plic@31
36     clockFrequency: 66000000
37
38 uart2: UART.MiV_CoreUART @ sysbus 0x70006000                             11
39     clockFrequency: 66000000
```

```
sysbus (SystemBus)
├─ clint (CoreLevelInterruptor)                                  6
│    <0x44000000, 0x4400FFFF>
├─ cpu (RiscV32)                                                 4
│    Slot: 0
├─ ddr (MappedMemory)                                            2
│    <0x80000000, 0x83FFFFFF>
├─ flash (MappedMemory)                                          1
│    <0x60000000, 0x6003FFFF>
├─ gpioInputs (MiV_CoreGPIO)                                     7
│  | <0x70002000, 0x700020A3>
│  ├─ user_switch_0 (Button)
│  │    Address: 0
│  ├─ user_switch_1 (Button)
│  │    Address: 1
│  └─ user_switch_2 (Button)
│       Address: 2
├─ gpioOutputs (MiV_CoreGPIO)                                    8
│  | <0x70005000, 0x700050A3>
│  ├─ led0 (LED)
│  │    Address: 0
│  └─ led1 (LED)
│       Address: 1
├─ plic (PlatformLevelInterruptController)                       5
│    <0x40000000, 0x43FFFFFF>
├─ timer0 (MiV_CoreTimer)                                        9
│    <0x70003000, 0x7000301B>
├─ timer1 (MiV_CoreTimer)                                        10
│    <0x70004000, 0x7000401B>
├─ uart (MiV_CoreUART)                                           3
│    <0x70001000, 0x70001017>
└─ uart2 (MiV_CoreUART)                                          11
     <0x70006000, 0x70006017>
```

The Board File

The Board File



1. Load the CPU file

2. Connect GPIO outputs to LEDs

3. Create LEDs

4. Create switches

5. Connect switch to GPIO

The Board File

SC workspace.examples - Renode/platforms/boards/miv-board.repl - Microsemi SoftConsole v6.0.0.116

File Edit Source Refactor Navigate Search Project Git Run Window Help

Project Explorer

- Renode
  - bin
  - licenses
  - platforms
    - boards
      - silabs
      - arduino_101-shield.repl
      - colibri-vf61.repl
      - mars_zx3-externals.repl
      - mars_zx3.repl
      - miv-board-additional-uarts.repl
      - miv-board.repl
      - quark_c1000-cc2520.repl
      - stm32f4_discovery-additional_gpios.repl
      - stm32f4_discovery-bb.repl
      - stm32f4_discovery-kit.repl
      - stm32f4_discovery.repl
      - stm32f7_discovery-bb.repl
      - tegra_externals.repl
      - tegra2.repl
      - tegra3.repl
      - versatile.repl
      - vexpress-externals.repl
      - vexpress.repl
      - zedboard-externals.repl
      - zedboard.repl
    - cpus
      - silabs
      - a20.repl
      - at91rm9200.repl
      - cc2538.repl
      - i386.repl
      - litex_vexriscv.repl
      - miv.repl
      - mpc5567.repl

miv.repl

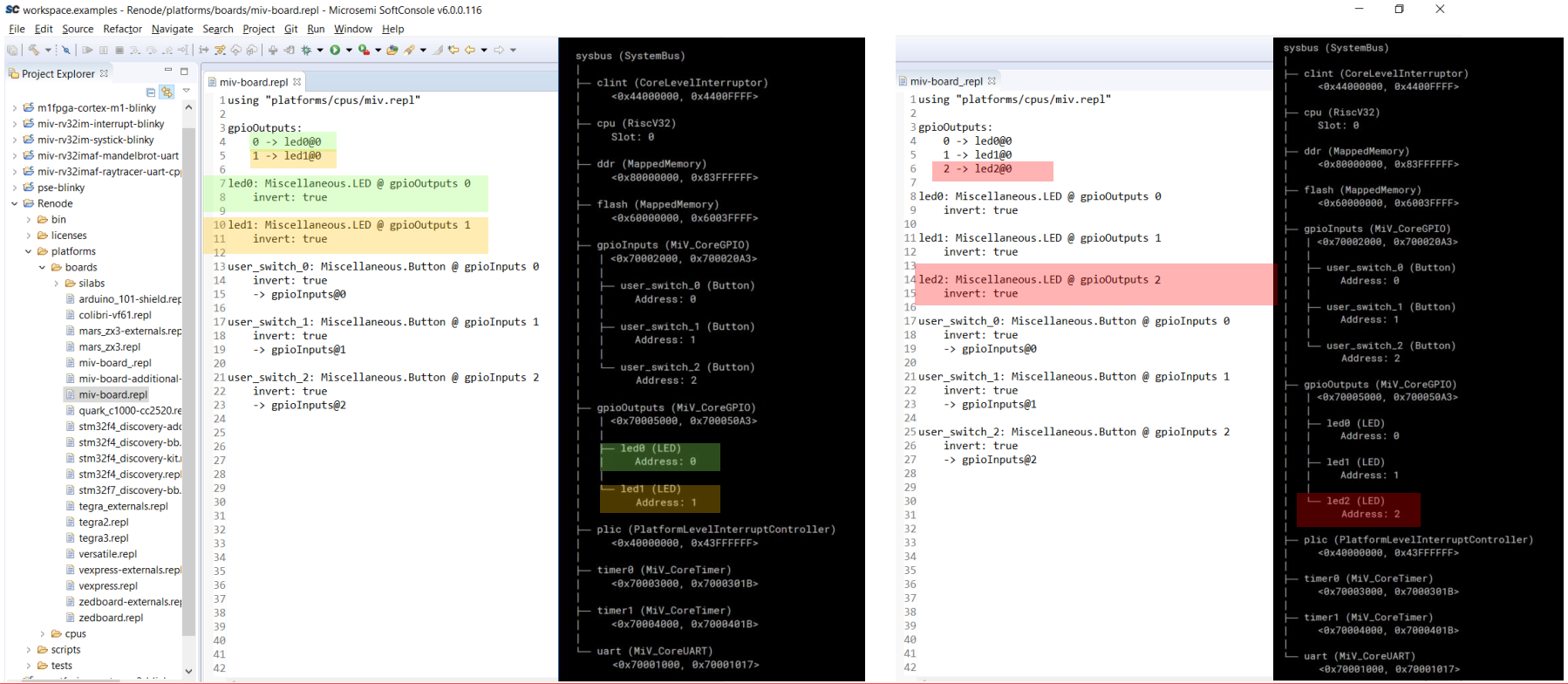
```
flash: Memory.MappedMemory @ sysbus 0x60000000
    size: 0x40000

ddr: Memory.MappedMemory @ sysbus 0x80000000
    size: 0x4000000

uart: UART.MiV_CoreUART @ sysbus 0x70001000
    clockFrequency: 66000000

cpu: CPU.RiscV32 @ sysbus
    cpuType: "rv32g"
    privilegeArchitecture: PrivilegeArchitecture.Priv1_09
    clint: clint

plic: IRQControllers.PlatformLevelInterruptController @ sysbus 0x40000000
    [0-3] -> cpu@[8-11]
    numberOfSources: 31
    prioritiesEnabled : false

