

Second Thursdays

- April 9 - Webinar 12: Simple Peripheral as Software Stimulus**
- May 14 - Webinar 13: Two Bare-Metal Applications on PolarFire® SoC**
- June 11 - Webinar 14: The PolarFire SoC Icicle Kit Model in Renode**
- July 9 - Webinar 15: Linux® on Renode**
- Aug. 13 - Webinar 16: Building Applications for Linux on PolarFire SoC**
- Sep. 10 - Webinar 17: Real-Time (AMP Mode) on PolarFire SoC**

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

Session 12 Simple Peripheral as Software Stimulus



A Leading Provider of Smart, Connected and Secure Embedded Control Solutions



SMART | CONNECTED | SECURE

Hugh Breslin, Design Engineer

Thursday Apr. 9, 2020

Supporting Content

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

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

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

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

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

Webinar 5: Add and Debug PolarFire SoC Models with Renode

Webinar 6: Add and Debug Pre-Existing Model in PolarFire SoC

Webinar 7: How to Write Custom Models

Webinar 8: What's New in SoftConsole v6.2

Webinar 9: Getting Started with PolarFire SoC

Webinar 10: Introduction to the PolarFire SoC Bare-Metal Library

Webinar 11: Handling Binaries

The screenshot shows the Microsemi website's navigation and content. At the top, there's a search bar and a navigation menu with links for 'Ordering', 'Company', 'Partners', and 'Support'. Below this is a banner for 'Libero SoC Design Suite v12.0' with bullet points: 'Unified design suite for PolarFire, IGLOO2, SmartFusion2 and RTG4 families', '60% runtime reduction for Timing and 20% runtime reduction for Power', and '25% runtime improvement for Place and Route'. A breadcrumb trail reads 'Home / Products & Services / FPGA & SoC / Mi-V RISC-V Ecosystem'. The main heading is 'Mi-V RISC-V Ecosystem', followed by a sub-navigation bar with links: 'Overview', 'Mi-V Partners', 'Tutorials', 'Renode Webinar Series' (highlighted with a red box), and 'Articles and News'. Below this, the section 'Getting Started with the RISC-V Based PolarFire™ SoC FPGA Webinar Series' is visible, with a paragraph of introductory text and a link to 'Click here to register.' Logos for 'Mi-V' and 'antmicro' are shown. At the bottom, the 'Webinar 1 (May 2): Discover Renode for PolarFire™ SoC Design and Debug' is listed, followed by a brief description of the session.

Agenda

- **Antmicro Tensor Flow Demo**
- **Demo**
 - PAC1934 on the Icicle Kit
 - Interfacing the Model with sysbus via I2C
 - Using the I2C Drivers
 - Reading Back Values
 - Monitoring Voltage and Current

Antmicro Tensor Flow Demo

Insights

[Sign up](#)

