Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Filière Systèmes industriels

Orientation Infotronique

Projet de diplôme 2014

Steve Devènes

Attitude and determination control system of a satellite with an FPGA

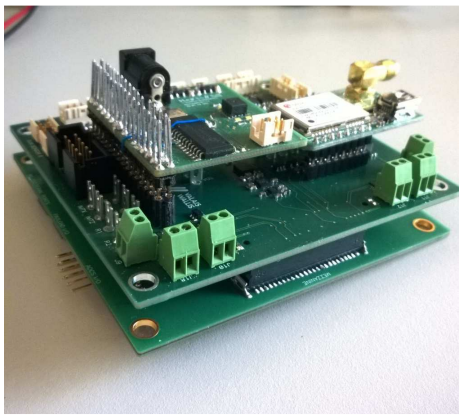
Professeur Christophe Bianchi

Sion, le 08 juillet 2014

Valais excellence

Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Système de pilotage d'un satellite avec FPGA



Diplômant/e Steve Devènes

Objectif du projet

Le but de ce projet est de développer un système d'acquisition et de contrôle de l'orientation d'un satellite de type cubSat, au moyen d'un circuit basé autour d'une FPGA.

Méthodes | Expériences | Résultats

Le système comprend un circuit électronique interfaçant cinq types de capteurs (capteur solaire, gyroscope, magnétomètre, GPS et caméra) et deux types d'actuateurs (magnéto-coupleur et roue de réaction).

Le but ultime est d'obtenir un circuit complètement autonome, qui recueille les informations des capteurs, les traite à l'aide d'algorithmes et pilote les actuateurs en conséquence pour stabiliser le satellite.

Pour l'heure ces algorithmes n'ont pas été implémentés, seule une application de démonstration développée sur QT Creator permet d'afficher les valeurs mesurées des capteurs et de piloter manuellement les différents actuateurs.

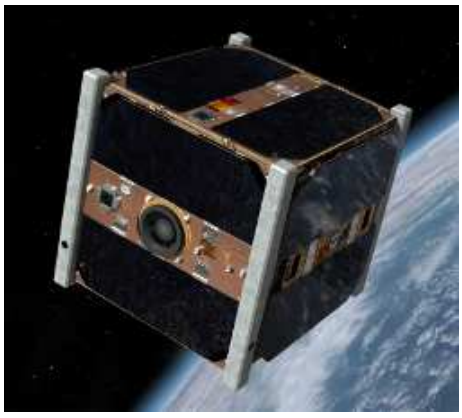
Travail de diplôme
| édition 2014 |

Filière
Systèmes industriels

Domaine d'application
Infotronique

Professeur responsable
Bianchi Christophe
Christophe.Bianchi@hevs.ch

Partenaire
Ecole polytechnique fédérale de Lausanne



Satellite SwissCube.
Source : http://swisscube.epfl.ch

☒ FSI ☐ FTV	Année académique / *Studienjahr* **2013/14**	No TD / *Nr. DA* **it/2014/61**
Mandant / *Auftraggeber* ☐ HES—SO Valais ☐ Industrie ☒ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **Steve Devènes** Professeur / *Dozent* **Christophe Bianchi**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja [1] ☒ non / nein	Expert / *Experte (données complètes)* **Stefano Rossi** EPFL | ENT CTS-GE | ELD 016 | Station 11 | 1015 Lausanne	

Titre / *Titel*

Attitude and determination control system of a Satellite with an FPGA

Description / *Beschreibung*

The goal of the project is to develop an ADCS (Attitude and Determination Control System) expertise in the framework of space project like CleanSpaceOne and CubeSat activities.

In order to fly an ADCS/CDMS (Control and Data Management System) FPGA board on a space mission, as technology development, we want to realize intermediate step consisting on :

- the implementation of FPGA interfaces to a realistic ADCS setup (6 Sun sunsors, 2 magnetometers; 3 magnetotorquers, 4 reaction wheels and possibly a star tracker)
- the implementation on a FPGA platform of a data processing via simple algorithms
- a performance comparison with solutions based on microcontroller.

Objectifs / *Ziele*

The objective of the bachelor project is :

- to mount and test the IF board developed during the semester work
- to modify the FPGA architecture of the motherboard in order to communicate with all interfaces
- to implement a simple regulation algorithm.

Signature ou visa / *Unterschrift oder Visum* Responsable de l'orientation *Leiter der Vertiefungsrichtung:*..... [1] Etudiant / Student :	Délais / *Termine* Attribution du thème / *Ausgabe des Auftrags:* 12.05.2014 Remise du rapport / *Abgabe des Schlussberichts:* 11.07.2014, 12:00 Expositions / *Ausstellungen der Diplomarbeiten:* 27 – 29.08.2014 Défense orale / *Mündliche Verfechtung:* Semaine | Woche 36

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.*
Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.

Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....

