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

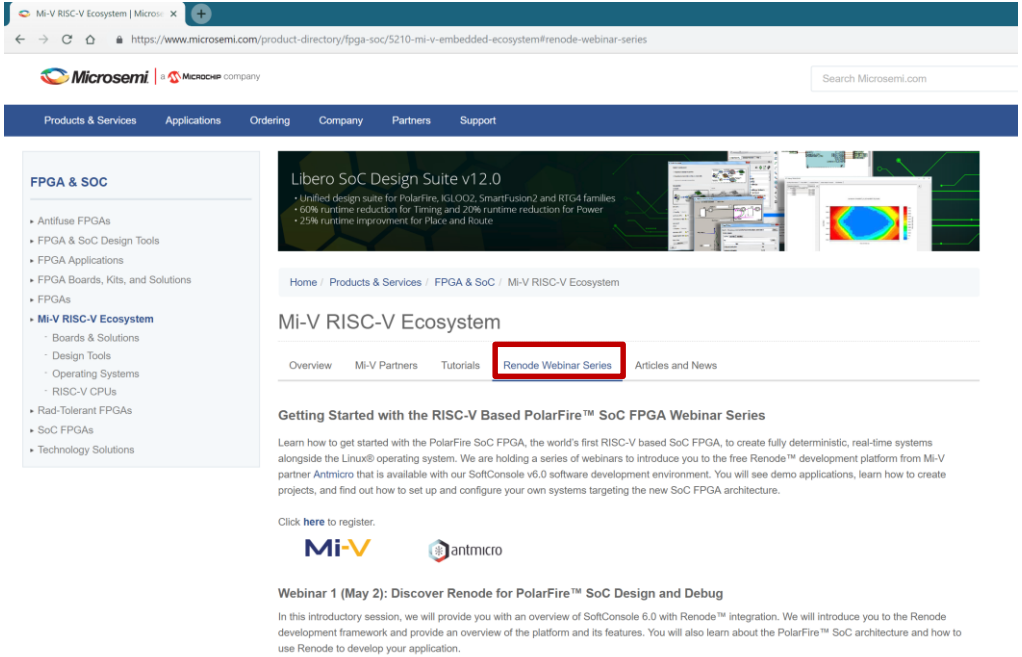
July 4 - Webinar 3: Learn to Debug a Bare-Metal PolarFire SoC Application with Renode

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

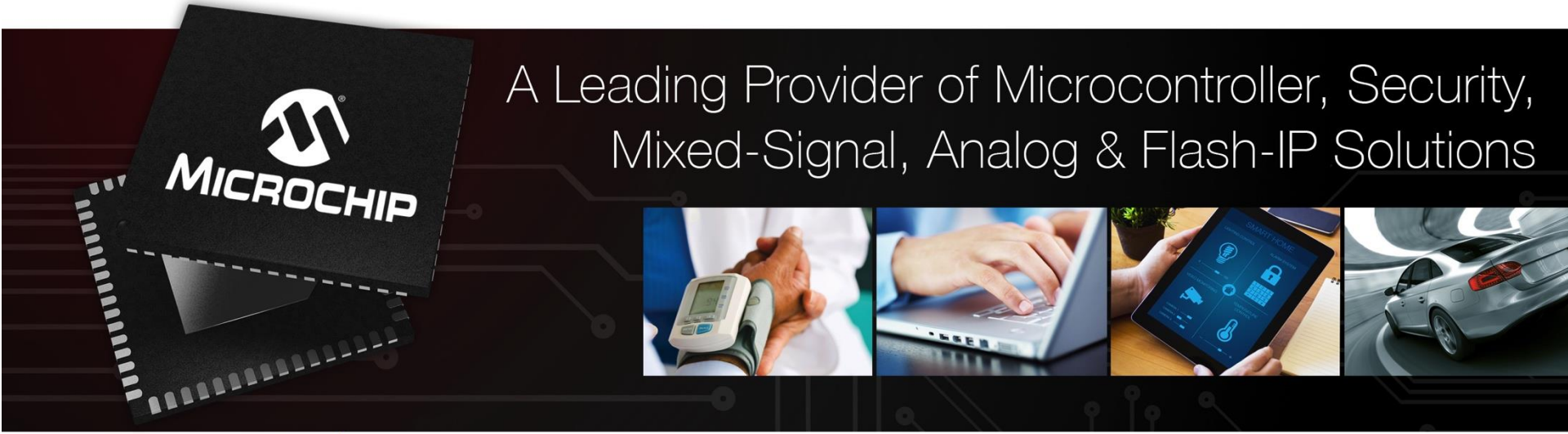
Sept. 5 - Webinar 5: Add and Debug PolarFire SoC Peripherals with Renode

Oct. 3 - Webinar 6: Add and debug a pre-existing peripheral in PolarFire SoC

Supporting Content



www.microsemi.com/Mi-V “Renode Webinar Series”



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

Session 4: Tips and Tricks for Even Easier PolarFire SoC Debug with Renode

Hugh Breslin, Embedded Linux Engineer

Thursday Aug. 1, 2019

Tips and Tricks for Even Easier PolarFire® SoC Debug with Renode

- **Recap**
- **Using the Renode console**
- **Using Renode logs and their output**

Recap

- **Webinar 1: Discover Renode for PolarFire® SoC Design and Debug**
  - Introduction to the Renode platform and PolarFire SoC
- **Webinar 2: How to Get Started with Renode for PolarFire SoC**
  - Installing SoftConsole with Renode on Windows® and Linux®
  - Demo of the Mi-V example projects
- **Webinar 3: Learn to Debug a Bare-Metal PolarFire SoC Application with Renode**
  - How to configure debug sessions for projects
  - How to run and debug the pse-blinky project

Recap

- Install SoftConsole v6.0 with Renode
- Able to launch the included sample projects and debug in Renode
- Set up debug configurations and configure launch groups for use with Renode
- Basic use of the Renode console
- Run PolarFire SoC applications on multiple harts

Using The Renode Console

Using the Renode Console

- **Renode commands and peripheral names are case sensitive**

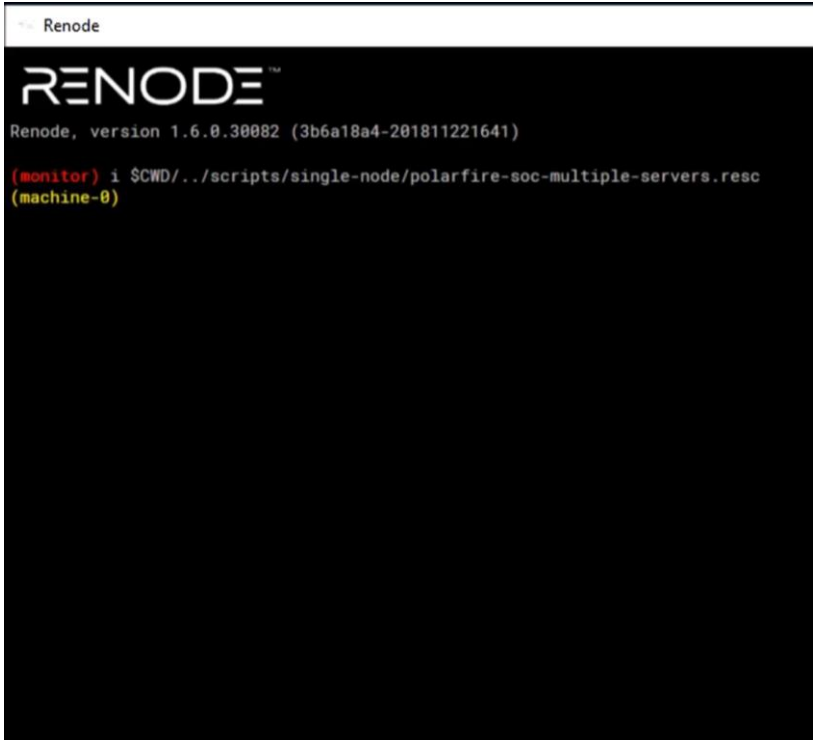
- **Incorrectly capitalizing a command or peripheral name will cause an error even if spelling is correct!**