Dismiss

Apache-2.0

Clone or download ▾

✓ Latest commit 25a11fd 14 days ago

19 days ago

4 months ago

14 days ago

last month

last month

19 days ago

last month

19 days ago

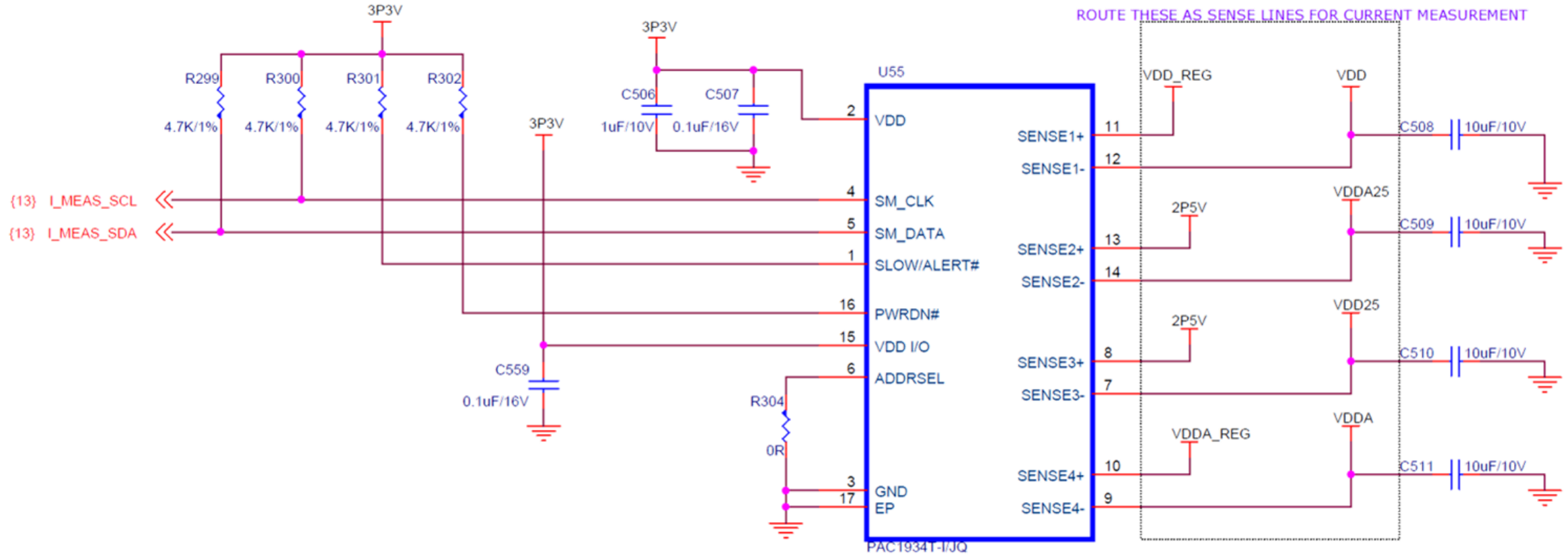
 README.md

Demo

PAC1934 on the Icicle Kit
Interfacing the Model with sysbus via I2C
Using the I2C Drivers
Reading Back Values
Monitoring Voltage and Current

PAC1934 on the Icycle Kit

Current Measurement 1



PAC1934 on the Icycle Kit

- **4 sense connections:**
 - VDD
 - VDDA
 - VDD25
 - VDDA25
- **Connected to I2C0**
- **PF SoC needs to configure I2C0 as a master and read registers from the PAC1934 as an I2C slave**

PAC1934 on the Icicle Kit

```
(machine-0) i2c0.PAC1934
```

The following methods are available:

- `Void` DebugLog (`String` message)
- `IEnumerable<Tuple<String,IGPIO>>` GetGPIOs ()
- `Machine` GetMachine ()
- `Boolean` HasGPIO ()
- `Void` Log (`LogLevel` type, `String` message)
- `Void` LogUnhandledRead (`Int64` offset)
- `Void` LogUnhandledWrite (`Int64` offset, `Int64` value)
- `Void` NoisyLog (`String` message)
- `Byte[]` Read (`Int32` count = 1)
- `Void` Reset ()
- `Void` update_AccCount ()
- `Void` update_AccData ()
- `Void` update_VBus ()
- `Void` update_VPower ()
- `Void` update_VSense ()
- `Void` Write (`Byte[]` data)

Usage:

```
sysbus.i2c0.PAC1934 MethodName param1 param2 ...
```

The following properties are available:

- `Int32` ACC_COUNT
available for 'get' and 'set'
- `Boolean` ALERT_CC
available for 'get' and 'set'
- `Boolean` ALERT_PIN
available for 'get' and 'set'

- `Int16` VSENSE2_AVG_bipolar
available for 'get' and 'set'
- `Int16[]` VSENSE2_AVG_bipolar_array
available for 'get'
- `Int16` VSENSE2_bipolar
available for 'get' and 'set'
- `UInt16` VSENSE3
available for 'get' and 'set'
- `UInt16` VSENSE3_AVG
available for 'get' and 'set'
- `UInt16[]` VSENSE3_AVG_array
available for 'get'
- `Int16` VSENSE3_AVG_bipolar
available for 'get' and 'set'
- `Int16[]` VSENSE3_AVG_bipolar_array
available for 'get'
- `Int16` VSENSE3_bipolar
available for 'get' and 'set'
- `UInt16` VSENSE4
available for 'get' and 'set'
- `UInt16` VSENSE4_AVG
available for 'get' and 'set'
- `UInt16[]` VSENSE4_AVG_array
available for 'get'
- `Int16` VSENSE4_AVG_bipolar
available for 'get' and 'set'
- `Int16[]` VSENSE4_AVG_bipolar_array
available for 'get'
- `Int16` VSENSE4_bipolar
available for 'get' and 'set'

