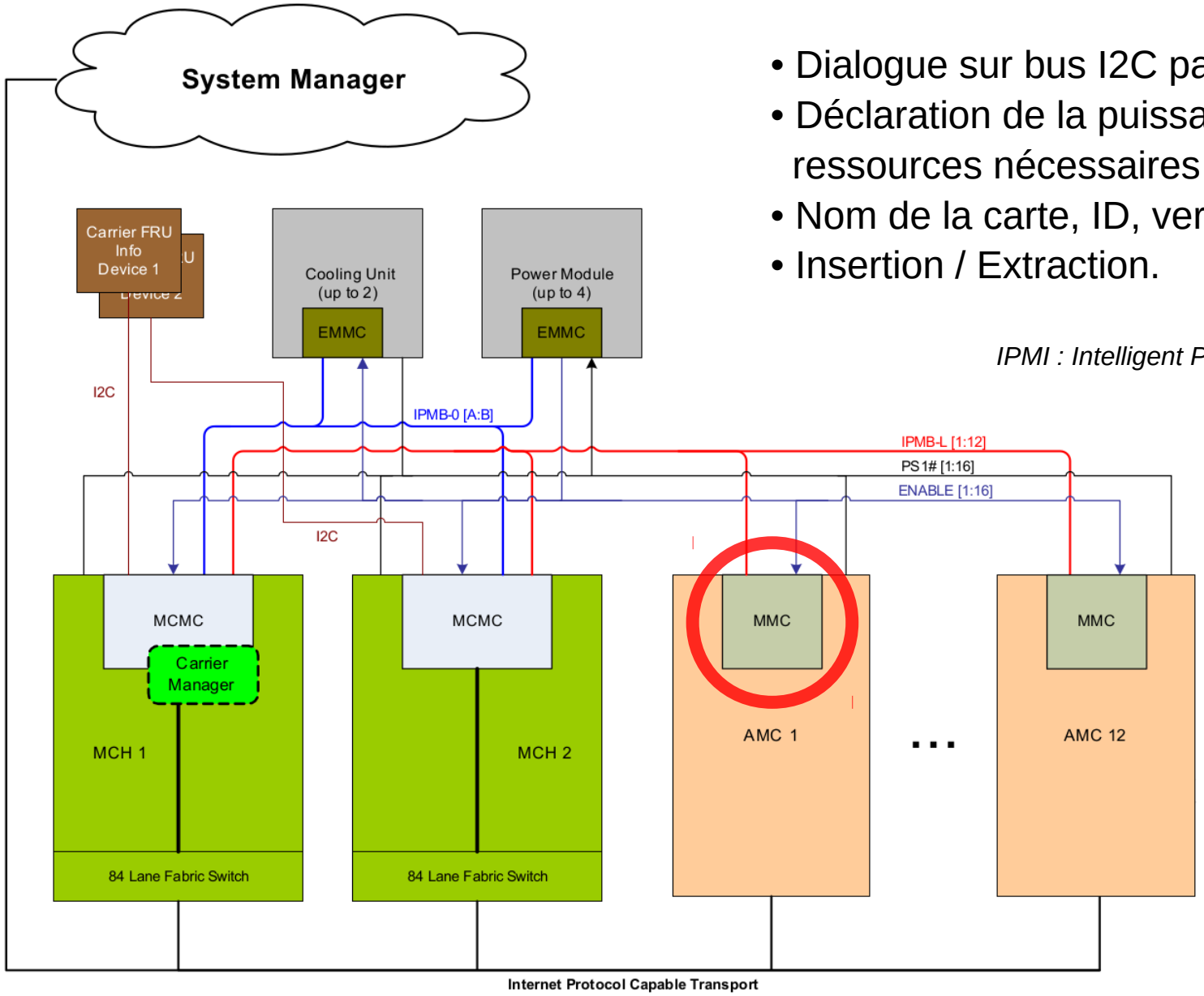
Mise en œuvre du MMC et fonctions de contrôle avancé par IPMI pour cartes AMC

Damien TOURRES
LPSC CNRS/IN2P3, Université Grenoble-Alpes

Le Module MMC



- Dialogue sur bus I2C par protocole IPMI.
- Déclaration de la puissance et des ressources nécessaires (fond de panier).
- Nom de la carte, ID, version...
- Insertion / Extraction.