e.g.:

sysbus.e51 – ok!

sysbus.E51 – not ok

Sysbus.e51 – not ok



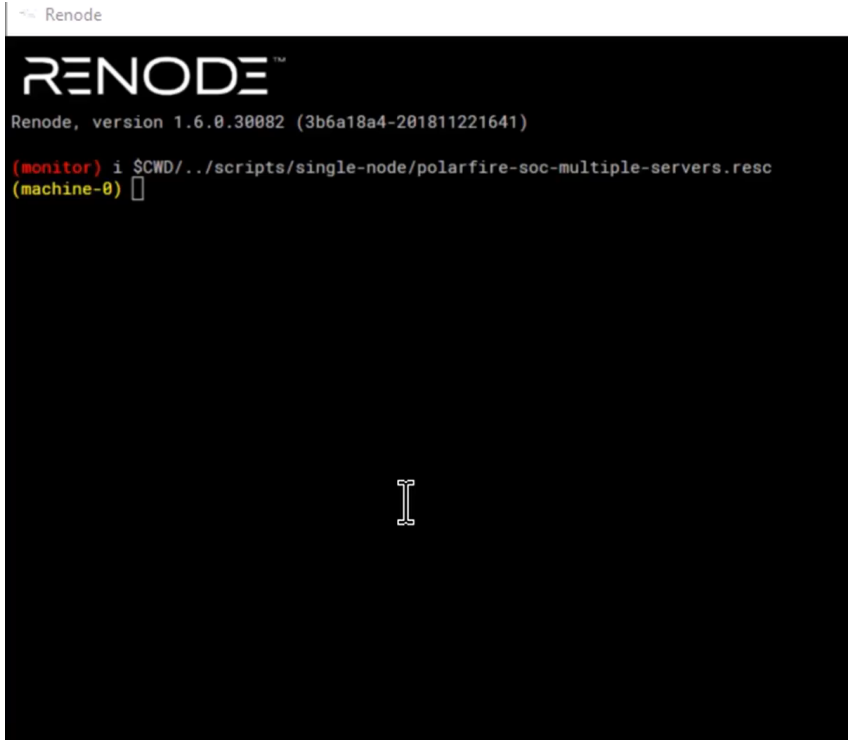
Using the Renode Console

Command: help

Use / Example: “help” “help [command]” “help logLevel”

Output: Prints available commands if no arguments and prints information about the command and usage examples if a command is given as an argument

Notes: Cannot be used for peripherals i.e. “help e51” will produce an error



Using the Renode Console

Command: peripherals

Use/Example: “peripherals”

Output: Prints peripherals connected to sysbus

Notes: Prints sysbus for the current machine

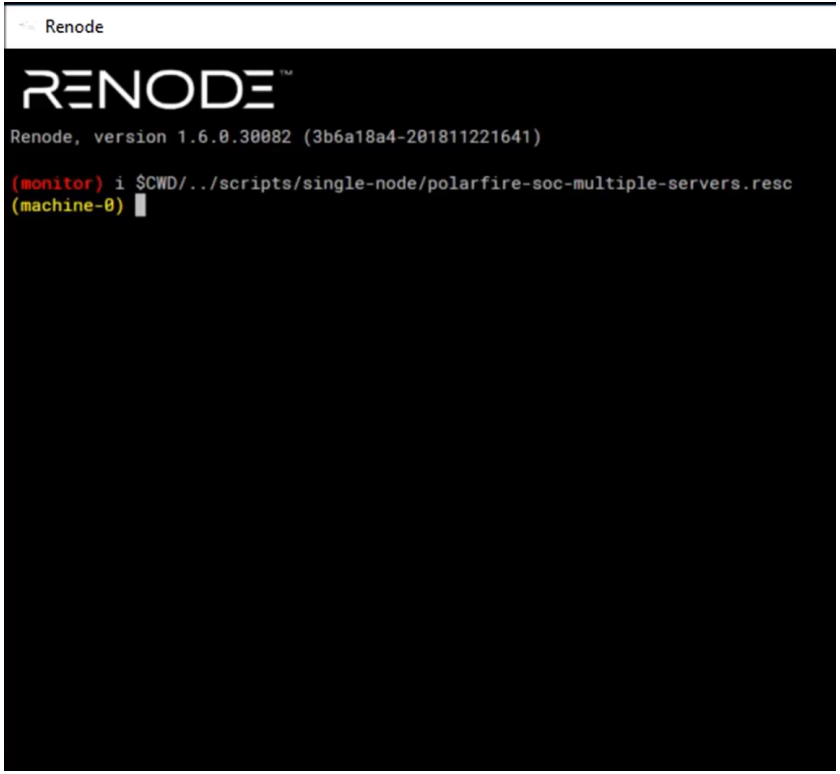
Using the Renode Console

Command: using sysbus

Use/Example: “using sysbus”

Output: N/A

Notes: Tells Renode to assume you’re using sysbus



Using the Renode Console

Command: [peripheral name]

Use/Example: “e51”

Output: Prints methods and properties available to be used with the peripheral

Notes: Methods can be called to set or change functionality of a peripheral (e.g. reset) and properties are values describing the peripheral or its state (e.g. Architecture or IsHalted)