Usage:

PAC1934 on the Icicle Kit

```
(machine-0) i2c0.PAC1934 VBUS1
0x03E8
(machine-0) i2c0.PAC1934 update_VBus
(machine-0) i2c0.PAC1934 VBUS1
0x03E9
(machine-0)
```

PolarFire-SoC-Icicle-Renode-emulation-platform [Program] /usr/bin/mono

```
14:04:32.5461 [INFO] Loaded monitor commands from: /home/miv/Microsemi_SoftConsole_v6.2/renode/./scripts/monitor.py
14:04:33.8674 [INFO] Including script: /home/miv/Microsemi_SoftConsole_v6.2/renode-microchip-mods/scripts/polarfire-soc-icicle-board.resc
14:04:38.6658 [ERROR] Script: Renode has been started successfully and is ready for a gdb connection. (This is not an error)
14:07:23.8613 [NOISY] i2c0.PAC1934: Updating VBUSn
14:07:23.8621 [NOISY] i2c0.PAC1934: Updating VBUSn_bipolar
14:07:23.8622 [NOISY] i2c0.PAC1934: Updating VBUSn_AVGs
```

PAC1934 on the Icicle Kit

- VBUSn is the voltage on Sensen
- VBUSn_AVG is the average of the last 8 readings
- VBUSn _AVG_array is an array of the last 8 results
- Update_VBus updates the VBUSn values and the average arrays

```
(machine-0) i2c0.PAC1934 VBUS1_AVG
0x00FA
(machine-0) i2c0.PAC1934 VBUS1_AVG_array
[
0x03EB, 0x03E9, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
]
(machine-0) i2c0.PAC1934 update_VBus
(machine-0) i2c0.PAC1934 VBUS1_AVG
0x0178
(machine-0) i2c0.PAC1934 VBUS1_AVG_array
[
0x03ED, 0x03EB, 0x03E9, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
]
(machine-0) i2c0.PAC1934 update_VBus
(machine-0) i2c0.PAC1934 VBUS1_AVG
0x01F6
(machine-0) i2c0.PAC1934 VBUS1_AVG_array
[
0x03EF, 0x03ED, 0x03EB, 0x03E9, 0x0000, 0x0000, 0x0000, 0x0000,
]
(machine-0) i2c0.PAC1934 update_VBus
(machine-0) i2c0.PAC1934 VBUS1_AVG
0x0273
(machine-0) i2c0.PAC1934 VBUS1_AVG_array
[
0x03ED, 0x03EF, 0x03ED, 0x03EB, 0x03E9, 0x0000, 0x0000, 0x0000,
]
(machine-0) i2c0.PAC1934 update_VBus
(machine-0) i2c0.PAC1934 VBUS1_AVG
0x02F1
(machine-0) i2c0.PAC1934 VBUS1_AVG_array
[
0x03ED, 0x03ED, 0x03EF, 0x03ED, 0x03EB, 0x03E9, 0x0000, 0x0000,
]
(machine-0) i2c0.PAC1934 update_VBus
(machine-0) i2c0.PAC1934 VBUS1_AVG
0x036E
(machine-0) i2c0.PAC1934 VBUS1_AVG_array
[
0x03EC, 0x03ED, 0x03ED, 0x03EF, 0x03ED, 0x03EB, 0x03E9, 0x0000,
]
(machine-0) i2c0.PAC1934 update_VBus
(machine-0) i2c0.PAC1934 VBUS1_AVG
0x03EC
(machine-0) i2c0.PAC1934 VBUS1_AVG_array
[
0x03EE, 0x03EC, 0x03ED, 0x03ED, 0x03EF, 0x03ED, 0x03EB, 0x03E9,
]
(machine-0)
```