IPMI : Intelligent Platform Management Interface

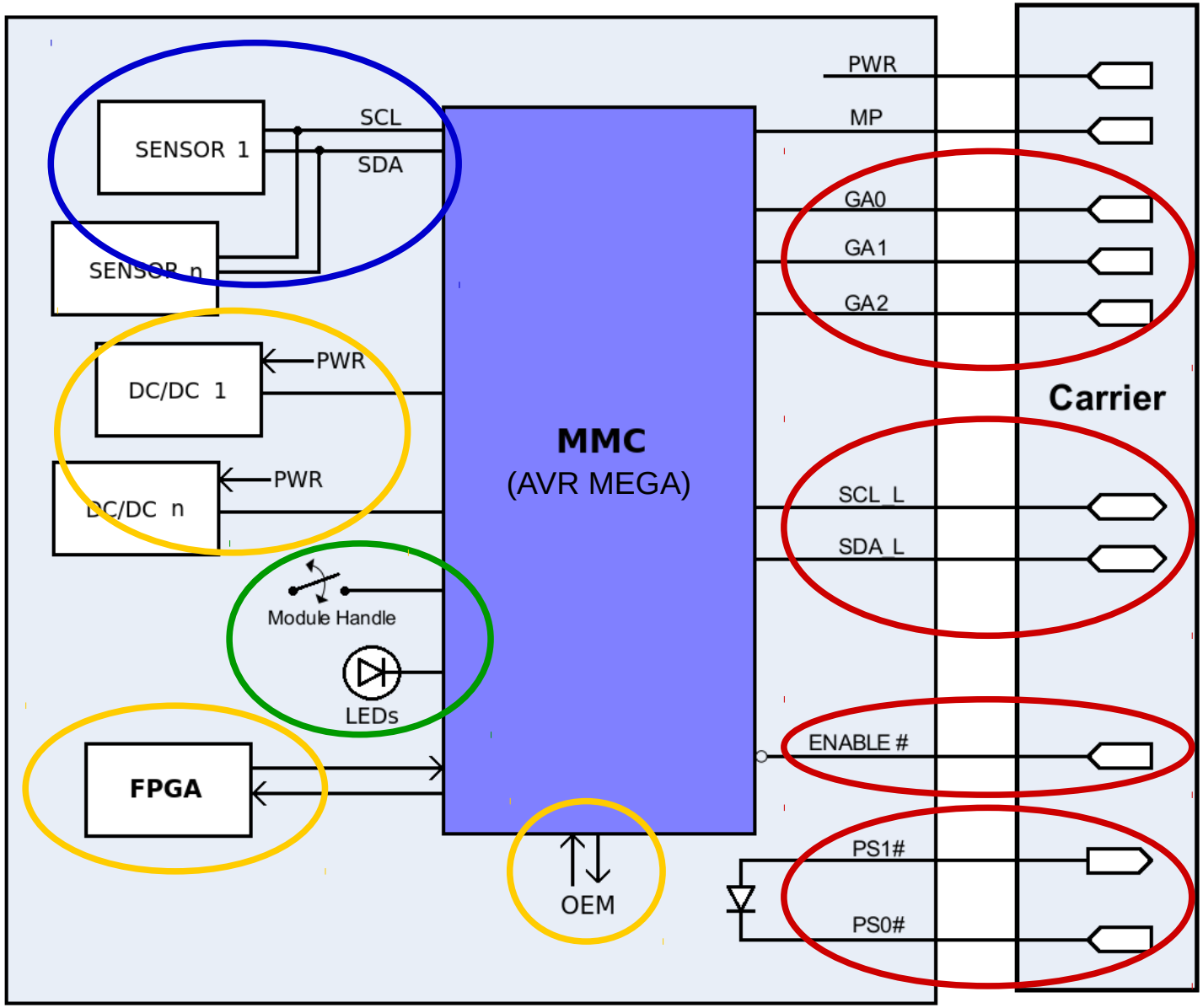
Le Module MMC

Fond de panier
- IPMI
- Adresse
- Insertion

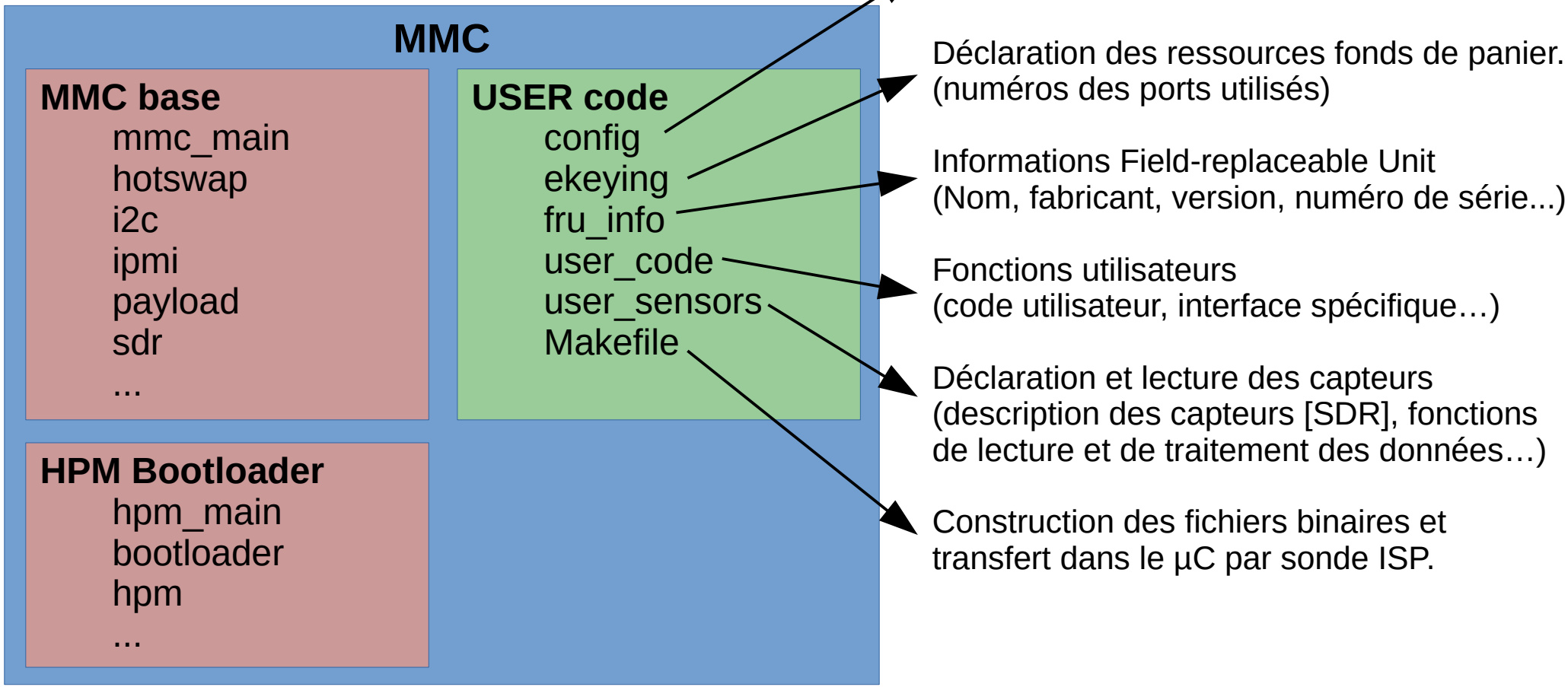
HotSwap
Blue Led

Sensors
- Alarmes
- Événements
- Refroidissement
- Surveillance

Custom
- Power Seq.
- FPGA(s)
- CI Spécifique
- RTM



Utilisation du Logiciel Embarqué MMC



Monitoring par script python et ipmi-tool

Commande : `control.py 192.168.100.127 sensor 3`

IP de la MCH

Lecture des capteurs

Slot AMC

```
ipmitool -H 192.168.100.127 -A NONE -T 0x82 -B 0 -t 0x76 -b 7 -m 0x20 sensor
```

Auth type

Bridge

Adresse I2C

Adresse locale

Slot ID = Slot AMC + 4
Adresse I2C = 2 x Slot AMC + 0x70

Monitoring de la carte MMC DevKit

```
-------------------------------------------------------------------------------------------------------------------------
| AMC HOT_SWAP     | 0x0         | discrete   | 0x0100| na        | na        | na        | na        | na        | na        |
| 12V              | 12.489      | Volts      | ok    | 9.039     | 9.522     | 10.005    | 14.007    | 14.490    | 14.973    |
| 3V3              | 3.156       | Volts      | ok    | 2.607     | 2.705     | 2.803     | 3.508     | 3.606     | 3.704     |
| 0.9V FPGA        | 0.902       | Volts      | ok    | 0.696     | 0.755     | 0.804     | 1.000     | 1.049     | 1.098     |