// Power/Reset/Clock/Interrupt
clint: IRQControllers.CoreLevelInterruptor  @ sysbus 0x44000000
    frequency: 66000000
    [0, 1] -> cpu@[3, 7]

gpioInputs: GPIOPort.MiV_CoreGPIO @ sysbus 0x70002000
    -> plic@29

gpioOutputs: GPIOPort.MiV_CoreGPIO @ sysbus 0x70005000

timer0: Timers.MiV_CoreTimer @ sysbus 0x70003000
    -> plic@30
    clockFrequency: 66000000

timer1: Timers.MiV_CoreTimer @ sysbus 0x70004000
    -> plic@31
    clockFrequency: 66000000

uart2: UART.MiV_CoreUART @ sysbus 0x70006000
    clockFrequency: 66000000
```

miv-board.repl

```
using "platforms/cpus/miv.repl"

gpioOutputs:
    0 -> led0@0
    1 -> led1@0

led0: Miscellaneous.LED @ gpioOutputs 0
    invert: true

led1: Miscellaneous.LED @ gpioOutputs 1
    invert: true

user_switch_0: Miscellaneous.Button @ gpioInputs 0
    invert: true
    -> gpioInputs@0

user_switch_1: Miscellaneous.Button @ gpioInputs 1
    invert: true
    -> gpioInputs@1

