Hands-On

IPMC Workshop

Fatih BELLACHIA, Thierry BOUEDO, Sébastien CAP, Nicolas DUMONT-DAYOT,

Sylvain LAFRASSE, Nicolas LETENDRE, Alexis VALLIER

1. Creating your Project Area

While working on ICARE, you should never have to modify the release code directly.
On the contrary, all your code will live in its own 'Project Area', relying on ICARE underneath.

To achieve this, you first need to set up the ICARE environment, then generating all the needed files with a provided script.

On your ICARE development station or VM, we are assuming ICARE is installed in $ICARE.

a. Setting up the ICARE Environment

```
% ICARE=/home/workshop/ICARE
% source $ICARE/releases/ICARE-00-03-00/admin/cmt/setup.sh
```

b. Generating the Files

```
% mkdir ~/workarea
% cd ~/workarea
% create_project_area.sh -a firstname.fullname@example.com -n
localhost -p KIWI
…
```

2. Hello World (blinky)

a. Setting up the project environment

```
% cd ~/workarea/KIWI/cmt
% source setup.sh
```

b. Compiling "`blinky`"

```
% cd ~/workarea/demo/cmt
% make
```

c. Checking Ethernet connectivity with IPMC

```
% ping <host>  (IP level)
% telnet <host> 7  (TCP echo)
…
```

d. Uploading "`blinky_IPMC`" to IPMC

```
% fwu -t IPMC -n <host> -f ../arm-gcc47-dbg/blinky_IPMC.bin
% fwu -t IPMC -n <host> -u  (wait ~25s)
```

3. KIWI Board

a. Configuration

In `~/workarea/config/config/ioconfig.h`, select your handle switch polarity to **normally closed** for example.

b. Describing Board's Information (FRU)

FRU informations are in `~/workarea/KIWI/share/data/KIWI.m4`

c. Describing Board's Sensors (SDR)

SDR information are in `~/workarea/KIWI/src/sdr_data.c`

d. Adding the missing code lines

Edit `~/workarea/KIWI/src/sensors.c` to add code lines to read **ADS1115** sensor (see `$ICARE/releases/ICARE-00-03-00/ads1115/ads1115/ads1115.h`) and to return the *raw* 8-bit ADC value .

e. Compiling

```
% cd ~/workarea/KIWI/cmt
% make rebuild
```

f. Uploading

```
% fwu -t IPMC -n <host> -f ../arm-gcc47-dbg/bmc_IPMC.bin
% fwu -t IPMC -n <host> -u  (wait ~25s)
% fwu -t IOIF -n <host> -f ../arm-gcc47-dbg/bmc_IOIF.bin
% fwu -t IOIF -n <host> -u  (wait ~25s)
```

g. Connecting to the Shelf Manager

Open an SSH session to your shelf-manager IP address. Default login is root without password.

```
# ./tools/dump_voltage.sh
# ./tools/dump_temperature.sh
```

4. Debugging

a. Uncomment line in `~/workarea/KIWI/cmt/requirements` to activate bus fault module

b. Compiling

```
% cd ~/workarea/KIWI/cmt
% make rebuild
```

c. Uploading

```
% fwu -t IPMC -n <host> -f ../arm-gcc47-dbg/bmc_IPMC.bin
% fwu -t IPMC -n <host> -u
```
(wait ~25s)

d. Getting the Program Counter (PC) and the Link Register (LR) from serial console

e. Finding the line which is throwing the bus fault and editing the source file

```
% arm-none-eabi-addr2line -e ../arm-gcc47-dbg/bmc_IPMC.elf
<program counter value>
% <editor> <filename>.c
```

f. Finding the callee and the caller with “Link Register value – 2”

```
% arm-none-eabi-addr2line -e ../arm-gcc47-dbg/bmc_IPMC.elf
<linker register value - 2>
% <editor> ../src/<filename>.c
```

5. Creating a TMP102 Driver

We will use the default configuration of TMP102 device ⇒ tmp102Config(…) do nothing and return 0
We will use the I2C Write/Read accesses to read temperature.

Table 6. Pointer Register Byte

P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	Register Bits	

Table 7. Pointer Addresses

P1	P0	REGISTER
0	0	Temperature Register (Read Only)
0	1	Configuration Register (Read/Write)
1	0	T_{LOW} Register (Read/Write)
1	1	T_{HIGH} Register (Read/Write)

Table 8. Byte 1 of Temperature Register(1)

D7	D6	D5	D4	D3	D2	D1	D0
T11 (T12)	T10 (T11)	T9 (T10)	T8 (T9)	T7 (T8)	T6 (T7)	T5 (T6)	T4 (T5)

(1) Extended mode 13-bit configuration shown in parenthesis.

