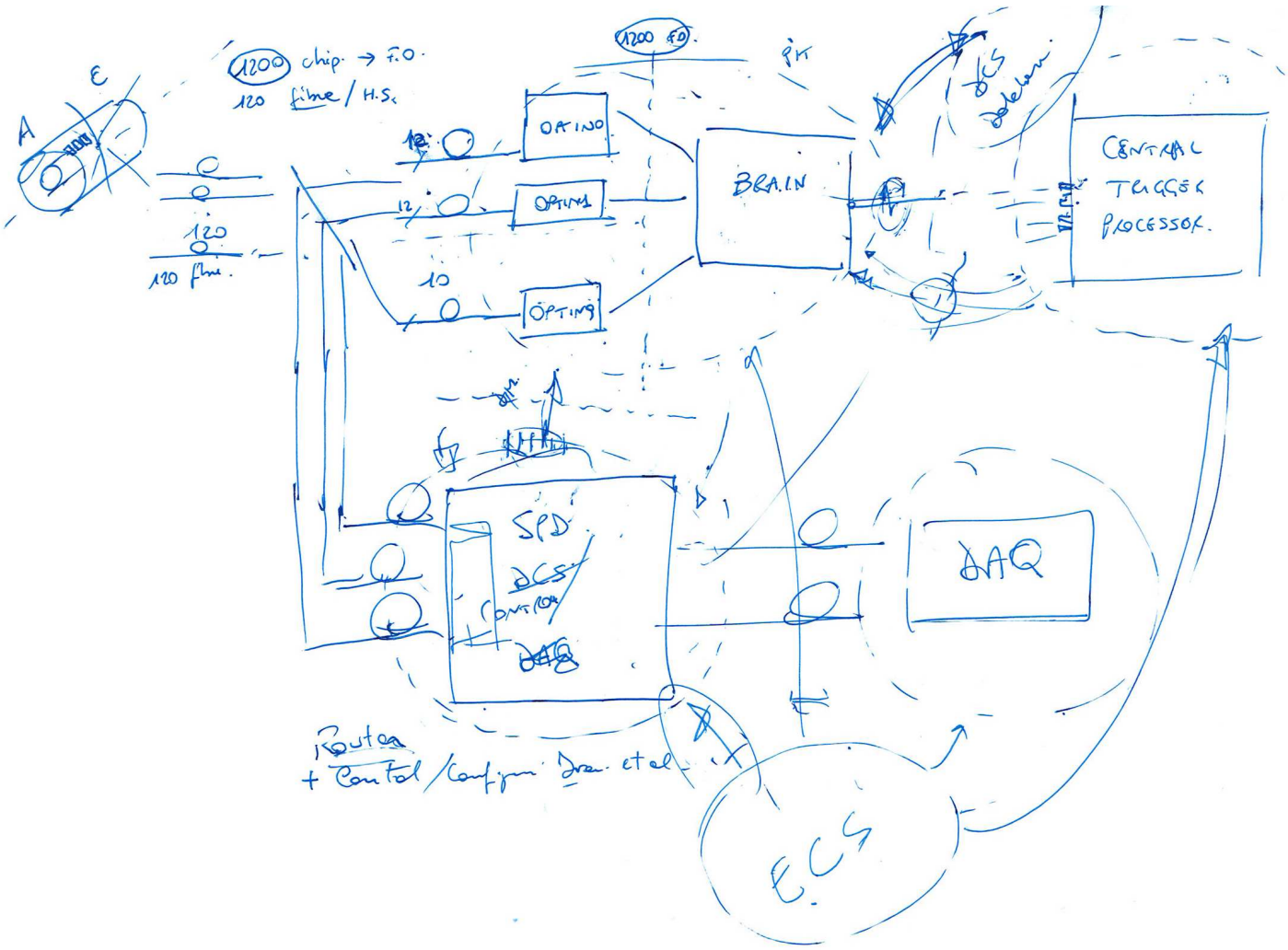
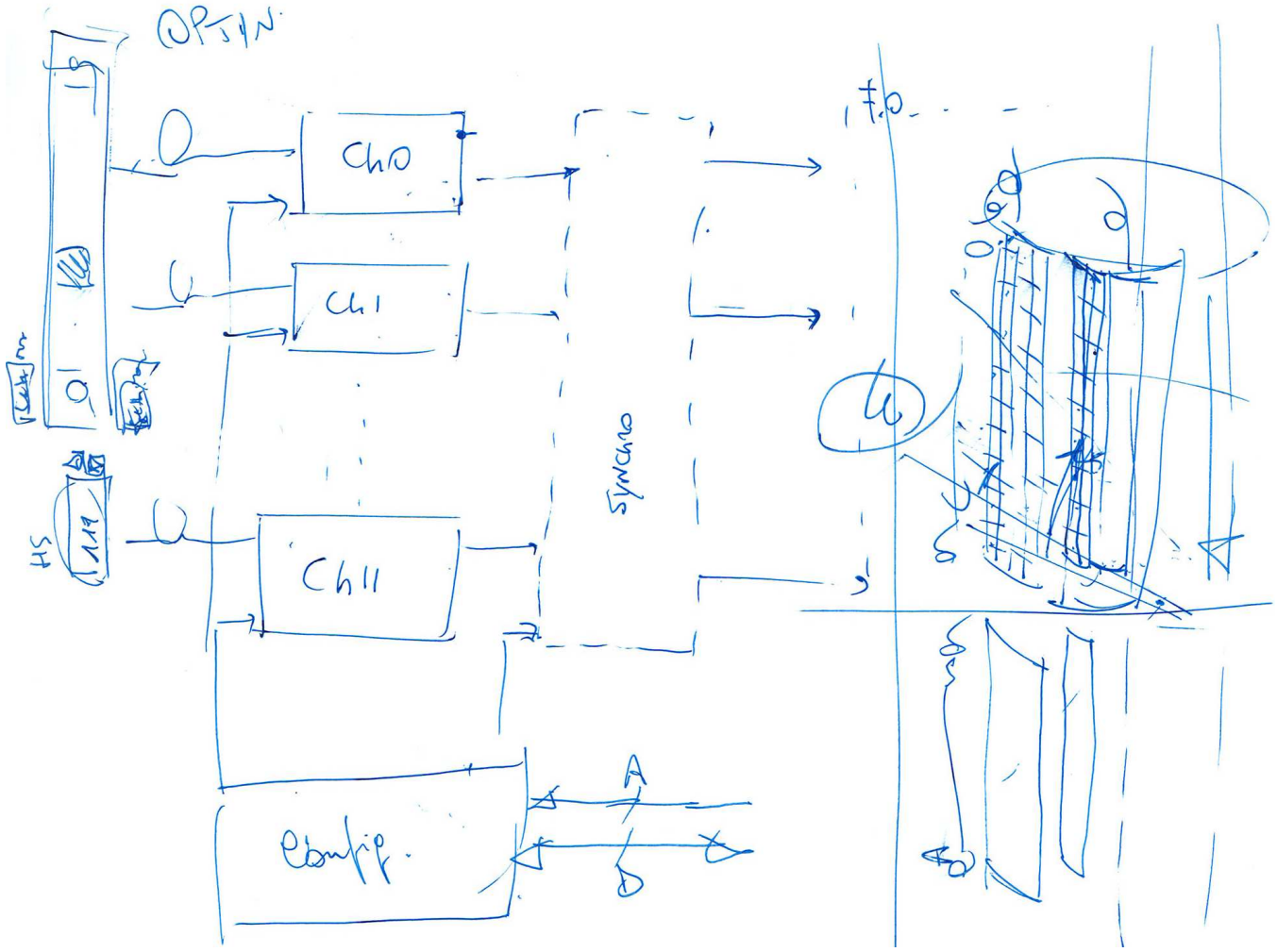
