IPMC SOFTWARE

Fatih Bellachia

Sébastien Cap, Nicolas Dumont-Dayot, Jasmin Fragnaud, Nicolas Letendre, Guy Perrot, Isabelle Wingerter

Software project name

INTELLIGENT PLATFORM MANAGEMENT CONTROLLER SOFTWARE

Monday 14th April 2014 - IPMC Meeting - LAPP

Specifications

The IPMC software solution is fully compliant with the following specifications:

- IPMI v1.5 *(document revision 1.1)* and some relevant subset of IPMI v2.0 *(document revision 1.0)*.
- PICMG 3.0 R3.0 *(AdvancedTCA™ base specification)*.
- AMC.0 R2.0 *(AdvancedMC™ base specification)*.

Software environment

Features

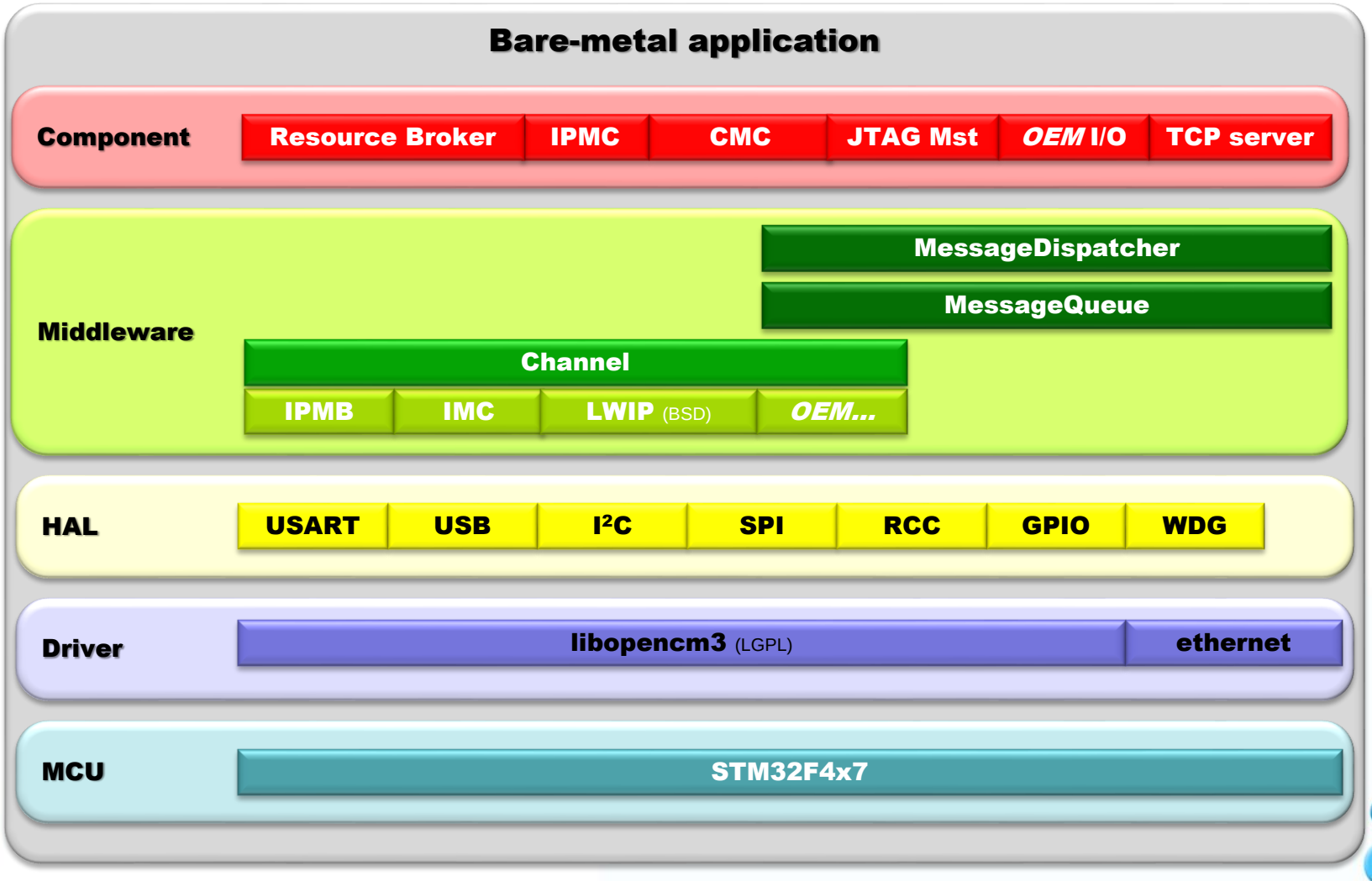
- Linux host development
- 32-bit ARM Cortex-M4 microcontroller
- Written in standard ANSI C
- GCC (4.7.0) tool chain
- Open Source Configuration Management environment: - CMT
- FRU (ATCA board) generation utility (using M4 preprocessor)
- OpenOCD (0.6.1) utility (Linux/Windows)
  - Need USB to JTAG interface Debug-Adapter-Hardware
    - Olimex ARM-USB-TINY-H
    - NGX technology
    - …

Software design

Focus

- Distributed bare-metal application
- Event (message) driven architecture
- The component (module) based design of the IPMC software source code allows the user to easily customize without modifying the existing code.