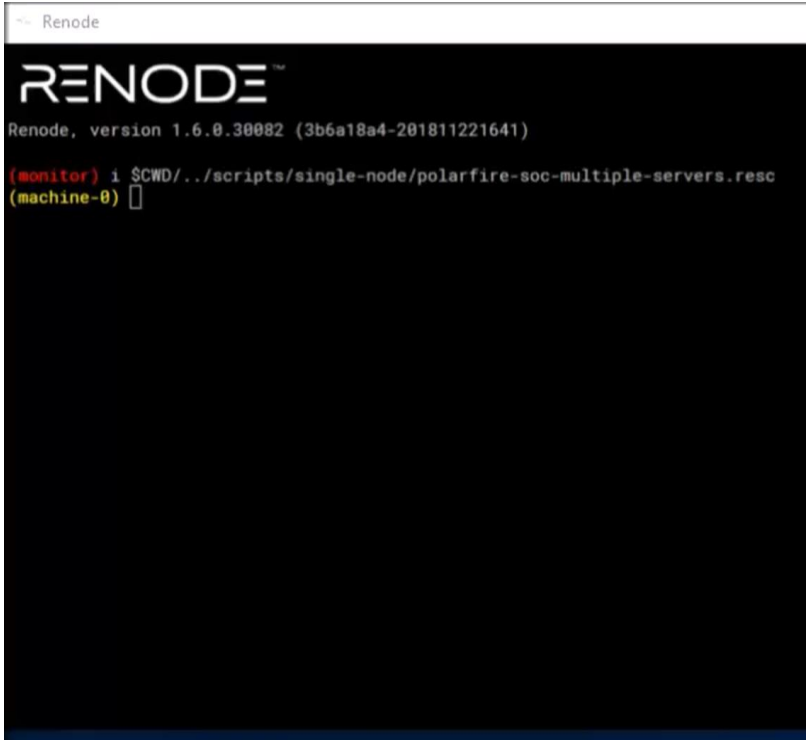
Using the Renode Console

Command: [peripheral name] [method] [args]

Use/Example: “e51 AddHook 0x60000000 @main.py”
“e51 GetRegistersValues”

Output: Depends on the return type of the method (if any)

Notes:



Using the Renode Console

Command: [peripheral name] [method] [args]

e.g.: - String GetCPUThreadName (Machine machine)

- [return value] {command} ([arg type] arg, [arg type] arg)

- **Void AddHook (UInt64 addr, String pythonScript)**
- **String DisassembleBlock (UInt32 addr, Uint32 flags = 0)**
- **Boolean HasGPIO()**
- **Boolean IsSetEvent(Int32 number)**

Void: No return value, String: Returns a string, Boolean: Returns true / false

Renode

```
RENODE™
Renode, version 1.6.0.30082 (3b6a18a4-201811221641)

(monitor) i $CWD/../scripts/single-node/polarfire-soc-multiple-servers.resc
(machine-0) sysbus.e51

The following methods are available:
 - Void AddHook (UInt64 addr, String pythonScript)
 - Void ClearHookAtBlockBegin ()
 - Void ClearPageAccessViaIo (UInt64 address)
 - Void ClearTranslationCache ()
 - Void DebugLog (String message)
 - String DisassembleBlock (UInt32 addr, UInt32 flags = 0)
 - Void Dispose ()
 - String GetCPUThreadName (Machine machine)
 - IEnumerable<Tuple<String,IGPIO>> GetGPIOs ()
 - Machine GetMachine ()
 - IEnumerable<CPURegister> GetRegisters ()
 - String[,] GetRegistersValues ()
 - RegisterValue GetRegisterUnsafe (Int32 register)
 - Boolean HasGPIO ()
 - Boolean IsInstructionSetEnabled (InstructionSet set)
 - Boolean IsSetEvent (Int32 number)
 - Void Log (LogLevel type, String message)
 - Void LogFunctionNames (Boolean value, String spaceSeparatedPrefixes = "")
 - Void LogUnhandledRead (Int64 offset)
 - Void LogUnhandledWrite (Int64 offset, Int64 value)
 - Void NoisyLog (String message)
 - Void OnGPIO (Int32 number, Boolean value)
 - Void Pause ()
 - Void RemoveAllHooks ()
 - Void RemoveHooksAt (UInt64 addr)
 - Void Reset ()
 - Void Resume ()
```

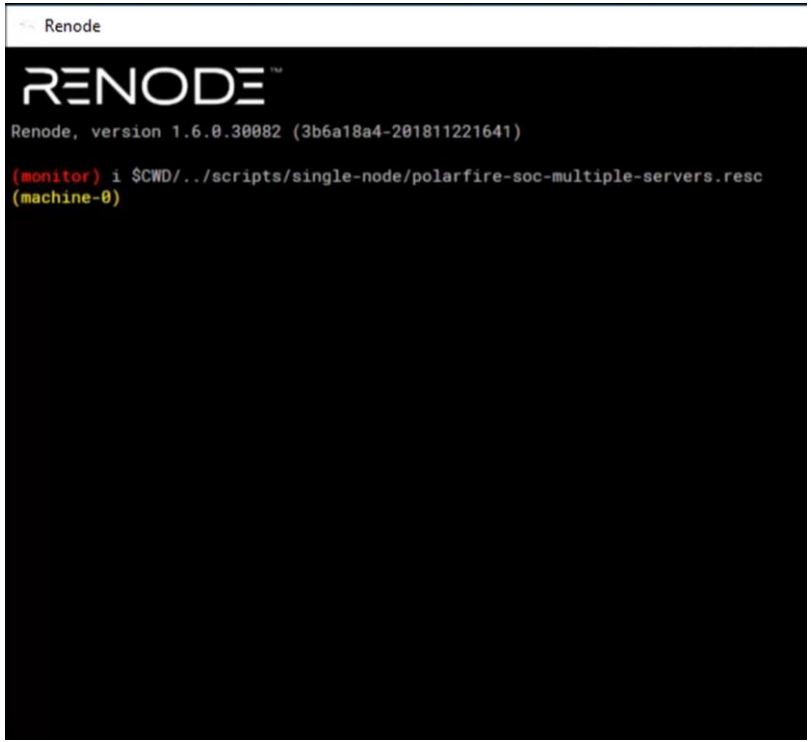
Using the Renode Console

Command: [peripheral name] [property] [args]

Use / Example: “e51 Architecture”, “e51 IsHalted”, “e51 IsHalted True”

Output: None if setting, will print value if using get

Notes: Some properties are read-only and some can be set during execution, you can tell as read-only will only have “get” available and read and write properties will have “get and set” available



Using the Renode Console

Command: [peripheral name] [property] [args]

- **String Architecture**

  Available for 'get'

- **String[] AvailableDisassemblers**

  Available for 'get'

- **Boolean ChainingEnabled**

  Available for 'get' and 'set'

'get': Returns the current value of the property

'set': Sets the value of the property

“available for ‘get’”: read only

“available for ‘get’ and ‘set’”: read and write

```
The following properties are available:
 - String Architecture
     available for 'get'
 - String[] AvailableDisassemblers
     available for 'get'
 - SystemBus Bus
     available for 'get'
 - Boolean ChainingEnabled
     available for 'get' and 'set'
 - Boolean DisableInterruptsWhileStepping
     available for 'get' and 'set'
 - String Disassembler
     available for 'get' and 'set'
 - Endianess Endianness
     available for 'get'
 - UInt64 ExecutedInstructions
     available for 'get'
 - ExecutionMode ExecutionMode
     available for 'get' and 'set'
 - String GDBArchitecture
     available for 'get'
 - UInt32 HartId
     available for 'get' and 'set'
 - UInt32 Id
     available for 'get'
```

Using the Renode Console

Command: [peripheral name].[sub-peripheral]

Use/Example: “gpio0.button0”

Output: Prints methods and properties available to be used with the peripheral

Notes: Methods can be called to set or change functionality of a peripheral (e.g reset) and properties are values describing the peripheral or its state (e.g Architecture or IsHalted)