| Current IN       | 0.304       | Amps       | ok    | 0.000     | 0.000     | 0.000     | 0.353     | 0.402     | 0.451     |
| Power IN         | 3.450       | Watts      | ok    | 0.000     | 0.000     | 0.000     | 3.450     | 4.140     | 4.485     |
| AMC Temp 1       | 25.000      | degrees C  | ok    | -20.000   | -10.000   | 0.000     | 60.000    | 70.000    | 80.000    |
| Counter          | 108.000     | unspecified| ok    | na        | na        | na        | na        | na        | na        |
-------------------------------------------------------------------------------------------------------------------------
```

Fonctions de contrôle avancé

Lecture d'un identifiant unique (MAC, UID...)

Commande : `control.py 192.168.100.127 frulist`

IP de la MCH

Identification des périphériques

```
AMC          : Kiss_AMC          ID=6    slot=2  IP=192.168.100.133 MAC=00:22:8f:03:00:54       MMC_fw=3.0
AMC          : MMC DevKit v1     ID=7    slot=3                                                  MMC_fw=3.0
FAN TRAY     : VT VT090          ID=41
FAN TRAY     : VT VT090          ID=40
MCH          : NMCH-CM           ID=0
POWER MODULE: NAT-PM-AC600       ID=50    site=0
```

```
ipmitool -H 192.168.100.127 -A NONE -T 0x82 -B 0 -t 0x74 -b 7 -m 0x20 raw 0x2E 0x1 0x00 0xA1 0x2E 0x84 0x06
```