TABLE DES MATIERES

1 CONTEXTE

Le département Système industriels de la HES collabore depuis de nombreuses années avec l'EPFL sur des projets dans le domaine de l'aérospatiale, notamment sur les Cubesats, ces petits satellites en forme de cube de $10cm^3$. Le premier de ces satellites entièrement conçu en Suisse fût nommé SwissCube, et depuis son lancement, de nombreuses autres missions utilisant ces satellites ont vu le jour, comme par exemple CubETH ou CleanSpaceOne.

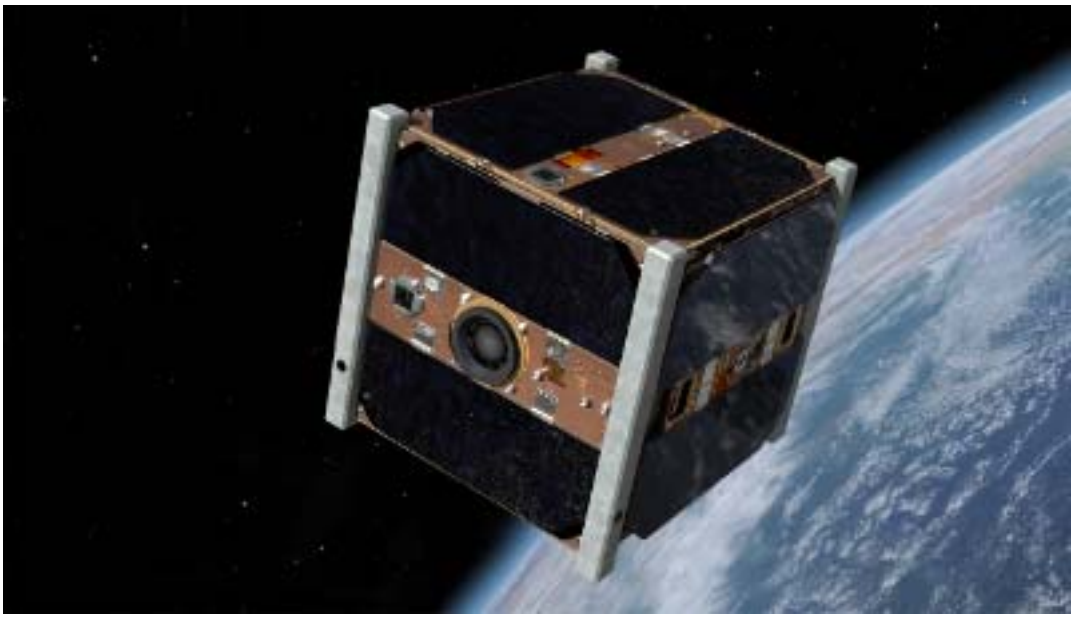


Figure 1 : Satellite SwissCube [1]

2 INTRODUCTION

Tous ces microsatellites ont été développés autour de microcontrôleur, afin de prouver que l'utilisation d'une FPGA dans un tel système est tout à fait viable, une carte de contrôle a été développée durant un travail de Master. Cette carte a été reprise pour le projet de Bachelor de Stéphane Lovejoy [2] qui consistait à développer des interfaces afin de piloter un certain nombre de capteurs et d'actuateurs, eux même présents sur les CubeSats et qui permettent d'orienter et stabiliser ces derniers.

Le but de ce travail de semestre consiste donc à récupérer, comprendre et modifier ce circuit d'interface afin de réaliser un ADCS* réaliste pour des missions tel que CubETH et CleanSpaceOne.

Dans ce rapport sera traité tout d'abord les modifications du cahier des charges selon les spécifications exigées par les projets CubETH et CleanSpaceOne, suivi par une analyse du travail de bachelor de Stéphane Lovejoy [2]. Ensuite un chapitre sera consacré aux différents capteurs et actuateurs, précédent une explication sur les différentes étapes du développement du nouveau circuit électronique. Après cela, la partie sur le protocole de communication utilisé dans le système sera traitée, suivie par l'implémentation du code VHDL ainsi que par la description de l'application de démonstration. Enfin viendra la partie des tests et des résultats, finalisée par une conclusion.

3 CAHIER DES CHARGES

Le but ultime consiste à développer un ADCS/CDMS* fonctionnant autour d'une FPGA.

Cahier des charges initial :

- Implémenter une interface FPGA pour une configuration réaliste d'un ADCS (six Sun-Sensors, deux Magnétotorquers, quatre Reaction Wheels et peut-être un StarTracker).
- Implémenter un code VHDL pour le traitement des données des capteurs en utilisant des algorithmes simples.
- Effectuer une comparaison de performances avec les solutions basées sur microcontrôleur.

Pour atteindre ces objectifs, plusieurs tâches intermédiaires doivent être effectuées au préalable :

- Analyser le système existant et identifier les modifications à y effectuer.
- Modifier le hardware/software existant pour une configuration réaliste d'un ADCS.
- Proposer et implémenter dans la FPGA de simple algorithme de stabilisation.

*ADCS : Attitude and Determination Control System, est un système de détermination et de contrôle d'orientation utilisé dans les satellites.