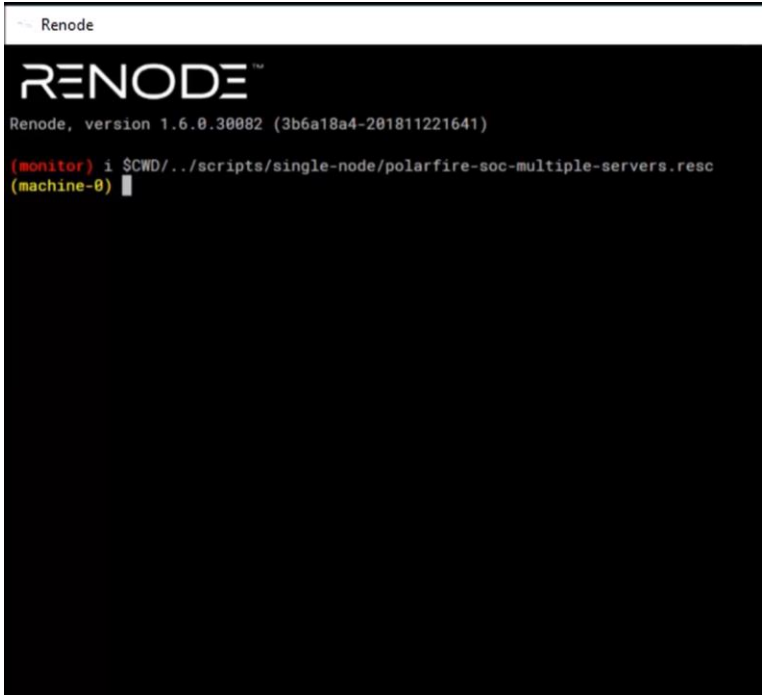
Using the Renode Console

Command: [peripheral name].[sub-peripheral] [method] [args]

Use / Example: “gpio0.button0 GetGPIOs”, “gpio0.button0 PressAndRelease”

Output: None if setting, will print value if using get

Notes: Some properties are read-only and some can be set during execution, you can tell as read-only will only have “get” available and read and write properties will have “get and set” available



Using the Renode Console

Command: logLevel [level] [peripheral]

Use / Example: “logLevel”, “logLevel -1”, “logLevel -1 sysbus.e51”

Output: No args: Peripherals currently being logged and their log levels

With args: No output

Notes:

Using Renode Logs and Their Output

Using Renode Logs and Their Output

- **There are several logging levels available:**
  - -1 Noisy
  - 0 Debug
  - 1 Info
  - 2 Warning
  - 3 Error

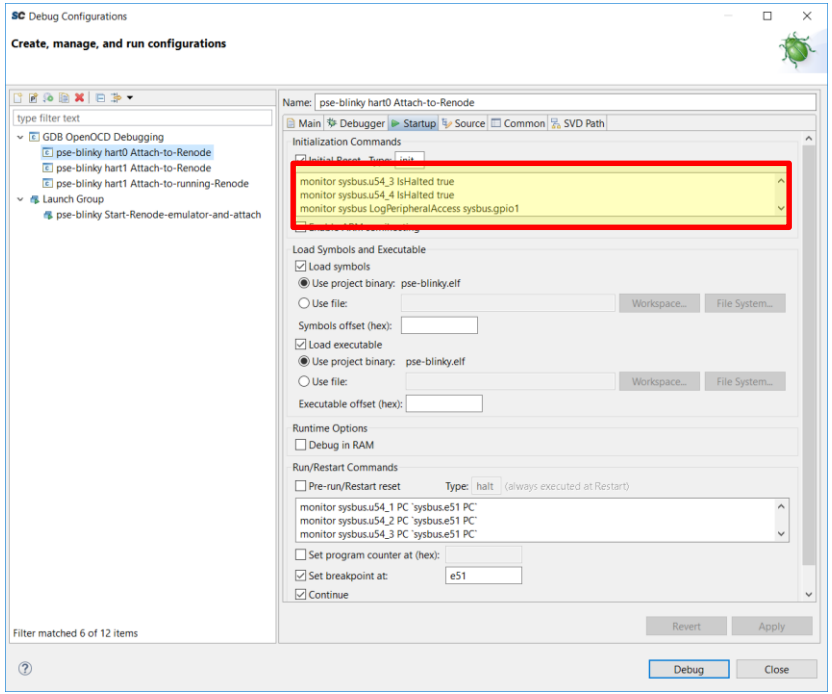
- **Each can capture different levels of data**

- **Peripherals will produce a different level of log data depending on what has caused the log (e.g. uart print vs fatal error)**

Using Renode Logs and Their Output

- **Logging (and other commands) can be set up automatically by passing the commands during start up**

- **E.G (pse-blinky debug configuration):**
  - monitor sysbus.u54_3 IsHalted true
  - monitor sysbus.u54_4 IsHalted true
  - monitor sysbus LogPeripheralAccess sysbus.gpio1
  - monitor logLevel -1 sysbus.gpio1



Using Renode Logs and Their Output

Logging can be set globally or set for individual peripherals

Renode

```
RENODE
Renode, version 1.6.0.30082 (3b6a18a4-201811221641)

(monitor) i $CWD/../scripts/single-node/polarfire-soc-multiple-servers.resc
(machine-0)
```

Renode

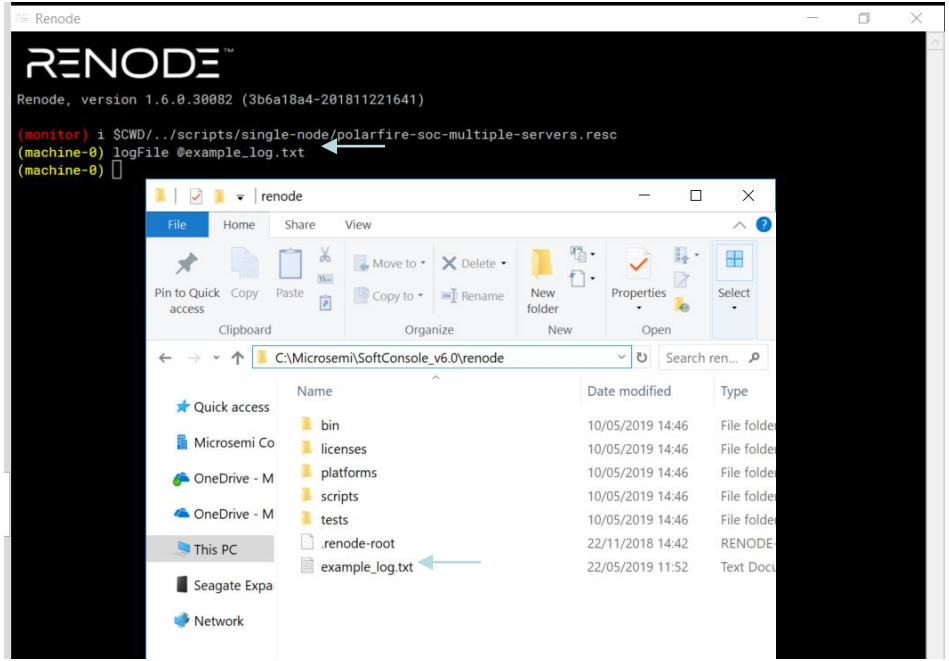
```
RENODE
Renode, version 1.6.0.30082 (3b6a18a4-201811221641)

(monitor) i $CWD/../scripts/single-node/polarfire-soc-multiple-servers.resc
(machine-0)
```