Auth type

Bridge

Adresse I2C du slot 2

Adresse locale

Commande RAW

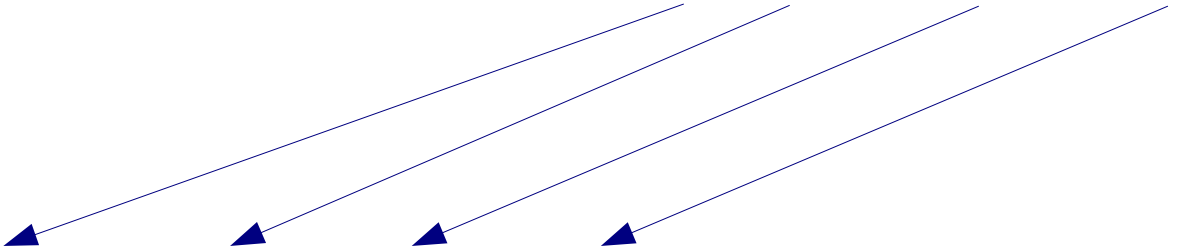
Commande raw :

0x2E : commande OEM (code utilisateur)
0x01 : lecture d'une mémoire
0x00 0xA1 0x2E : IANA IN2P3
0x84 : adresse à lire
0x06 : nombre d'octet à lire

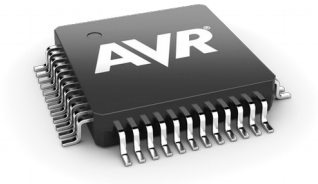
Fonctions de contrôle avancé

Commande IPMI OEM

```
ipmitool -H 192.168.100.127 -A NONE -T 0x82 -B 0 -t 0x74 -b 7 -m 0x20 raw 0x2E 0x1 0x00 0xA1 0x2E 0x84 0x06
```



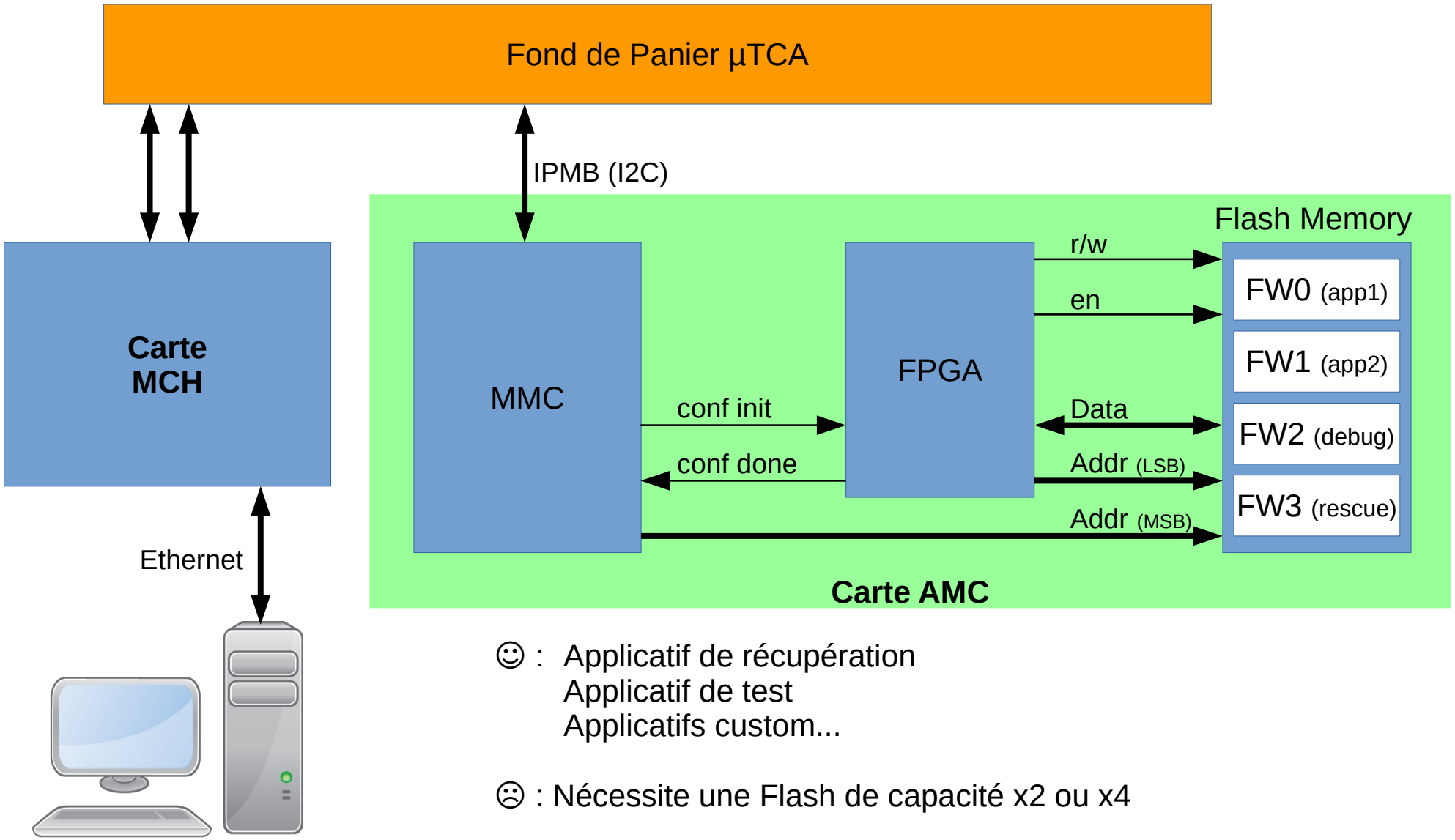
USER_CODE.C :




```
ipmi_oem_user(uint8 cmd, uint8 *iana, uint8 *data, uint8 size, uint8 *buf, uint8 *error)
{
        if( !iana_is_ok(iana, 0x00A12E))
        {
                *error = IPMI_CC_PARAM_OUT_OF_RANGE;
                return(0);
        }
        if(cmd == 0x01)
        {
                buf[0]=0x00 ; buf[1]=0xA1 ; buf[2]=0x2E ;        // IANA
                write_I2C(MEM_ADR, data[0]);                     // adresse des données
                read_I2C(MEM_ADR, data[1], &buf[3]);             // lecture de 6 octets
                *error = IPMI_CC_OK;
                return 9;                                        // IANA : 3octets
        }                                                        // MAC : 6octets
        else
        {
                *error = IPMI_CC_INV_CMD;
                return(0);
        }
}
```