Software architecture



FRU generator tool

- FRU tool (M4 based) generating corresponding C structures

```
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, Fermilab)
IPMI_BOARD_PRODUCT(FRU_ID, Pulsar IIb)
IPMI_BOARD_PART_NUMBER(FRU_ID, Pulsar IIb)
IPMI_BOARD_MFG_DATE(FRU_ID, 0xA0, 0x72, 0x92)
IPMI_BOARD_SERIAL_NUMBER(FRU_ID, 1)
IPMI_BOARD_FRU_FILE_ID(FRU_ID, fru_data.bin)
dnl
dnl IPMI_PRODUCT_INFO_AREA
dnl
IPMI_PRODUCT_MANUFACTURER(FRU_ID, Fermilab)
IPMI_PRODUCT_PART_NUMBER(FRU_ID, Pulsar IIb)
IPMI_PRODUCT_PRODUCT(FRU_ID, Pulsar IIb)
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
    0,    // 7:4 - reserved, write as 0000b
    0,    // Internal Use Area Starting Offset (in multiples of 8 bytes)
          // 00h indicates that this area is not present
    0,    // Chassis Info Area Starting Offset (in multiples of 8 bytes)
          // 00h indicates that this area is not present
    1,    // Board Area Starting Offset (in multiples of 8 bytes)
          // 00h indicates that this area is not present
    8,    // Product Info Area Starting Offset (in multiples of 8 bytes)
          // 00h indicates that this area is not present
    14,    // MultiRecord Area Starting Offset (in multiples of 8 bytes)
          // 00h indicates that this area is not present
    0,    // PAD, write as 00h
    0xE8, // Common Header Checksum (zero )
  },

  { /*--- BOARD INFO AREA ---*/

    0x01,              //  1 Board Area Format Version
                       //    7:4 - reserved, write as 0000b
                       //    3:0 - format version number = 1h for this
                              specification.
    7,                 //  1 Board Area Length (in multiples of 8 bytes)
    25,                //  1 Language Code (See section 15)
    0xA0, 0x72, 0x92, //  3 Mfg. Date / Time
                       //    Number of minutes from 0:00 hrs 1/1/96, LSbyte
                              first (little endian)
    0xC8,              //  1 Board Manufacturer type/length byte
                       //  P Board Manufacturer bytes
    'F', 'e', 'r', 'm', 'i', 'l', 'a', 'b',

    0xCA,              //  1 Board Product Name type/length byte
                       //  Q Board Product Name bytes
    'P', 'u', '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*
```

Module

- The software "module" concept allows user to extend functionalities of IPMC without modifying existing code.
- User need to provide 3 entry point functions and information.
  - Init
  - Cleanup
  - Process
- E.g.

```
/*---------------------------------------------------------------------------*/
bool led0_init(void)
/*---------------------------------------------------------------------------*/
{
  LED_Off(LED_0);
  hTask = TimerTaskCreate(led0_blink, NULL, DELAY);

  return true;
}

/*---------------------------------------------------------------------------*/
void led0_process(void *pvArg)
/*---------------------------------------------------------------------------*/
{
  printf("\rLEDs [%c", bOn? '*': '\'');
  fflush(stdout);
}

MODULE_BEGIN_DECL(led0)
  MODULE_NAME("Blinky LED 0")
  MODULE_AUTHOR("fatih.bellachia@lapp.in2p3.fr")
  MODULE_INIT(led0_init)
  MODULE_CLEANUP(NULL)
  MODULE_PROCESS(led0_process)
MODULE_END_DECL
```

EEPROM

- The Framework support EEPROM M24256
- For an unsupported EEPROM the user must *overwriting* the following functions:
  - bool prom_init(void)
  - bool prom_reset(void)
  - int prom_erase(unsigned addr, int len)
  - bool prom_busy(void)
  - int prom_read(void *dst, unsigned src_addr, int len)
  - int prom_write(unsigned dst_addr, void *src, int len)

Sensors

- The Framework support the following sensors
  - AD7414
  - LTC4151
  - LTC2499
  - IQ65033QMA10
- Register your 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

E-Keying

- Register your backplane channel with ekeying library e.g.

```
/*-----------------------------------------------------------------------------*/
bool set_port_state_callback(linkDescriptor linkInfo, char state)
/*-----------------------------------------------------------------------------*/
{
  if (linkInfo == 0x00001101)
    do something…
  else if (linkInfo == 0x00001102)
    do something else…

  …

  return true;
}

/*-----------------------------------------------------------------------------*/
bool my_init(void)
/*-----------------------------------------------------------------------------*/
{
  …
  EKRegisterFunc(0x00001101, set_port_state_callback);
  EKRegisterFunc(0x00001102, set_port_state_callback);

  …

  return true;
}
```

Base I/F – channel 1

Base I/F – channel 2

Software status

	Unit Testing	Integration Testing	System Testing	Comment
HAL	✓	✓	✓	
IPMB	✓	✓	✓	
IMC	✓	✓	✓	
lwip	✓	✓	✓	
Channel	✓	✓	✓	
MessageQueue	✓	✓	✓	
MessageDispatcher	✓	✓	✓	
	✓	✓	✓	
FRU/SDR storage	✓	✓	✓	M24256
E-Keying (backplane)	✓	✓	.	
Sensors	✓	work in progress...	.	AD7414 LTC2499 LTC4151
Module	✓	work in progress...	.	
IPMC	✓	✓	.	
JTAG master	✓	.	.	
Firmware upgrade	✓	.	.	

Software status

	Unit Testing	Integration Testing	System Testing	Comment
CMC	✓	.	.	
Watchdog	✓	.	.	
E-Keying (Carrier)	.	.	.	

Extra features

- JTAG master (e.g. upgrade of ATCA blade firmware)
  - **SVF** player
- IPMC firmware Upgrade via TCP/IP (e.g. Base Interface)