user_switch_2: Miscellaneous.Button @ gpioInputs 2
    invert: true
    -> gpioInputs@2
```

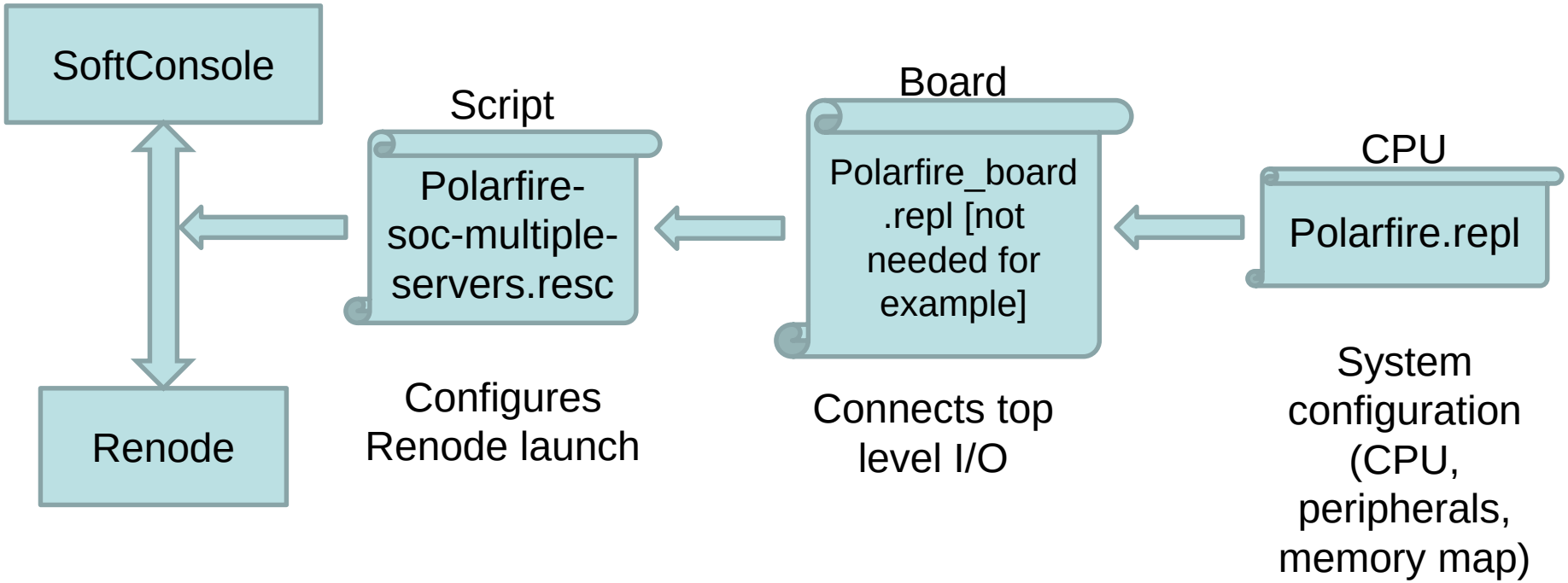
```
sysbus (SystemBus)
    |
    ├─ clint (CoreLevelInterruptor)
    |     <0x44000000, 0x4400FFFF>
    |
    ├─ cpu (RiscV32)
    |     Slot: 0
    |
    ├─ ddr (MappedMemory)
    |     <0x80000000, 0x83FFFFFF>
    |
    ├─ flash (MappedMemory)
    |     <0x60000000, 0x6003FFFF>
    |
    ├─ gpioInputs (MiV_CoreGPIO)
    |   | <0x70002000, 0x700020A3>
    |   |
    |   ├─ user_switch_0 (Button)
    |   |     Address: 0
    |   |
    |   ├─ user_switch_1 (Button)
    |   |     Address: 1
    |   |
    |   └─ user_switch_2 (Button)
    |         Address: 2
    |
    ├─ gpioOutputs (MiV_CoreGPIO)
    |   | <0x70005000, 0x700050A3>
    |   |
    |   ├─ led0 (LED)
    |   |     Address: 0
    |   |
    |   ├─ led1 (LED)
    |   |     Address: 1
    |
    ├─ plic (PlatformLevelInterruptController)
    |     <0x40000000, 0x43FFFFFF>
    |
    ├─ timer0 (MiV_CoreTimer)
    |     <0x70003000, 0x7000301B>