Fonctions de contrôle avancé

Sélection de l'applicatif du FPGA par IPMI

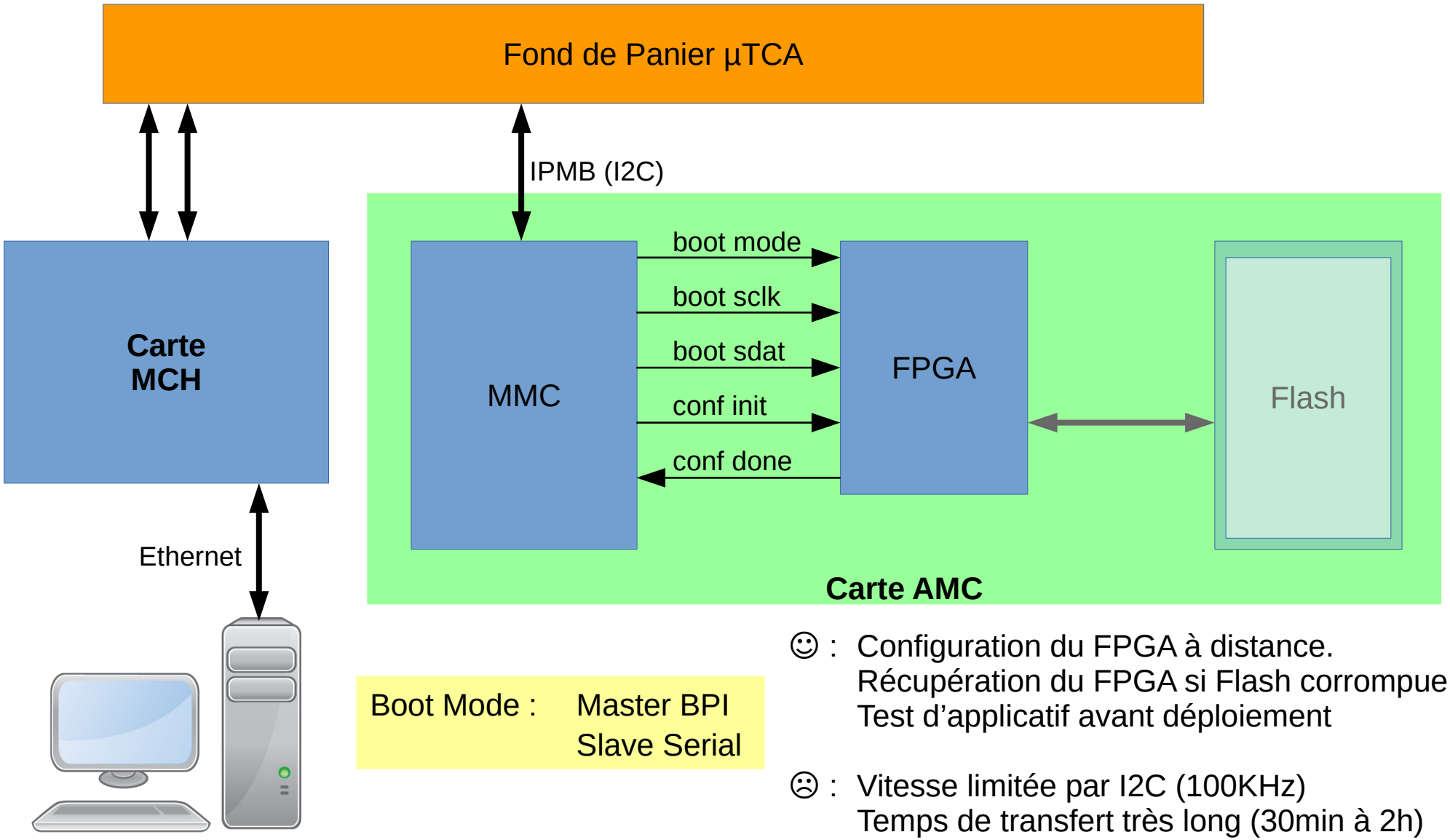


☺ : Applicatif de récupération
Applicatif de test
Applicatifs custom...