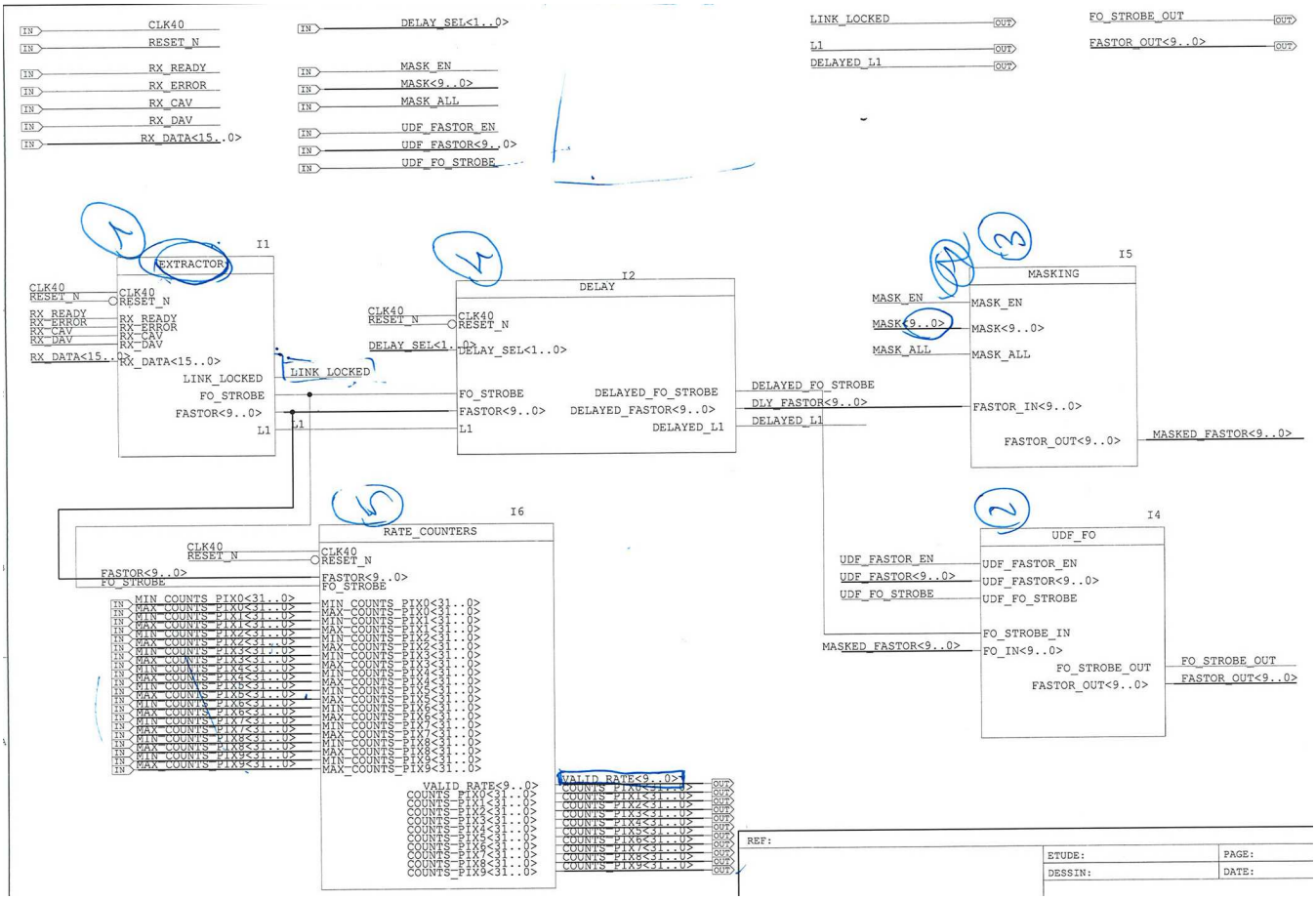
Reunion ALICE SPD 2007-09-24
Point sur les nouvelles fonctionnalités a ajouter au driver