    |
    ├─ timer1 (MiV_CoreTimer)
    |     <0x70004000, 0x7000401B>
    |
    └─ uart (MiV_CoreUART)
          <0x70001000, 0x70001017>
```

The Board File

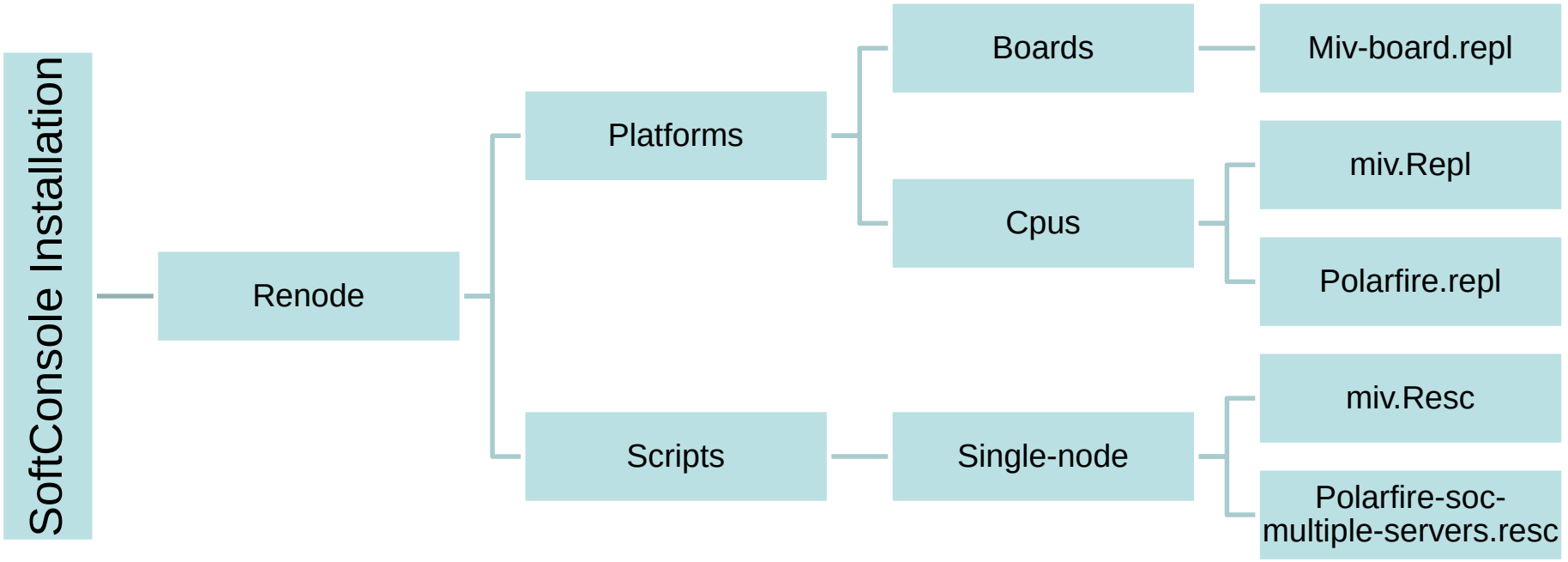


Summary

Summary

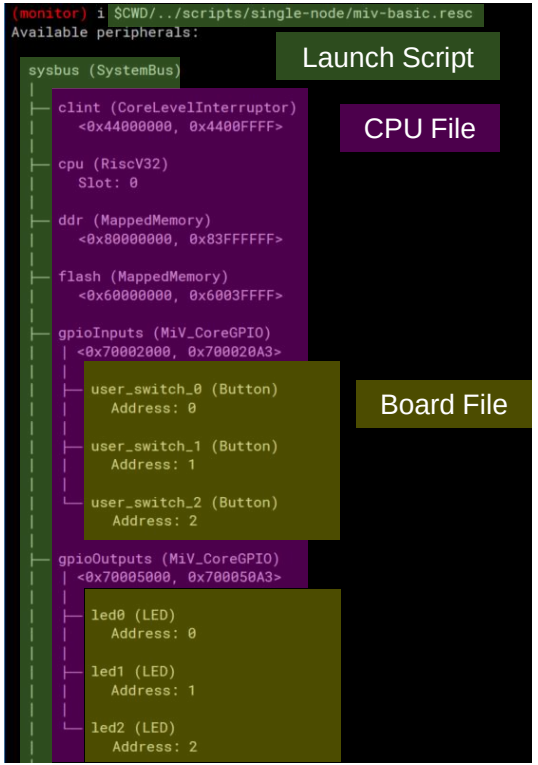

SoftConsole
Renode
Script
Polarfire-soc-multiple-servers.resc
Configures Renode launch
Board
Polarfire_board.repl [not needed for example]
Connects top level I/O
CPU
Polarfire.repl
System configuration (CPU, peripherals, memory map)

Summary


SoftConsole Installation
Renode
Platforms
Boards
Miv-board.repl
Cpus
miv.Repl
Polarfire.repl
Scripts
Single-node
miv.Resc
Polarfire-soc-multiple-servers.resc

Summary

1. **Launch the Renode external tool configuration**
2. **Renode starts**
   A. Loads the script file passed to it as an argument
   B. Script creates a machine
   C. Script tells Renode to load a board file
   D. Board file tells Renode to load a CPU file
   E. Renode loads the CPU file
   F. Renode loads the board file
   G. Script tells Renode to start the GDB server