☹ : Nécessite une Flash de capacité x2 ou x4

Fonctions de contrôle avancé

Configuration du FPGA par IPMI



Boot Mode : Master BPI
Slave Serial

☺ : Configuration du FPGA à distance.
Récupération du FPGA si Flash corrompue.
Test d'applicatif avant déploiement

☹ : Vitesse limitée par I2C (100KHz)
Temps de transfert très long (30min à 2h)

Fonctions de contrôle avancé

Configuration du FPGA par IPMI : Compresser l'image

☹ : Vitesse limitée par I2C (100KHz)
Temps de transfert très long

☺ : Compression du fichier de configuration.
Sécurisation du transfert.

☹ : Ressources µC (CPU et RAM) limitées

☺ : Algorithme de décompression léger.
Compression basée sur **RLE**.

blanc	blanc	blanc	blanc	blanc	blanc	blanc	blanc	blanc	noir	noir	noir	noir	blanc	blanc	blanc

16 mots

RLE

9	blanc	4	noir	3	blanc

6 mots

blanc	blanc	blanc	blanc	blanc	blanc	blanc	blanc	blanc	noir	bleu	gris	vert	blanc	blanc	blanc

16 mots

☹ ☹ ☹ ☹

RLE

9	blanc	1	noir	1	bleu	1	gris	1	vert	3	blanc

12 mots

Fonctions de contrôle avancé

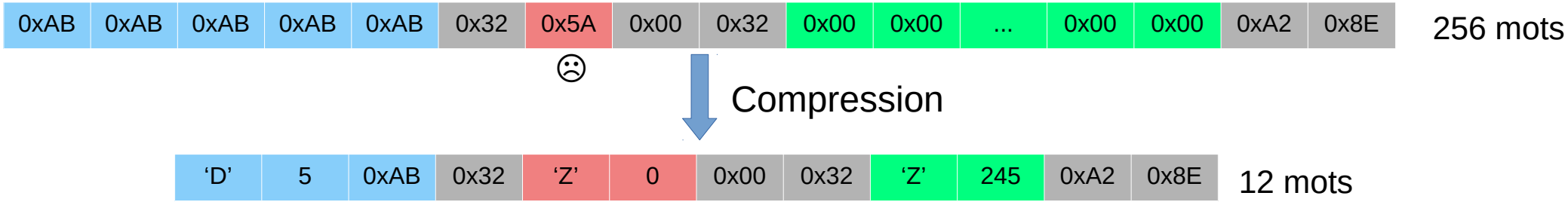
Configuration du FPGA par IPMI : optimiser la compression

☹ : Tous les codes binaires sont utilisés. → Définir des codes binaires réservés pour les commandes.

☹ : Les mots réservés doivent être codés !

☹ : Les codes des commandes sont-il optimales ? → ☺ : Définir les codes commandes dynamiques pour limiter le codage des mots réservés.

Commande ‘Z’ (0x5A) : compression des 0x00
Commande ‘D’ (0x44) : compression des 0xXX



Fonctions de contrôle avancé

Configuration du FPGA par IPMI : efficacité de la compression

Kintex KU40 :

- image non compressée : 16Mo
- image compressée :
  - RLE : 7,25Mo (x2,2)
  - *ZIP : 4,8Mo (x3,3)*

Kintex 7K70T :

- image non compressée : 3Mo
- image compressée :
  - RLE : 1,48Mo (x2,02)
  - *ZIP : 0,90Mo (x3,3)*

Temps de transfert *(débit utile ~22kbit/s)* :

- image KU40 brute : 97 minutes
- image KU40 (RLE) : 44 minutes
- image 7K70T brute : 18 minutes
- image 7K70T (RLE) : 9 minutes

