IPMC/ICARE
Framework

Karen Hayrapetyan

Queen Mary
University of London

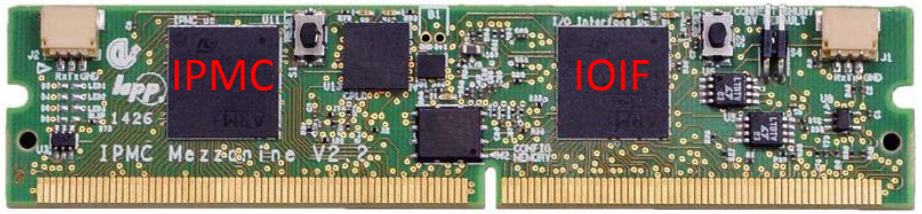
Outline

- LAPP IPMC Hardware
- Software environment and firmware tools
- New Firmware (version 5.2) for IPMC
- IPMC Firmware upgrade
- ICARE software Framework
- FRU/SDR records and generator tools
- eFEX sensors and FRU/SDR
- New CERN IPMC design - Pigeon Point IPMC solution

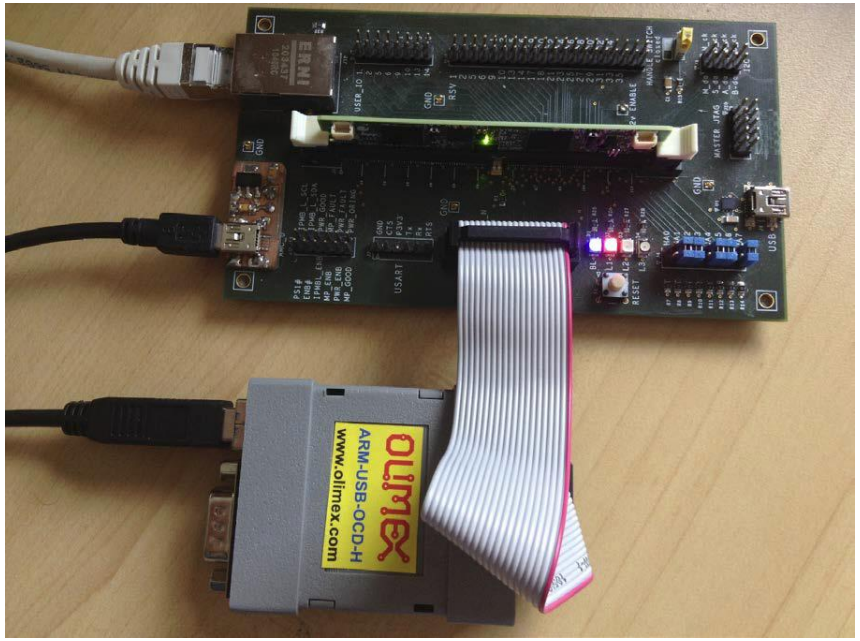
LAPP IPMC Hardware

IPMC Serial Port

IOIF Serial Port



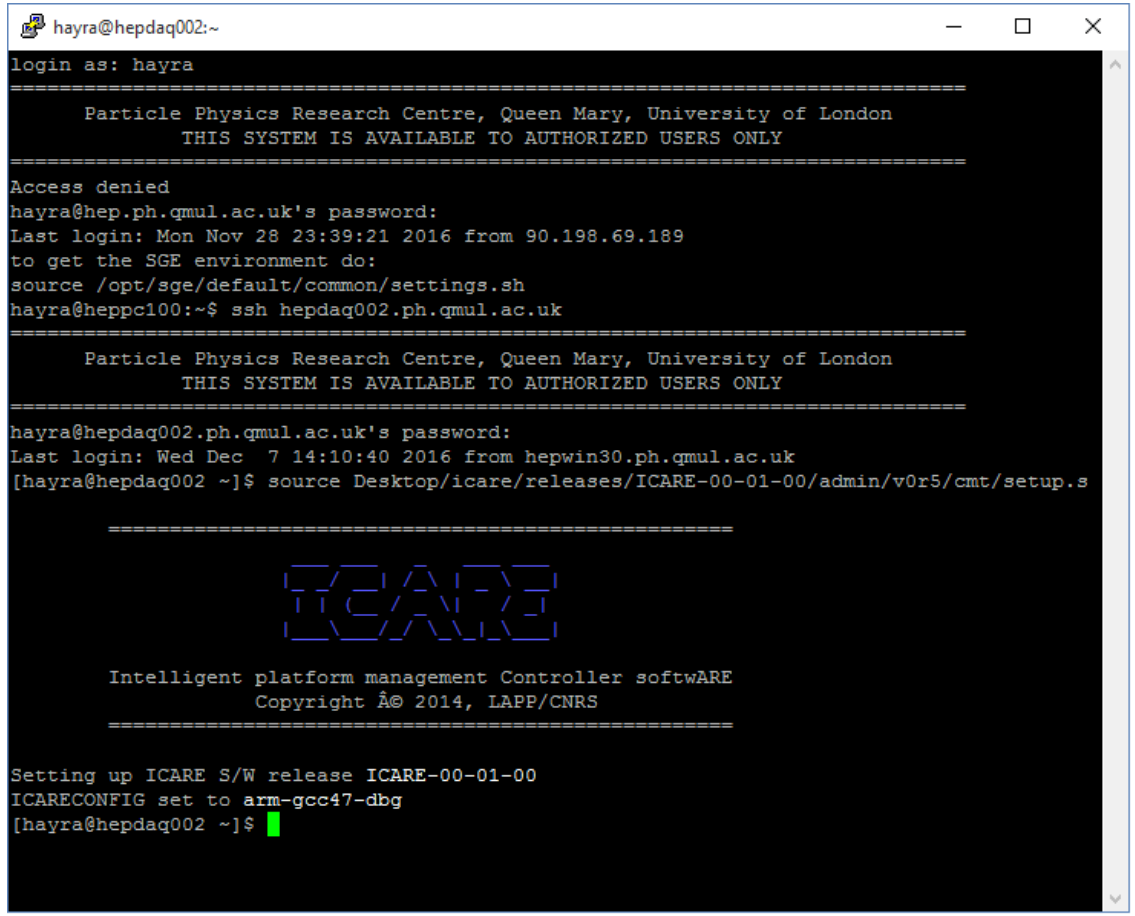
LAPP IPMC



IPMC setup at QMUL

- The IPMC has 2 micro-controllers called IOIF and IPMC
- Each micro-controller has an internal flash memory for storing the firmware and One Time Programming memory for storing permanent information
- There is also an external flash memory which is accessible by the two micro-controllers through a CPLD
- PC communicates with the IPMC through the IPMC JTAG chain and through the serial port of each micro-controller on the IPMC
- New firmware (version 5.2) allows Ethernet access to JTAG chain
- Scientific Linux 6 or 7 required to program IPMC firmware

