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

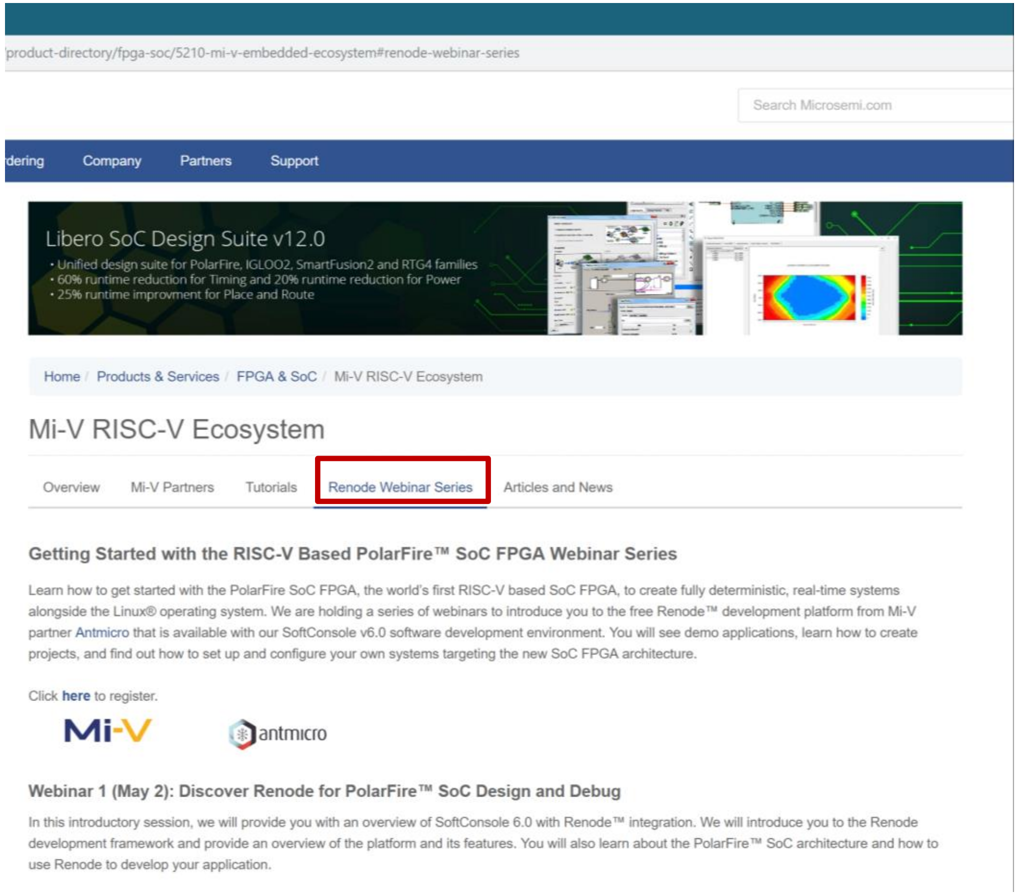
Hugh Breslin, Design Engineer

Thursday Apr. 9, 2020

SMART | CONNECTED | SECURE

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

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

GitHub - antmicro/litex-vexriscv-tensorflow-lite-demo: TF Lite demo on LiteX/VexRiscv soft RISC-V SoC on a Digilent Arty board - Mozilla Firefox

https://github.com/antmicro/litex-vexriscv-tensorflow-lite-demo

Why GitHub? Enterprise Explore Marketplace Pricing Search Sign in Sign up

antmicro / litex-vexriscv-tensorflow-lite-demo

Watch 11 Star 12 Fork 2

Code | Issues 7 | Pull requests 0 | Actions | Projects 0 | Security | Insights

Join GitHub today

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

TF Lite demo on LiteX/VexRiscv soft RISC-V SoC on a Digilent Arty board https://antmicro.com/blog/2019/12/tfl...