Interfacing the model with sysbus using I2C

- The PAC1934 model will be connected to the I2C0 peripheral which is connected to sysbus
- Create a “models” folder in the [SC_install]/renode-microchip-mods/ directory
- Include the model in the platform .resc file

```
include @models/PAC1934.cs
machine LoadPlatformDescriptionFromString "PAC1934: Sensors.PAC1934
@ i2c0 0"
```

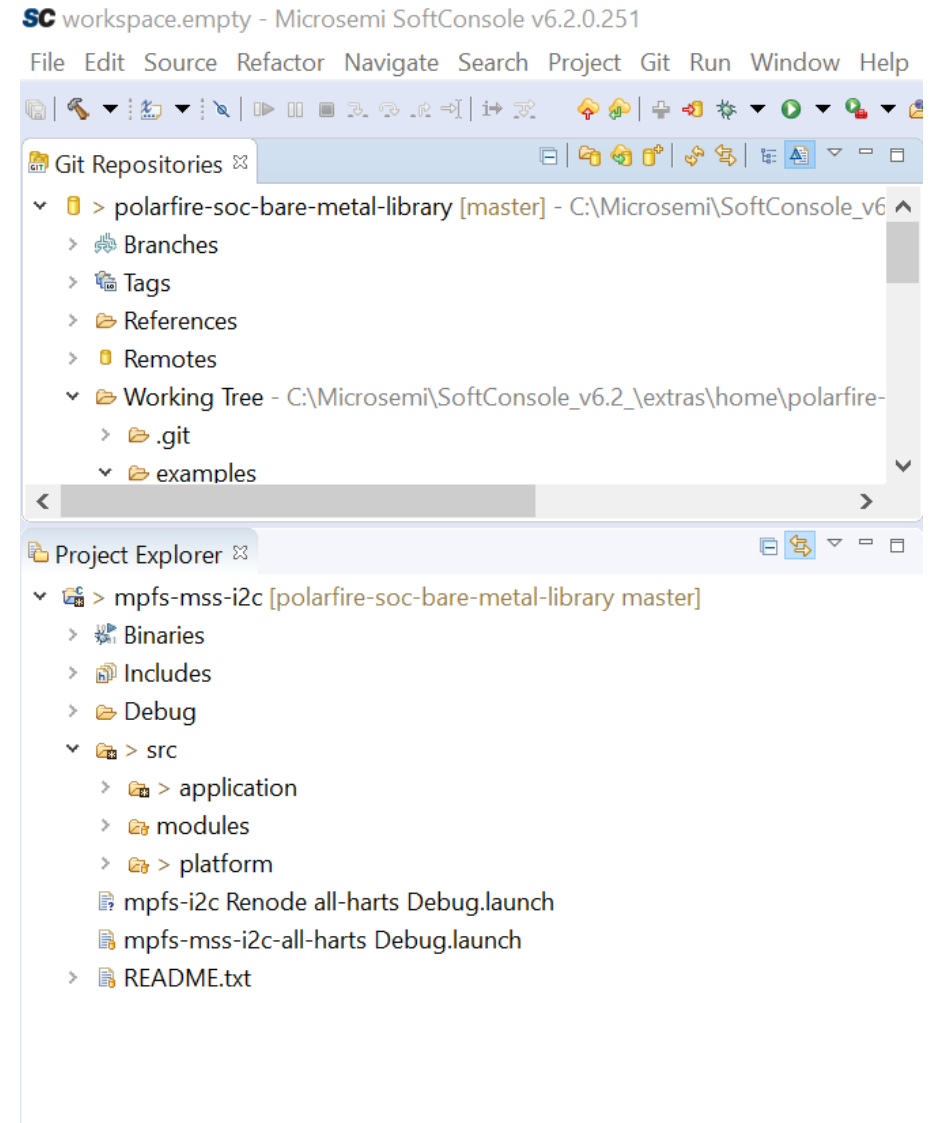
Interfacing the model with sysbus using I2C

- PAC1934 is now connected to i2c0 at address 0

```
— i2c0 (PSE_I2C)
  | <0x2010A000, 0x2010AFFF>
  |
  |— PAC1934 (PAC1934)
    Address: 0
```