Using Renode Logs and Their Output

- **Logs can be directed to a file**
- **The log file is created if it doesn't exist and existing logs are overwritten**

SoftConsole Installation — Renode — Log.txt



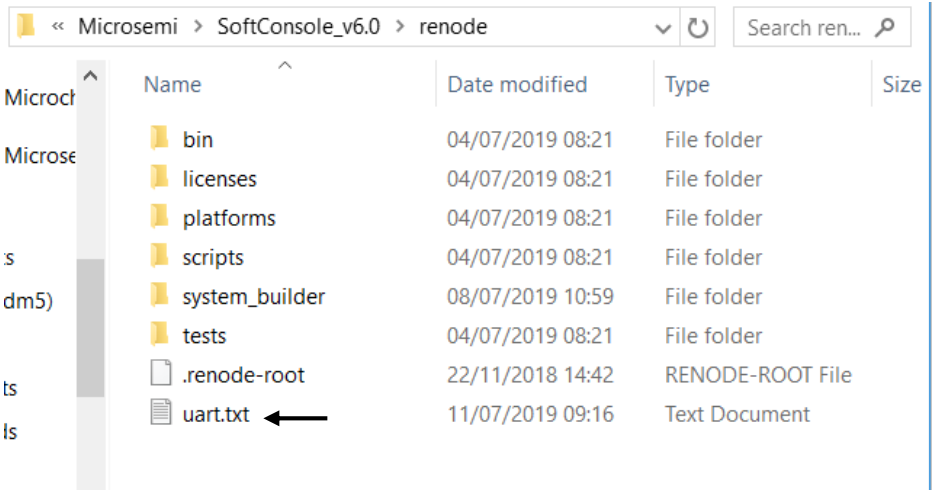
Using Renode Logs and Their Output

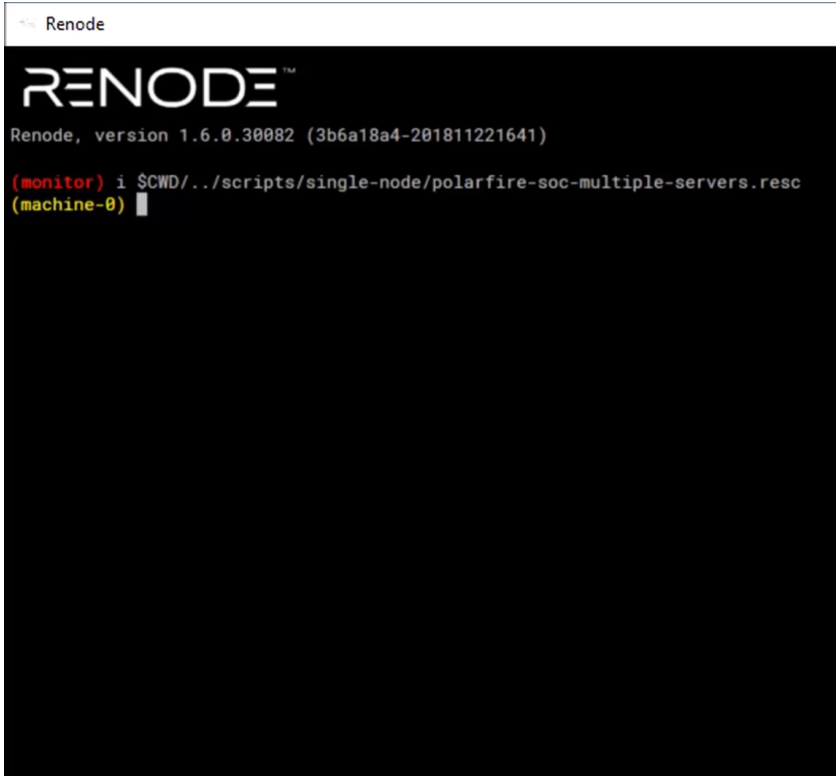
Example:

logFile @uart.txt

logLevel 0 sysbus.mmuart0

Sysbus LogPeripheralAccess sysbus.mmuart0



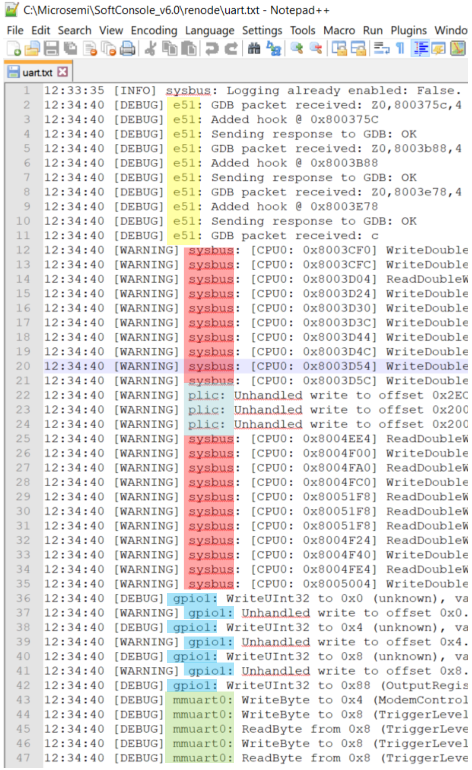


Using Renode Logs and Their Output

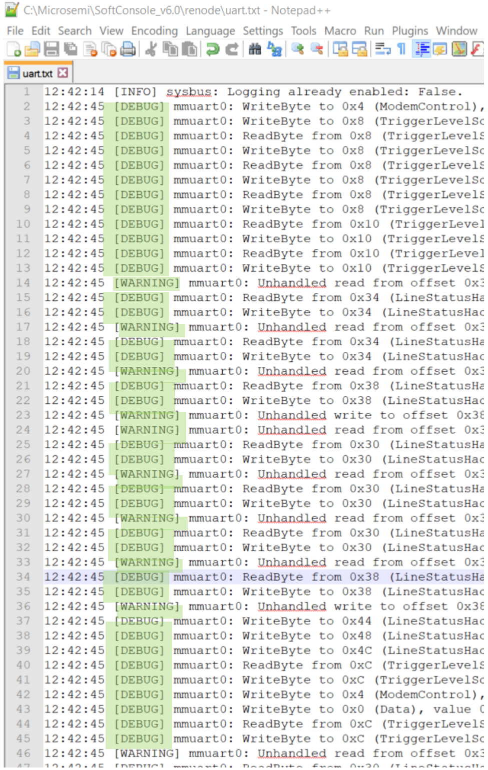
logLevel 3 file

```
Renode, version 1.6.0.30082 (3b6a18a4-201811221641)

(monitor) i $CWD/../scripts/single-node/polarfire-soc-multiple-servers.resc
(machine-0)
```