   H. Script tells Renode to print the started message in the console
3. **SoftConsole starts GDB**



Summary:
Available Peripherals Mi-V system

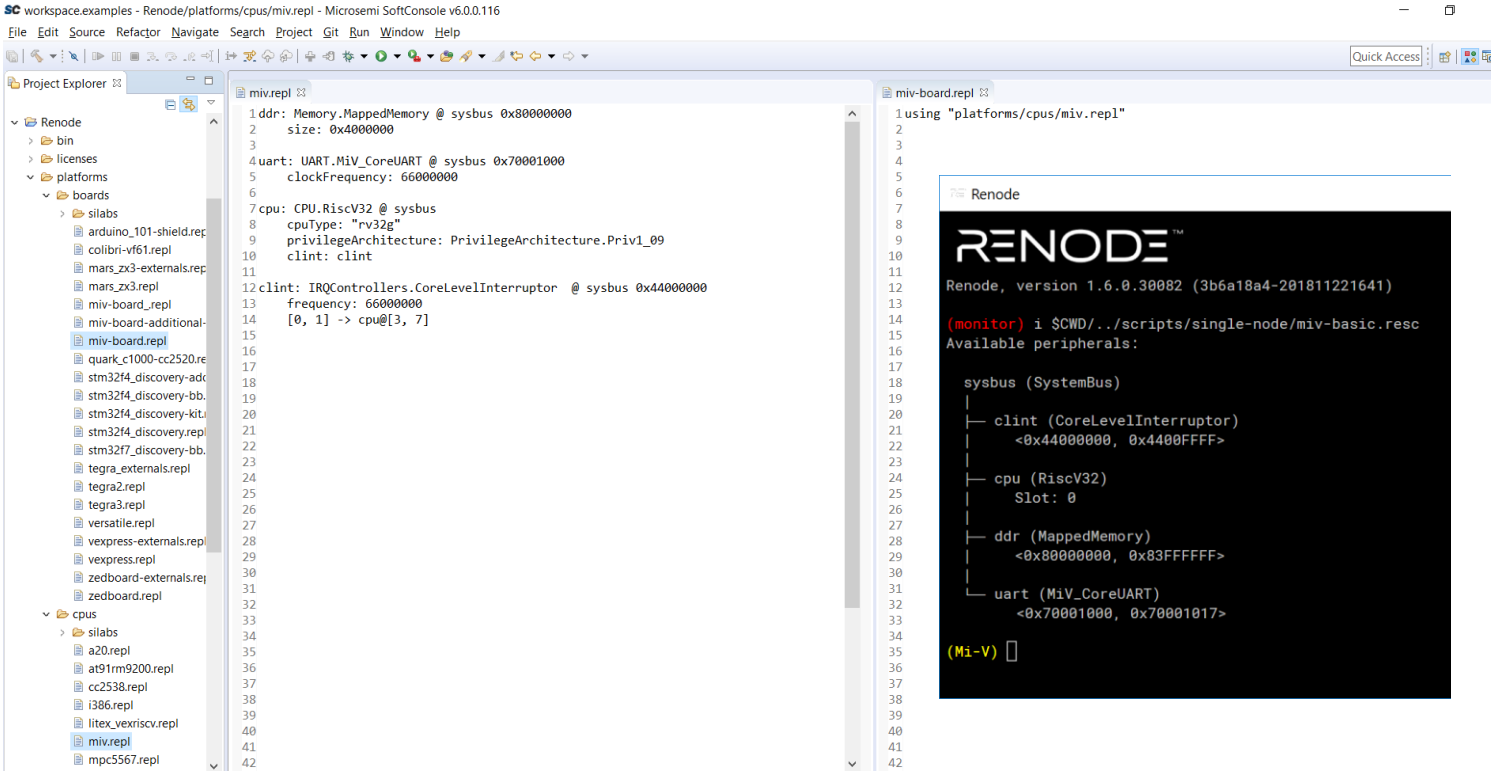
Memory	Cores	Peripherals
DDR	RV32G	CoreUART
Flash		PLIC
		CLINT
		CoreGPIO
		CoreTimer

Summary:
Available Peripherals PolarFire SoC system

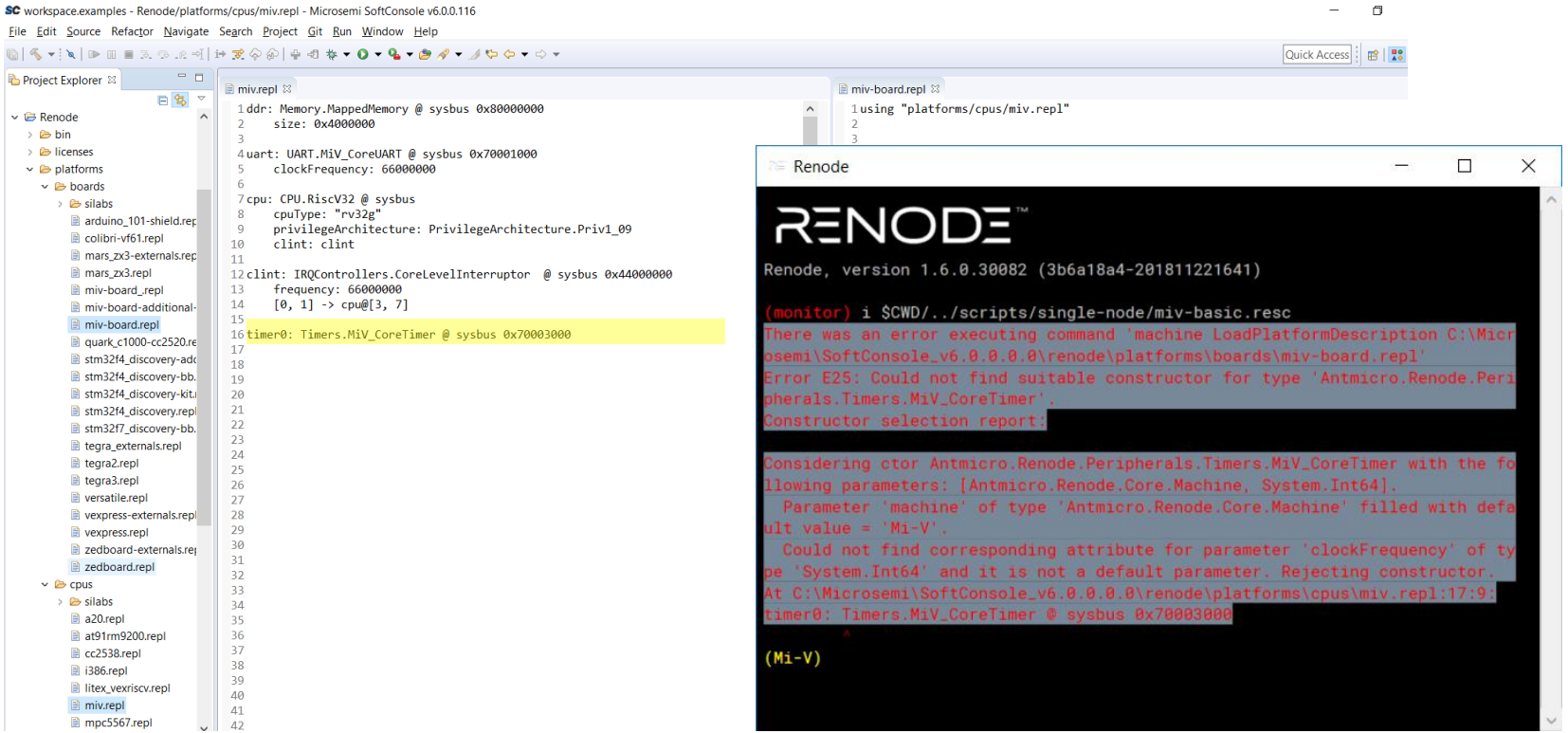
Cores	Memory	Peripherals	Peripherals contd.
RV64IMAC	DDR	CLINT	Cadence GEM (MAC)
RV64GC	Flash	PLIC	PSE GPIO
	PSE eNVM	NS16550 UART	PSE RTC
		PSE SD Controller	PSE Timer
		PSE SPI	PSE USB
		PSE I2C	PSE PCI
		PSE CAN	

Debugging Adding Peripherals

Debugging Adding Peripherals

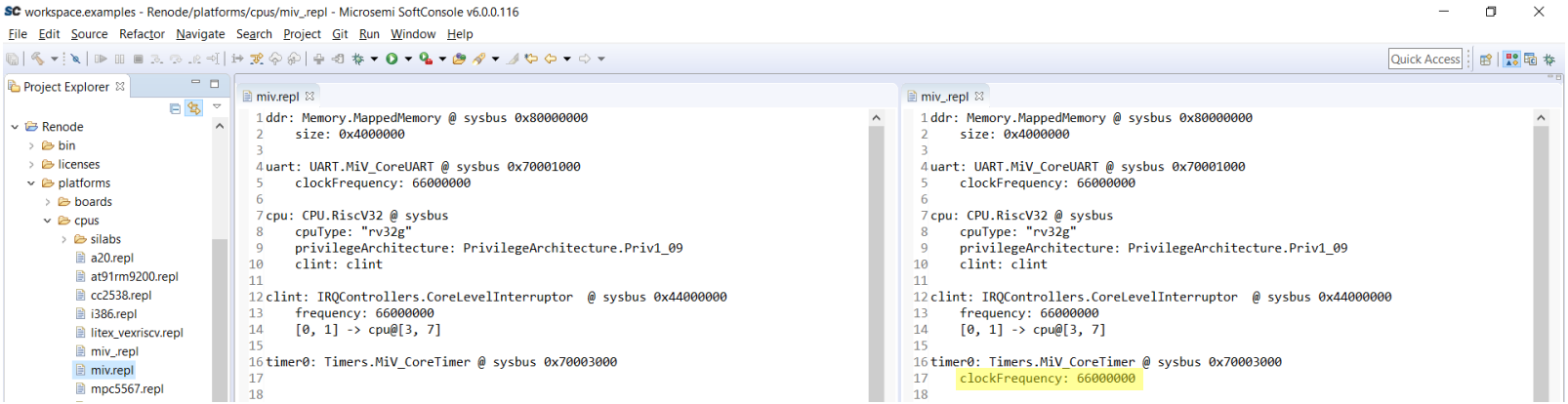


Debugging Adding Peripherals

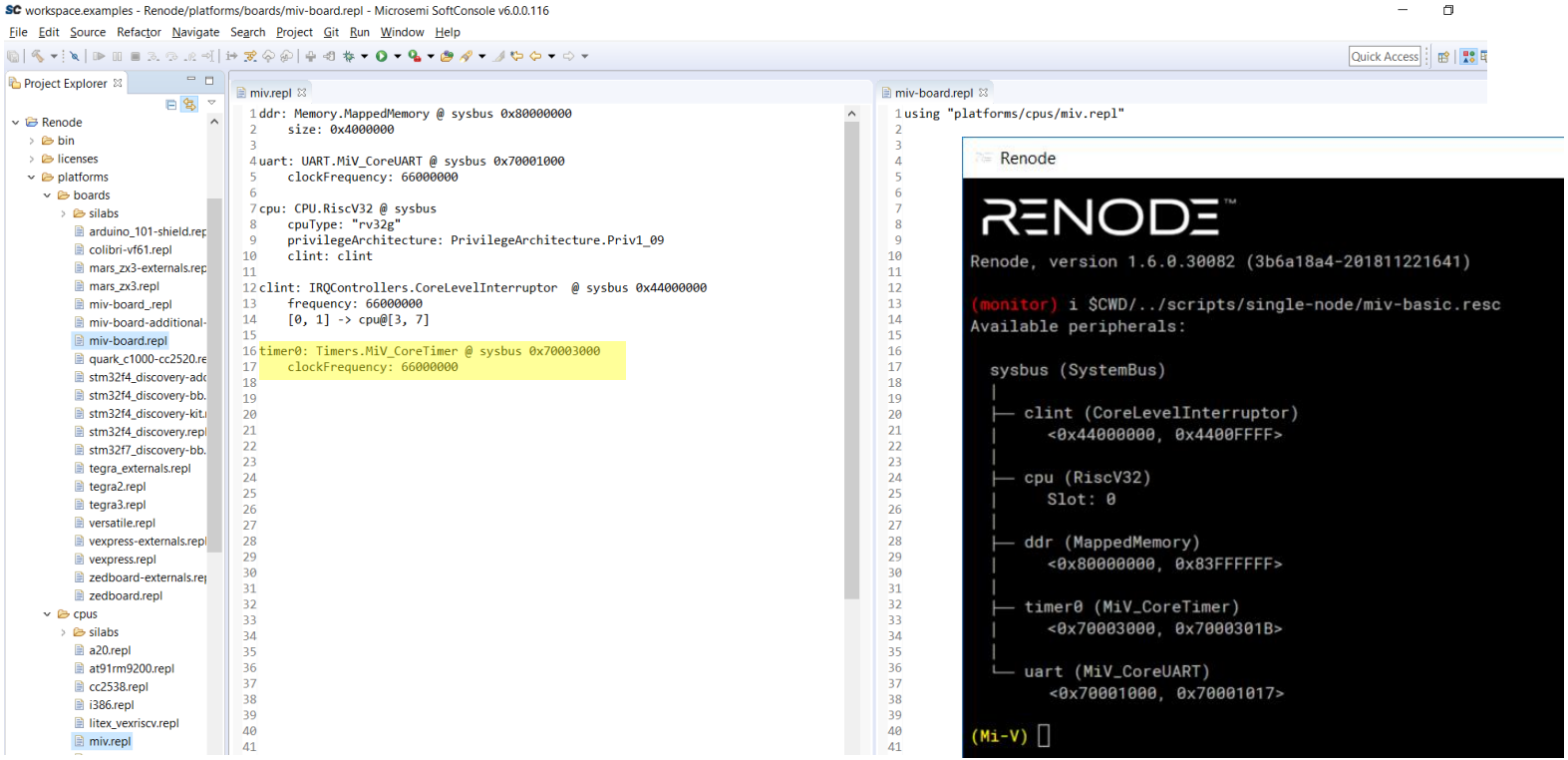


Debugging Adding Peripherals

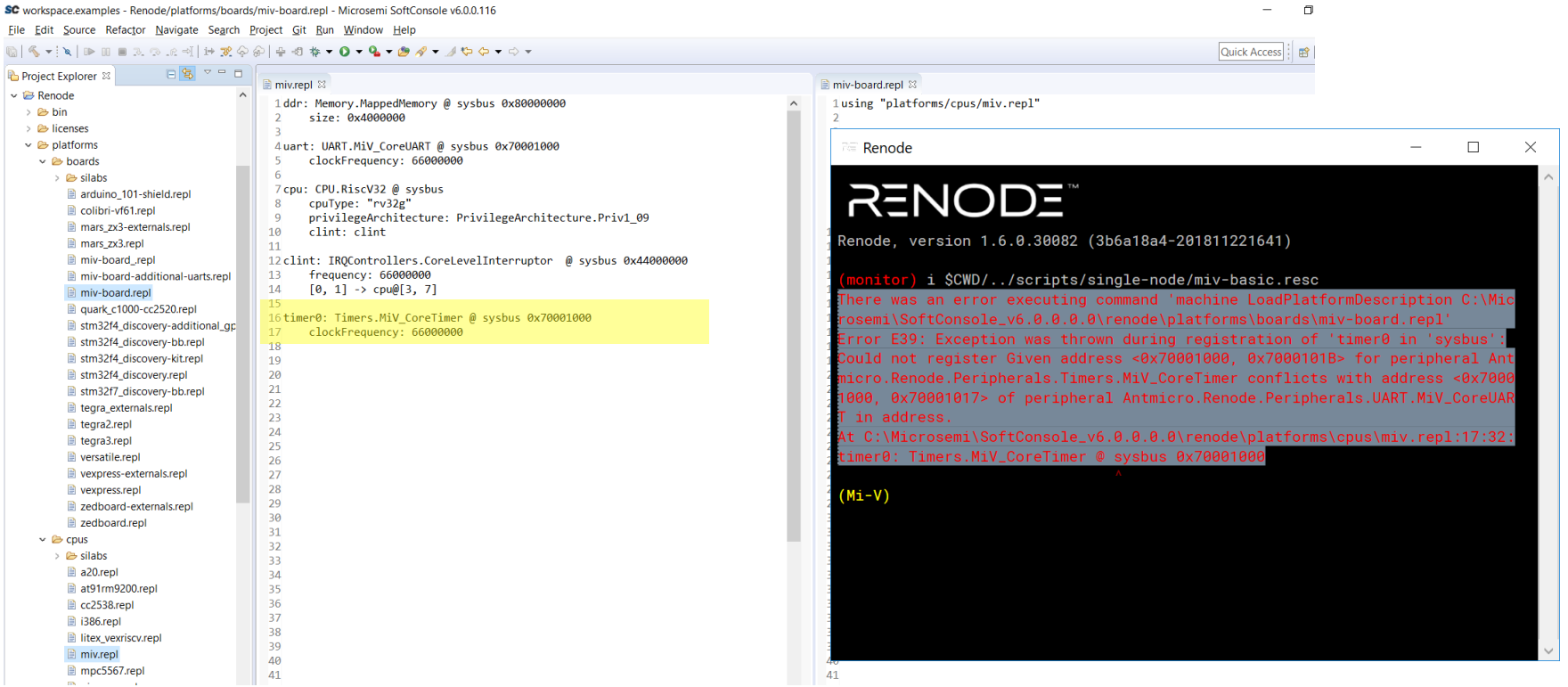
1. There was an error executing command 'machine LoadPlatformDescription C:\Microsemi\SoftConsole_v6.0.0.0.0\renode\platforms\boards\miv-board.repl'