Using the I2C Drivers

- The I2C driver examples can be found in the PolarFire SoC Bare Metal Library
- From the data sheet a write with no data will set the register pointer in the PAC1934
- Using the I2C write read function the register pointer can be set and the register read



Reading Back Values

- Values are read back as bytes and some registers are different sizes

```
if (reg == 0x3 | reg == 0x4 | reg == 0x5 | reg == 0x6){
    value = g_master_rx_buf[7];
    value = value << 8;
    value = value + g_master_rx_buf[6];
    value = value << 8;
    value = value + g_master_rx_buf[5];
    value = value << 8;
    value = value + g_master_rx_buf[4];
    value = value << 8;
    value = value + g_master_rx_buf[3];
    value = value << 8;
    value = value + g_master_rx_buf[2];
    value = value << 8;
    value = value + g_master_rx_buf[1];
    value = value << 8;
    value = value + g_master_rx_buf[0];
}
else if (reg == 0x7 | reg == 0x8 | reg == 0x9 | reg == 0xA | reg == 0xi
    value = g_master_rx_buf[1];
    value = value << 8;
    value = value + g_master_rx_buf[0];
} else if (reg == 0x17 | reg == 0x18 | reg == 0x19 | reg == 0x1A){
    value = g_master_rx_buf[3];
    value = value << 8;
    value = value + g_master_rx_buf[2];
    value = value << 8;
    value = value + g_master_rx_buf[1];
    value = value << 8;
    value = value + g_master_rx_buf[0];
} else if (reg == 0xFD | reg == 0xFE | reg == 0xFF){
    value = g_master_rx_buf[0];
}
return(value);
```

0x00	W	1 byte	REFRESH
0x01	R/W	1 byte	CTRL
0x02	R	3 bytes	ACC_COUNT
0x03	R	6 bytes	VPOWER1_ACC
0x04	R	6 bytes	VPOWER2_ACC
0x05	R	6 bytes	VPOWER3_ACC
0x06	R	6 bytes	VPOWER4_ACC
0x07	R	2 bytes	VBUS1
0x08	R	2 bytes	VBUS2
0x09	R	2 bytes	VBUS3
0x0A	R	2 bytes	VBUS4
0x0B	R	2 bytes	VSENSE1
0x0C	R	2 bytes	VSENSE2
0x0D	R	2 bytes	VSENSE3
0x0E	R	2 bytes	VSENSE4
0x0F	R	2 bytes	VBUS1_AVG
0x10	R	2 bytes	VBUS2_AVG
0x11	R	2 bytes	VBUS3_AVG
0x12	R	2 bytes	VBUS4_AVG
0x13	R	2 bytes	VSENSE1_AVG
0x14	R	2 bytes	VSENSE2_AVG
0x15	R	2 bytes	VSENSE3_AVG
0x16	R	2 bytes	VSENSE4_AVG
0x17	R	4 bytes	VPOWER1
0x18	R	4 bytes	VPOWER2
0x19	R	4 bytes	VPOWER3
0x1A	R	4 bytes	VPOWER4
0x1C	R/W	1 byte	CHANNEL_DIS
0x1D	R/W	1 byte	NEG_PWR
0x1E	W	1 byte	REFRESH_G
0x1F	W	1 byte	REFRESH_V
0x20	R/W	1 byte	SLOW
0x21	R	1 byte	CTRL_ACT
0x22	R	1 byte	CHANNEL_DIS_ACT
0x23	R	1 byte	NEG_PWR_ACT
0x24	R	1 byte	CTRL_LAT
0x25	R	1 byte	CHANNEL_DIS_LAT
0x26	R	1 byte	NEG_PWR_LAT
0xFD	R	1 byte	PID
0xFE	R	1 byte	MID
0xFF	R	1 byte	REV