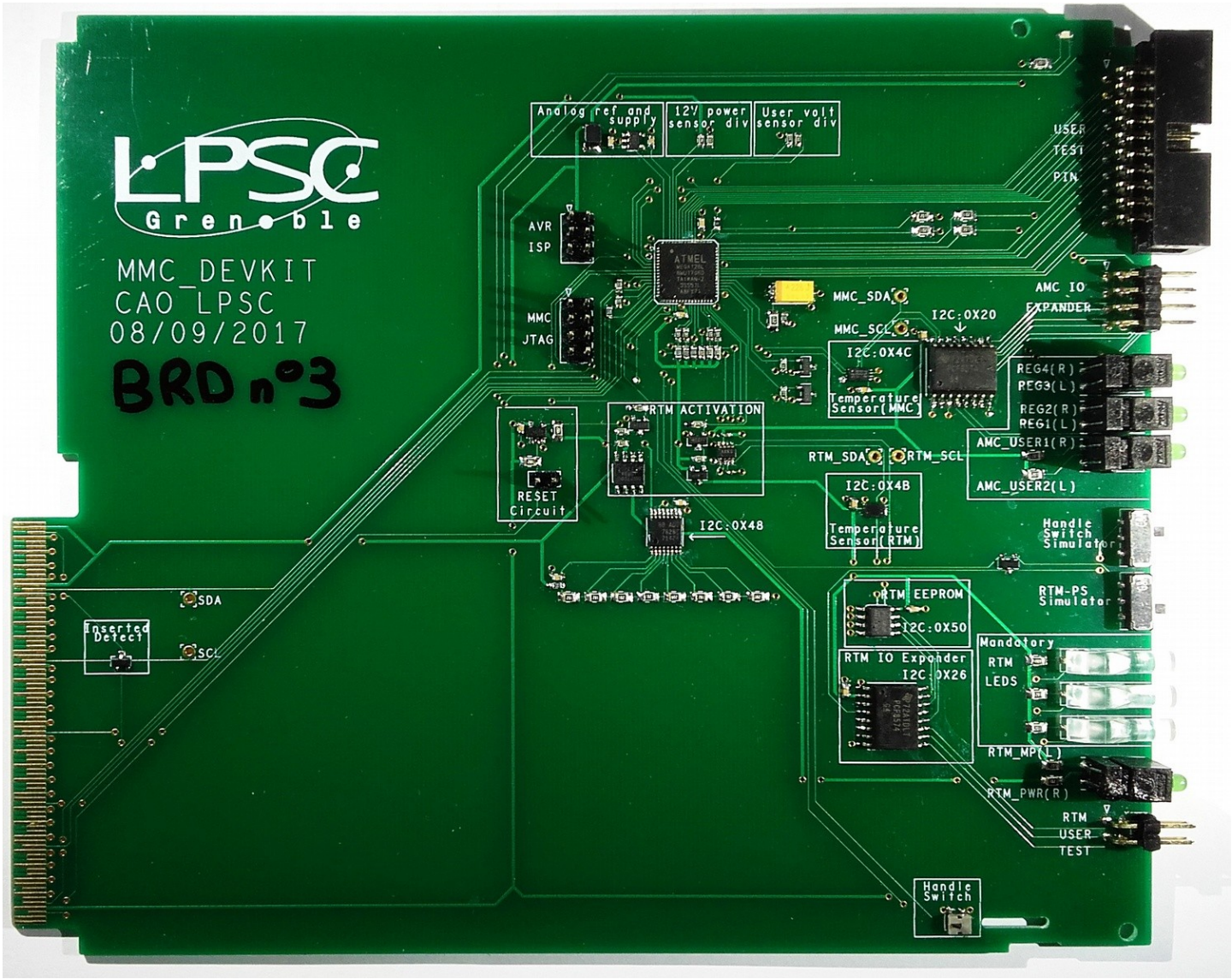
- Réduction du temps de transfert d'un facteur 2.
- Décompression RLE légère et rapide.
- Fonctionnalité ne demandant aucune ressource HW supplémentaire.
- Dépannage à distance.

Utilisation : `FPGADownloader 192.168.100.127 3 fpga.bit`

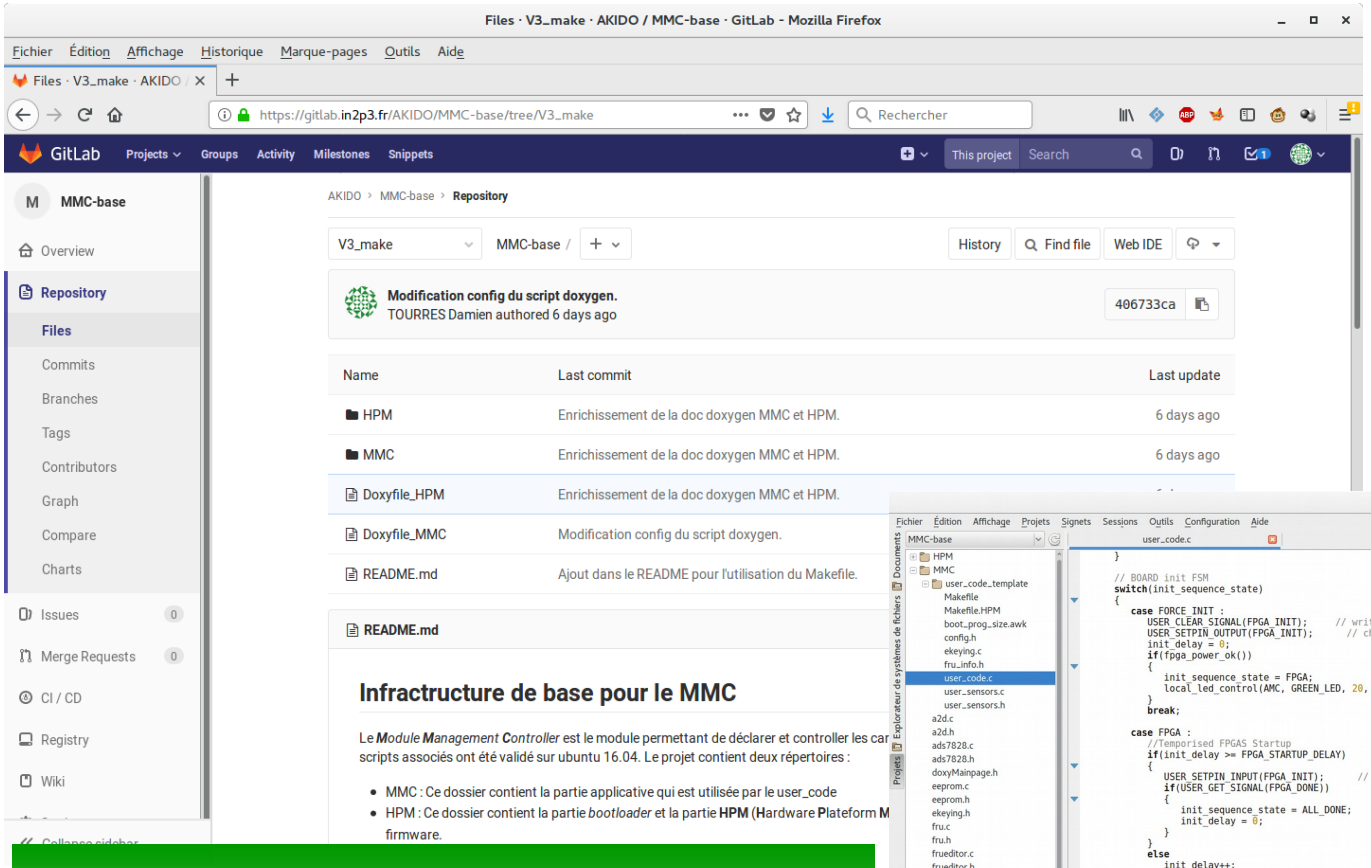
IP de la MCH — Slot AMC — Fichier de configuration