Software Environment



- Scientific Linux host development
- 32-bit ARM Cortex-M4 microcontroller
- Written in standard ANSI C
- 32-bit GCC compiler (4.7.0) tool chain
- Open Source Configuration Management Environment: CMT
- FRU generation utility (using M4 preprocessor)
- New FRU/SDR generation GUI (using Qt 5)
- OpenOCD (0.9.0) utility (Linux/Windows)
  - Requiers USB to JTAG interface Debug-Adapter-Hardware
    - Olimex ARM-USB-OCD-H
    - NGX technology

New Firmware and Tools

ICARE-00-02-00 available since January 2017

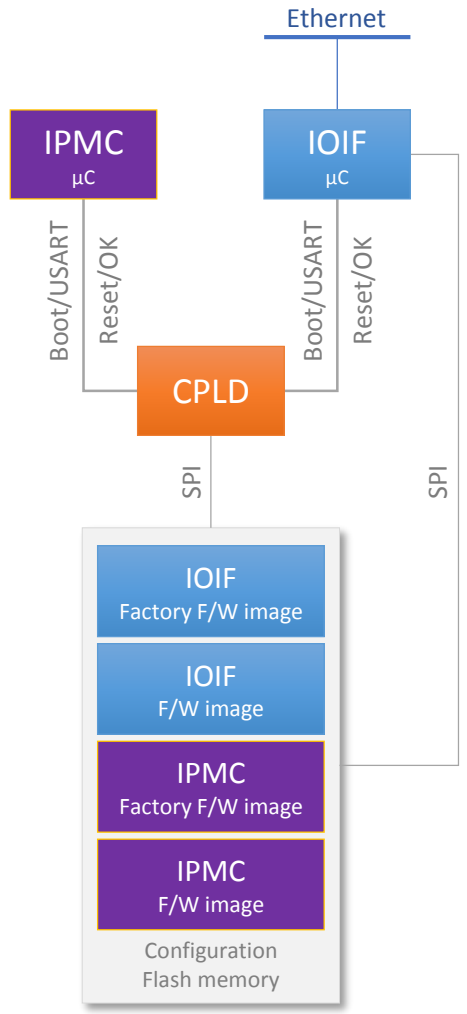
- tools for compiling, loading, debugging and executing the IPMC software.
- source code (ICARE framework) for the IPMC.
- Some examples for sensors, SDR/FRU…
- New FRU/SDR generator (GUI) available

IPMC firmware

- Management of ATCA blade
  - FRU/SDR, Sensors, Backplane E-Keying.
- IPMC firmware upgrade (proprietary solution) over TCP/IP on 10/100 MbE (lwip).
- Management of AMC
- JTAG Master: Xilinx Virtual Cable server (Xilinx iMPACT and VIVADO s/w)
- Version Information available via OpenOCD or by a command from the Shelf Manager