Monitoring Voltage and Current

- Write to the “Refresh” register to cause an update
- Read back the value in each register

```
while(1){
    MSS_UART_polled_tx_string(gp_sys_uart, (const uint8_t *)"\n\r\n\r Sensor refresh");
    sensor_reg_check(REFRESH_REG);
    VPowerACC[0] = sensor_reg_check(VPOWER1_ACC_REG);
    VPowerACC[1] = sensor_reg_check(VPOWER2_ACC_REG);
    VPowerACC[2] = sensor_reg_check(VPOWER3_ACC_REG);
    VPowerACC[3] = sensor_reg_check(VPOWER4_ACC_REG);
    VBus[0] = sensor_reg_check(VBUS1_REG);
    VBus[1] = sensor_reg_check(VBUS2_REG);
    VBus[2] = sensor_reg_check(VBUS3_REG);
    VBus[3] = sensor_reg_check(VBUS4_REG);
    VSense[0] = sensor_reg_check(VSENSE1_REG);
    VSense[1] = sensor_reg_check(VSENSE2_REG);
    VSense[2] = sensor_reg_check(VSENSE3_REG);
    VSense[3] = sensor_reg_check(VSENSE4_REG);
    VBusAvg[0] = sensor_reg_check(VBUS1_AVG_REG);
    VBusAvg[1] = sensor_reg_check(VBUS2_AVG_REG);
    VBusAvg[2] = sensor_reg_check(VBUS3_AVG_REG);
    VBusAvg[3] = sensor_reg_check(VBUS4_AVG_REG);
    VSenseAvg[0] = sensor_reg_check(VSENSE1_AVG_REG);
    VSenseAvg[1] = sensor_reg_check(VSENSE2_AVG_REG);
    VSenseAvg[2] = sensor_reg_check(VSENSE3_AVG_REG);
    VSenseAvg[3] = sensor_reg_check(VSENSE4_AVG_REG);
    VPower[0] = sensor_reg_check(VPOWER1_REG);
    VPower[1] = sensor_reg_check(VPOWER2_REG);
    VPower[2] = sensor_reg_check(VPOWER3_REG);
    VPower[3] = sensor_reg_check(VPOWER4_REG);
}
```

Monitoring Voltage and Current

- Print the arrays to a UART monitor
- Raise interrupts for the U54s to deal with high / low values

```
drawUIUpdate(&VPowerACC, &VBus, &VSense, &VBusAvg, &VSenseAvg, &VPower);
blankLine();

if (VBus[0] < 990 | VBus[1] < 2475 | VBus[2] < 2400 | VBus[3] < 965){
    raise_soft_interrupt(1u);
    MSS_UART_polled_tx_string(gp_sys_uart, (const uint8_t *)"\n\r\n\r Soft IRQ U54_1");
    MSS_UART_polled_tx_string(gp_usr_uart, (const uint8_t *)"+ Alert raised on U54_1");
    MSS_UART_polled_tx_string(gp_usr_uart, (const uint8_t *)"\n\r");
}
if (VBus[0] > 1150 | VBus[1] > 2595 | VBus[2] > 2700 | VBus[3] > 1275){
    raise_soft_interrupt(2u);
    MSS_UART_polled_tx_string(gp_sys_uart, (const uint8_t *)"\n\r\n\r Soft IRQ U54_2");
    MSS_UART_polled_tx_string(gp_usr_uart, (const uint8_t *)"+ Alert raised on U54_2");
    MSS_UART_polled_tx_string(gp_usr_uart, (const uint8_t *)"\n\r");
}
if (VSense[0] < 4850 | VSense[1] < 9700 | VSense[2] < 9650 | VSense[3] < 5000){
    raise_soft_interrupt(3u);
    MSS_UART_polled_tx_string(gp_sys_uart, (const uint8_t *)"\n\r\n\r Soft IRQ U54_3");
    MSS_UART_polled_tx_string(gp_usr_uart, (const uint8_t *)"+ Alert raised on U54_3");
    MSS_UART_polled_tx_string(gp_usr_uart, (const uint8_t *)"\n\r");
}
if (VSense[0] > 5250 | VSense[1] > 13000 | VSense[2] > 12500 | VSense[3] > 6000){
    raise_soft_interrupt(4u);
    MSS_UART_polled_tx_string(gp_sys_uart, (const uint8_t *)"\n\r\n\r Soft IRQ U54_4");
    MSS_UART_polled_tx_string(gp_usr_uart, (const uint8_t *)"+ Alert raised on U54_4");
    MSS_UART_polled_tx_string(gp_usr_uart, (const uint8_t *)"\n\r");
}
blankLine();
drawBar();
```