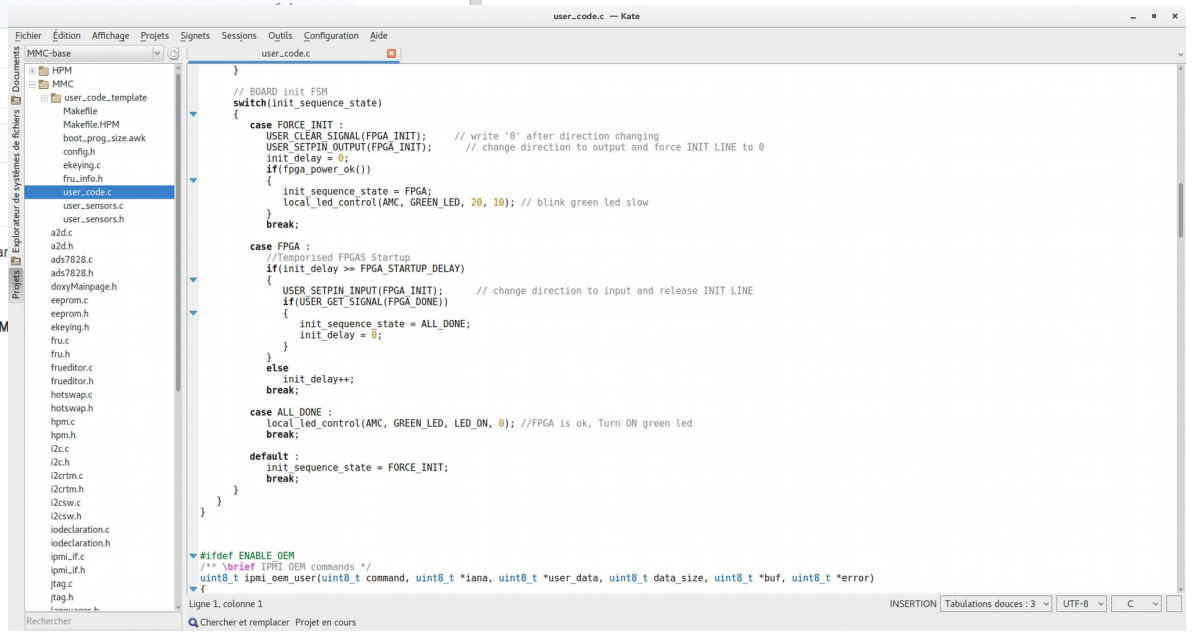
Carte MMC DevKit et Design de référence

- Faible coût : < 80€ HT (pcb DF + composants)
- 2x ADC ADS7828 I2C (monitoring tensions)
- 2x Capteurs LM75 I2C (monitoring température)
- 2x I/O EXP PCF8574 (I/O supplémentaires)
- Simulateur RTM
- points et connecteurs de test.
- Schéma de référence pour les nouveaux design



Support et Documentation





- Code documenté DOXYGEN.
- Projet GITLAB IN2P3 (licence GPL).
  https://gitlab.in2p3.fr/AKIDO/MMC-base
  https://gitlab.in2p3.fr/DAQGEN/xTCA-tools
- Support pour l'IN2P3 (réseau DAQ).