27 commits | 5 branches | 0 packages | 0 releases | 5 contributors | Apache-2.0

Branch: master | New pull request | Find file | Clone or download

PiotrZierhoffer Fix path to binary in Renode script ... Latest commit 25a11fd 14 days ago

binaries/magic_wand	Add magic_wand prebuilts	19 days ago
litex-buildenv @ e676436	Add submodules	4 months ago
renode	Fix path to binary in Renode script	14 days ago
tensorflow @ 0a1a7a6	Update tensorflow	last month
.gitmodules	remove Zephyr submodule	last month
.travis.yml	west: switch branch	19 days ago
LICENSE	Add LICENSE	last month
README.md	west: switch branch	19 days ago

README.md

Demo

PAC1934 on the Icicle Kit

Interfacing the Model with sysbus via I2C

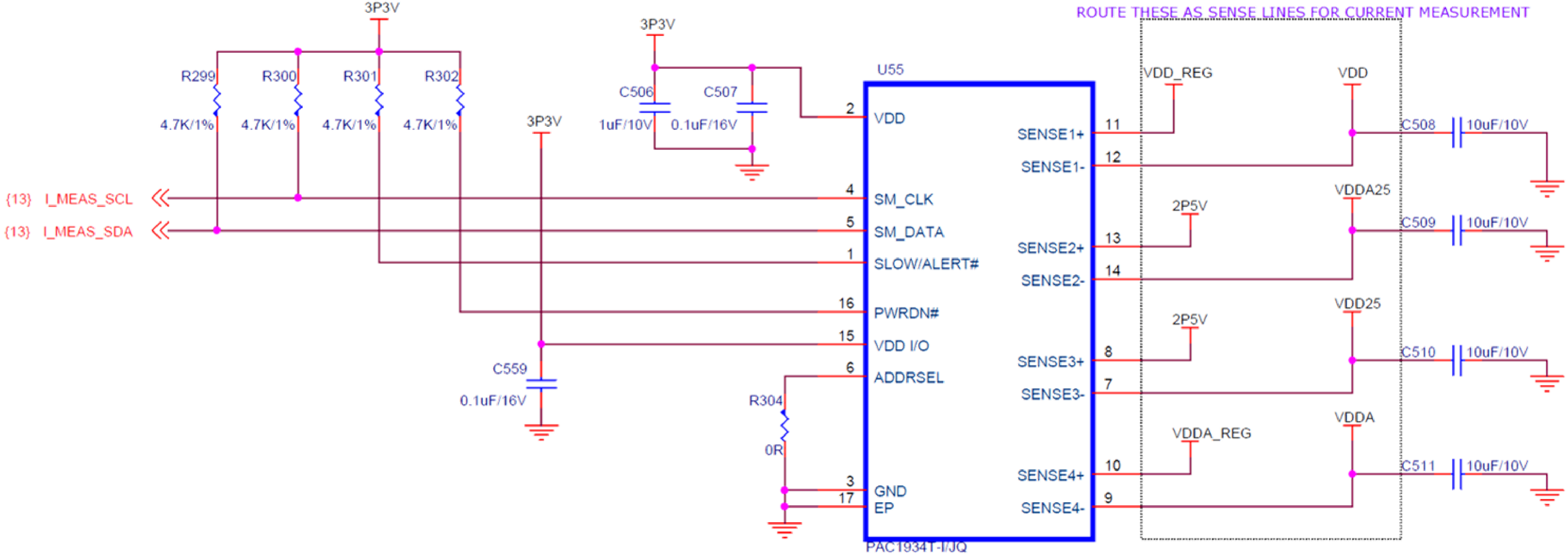
Using the I2C Drivers

Reading Back Values

Monitoring Voltage and Current

PAC1934 on the Icicle Kit

Current Measurement 1



PAC1934 on the Icicle 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 LoadPlatformDesciptionFromString "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
│
```

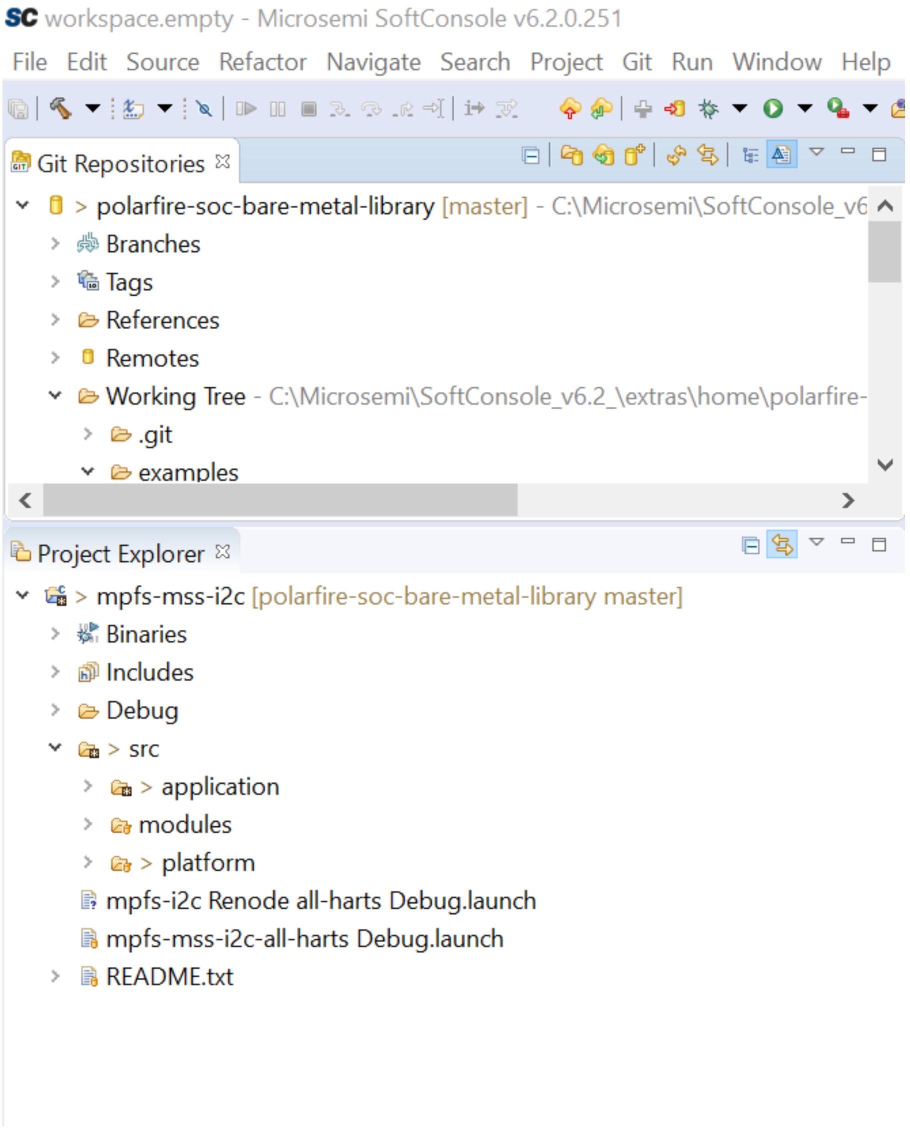

MICROCHIP

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**

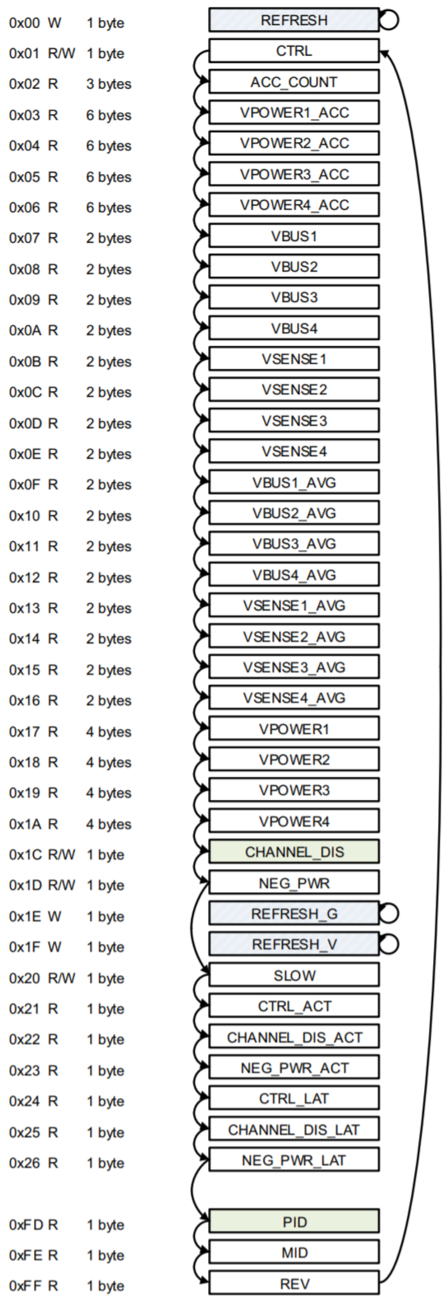
- **Using the I2C write read function the register pointer can be set and the register read**



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 == 0xB
    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);
```