Firmware Upgrade

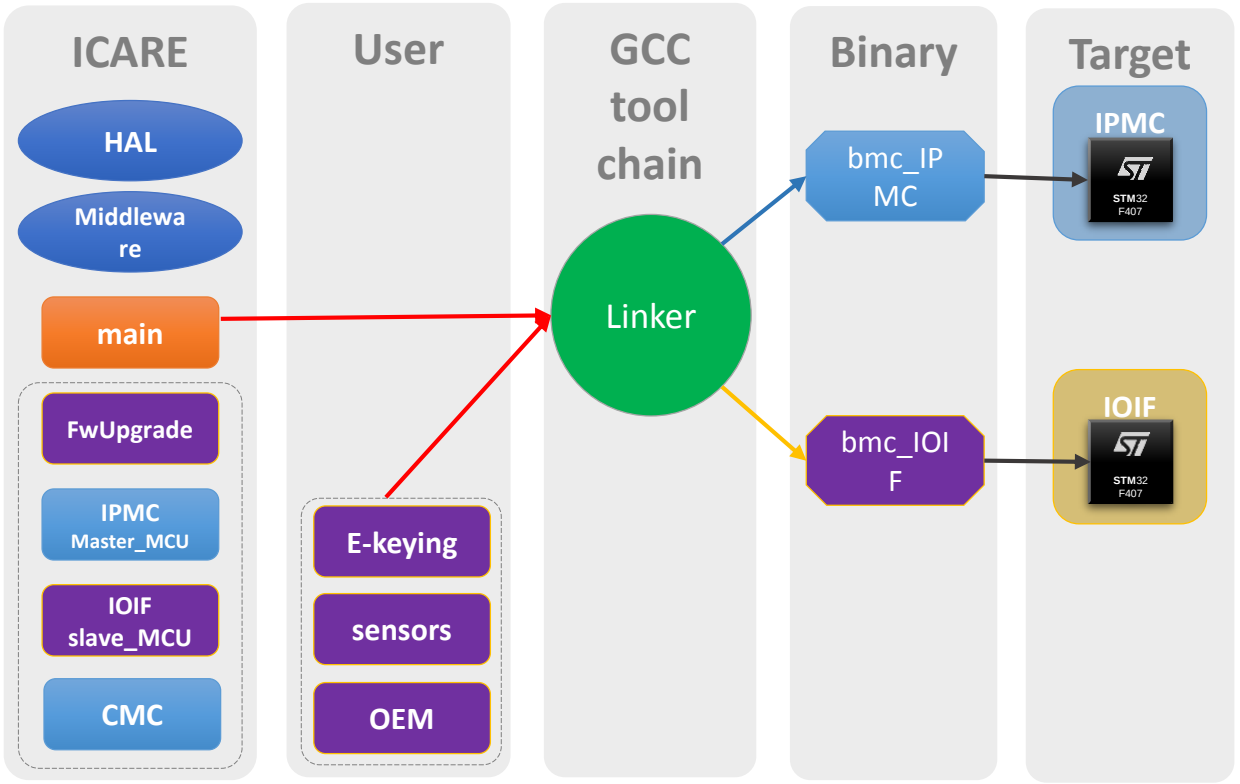
- Factory Firmware is initially stored in both the configuration flash memory and micro-controllers internal flash memory.
- The factory firmware always remains available in the configuration flash memory.
- The micro-controllers will revert to it after 3 failing consecutive attempts of upgrading the internal flash memory.
- Firmware upgrade requires new firmware in the IPMC CPLD.
  - Version 5.2 is available from the ICARE twiki page.
  - Requires the USB-JTAG Adaptor and OpenOCD tool for upgrading.
  - Version 5.2 allows Ethernet firmware upgrade (tested)



Point-to-Point fabric E-Keying and Clock E-Keying is not yet implemented

Queen Mary University of London

ICARE Software Framework



- ICARE framework 00-02-00 release project structure
- Template projects available
- User modifications needed for specific hardware
  - I/O pin polarity ioconfig.h
  - Extra board information boardconfig.h
  - E-Keying cannels p2p_ekeying.c
  - Sensors sensors.c
  - Configuration cfg_data.h

FRU/SDR Record

- **FRU (Field Replaceable Unit) Information**
  - Description (manufacturer, product name, model, serial number)
  - Description of connections to the backplane (E-Keying). Type of protocol, on which channel
  - Power requirement
- **SDR (Sensor Data Record) describes the configuration of the "sensors" of FRU**
  - Type, location and number of sensors
  - Warning threshold
  - Ability to generate an event
  - Interpretation of data
  - Compulsory temperature sensor
- **SEL (System Event Log)**
  - Keep a trace of the different events of the sensors
  - Obligatory for the Shelf Manager, not for the IPMC

FRU management

Activation/De-activation state machine

M0 (not installed)
M1 (installed)
- SM read the FRU information
  - Current
  - E-keying
  - Clock

M2 (activation requested)
M3 (activation in progress)
- Enable power
- E-keying control
- Clock configuration

M4 (active)
- Sensor, Power, Handle switch monitoring

M5 (Deactivation requested)
- E-keying de-activation
- Clock de-activation

M6 (Deactivation in progress)
- Disable power

Queen Mary University of London

EEPROM

- Used to store FRU/SDR information (generated corresponding C structures)
- The Framework support EEPROM M24256
- **By default FRU/SDR data are stored in 'IOIF' MCU memory on IPMC**

Sensor Reading

- Create library (i.e. package) to allow to read sensors
  - <xxxx>Init(…)
    - Do some initializations
    - Create periodic task (e.g. ~ 1s) which allows to read sensor values from microcontroller
  - <xxxx>Read(…)
    - Implements a function to read sensor value from array according to sensor ID

Sensor integration

- The Framework support the following sensors
  - AD7414 - I2C Temperature sensor
  - LTC4151 - High Voltage I2C Current and Voltage Monitor
  - LTC2499 - I2C 24-Bit 8-/16-Channel Delta Sigma ADC
  - IQ65033QMA10 - I2C ATCA Power Interface Module
- Registering sensor with the ResourceBroker library e.g.:

```
/*-------------------------------------------------------------------------*/
bool SensorsInit()
/*-------------------------------------------------------------------------*/
{
  RBResource_t stResource;

  // AD7414 Temperature
  stResource.ucChannelId  = CHANNEL_I2C_SENSOR;
  stResource.ucAddress    = SDR_AD7414_I2C_ADDR;
  stResource.ucIdentifier = SDR_NUM_AD7414_TEMP;
  stResource.fnInit       = InitSensorAD7414;
  stResource.fnRead       = ReadSensorAD7414;
  stResource.fnWrite      = WriteSensorAD7414;

  if (ResourceBrokerAddResource("AD7414 Temp", &stResource) == false)
    return false;

  …

  return true;
}
```