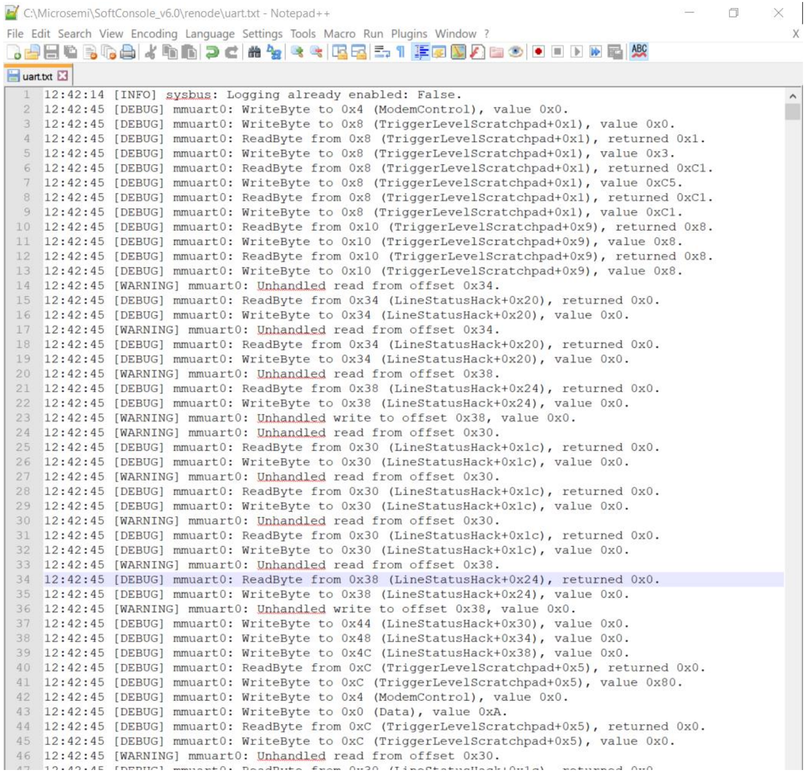
```
12:33:35 [INFO] sysbus: Logging already enabled: False.
12:34:40 [DEBUG] e51: GDB packet received: Z0,800375c,4
12:34:40 [DEBUG] e51: Added hook @ 0x800375C
12:34:40 [DEBUG] e51: Sending response to GDB: OK
12:34:40 [DEBUG] e51: GDB packet received: Z0,8003b88,4
12:34:40 [DEBUG] e51: Added hook @ 0x8003B88
12:34:40 [DEBUG] e51: Sending response to GDB: OK
12:34:40 [DEBUG] e51: GDB packet received: Z0,8003e78,4
12:34:40 [DEBUG] e51: Added hook @ 0x8003E78
12:34:40 [DEBUG] e51: Sending response to GDB: OK
12:34:40 [DEBUG] e51: GDB packet received: c
12:34:40 [WARNING] sysbus: [CPU0: 0x8003CF0] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8003CFC] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8003D04] ReadDoubleW
12:34:40 [WARNING] sysbus: [CPU0: 0x8003D24] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8003D30] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8003D3C] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8003D44] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8003D4C] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8003D54] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8003D5C] WriteDouble
12:34:40 [WARNING] plic: Unhandled write to offset 0x2EC
12:34:40 [WARNING] plic: Unhandled write to offset 0x200
12:34:40 [WARNING] plic: Unhandled write to offset 0x200
12:34:40 [WARNING] sysbus: [CPU0: 0x8004EE4] ReadDoubleW
12:34:40 [WARNING] sysbus: [CPU0: 0x8004F00] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8004FA0] ReadDoubleW
12:34:40 [WARNING] sysbus: [CPU0: 0x8004FC0] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x80051F8] ReadDoubleW
12:34:40 [WARNING] sysbus: [CPU0: 0x80051F8] ReadDoubleW
12:34:40 [WARNING] sysbus: [CPU0: 0x80051F8] ReadDoubleW
12:34:40 [WARNING] sysbus: [CPU0: 0x8004F24] ReadDoubleW
12:34:40 [WARNING] sysbus: [CPU0: 0x8004F40] WriteDouble
12:34:40 [WARNING] sysbus: [CPU0: 0x8004FE4] ReadDoubleW
12:34:40 [WARNING] sysbus: [CPU0: 0x8005004] WriteDouble
12:34:40 [DEBUG] gpio1: WriteUInt32 to 0x0 (unknown), va
12:34:40 [WARNING] gpio1: Unhandled write to offset 0x0.
12:34:40 [DEBUG] gpio1: WriteUInt32 to 0x4 (unknown), va
12:34:40 [WARNING] gpio1: Unhandled write to offset 0x4.
12:34:40 [DEBUG] gpio1: WriteUInt32 to 0x8 (unknown), va
12:34:40 [WARNING] gpio1: Unhandled write to offset 0x8.
12:34:40 [DEBUG] gpio1: WriteUInt32 to 0x88 (OutputRegis
12:34:40 [DEBUG] mmuart0: WriteByte to 0x4 (ModemControl
12:34:40 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevel
12:34:40 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLeve
12:34:40 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevel
12:34:40 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLeve
```



```
12:42:14 [INFO] sysbus: Logging already enabled: False.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x4 (ModemControl),
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelSc
12:42:45 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLevelS
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelSc
12:42:45 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLevelS
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelSc
12:42:45 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLevelS
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelSc
12:42:45 [DEBUG] mmuart0: ReadByte from 0x10 (TriggerLevel
12:42:45 [DEBUG] mmuart0: WriteByte to 0x10 (TriggerLevelS
12:42:45 [DEBUG] mmuart0: ReadByte from 0x10 (TriggerLevel
12:42:45 [DEBUG] mmuart0: WriteByte to 0x10 (TriggerLevelS
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x3
12:42:45 [DEBUG] mmuart0: ReadByte from 0x34 (LineStatusHa
12:42:45 [DEBUG] mmuart0: WriteByte to 0x34 (LineStatusHac
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x3
12:42:45 [DEBUG] mmuart0: ReadByte from 0x34 (LineStatusHa
12:42:45 [DEBUG] mmuart0: WriteByte to 0x34 (LineStatusHac
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x3
12:42:45 [DEBUG] mmuart0: ReadByte from 0x38 (LineStatusHa
12:42:45 [DEBUG] mmuart0: WriteByte to 0x38 (LineStatusHac
12:42:45 [WARNING] mmuart0: Unhandled write to offset 0x38
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x3
12:42:45 [DEBUG] mmuart0: ReadByte from 0x30 (LineStatusHa
12:42:45 [DEBUG] mmuart0: WriteByte to 0x30 (LineStatusHac
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x3
12:42:45 [DEBUG] mmuart0: ReadByte from 0x30 (LineStatusHa
12:42:45 [DEBUG] mmuart0: WriteByte to 0x30 (LineStatusHac
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x3
12:42:45 [DEBUG] mmuart0: ReadByte from 0x30 (LineStatusHa
12:42:45 [DEBUG] mmuart0: WriteByte to 0x30 (LineStatusHac
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x3
12:42:45 [DEBUG] mmuart0: ReadByte from 0x38 (LineStatusHa
12:42:45 [DEBUG] mmuart0: WriteByte to 0x38 (LineStatusHac
12:42:45 [WARNING] mmuart0: Unhandled write to offset 0x38
12:42:45 [DEBUG] mmuart0: WriteByte to 0x44 (LineStatusHac
12:42:45 [DEBUG] mmuart0: WriteByte to 0x48 (LineStatusHac
12:42:45 [DEBUG] mmuart0: WriteByte to 0x4C (LineStatusHac
12:42:45 [DEBUG] mmuart0: ReadByte from 0xC (TriggerLevelS
12:42:45 [DEBUG] mmuart0: WriteByte to 0xC (TriggerLevelSc
12:42:45 [DEBUG] mmuart0: WriteByte to 0x4 (ModemControl),
12:42:45 [DEBUG] mmuart0: WriteByte to 0x0 (Data), value
12:42:45 [DEBUG] mmuart0: ReadByte from 0xC (TriggerLevelS
12:42:45 [DEBUG] mmuart0: WriteByte to 0xC (TriggerLevelSc
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x3
```