Monitoring Voltage and Current

The screenshot displays a Renode virtual machine environment with four terminal windows. The top-left window, titled 'Renode', shows the command-line interface where the GDB server is started on port 3333. The top-right window, titled 'machine-0:sysbus.mmuart0', shows repeated 'I2C0 Transfer completed.' messages and 'Data Received to Master Receive Buffer is:' prompts. The bottom-right window, titled 'machine-0:sysbus.mmuart1', displays the output of an I2C sensor demo, including manufacturer and product ID checks. The bottom-left window, titled 'machine-0:sysbus.mmuart3', is currently empty. The bottom status bar indicates the Renode version (1.8.2.29648) and the current time (13:50).

```
Renode
Renode, version 1.8.2.29648 (c2650b65-201911211626)
(monитор) $GDB_SERVER_PORT=3333
(monитор) path add @/home/miv/Microsemi_SoftConsole_v6.2/renode
Current 'PATH' value is: /home/miv/Microsemi_SoftConsole_v6.2/renode/bin;/home/miv/Microsemi_SoftConsole_v6.2/renode/bin;/home/miv/Microsemi_SoftConsole_v6.2/renode/bin
(monитор) include @scripts/polarfire-soc-generic-board.resc
Starting GDB server on port: 3333
(machine-0) []

machine-0:sysbus.mmuart0
I2C0 Transfer completed.
Data Received to Master Receive Buffer is:
I2C0 Transfer completed.
I2C0 Transfer completed.
I2C0 Transfer completed.
Data Received to Master Receive Buffer is:
I2C0 Transfer completed.
I2C0 Transfer completed.
Data Received to Master Receive Buffer is:

machine-0:sysbus.mmuart1
*****
* PF SoC I2C Sensor Demo
*****
* Checking device Manufacturer ID
Device Manufacturer ID is: 93
* Checking device Product ID
Device Product ID is: 91
* Checking device rev
Device rev is: 3

machine-0:sysbus.mmuart3
[]

machine-0:sysbus.mmuart2
[]
```

Monitoring Voltage and Current

The image shows a multi-windowed application running on a Linux desktop. The main window is titled "Renode" and displays the Renode logo and version information (1.8.2.29648). Below this, there is a command prompt showing the execution of a script to start a GDB server on port 3333. The script sets the path to the Microsemi SoftConsole v6.2 binaries and includes a configuration file for the PolarFire SoC generic board. The prompt then shows the start of the GDB server and the connection to a machine window.

There are five other windows, each representing a different machine in the system:

- machine-0:sysbus.mmuart0**: Displays I2C transfer logs. It shows data received to the master receive buffer and the completion of I2C transfers. It also shows the start of a soft IRQ for U54_3.
- machine-0:sysbus.mmuart1**: Displays a table of sensor data. The table has four columns: VPowerACC1, VPowerACC2, VPowerACC3, and VPowerACC4. The data is as follows:

VPowerACC1	VPowerACC2	VPowerACC3	VPowerACC4
[20013007]	[99862643]	[100414401]	[22992146]
mv	mv	mv	mv
VBus1	VBus2	VBus3	VBus4
[1001]	[2490]	[2509]	[1042]
mv	mv	mv	mv
VSense1	VSense2	VSense3	VSense4
[5000]	[9988]	[9981]	[5491]
mv	mv	mv	mv
VBus1Avg	VBus2Avg	VBus3Avg	VBus4Avg
[500]	[1248]	[1256]	[522]
mv	mv	mv	mv
VSense1Avg	VSense2Avg	VSense3Avg	VSense4Avg
[2500]	[4997]	[4994]	[2750]
mv	mv	mv	mv
VPower1	VPower2	VPower3	VPower4
[5005000]	[24870120]	[25042329]	[5721622]
mv	mv	mv	mv
- machine-0:sysbus.mmuart2**: Displays a table of sensor data. The table has four columns: VPowerACC1, VPowerACC2, VPowerACC3, and VPowerACC4. The data is as follows:

VPowerACC1	VPowerACC2	VPowerACC3	VPowerACC4
[20013007]	[99862643]	[100414401]	[22992146]
mv	mv	mv	mv
VBus1	VBus2	VBus3	VBus4
[1001]	[2490]	[2509]	[1042]
mv	mv	mv	mv
VSense1	VSense2	VSense3	VSense4
[5000]	[9988]	[9981]	[5491]
mv	mv	mv	mv
VBus1Avg	VBus2Avg	VBus3Avg	VBus4Avg
[500]	[1248]	[1256]	[522]
mv	mv	mv	mv
VSense1Avg	VSense2Avg	VSense3Avg	VSense4Avg
[2500]	[4997]	[4994]	[2750]
mv	mv	mv	mv
VPower1	VPower2	VPower3	VPower4
[5005000]	[24870120]	[25042329]	[5721622]
mv	mv	mv	mv
- machine-0:sysbus.mmuart3**: Displays a table of sensor data. The table has four columns: VPowerACC1, VPowerACC2, VPowerACC3, and VPowerACC4. The data is as follows:

VPowerACC1	VPowerACC2	VPowerACC3	VPowerACC4
[20013007]	[99862643]	[100414401]	[22992146]
mv	mv	mv	mv
VBus1	VBus2	VBus3	VBus4
[1001]	[2490]	[2509]	[1042]
mv	mv	mv	mv
VSense1	VSense2	VSense3	VSense4
[5000]	[9988]	[9981]	[5491]
mv	mv	mv	mv
VBus1Avg	VBus2Avg	VBus3Avg	VBus4Avg
[500]	[1248]	[1256]	[522]
mv	mv	mv	mv
VSense1Avg	VSense2Avg	VSense3Avg	VSense4Avg
[2500]	[4997]	[4994]	[2750]
mv	mv	mv	mv
VPower1	VPower2	VPower3	VPower4
[5005000]	[24870120]	[25042329]	[5721622]
mv	mv	mv	mv
- machine-0:sysbus.mmuart4**: Displays a table of sensor data. The table has four columns: VPowerACC1, VPowerACC2, VPowerACC3, and VPowerACC4. The data is as follows:

VPowerACC1	VPowerACC2	VPowerACC3	VPowerACC4
[20013007]	[99862643]	[100414401]	[22992146]
mv	mv	mv	mv
VBus1	VBus2	VBus3	VBus4
[1001]	[2490]	[2509]	[1042]
mv	mv	mv	mv
VSense1	VSense2	VSense3	VSense4
[5000]	[9988]	[9981]	[5491]
mv	mv	mv	mv
VBus1Avg	VBus2Avg	VBus3Avg	VBus4Avg
[500]	[1248]	[1256]	[522]
mv	mv	mv	mv
VSense1Avg	VSense2Avg	VSense3Avg	VSense4Avg
[2500]	[4997]	[4994]	[2750]
mv	mv	mv	mv
VPower1	VPower2	VPower3	VPower4
[5005000]	[24870120]	[25042329]	[5721622]
mv	mv	mv	mv

The bottom of the image shows a taskbar with several icons, including a terminal, a file explorer, and a web browser. The system clock in the bottom right corner shows the time as 13:50.

Monitoring Voltage and Current

- E51 monitors the PAC1934
- E51 prints system messages (e.g irq to hart or I2C event) to mmuart 0
- E51 prints readings to mmuart1
- U54_x handles voltage / current event

Demo

Agenda

- **Antmicro Tensor Flow Demo**
- **Demo**
 - PAC1934 on the Icicle Kit
 - Interfacing the Model with sysbus via I2C
 - Using the I2C Drivers
 - Reading Back Values
 - Monitoring Voltage and Current

Thank you!

Any questions?

Second Thursdays

- April 9 - Webinar 12: Simple Peripheral as Software Stimulus**
- May 14 - Webinar 13: Two Bare-Metal Applications on PolarFire® SoC**
- June 11 - Webinar 14: The PolarFire SoC Icicle Kit Model in Renode**
- July 9 - Webinar 15: Linux® on Renode**
- Aug. 13 - Webinar 16: Building Applications for Linux on PolarFire SoC**
- Sep. 10 - Webinar 17: Real-Time (AMP Mode) on PolarFire SoC**