Channel #

I2C address

Unique sensor ID

User's functions

SDR - Sensor Data Record

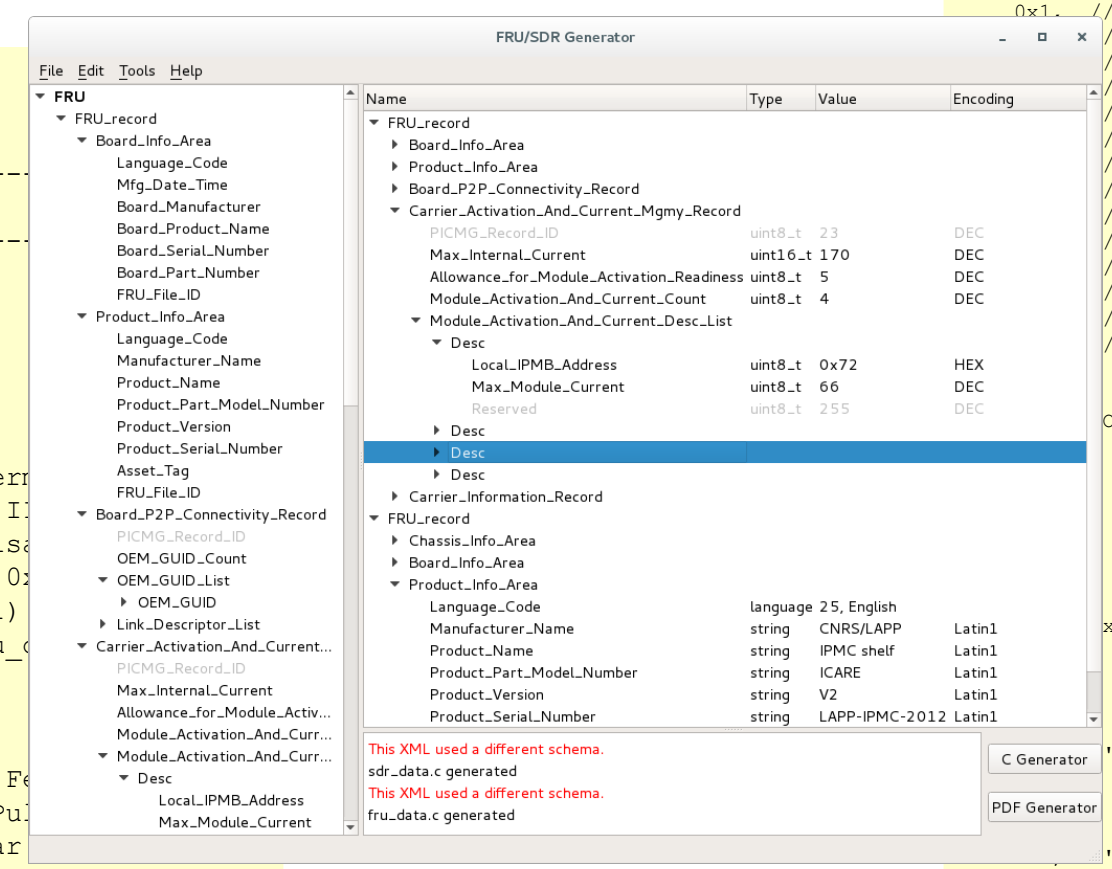
- Channel ID
  The channel IDs are defined in channel/channel.h
  CHANNEL_I2C_MGT (managed by IPMC)
  CHANNEL_I2C_SENSOR (managed by IPMC)
  CHANNEL_I2C_USER_IO
  CHANNEL_I2C_IPM_IO
  - <Full|Compact> Sensor Record
    set BYTE 7 [4:7] channel_num
- Sensor Number
  Unique number identifying the sensor *(e.g. see cfg_data.h)*
  - <Full|Compact> Sensor Record
    set BYTE 8 [0:7] sensor_number
- I2C address of sensor
  - Compact Sensor Record
    set BYTE 31 [0:7] OEM - Reserved for OEM use
  - Full Sensor Record
    set BYTE 47 [0:7] OEM - Reserved for OEM use

```
/ ================================================================================
// Core temperature sensor record
// ================================================================================
#define SDR_AD7414_TEMP_STR_LEN 9
FullSensorRecord_t sdr_ad7414_temp = { // MANDATORY LF 20110511
 /* RECORD HEADER */
 0, 0,                  // 1-2 Record ID (filled in at startup)
 SDR_VERSION,           // 3 SDR Version - Version of the Sensor Model
 FULL_SENSOR_RECORD_TYPE,       // 4 Record Type Number
 // BYTE 5 - record_len - Number of remaining record bytes following
 SDR_LEN(FullSensorRecord_t, SDR_AD7414_TEMP_STR_LEN),

 /* RECORD KEY BYTES */
 // BYTE 6
 0,   //  [0] id_type - 0b = owner_id is IPMB Slave Address, 1b = system software ID
 0,   // [1:7] owner_id - 7-bit I2C Slave, fill during init

 // BYTE 7
 0,   // [0:1] sensor_owner_lun
 0,   // [2:3] reserved
 CHANNEL_I2C_SENSOR,    // [4:7] channel_num

 SDR_NUM_AD7414_TEMP,  // 8 sensor number

 /* RECORD BODY BYTES */
 PICMG_ENTITY_FRONT_BOARD, // 9 entity_id - see PICMG 3.0 $3.4.3.1
 // BYTE 10
 //  [0:6] entity_instance - 60h-7Fh device-relative Entity Instance.
 ENTITY_0_INSTANCE,
 //   [7] entity_type - 0b = treat entity as a physical entity, 1b = logical entity
 ENTITY_TYPE_PHYSICAL,

 // BYTE 11 - Sensor initialization
 1,   // [0] sensor scanning enabled
 1,   // [1] event generation enabled
 0,   // [2] init sensor type and event/reading type code
 0,   // [3] enable hysteresis
 0,   // [4] enable thresholds
 1,   // [5] enable events