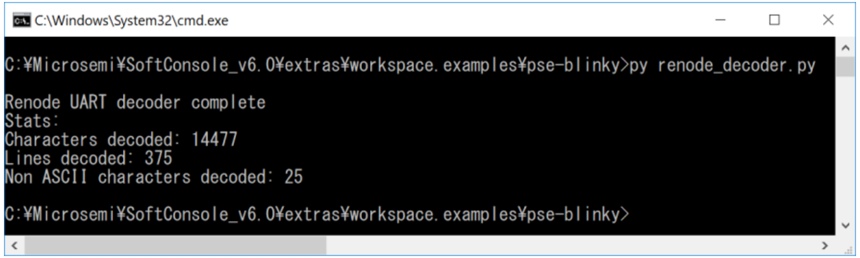
Using Renode Logs and Their Output

- Now there's a log producing a very large output file
- Only want to actually look at a small part of it / check if a value appears
- Use a script to parse it!



```
12:42:14 [INFO] sysbus: Logging already enabled: False.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x4 (ModemControl), value 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelScratchpad+0x1), value 0x0.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLevelScratchpad+0x1), returned 0x1.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelScratchpad+0x1), value 0x3.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLevelScratchpad+0x1), returned 0xC1.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelScratchpad+0x1), value 0xC5.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLevelScratchpad+0x1), returned 0xC1.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelScratchpad+0x1), value 0xC1.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x10 (TriggerLevelScratchpad+0x9), returned 0x8.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x10 (TriggerLevelScratchpad+0x9), value 0x8.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x10 (TriggerLevelScratchpad+0x9), returned 0x8.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x10 (TriggerLevelScratchpad+0x9), value 0x8.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x34.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x34 (LineStatusHack+0x20), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x34 (LineStatusHack+0x20), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x34.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x34 (LineStatusHack+0x20), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x34 (LineStatusHack+0x20), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x38.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x38 (LineStatusHack+0x24), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x38 (LineStatusHack+0x24), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled write to offset 0x38, value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x30.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x30 (LineStatusHack+0x1c), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x30 (LineStatusHack+0x1c), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x30.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x30 (LineStatusHack+0x1c), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x30 (LineStatusHack+0x1c), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x30.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x30 (LineStatusHack+0x1c), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x30 (LineStatusHack+0x1c), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x38.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x38 (LineStatusHack+0x24), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x38 (LineStatusHack+0x24), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled write to offset 0x38, value 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x44 (LineStatusHack+0x30), value 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x48 (LineStatusHack+0x34), value 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x4C (LineStatusHack+0x38), value 0x0.
12:42:45 [DEBUG] mmuart0: ReadByte from 0xC (TriggerLevelScratchpad+0x5), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0xC (TriggerLevelScratchpad+0x5), value 0x80.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x4 (ModemControl), value 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x0 (Data), value 0xA.
12:42:45 [DEBUG] mmuart0: ReadByte from 0xC (TriggerLevelScratchpad+0x5), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0xC (TriggerLevelScratchpad+0x5), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x30.
```

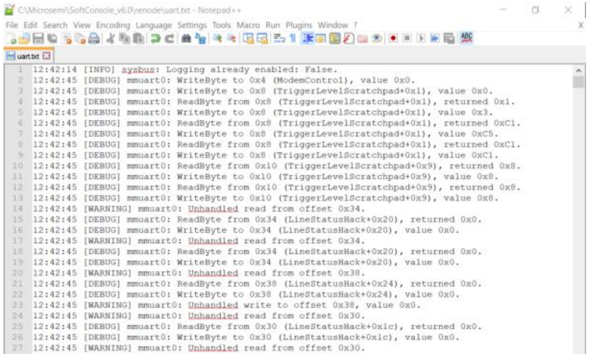
Using Renode Logs and Their Output



```
C:¥Microsemi¥SoftConsole_v6.0¥extras¥workspace.examples¥pse-blinky>py renode_decoder.py

Renode UART decoder complete
Stats:
Characters decoded: 14477
Lines decoded: 375
Non ASCII characters decoded: 25

C:¥Microsemi¥SoftConsole_v6.0¥extras¥workspace.examples¥pse-blinky>
```



```
12:42:14 [INFO] sysbus: Logging already enabled: False.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x4 (ModemControl), value 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelScratchpad+0x1), value 0x0.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLevelScratchpad+0x1), returned 0x1.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelScratchpad+0x1), value 0x3.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLevelScratchpad+0x1), returned 0xC1.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelScratchpad+0x1), value 0xC5.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x8 (TriggerLevelScratchpad+0x1), returned 0xC1.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x8 (TriggerLevelScratchpad+0x1), value 0xC1.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x10 (TriggerLevelScratchpad+0x9), returned 0x8.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x10 (TriggerLevelScratchpad+0x9), value 0x8.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x10 (TriggerLevelScratchpad+0x9), returned 0x8.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x10 (TriggerLevelScratchpad+0x9), value 0x8.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x34.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x34 (LineStatusHack+0x20), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x34 (LineStatusHack+0x20), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x34.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x34 (LineStatusHack+0x20), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x34 (LineStatusHack+0x20), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x38.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x38 (LineStatusHack+0x24), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x38 (LineStatusHack+0x24), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled write to offset 0x38, value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x30.
12:42:45 [DEBUG] mmuart0: ReadByte from 0x30 (LineStatusHack+0x1c), returned 0x0.
12:42:45 [DEBUG] mmuart0: WriteByte to 0x30 (LineStatusHack+0x1c), value 0x0.
12:42:45 [WARNING] mmuart0: Unhandled read from offset 0x30.
```