2. Error E25: Could not find suitable constructor for type 'Antmicro.Renode.Peripherals.Timers.MiV_CoreTimer'.
    A. Constructor selection report:
    B. Considering ctor Antmicro.Renode.Peripherals.Timers.MiV_CoreTimer with the following parameters: [Antmicro.Renode.Core.Machine, System.Int64].
    C. Parameter 'machine' of type 'Antmicro.Renode.Core.Machine' filled with default value = 'Mi-V'.
    D. Could not find corresponding attribute for parameter 'clockFrequency' of type 'System.Int64' and it is not a default parameter. Rejecting constructor.
    E. At C:\Microsemi\SoftConsole_v6.0.0.0.0\renode\platforms\cpus\miv.repl:17:9:
    F. timer0: Timers.MiV_CoreTimer @ sysbus 0x70003000



Debugging Adding Peripherals

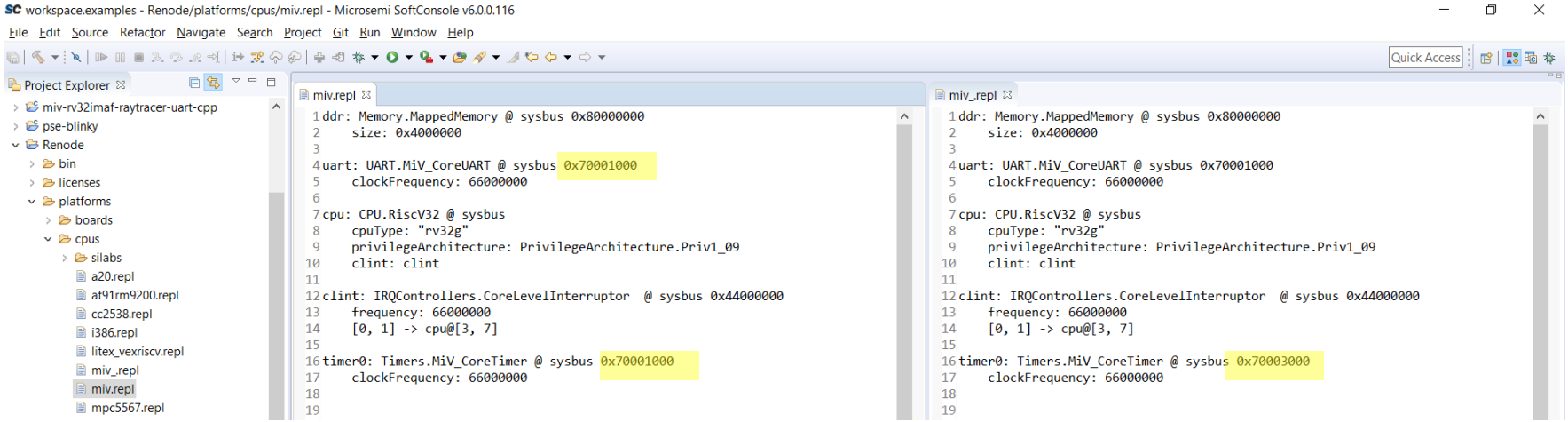


Debugging Adding Peripherals

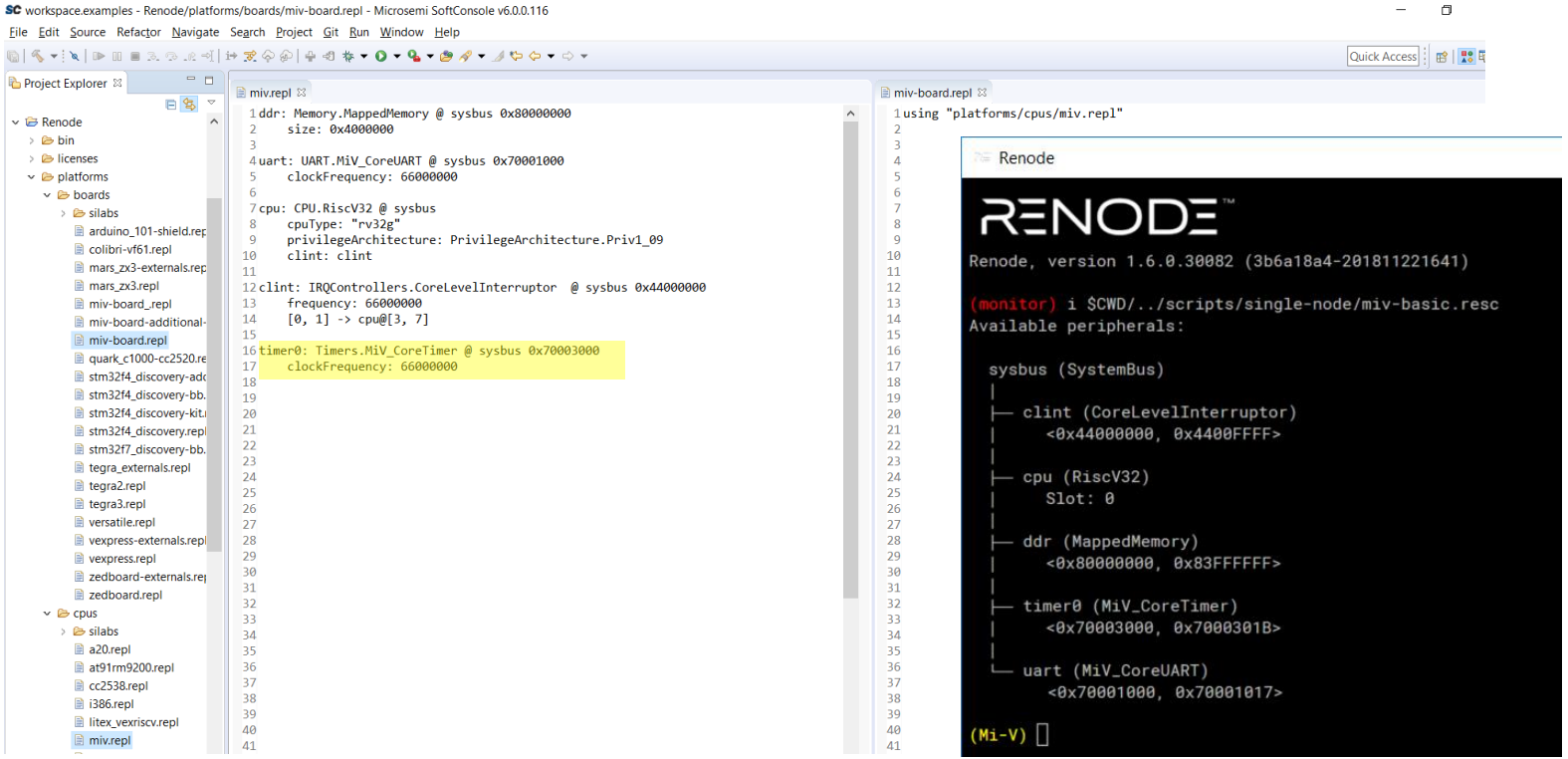


Debugging Adding Peripherals

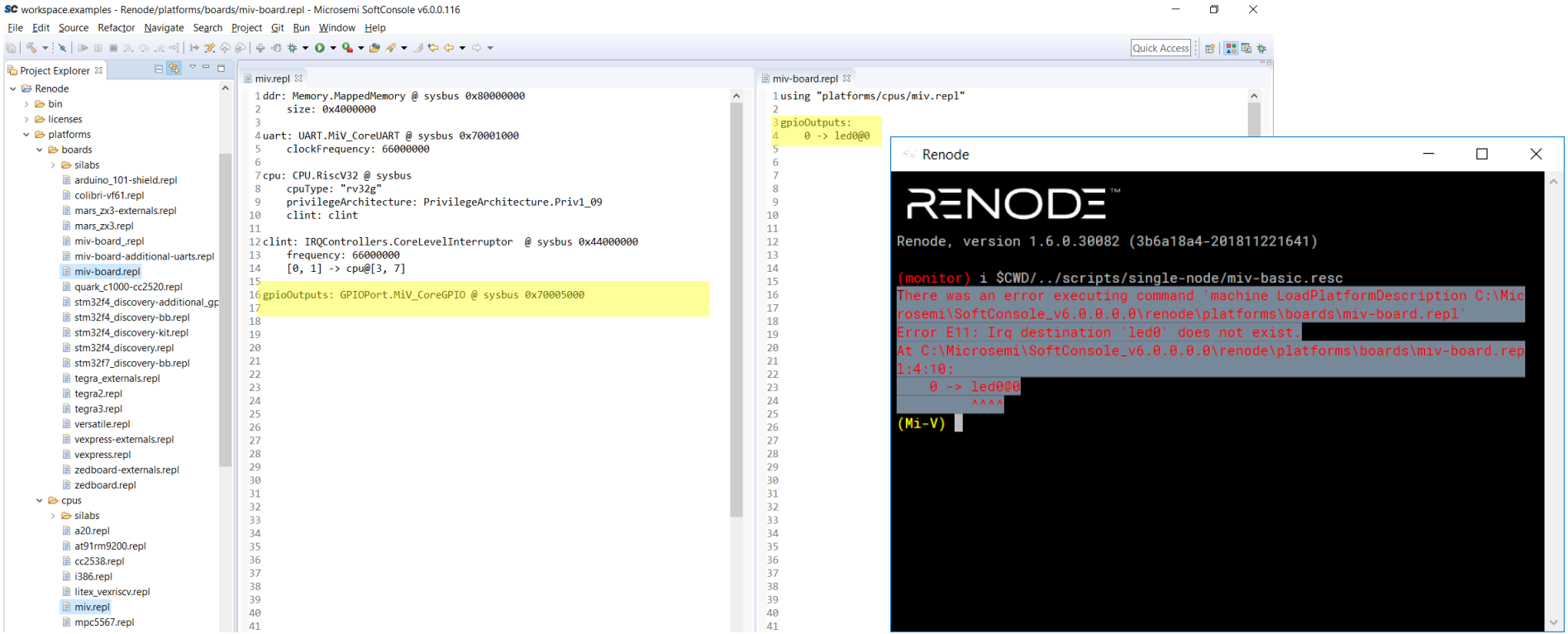
1. There was an error executing command 'machine LoadPlatformDescription C:\Microsemi\SoftConsole_v6.0.0.0.0\renode\platforms\boards\miv-board.repl'