 1,   // [6] enable scanning
 0,   // [7] settable

 // BYTE 12 - Sensor capabilities
 0,   // [0:1] event_msg_control
 2,   // [2:3] sensor_threshold_access
 3,   // [4:5] sensor_hysteresis_support
 1,   //  [6] sensor_manual_support
 0,   //  [7] ignore_sensor

 ST_TEMPERATURE, // 13 sensor_type
 0x01,          // 14 event_type_code
 0xF0, 0x0F,    // 15,16 assertion event mask, LSB first
 0xF0, 0x0F,    // 17,18 deassertion event mask, LSB first
 0x3F, 0x00,    // 19,20 reading_mask, LSB first

 // BYTE 21
 0,   //  [0] percentage - 0b
 0,   // [1:2] modifier_unit - 00b = none
 3,   // [3:5] rate_unit - 011b = every second
 2,   // [6:7] Analog (numeric) Data Format - 2's complement (signed)

 1,       // 22 sensor_units2
 0,       // 23 sensor_units3
 0,       // 24 linearization
 1,       // 25      M_lsb (8 bits)
 0,       // 26 [0:5] M Tolerance
 0,       //   [6:7] M_msb
 0,       // 27      B_lsb (8 bits)
 0,       // 28 [0:5] B Accuracy_lsb
 0,       //   [6:7] B_msb
 0,       // 29 [0:1] Sensor Direction
 0,       //   [2:3] Accuracy exp
 0,       //   [4:7] Accuracy_msb
 0,       // 30 [0:3] B exponent
 0,       //   [4:7] R exponent
 0,       // 31 Analog characteristic flags
 25,      // 32 Nominal Reading
 (uchar)  55, // 33 Normal Maximum - Given as a raw value
 (uchar) -40, // 34 Normal Minimum - Given as a raw value
 (uchar) 125, // 35 Sensor Maximum Reading
 (uchar) -55, // 36 Sensor Minimum Reading
 (uchar)  75, // 37 Upper non-recoverable Threshold
 (uchar)  55, // 38 Upper critical Threshold
 (uchar)  45, // 39 Upper non-critical Threshold
 (uchar) -55, // 40 Lower non-recoverable Threshold
 (uchar) -55, // 41 Lower critical Threshold
 (uchar) -55, // 42 Lower non-critical Threshold
 2,       // 43 Positive-going Threshold Hysteresis value
 2,       // 44 Negative-going Threshold Hysteresis value
 { 0, 0 },  // 45-46 reserved. Write as 00h
 SDR_AD7414_I2C_ADDR,       // 47 OEM - Reserved for OEM use

 // BYTE 48
 SDR_AD7414_TEMP_STR_LEN, // [0:4] length of following data, in characters
 0,             //  [5] reserved
 3,             // [6:7] id string type, 3 <=> 8-bit ASCII + Latin 1
 // BYTE 49+ Sensor ID String bytes
 { 'C', 'o', 'r', 'e', ' ', 'T', 'e', 'm', 'p' }   // sensor id string
};
```