OPTION
Ch0
Ch1
Ch11
Sincro
Ampli.
A
D



1200 chips -> FAST OR sur 120 chips. 120 fibres : groupées par 12. 10 cartes OPTIN au total : chaque carte est branchées a 12 fibres. 120 fibres arrivent du détecteur.
10 chips pour chaque fibre.

Fonction OPTIN : prélever des fibres 1 bit / chip : 1200 chips (Fast OR). Sur chaque fibre, 10 bits pour 10 chips (qui vient de 10 chips).

Sortie de tous les OPTIN : BRAIN board pour générer 1 output. Connecte au Central Trigger Processor.

Les fibres arrivent aussi en parallèle sur un autre système : SPD Control / DAQ (router de Marian et system de control et config de Ivan).

Notre système est passif (pas action sur le détecteur). Si on détecte un problème, il faudra communiquer sur SPD Control / DAQ, et sur l'ECS (qui gère tout le système) qui fonction en State machine.

Notre système peut aussi parler avec le DCS database.

CTP : se référer avec Marian Krivda éventuellement Villalobos Orlando.

<u>Détail du Hardware :</u>

Définir une convention de noms dans les fonctions.

- Définir deux fonctions : setOutputMode et getOutputMode.
- Définir : getListChannelsActives : ECS demande une config (avec la liste des channels actives), je récupère la liste avec cette fonction, puis je compare. Selon le résultat de la comparaison., je me met dans un état donné.

- Quand on demande une config, il faut lire vis-à-vis du SPD si la config du détecteur est en cours, terminée… et une fois la config terminée par le SPD, on va lire le HW l'état des voies. : interaction avec IVAN : PitGetSPDConfigurationStatus (ex : UNCONFIGURED, CONFIGURING, CONFIGURED).

- Carte OPTIN :
    - on arrive avec 12 fibres en entrée du système. Cote hw, on fait référence a la carte, channel et adresse. Correspondance module / channels physiques doit etre avec un tableau dans la DB. Voir avec Ivan et Gianluca la correspondance entre module et channels. On utilisera cette correspondance sous forme d'une table dans la DB pour d'éventuelles modifications de config.

    - 12 channels en entrée d'une carte OPTIN. a l'intérieur, système de synchro des entrées (des 12 channels). Avoir des fonctions qui en entrée module détecteur, en sortie adresse de base (grâce au fichier de correspondance dans la DB).

- Block Extractor :
    - A l'intérieur d'une channel : un bloc 'extractor' reçoit la fibre : sort une info si la communication est bien en place (LINK_LOCKED). : créer une fonction PIT_GET_STATUS_CHANNEL(int channelNum) ou GEThalfstateChannel. GetStatusModule (reçoit un module du détecteur) _> DB -> CARTE OPTIN ET CHANNEL -> fonctionbqui retourne l'adresse de base. Ensuite on utilise un offset qui indique dans la carte ou chercher l'information sur le status.

- Sur les états qu'on a besoin de vérifier :
    - état du FPGA brain(unconfigued, configured, configuring)
    - états des voies (voir juste avant)

- Block Délai :
    - sur chaque channel, possibilité de lire et configurer un délai. (getDelayChannel, setDelayChanel). De notre cote ce sera manuel. Peut etre automatise (lecture delay des channels et reconfiguration des délais). On peut envisager une classe channel avec les infos et ses méthodes (setDelay…)…

- Block Masking :

    - Set/getMask. 10 bits par channel. Pour chaque channel on va masquer des entrées. Position du bit en correspondance entre un chip dans la channel.

- Block Rate Counter :
    - Rate counter : 10 bits : si un bit actif, état error et possibilité de masquer le bit correspondant. Me dit si le chip est dans un bon rate ou pas. Et possibilité de lire la valeur du rate. pitVerifyRateChannel et pitGetCounterValue pour un certain chip. Fonction : configureRateCounter qui va fixer une limite (imposée par les physiciens) dans le HW. Une ou deux limites (rate min et rate max). Une fois configure, le hw va vérifier automatiquement si le rate est > a la limite et le présenter sur le register VALID_RATE (si bit = 1 -> chip a désactiver).
    - Si un problème (un bit dans le rate counter), on le signale (état WRONG_RATE).

- Block UDF_FO (User defined Fast OR) :
    - possibilité d'activer une sortie d'un chip de la carte optin pour faire des tests.