2. Error E39: Exception was thrown during registration of 'timer0 in 'sysbus':
    1. Could not register Given address <0x70001000, 0x7000101B> for peripheral Antmicro.Renode.Peripherals.Timers.MiV_CoreTimer conflicts with address <0x70001000, 0x70001017> of peripheral Antmicro.Renode.Peripherals.UART.MiV_CoreUART in address.
    2. At C:\Microsemi\SoftConsole_v6.0.0.0.0\renode\platforms\cpus\miv.repl:17:32:
    3. timer0: Timers.MiV_CoreTimer @ sysbus 0x70001000



Debugging Adding Peripherals

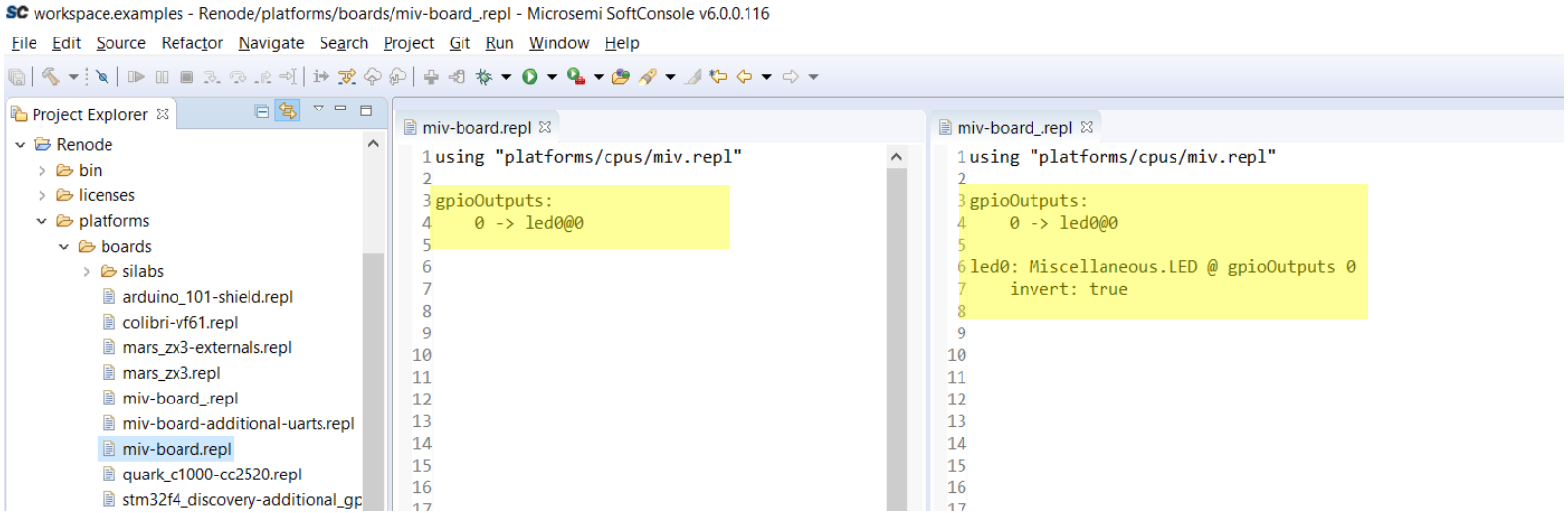


Debugging Adding Peripherals

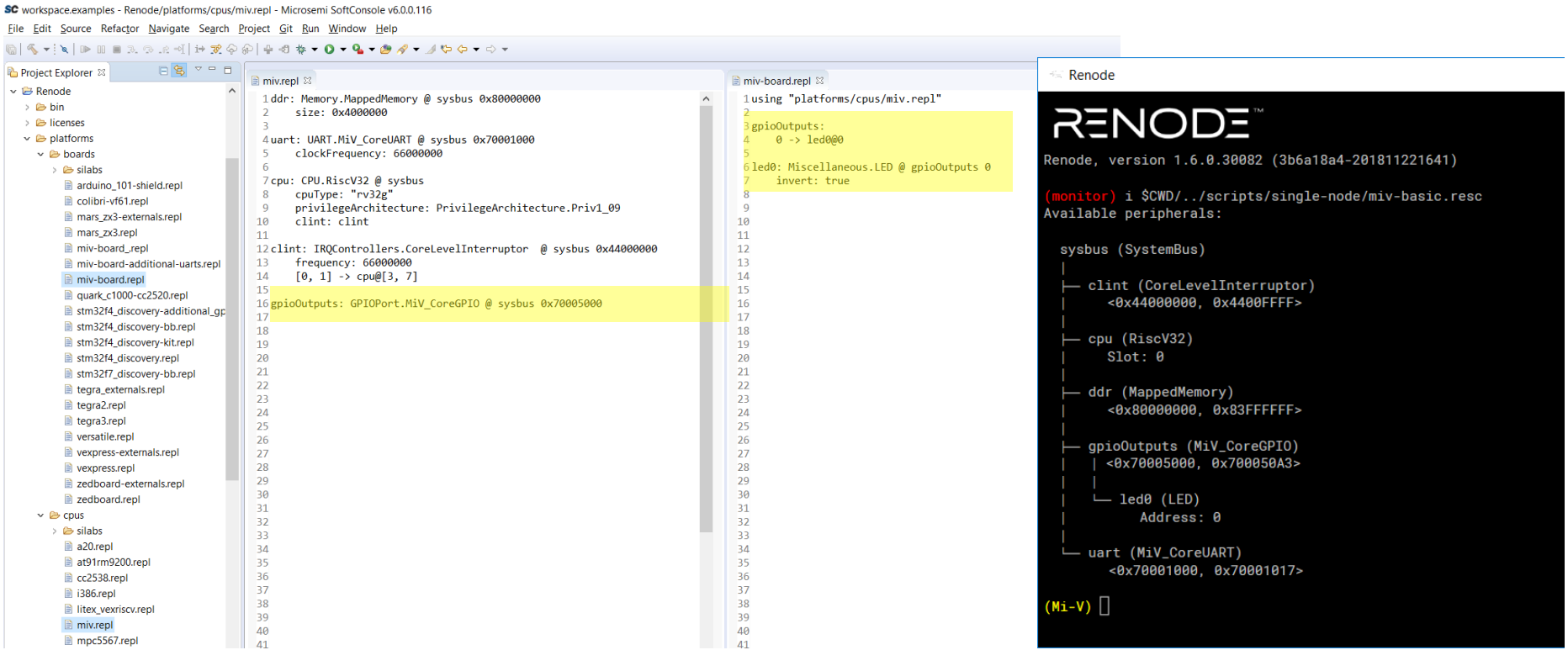


Debugging Adding Peripherals

1. There was an error executing command 'machine LoadPlatformDescription C:\Microsemi\SoftConsole_v6.0.0.0.0\renode\platforms\boards\miv-board.repl'
2. Error E11: Irq destination 'led0' does not exist.
    1. At C:\Microsemi\SoftConsole_v6.0.0.0.0\renode\platforms\boards\miv-board.repl:4:10:
    2. 0 -> led0@0
    3. ^^^^



Debugging Adding Peripherals



First Thursdays