Example Temperature sensor SDR record

FRU Generator utility (GUI)

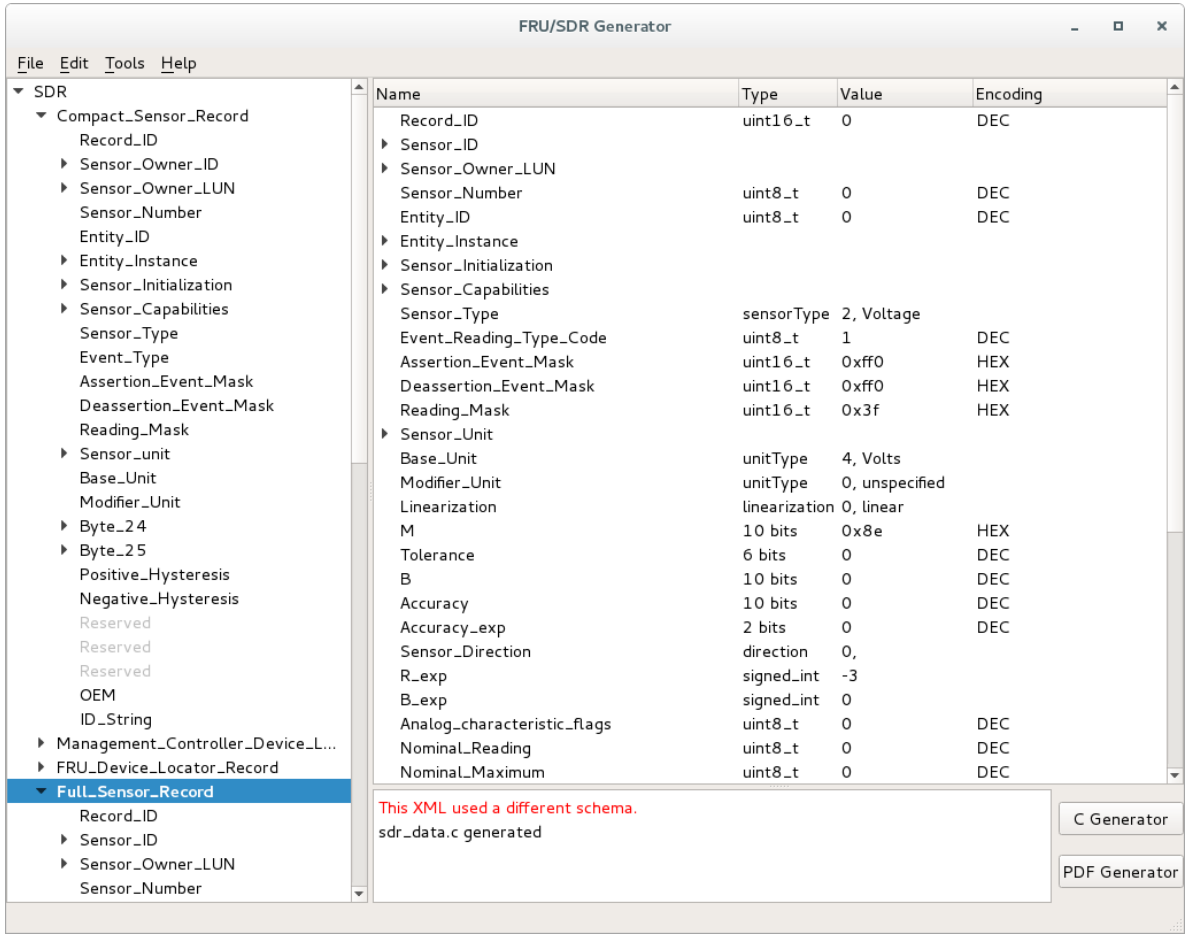
```
divert(`-1')
include(`FRU.m4')
dnl
dnl --------------------------------
dnl FRU #0
dnl --------------------------------
dnl
set_FRU_ID(0)
FRU_INIT(FRU_ID)
dnl
dnl IPMI_BOARD_INFO_AREA
dnl
IPMI_BOARD_MANUFACTURER(FRU_ID, Ferr
IPMI_BOARD_PRODUCT(FRU_ID, Pulsar I
IPMI_BOARD_PART_NUMBER(FRU_ID, Pulsa
IPMI_BOARD_MFG_DATE(FRU_ID, 0xA0, 0x
IPMI_BOARD_SERIAL_NUMBER(FRU_ID, 1)
IPMI_BOARD_FRU_FILE_ID(FRU_ID, fru_d
dnl
dnl IPMI_PRODUCT_INFO_AREA
dnl
IPMI_PRODUCT_MANUFACTURER(FRU_ID, Fe
IPMI_PRODUCT_PART_NUMBER(FRU_ID, Pul
IPMI_PRODUCT_PRODUCT(FRU_ID, Pulsar
IPMI_PRODUCT_VERSION(FRU_ID, 1)
IPMI_PRODUCT_SERIAL_NUMBER(FRU_ID, 1)
IPMI_PRODUCT_ASSET_TAG(FRU_ID)
IPMI_PRODUCT_FRU_FILE_ID(FRU_ID)
…
```




```
static fru0_data_t fru0_data = {

  { /*--- COMMON HEADER ---*/

    0x1,   // 3:0 - format version number = 1h for this specification
           / 7:4 - reserved, write as 0000b
           / Internal Use Area Starting Offset (in multiples of 8 bytes)
           / 00h indicates that this area is not present
           / Chassis Info Area Starting Offset (in multiples of 8 bytes)
           / 00h indicates that this area is not present
           / Board Area Starting Offset (in multiples of 8 bytes)
           / 00h indicates that this area is not present
           / Product Info Area Starting Offset (in multiples of 8 bytes)
           / 00h indicates that this area is not present
           // MultiRecord Area Starting Offset (in multiples of 8 bytes)
           / 00h indicates that this area is not present
           / PAD, write as 00h
           / Common Header Checksum (zero )


           OARD INFO AREA ---*/

                    //  1 Board Area Format Version
                    //    7:4 - reserved, write as 0000b
                    //    3:0 - format version number = 1h for this
                          specification.
                    //  1 Board Area Length (in multiples of 8 bytes)
                    //  1 Language Code (See section 15)
    x72, 0x92,      //  3 Mfg. Date / Time
                    //    Number of minutes from 0:00 hrs 1/1/96, LSbyte
                          first (little endian)
                    //  1 Board Manufacturer type/length byte
                    //  P Board Manufacturer bytes
    ', 'r', 'm', 'i', 'l', 'a', 'b',

                    //  1 Board Product Name type/length byte
                    //  Q Board Product Name bytes
    ', 'l', 's', 'a', 'r', ' ', 'I', 'I', 'b',

    0xC1,           //  1 Board Serial Number type/length byte*
                    //  N Board Serial Number bytes*
    '1',

    0xCA,           //  1 Board Part Number type/length byte
                    //  M Board Part Number bytes
    'P', 'u', 'l', 's', 'a', 'r', ' ', 'I', 'I', 'b',

    0xCC,           //  1 FRU File ID type/length byte*
```