MICROCHIP

Monitoring Voltage and Current

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

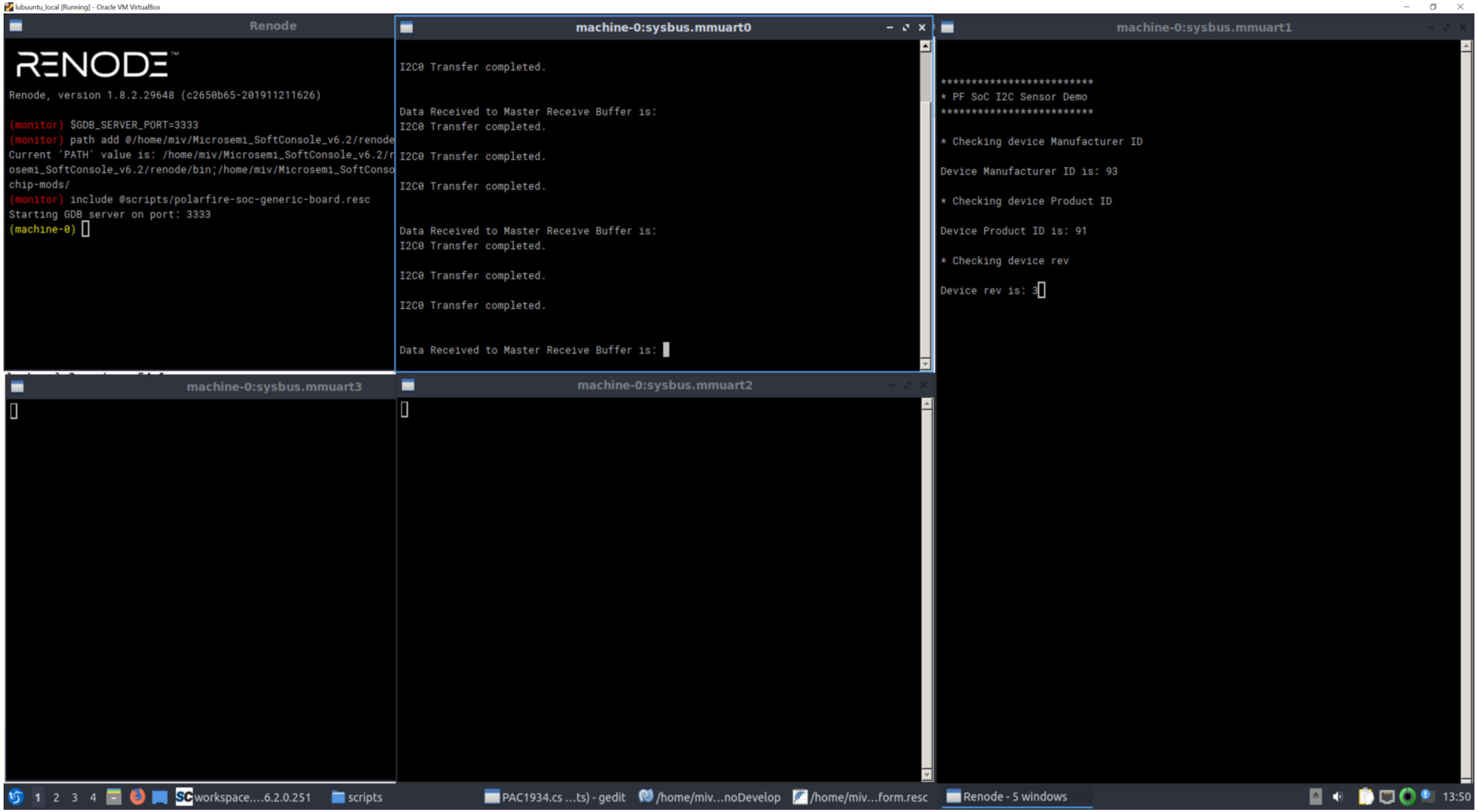
```
while(1){
    MSS_UART_polled_tx_string(gp_sys_uart, (const uint8_t *)"\n\r\n\r Sensor refresh");
    sensor_reg_check(REFRESH_REG);
    VPowerACC[0] = sensor_reg_check(VPOWER1_ACC_REG);
    VPowerACC[1] = sensor_reg_check(VPOWER2_ACC_REG);