Table 9. Byte 2 of Temperature Register(1)

D7	D6	D5	D4	D3	D2	D1	D0
T3 (T4)	T2 (T3)	T1 (T2)	T0 (T1)	0 (T0)	0 (0)	0 (0)	0 (1)

(1) Extended mode 13-bit configuration shown in parenthesis.

a. Creating `tmp102` package

```
% cd ~/workarea
% cmt create tmp102 v0r1
```

b. Creating `tmp102` directory and adding header file `tmp102.h` (see Annex 1)

```
% cd ~/workarea/tmp102
% mkdir tmp102
% cd tmp102
% <editor> tmp102.h
…
```

c. Writing `tmp102Config(…)` and `tmp102Read(…)` functions

```
% cd ~/workarea/tmp102/src
% <editor> tmp102.c
…
```

d. Editing `requirements` file and adding CMT statements (see Annex 2)

```
% cd ~/workarea/tmp102/cmt
% <editor> requirements
```

e. Compiling `tmp102` library

```
% cd ~/workarea/tmp102/cmt
% make
```

f. Compiling KIWI project

```
% cd ~/workarea/KIWI/cmt
% make rebuild
```

g. Uploading

```
% fwu -t IPMC -n <host> -f ../arm-gcc47-dbg/bmc_IPMC.bin
% fwu -t IPMC -n <host> -u (wait ~25s)
% fwu -t IOIF -n <host> -f ../arm-gcc47-dbg/bmc_IOIF.bin
% fwu -t IOIF -n <host> -u (wait ~25s)
```

Annex 1: tmp102.h

~~~~~~ ✂ ~~~~~~

```
/* Includes -------------------------------------------------------------------*/
#include <stdint.h>
#include <stdbool.h>

/*
 * ICARE
 */
#include <i2c/i2clib.h>

#ifndef __TMP102_H__
#define __TMP102_H__

/* Exported constants --------------------------------------------------------*/
  /* I2C slave address */
#define TMP102_SLAVE_ADDR_GND_PIN 0x48 // 1001000 (7-bit address)
#define TMP102_SLAVE_ADDR_VDD_PIN 0x49 // 1001001 (7-bit address)
#define TMP102_SLAVE_ADDR_SDA_PIN 0x4A // 1001010 (7-bit address)
#define TMP102_SLAVE_ADDR_SCL_PIN 0x4B // 1001011 (7-bit address)

/* Exported functions --------------------------------------------------------*/
extern int  tmp102Config (uint32_t i2cChannel, uint8_t address);
extern bool tmp102Read   (uint32_t i2cChannel, uint8_t address, int16_t *sValue);

#endif // __TMP102_H__
```

~~~~~~

Annex 2: requirements

~~~~~~ ✂ ~~~~~~

```
public

use ICAREPolicy *
use rcc v0r*
use i2c v3r*

include_dirs $(tmp102_root)

macro tmp102_use_arm_linkopts "-L$(TMP102ROOT)/${ICARECONFIG} -ltmp102"

private

#-----------------------------------------------------------------------------
# ARM libraries
#-----------------------------------------------------------------------------
macro_remove arm_cflags "-pedantic"

document arm-library tmp102 tmp102.c
```

~~~~~~

Annex 3: Virtual Machine Readme

```
Welcome to the IPMC Workshop @ CERN (9-11 Oct. 2018) !

This file describes how to use the material present on the USB key to set up
the LAPP IPMC ICARE Software DekKit in 3 steps. Those instructions and the VM
were successfully tested on macOS 10.13.6 and Windows 10 (assuming
virtualization is enabled in your BIOS).

1) Install VirtualBox
You will find the appropriate installer in the 'VirtualBox' directory, grouped
by OS and distributions. Just use the usual installation technique of your OS
of choice.

2) Install VirtualBox Extension Pack
Once installed, please add the corresponding VirtualBox Extension Pack,
according to your exact version of VirtualBox. To do so, please launch
VirtualBox, go to its Preference pane, then click on the 'Extension' tab and
load the Extension Pack file from the VirtualBox directory using the '+' icon.

3) Load and Use the VM
From within VirtualBox application, please use 'Import Appliance...' from the
'File' menu to select the 'LAPP_IPMC.ova' file in the 'Documents' folder. This
may take a while to complete.

Once done, you can start the VM and log in through SSH:
# ssh -Y workshop@localhost -p 2222

The password is the username.

4) (Optionally) Install SSH on Windows
We also provide you with the latest version of Putty in case you don't have an
SSH client already installed on your system.

5) VM Details
The provided VM runs CERN SLC 6.10 OS.
It includes ICARE 00-03-00 release.

The main user name and password is 'workshop' without quotes.
The root password is the same as the main user (but should not be needed).
The keyboard layout is QWERTY.

There are 3 Network Port Forwarding set up:
- Host port 2222 is redirected to the VM port 22 for SSH access;
- Host port 3333 is redirected to the VM port 3333 for GDB access;
- Host port 4444 is redirected to the VM port 4444 for OpenOCD access;
You may need to open your host firewall accordingly to support remote accesses.

There are also 2 USB Device Filters in place, to automatically attach the VM to
the JTAG adapters plugged in your host computer:
- Luminary Micro ICDI Board (Vendor ID:0403 / Product ID:bcda);
- Olimex OpenOCD JTAG ARM-USB-OCD-H (Vendor ID:15ba / Product ID:002b);

Provided OpenOCD server can then be started using: ~workshop/OpenOCD/start.sh
```