Generating corresponding C structures for EEPROM

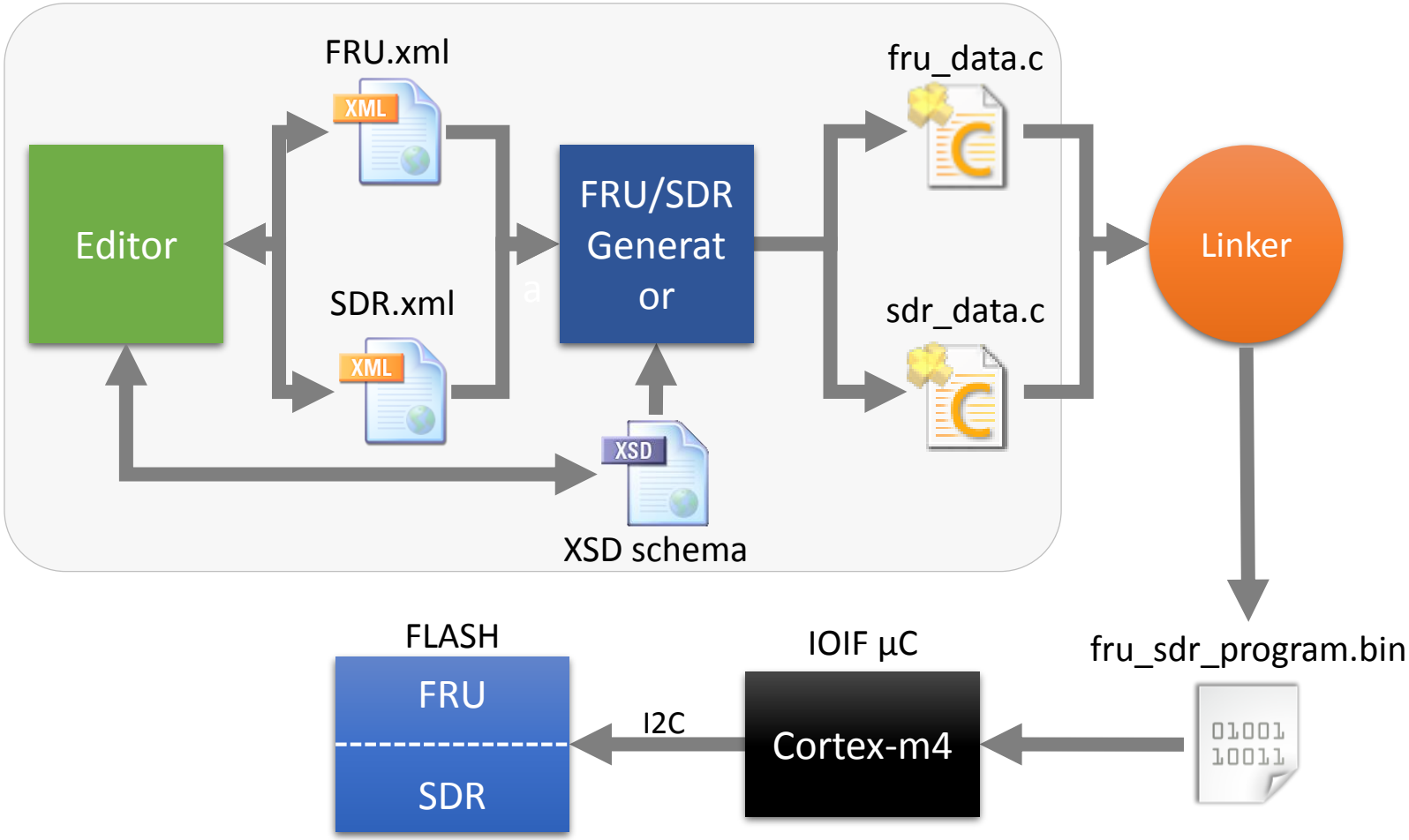
SDR Generator utility (GUI)




```
#include <dataStorage/sdr_data.h>
#include <channel/channel.h>

// User's header file with specific defines
#include "cfg_data.h"

#define SDR1_ID_STR_LEN                 15
MGMTCtrlDeviceLocatorRecord_t sdr_mgmt_ctrl_locator = { // MANDATORY LF
20110511
  /* RECORD HEADER */
  0, 0,                                 // 1-2 Record ID (filled in at
startup)
  SDR_VERSION,                          // 3   Version of the Sensor Model
  MGMT_CTRL_DEVICE_LOCATOR_RECORD_TYPE, // 4   Record Type Number
  // BYTE 5 - record_len - Number of remaining record bytes following
  SDR_LEN(MGMTCtrlDeviceLocatorRecord_t, SDR1_ID_STR_LEN),

  /* RECORD KEY BYTES */
  // BYTE 6
  0,    //   [0] reserved - set as 0b
  0,    // [1:7] owner_id - 7-bit I2C Slave, fill during init

  // BYTE 7 Channel Number
        // [7:4] - reserved (NOT IMPLEMENTED, only bits 0:3 specified as a
whole byte)
  0,    // [3:0] - Channel number for the channel that the management
controller is on.
        //         Use 0h for the primary BMC. (New byte for IPMI v1.5.
Note this
        //         addition causes some of the following byte offsets to
be pushed down
        //         when compared to the IPMI v1.0 version of this record.)

  // BYTE 8 Power State Notification, Global Initialization
  0,    // [0:1] ctrl_init
  0,    //   [2] log_init_agent_errs - 1b = Log Initialization Agent
errors
  0,    //   [3] ctrl_logs_init_errs - 1b = Controller logs Initialization
  0,    //   [4] reserved
  0,    //   [5] reserved
  0,    //   [6] acpi_dev_pwr_st_notify_req - 0b = no ACPI Device Power
State notification required
  0,    //   [7] acpi_sys_pwr_st_notify_req - 0b = no ACPI System Power
State notification required
```

generating corresponding C structures for EEPROM

FRU/SDR Generator utility (GUI)



- FRU and SDR generation utility
- M4 file templates available
  - using M4 preprocessor
  - GUI interface
- User modifications needed for specific hardware
- Generates fru_data.c and sdr_data.c C structures for EEPROM

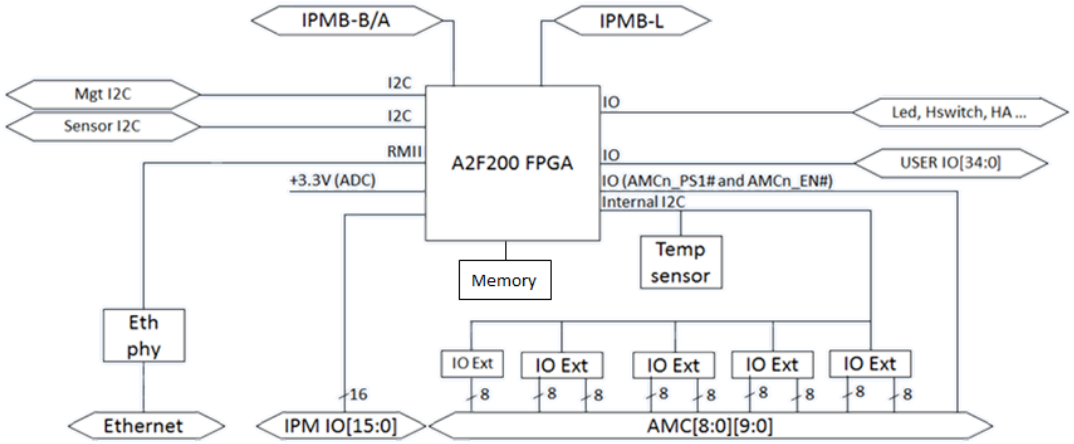
I2C sensors for eFEX

- LM82 Temperature sensor / Sensor-I2C Bus /
- LTC2499 8/16 Channel Delta Sigma ADC / Sensor-I2C Bus /
  3.3V, 2.5V, 1.8V, 1.2V, 1.05V current monitor