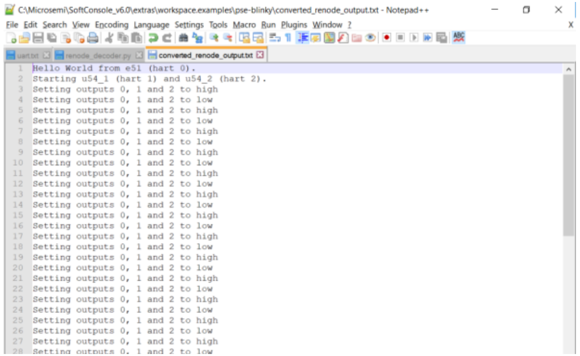
```python
import os
import sys

def renode_uart_decode():
    if os.path.isfile("../../../renode/uart.txt"):
        uart_file = open("../../../renode/uart.txt", "r")
        output_file = open("./converted_renode_output.txt", "w+")
        buffer = ""
        chars = 0
        new_lines = 0
        non_ascii_chars = 0

        for line in uart_file:
            chars = chars + 1
            if "mmuart0: WriteByte" in line:
                temp = line.strip()
                temp = temp.split("value 0x")
                temp = temp[1]
                temp = temp.split(".")
                temp = str(temp[0])
                hex = int(temp,16)
                if not hex < 31 and not hex > 127:
                    buffer = buffer + str(bytearray.fromhex(temp).decode())
                elif hex == 10:
                    output_file.write(buffer + "\n")
                    buffer = ""
                    new_lines = new_lines + 1
                elif not hex == 13:
                    buffer = buffer
                    non_ascii_chars = non_ascii_chars + 1
        output_file.close()

        print("\nRenode UART decoder complete")
        print("Stats:")
        print("Characters decoded: " + str(chars))
        print("Lines decoded: " + str(new_lines))
        print("Non ASCII characters decoded: " + str(non_ascii_chars))
    else:
        print("File not found!")
        return

renode_uart_decode()
```

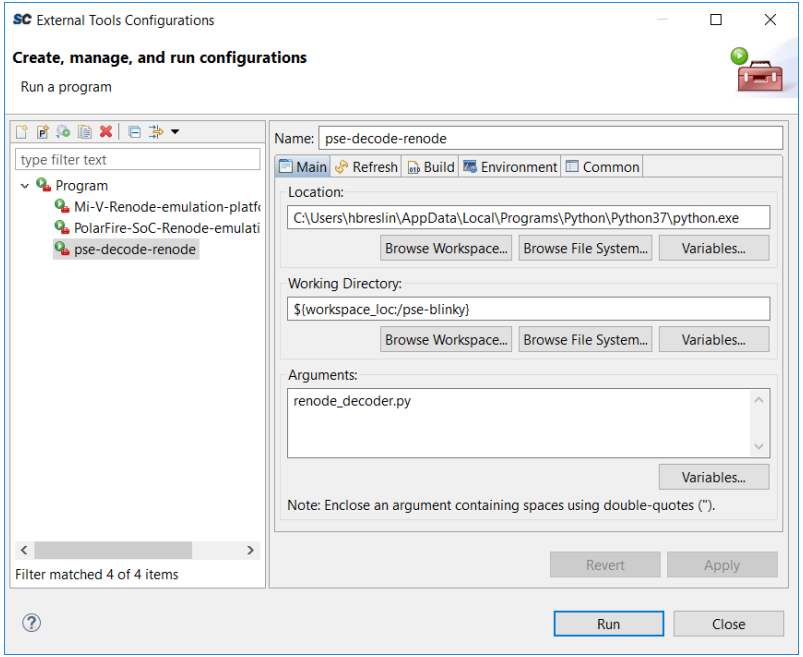


```
Hello World from e51 (hart 0).
Starting u54_1 (hart 1) and u54_2 (hart 2).
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
Setting outputs 0, 1 and 2 to high
Setting outputs 0, 1 and 2 to low
```

Using Renode Logs and Their Output

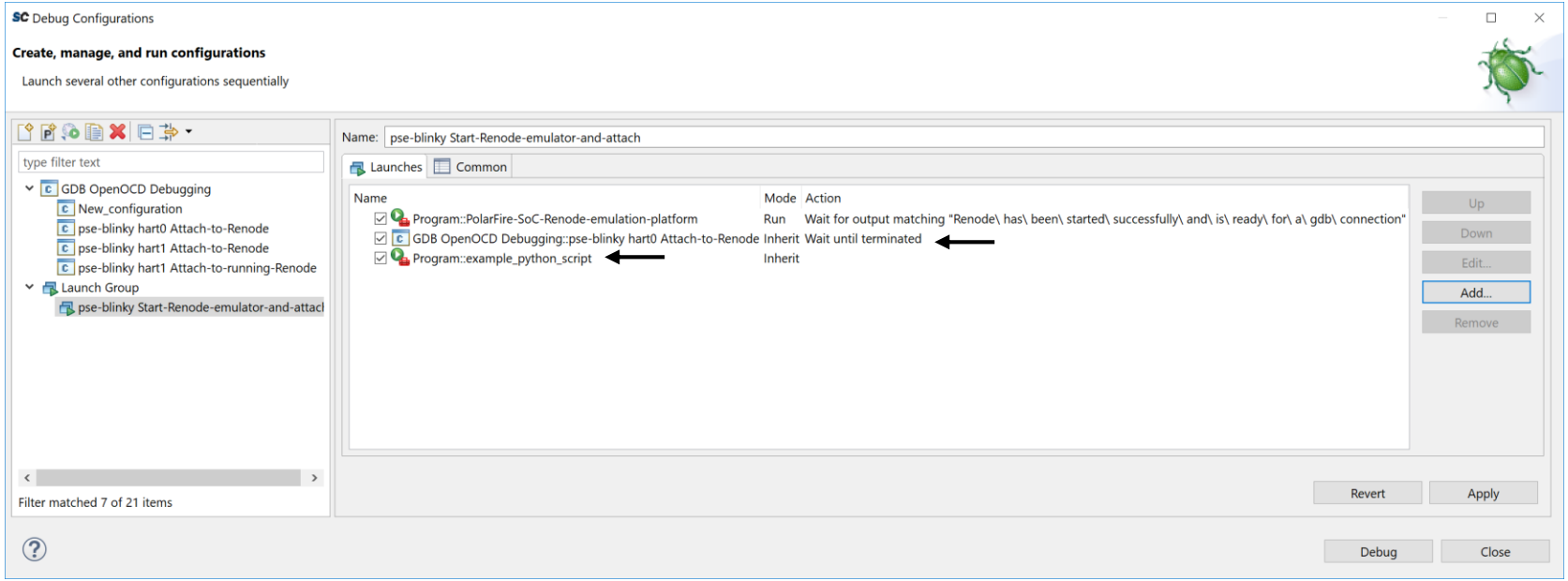
- **Python scripts can be added as external tools**
  - Add the script to a launch group and run it after your debug session has been terminated

- **Add the Python script as a hook in Renode**



Using Renode Logs and Their Output

- **The python external tool can be called as part of a launch group**
- **They will then run from the Console in SoftConsole**



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 a pre-existing peripheral in PolarFire SoC

We are planning our next series and would like your input on the content presented so far and what you would like to see covered

Thank You!

Questions?