May 2 - Webinar 1: Discover Renode for PolarFire® SoC Design and Debug

June 6 - Webinar 2: How to Get Started with Renode for PolarFire SoC

July 4 - Webinar 3: Learn to Debug a Bare-Metal PolarFire SoC Application with Renode

Aug. 1 - Webinar 4: Tips and Tricks for Even Easier PolarFire SoC Debug with Renode

Sept. 5 - Webinar 5: Add and Debug PolarFire SoC Peripherals with Renode

Oct. 3 - Webinar 6: Add and Debug and Pre-Existing Peripheral in PolarFire SoC

Nov. 7 - Webinar 7: How to write custom models – filters, offloading, acceleration etc

Dec. 5 - Webinar 8: Handling Binaries

Contd.

Second Thursdays

Jan. 9 - Webinar 9: Run Linux on Renode (PolarFire SoC Model as a Quad-core SMP) – this is not a Linux / Buildroot tutorial

Feb. 13 - Webinar 10: Build applications for Linux on PolarFire SoC

Mar. 12 - Webinar 11: Introduction to PolarFire SoC MSS Configuration and Software Flow

Apr. 9 - Webinar 12: Two baremetal Applications on PolarFire SoC

May 14 - Webinar 13: Linux + Real-Time (AMP Mode) on PolarFire SoC

Thank You