- Priorité dans les blocs (vis-à-vis de ce qui y a réaliser) :
    1. ☐ Extractor : Evaluation de l'état des channels.
    2. ☐ UDF
    3. ☐ Masking.
    4. ☐ Delay
    5. ☐ Counters.

Conclusion a ce point :

Liste des différents blocks de l'expérience :

OPTIN :
- Nombre de cartes OPTIN = 10
- Une carte OPTIN a 12 fibres en entrée (channels)
- Chaque fibre (channel) et reliée a 10 chips
- TOTAL : 10 * 12 * 10 = 1200 chips.
- Unite de base = chip. On a 1200 pixel chips.

BRAIN :
- Reçoit en entrée les sorties issues des OPTIN.
- Génère en sortie un signal qui sera émis au CTP.

SPD Control /DAQ
- Reçoit aussi toutes les 1200 sorties du détecteur.
- Notre interface sera de récupérer l'état de configuration des voies.

CTP (Central Trigger Processor)
- Recevra la sortie du BRAIN.
- Il pourra envoyer une commande a notre driver pour établir une configuration de sortie (SIGNATURE…)

ECS
- Gère l'ensemble. Va envoyer au driver un numéro de configuration, et celui-ci ira chercher dans la DB tous les paramètres relatifs a cette configuration.

Liste des TODO (non classes en priorité)

1. Communication entre le BRAIN et le CTP

- ☐ Définir deux fonctions : **void pitSetOutputMode(int mode)** et **int pitGetOutputMode(void)** qui permettent de définir le mode de sortie et lire le mode de sortie courant. La liste des modes possibles sont : TOGGLE (0101010…), RANDOM et SIGNATURE (bits emis en serie), OFF, NORMAL.
- Le CTP enverra une commande pour initialiser un mode de sortie.
- Communication DIM entre le CTP et notre DRIVER.

2. Communication entre le BRAIN et l'ECS :

- L'utilisateur va demander, via l'ECS une version de configuration, cette version sera relative a un firmware du FPGA et un ensemble de paramètres telles que la liste des channels actives… Le DRIVER récupère la liste des channels actives, puis lit sur le SPD Control l'etat de configuration des channels (CONFIGURING, CONFIGURED, NOT CONFIGURED) et si les channels sont CONFIGURED, le DRIVER va lire l'etat des channels sur les OPTIN et comparer a ce qui est requis, la sortie sera un etat de notre système.
- ☐ Définir une fonction **int* pitGetListChannelActive()** qui va indiquer la liste des channels actives en entrée des cartes OPTIN.
- ☐ Définir une fonction **int pitGetSpdConfigDetector(void)** qui retournera l'état de configuration des voies (configurees, configuration en cours, non configurée). Demander a Ivan de quelle maniere sont publiees l'etat de configuration.

3. Carte OPTIN :

- ☐ Définir une fonction **int pitChannelFromModule(int channel)** qui va faire la correspondance entre un numéro de module en entrée et un numéro de channel physique dans une carte OPTIN. Cette correspondance se realisera a partir d'une table stockee dans la DB en fonctions d'une regle de numerotation definir dans d'autres parties du projet (SPD Control par exemple).
- ☐ Definir une **fonction bool pitGetStatusChannel(int channelNumber)** qui va lire sur le hw l'etat d'une channel ou une fonction **pitGetStatusModule(int module)** qui fera la meme chose mais a partir d'un module (donc une conversion 'pitGetStatusChannel' est a faire).
- ☐ Definir une fonction **int pitGetDelayChannel(int channel)** et **void pitSetDelayChannel(int chanel, int delay)** qui permettent de lire et ecrire un delay a appliquer sur une channel.
- ☐ Definir une fonction **int pitGetMaskChannels()** et **void pitSetMaskChannel(int channel, int masque)** pour lire et appliquer un masque sur les channels. Un masque

contient 10 bits pour 10 channels. Une table dans la DB fera la correspondance entre les bits et les channels dans l'OPTIN.