- PIM400KZ ATCA Board Power Input Module / MGNT–I2C Bus /
  48V management
  Temperature
- QBDW033A0B DC/DC converter / MGNT–I2C Bus /
  48V to 12V management
  Temperature
- MDT040A0X DC-DC Power Module / MGNT–I2C Bus /
  12V to 1V management
  Temperature
- M24256 EEPROM / MGNT–I2C Bus/

CERN IPMC - General overview

Second revision of the CERN IPMC mezzanine



CERN IPMC functional block diagram.

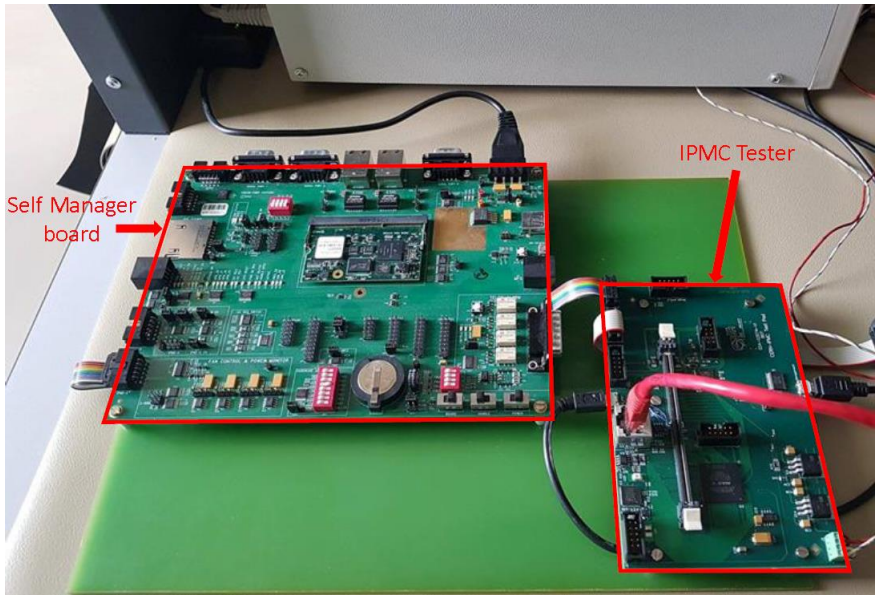
- **Adaptation of the Pigeon Point IPMC solution**
- Mezzanine card was designed at CERN
- DIMM-DDR3 VLP form factor
- Compatible with already designed AdvancedTCA board
- Follows the LAPP IPMC specification

Supported features

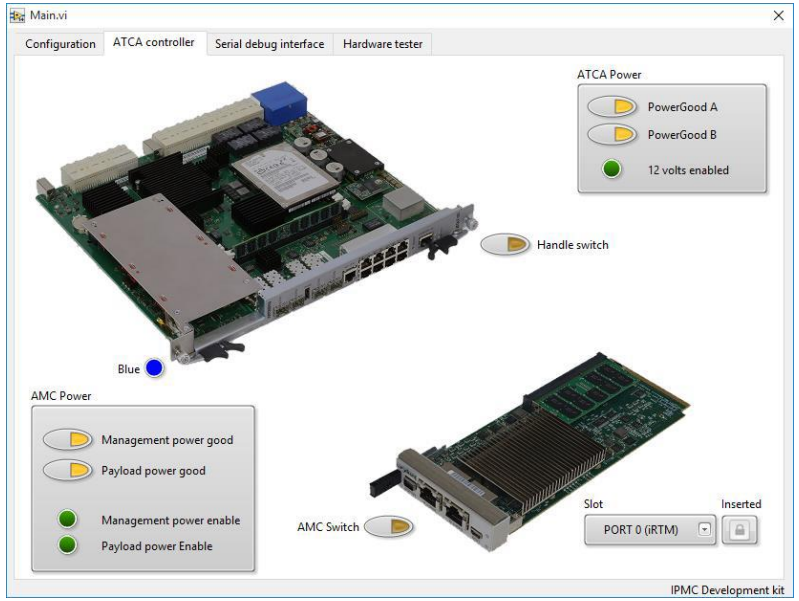
- AdvancedTCA Rev.3.0 standard
  Hot swap (FRU info, handle switch, LEDs, Hardware address, etc.)
  Sensor monitoring (SDR, measurement, events, etc.)
  Rear Transition Module (intelligent and non-intelligent RTM)
- AMC standard (up to 9 AMCs)
- Ethernet interface (RMCP/RMCP+, TPC/IP, UDP, Telnet)
- Serial interface (SoL or debug interface)
- User I/Os (35 User I/Os + 16 IPM I/Os)
- JTAG Master (Xilinx Virtual Cable daemon)

- **Additional tools to simplify the firmware configuration**

CERN IPMC - General overview



Hardware test setup



LabVIEW interface

XML configuration file:
FRU information (Device ID, Manufacturer info., Product info.)
LAN configuration (MAC address, Default IP, slot specific IP, Gateway, Netmask)
AMCs (AMC Sites, Physical port, Maximum current)
iRTM (Physical port, I2C address, Maximum current)
Sensors (Name, Thresholds, Custom fields)
Non-intelligent RTM, E-Keying

Queen Mary
University of London

Thank you