    VPowerACC[2] = sensor_reg_check(VPOWER3_ACC_REG);
    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**

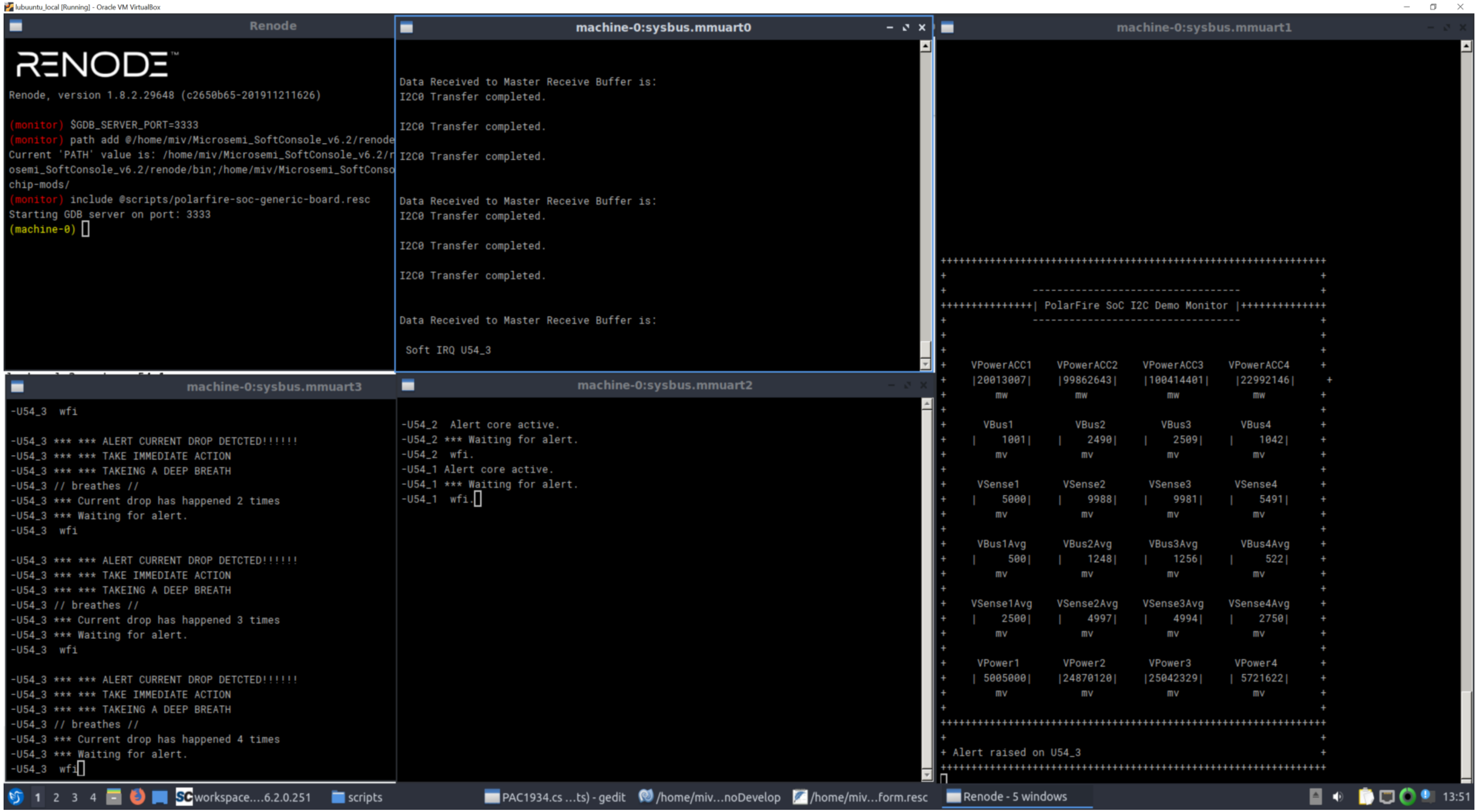
```
drawUIUpdate(&VPowerACC, &VBus, &VSense, &VBusAvg, &VSenseAvg, &VPower);
blankLine();

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



MICROCHIP

Monitoring Voltage and Current



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