- ☐ Définir une fonction **pitConfigureRateCounter(int limit)** qui permettra d'envoyer au hw une limite pour que le hw définisse automatiquement que le 'rate counter' est trop élevée cad qu'une voie est active anormalement trop longtemps.
- ☐ Définir une fonction **pitVerifyRateChannel** qui va lire les status issus du hardware. Si le hw a détecte que des voies on depassees le seuil 'rate limit' des bits seront actives que cette fonction va lire.
- ☐ Définir une fonction **pitForceChannel** qui va forcer une channel a l'etat actif. Le block hw concerne est UDF_FO.

Il est intéressant d'implanter des classes relatif a chaque block de l'électronique. L'accès a l'ensemble des registres se verra facilite (seule l'adresse de base va changer).

Definition des classes:

CHANGEMENT DE NOMENCLATURE :
PIT-> 10 OPTIN -> 12 LINK (au lieu de channel) -> 10 FO Channel (au lieu de chip) : unité de base.

Class PIT

Fonctions membres:
- pitOptinChannelFromModule(entrée :numéro du module, sorties : nr optin, nr channel dans optin) : renvoie un num. optin et de channel dans l'optin.

ou, on peut diviser la function en 2:

- pitChannelNrFromModule(in module, out : channel nr)
- pitOptinNrFromModule(in module, out: optin nr)

- Vérifier que l'alimentation est on : vérifier la commi SIU.

A l'initialisation de la classe, lire la table de correspondance dans la DB une fois pour toute.

Class PROCESSING_FPGA

Fonctions membres:
- setOutput(enum outputType) : changement du mode de la sortie (SIGNATURE…)
- enum outputType getOutput() : lit le mode de la sortie
- checkFirmwareVersionNumber() : lit le numero de version.
- isProgrammed() : verifie si le FPGA est programme.
- returnToDefault().

Class OPTIN

10 instances de cette classe seront crees lors de l'initialisation de la classe PIT. Lors de l'initialisation, on appliquera une adresse de base qui sera specifique a chaque instance. Définir les adresses de bases dans des constantes.

Fonctions membres :

- OptinRefreshLinkStatus() lecture des reg de la carte optin qui retourne les 12 bits. Permet la MAJ des data members linkActive (acces au hard).
- setMinRateInnerLayer(in : chip number, valeur))
- setMaxRateInnerLayer
- getMin/MaxInnerLayer(in : chip number)
- setMin/MaxCountOuterLayer()
- getMin/MaxCountOuterLayer()
- setCountingPeriod (periode de mesure pour le rate : utile pour l'electronique)
- getCountingPeriod
- checkConnected() : verification si l'optin est branchee : on peut faire ca en lisant la version. Effectuer cela a la creation de l'instance.

Paramètres :

Version : à l'initialisation, lire la version de la carte.

Classe LINK (anciennement channel)

Correspond a une fibre optique qui contient 10 FO channels (anciennement chip)

Fonctions membres :

- isActive() : lecture sur le hw de l'etat actif/inactif du link.
- getDelay()
- setDelay(enum delai)
- refreshRateValidity()

Paramètres :

- bool linkActive.
- enum delai delay (delai = 0,1,2,3 TBC)

Class FOChannel (anciennement chip)

Une FO Channel est l'unite de base : 1 voie.

Fonctions membres:

- bool isMasked()
- void setMask()
- void clearMask()
- bool isForced()
- void forceHigh() (force active)
- void releaseForce()
- getRate() indique la valeur de la limite.
- bool isValid() (rate)

Exemple :
Pour récupérer l'état de la channel 8 de l'optin 6 :Obj_Card[5].optinchannel[7].isActive

listDCSConfigureChannel[] : array qui contient les resultats.
getListDCSConfigureChannels() : Communication avec le SPD Control. Voir avec Ivan comment sont publiees ces resultats par le SPD Control.

Il est intéressant de publier certaines informations de manière périodique au client PVSS :

- état des voies
- TBD

Au démarrage, constructeur de la classe principale :

- On vérifie la communication avec la SIU
- On vérifie la version du FPGA