*CDMS : Control and Data Management System, est un système qui fait office de cerveau dans un satellite. Un de ses rôles est de recueillir les différentes informations des sous systèmes tel l'ADCS, de les traiter, et de piloter en conséquence l'ensemble du satellite.

4 ANALYSE

4.1 Etat actuel du développement

4.1.1 Circuit d'interface

Le travail de Stephane Lovejoy [2] se compose d'un système utilisant deux cartes électronique comme le montre la figure 2. L'ensemble du système est piloté via l'UART par une application sur PC.

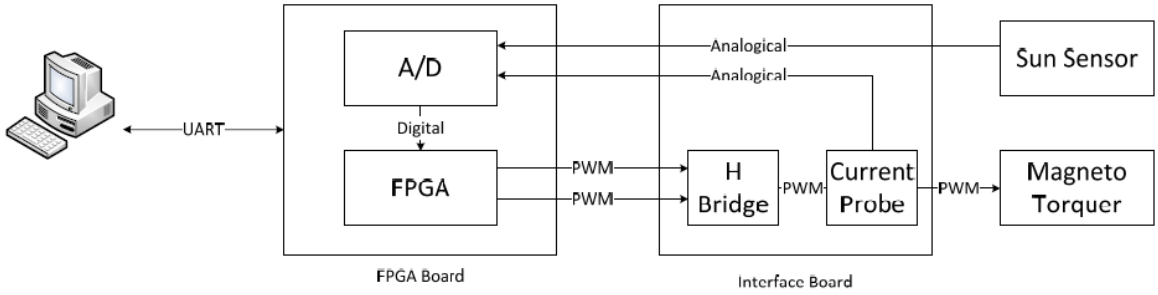


Figure 2 : Schéma bloc CDMS [2]

La première carte contenant la FPGA (figure 3) a été développée durant le travail de Master de Bastien Praplan [3]. Mis à pars la FPGA, une Spartan-6 XC6SLX150, la carte se compose de deux mémoire flash de 128MB (RC28F00AM29EW) et de 16MB (W25Q128FVE), d'un convertisseur analogique/digitale (ADS8028), d'une mémoire SDRAM (MT48LC4M16A2) et de divers connecteurs pour les commandes et interfaces.

Figure 3 : Photo carte FPGA [2]

La deuxième carte permet de faire l'interface entre la FPGA et les différents capteurs et actuateurs. Après analyse, il s'avère qu'elle a été prévue pour connecter uniquement un seul Sun-Sensor ainsi qu'un seul Magnéto-Torquer. De ce fait cette carte doit être revue afin de pouvoir y intégrer les capteurs et actuateurs manquants.

4.1.2 Code FPGA

La figure 4 montre une vue simplifiée du circuit interne de la FPGA développé par Stephane Lovejoy [2], celui-ci se compose de quatre blocs principaux.

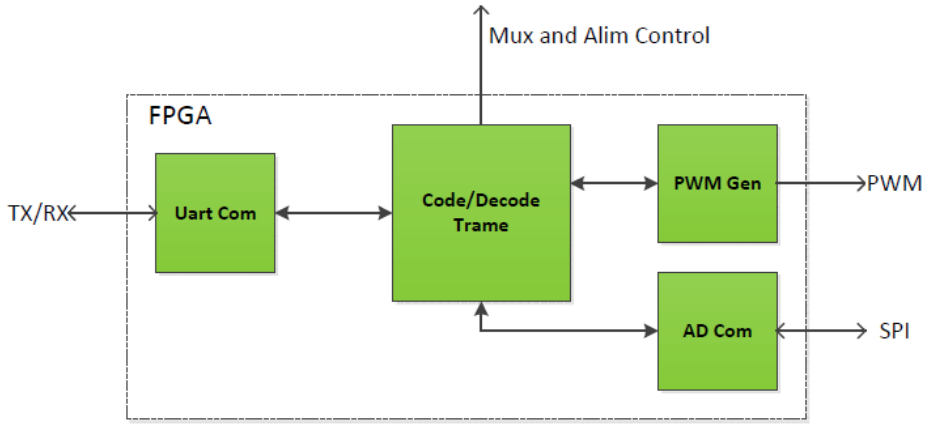


Figure 4 : Schéma-bloc précédent code FPGA

Le premier bloc est utilisé pour la gestion de la communication UART avec le PC, son rôle est d'une part, désérialiser les trames provenant du PC et de les envoyer sous forme de paquets huit bits au bloc suivant, et de l'autre, construire des trames UART avec les données provenant du système.

Le bloc PWM Gen permet de générer un signal PWM selon une valeur de Duty-Cycle donné. Il est utile pour la commande du Magnéto-Torquer.

Le bloc AD Com permet de piloter le convertisseur AD en utilisant le protocole SPI. Il suffit d'indiquer à ce bloc le canal de l'AD à convertir ainsi qu'un signal de start, puis les valeurs converties sont retournées quelque temps après. La figure 5 contient un exemple de trame SPI pour initialiser une conversion sur l'AD.

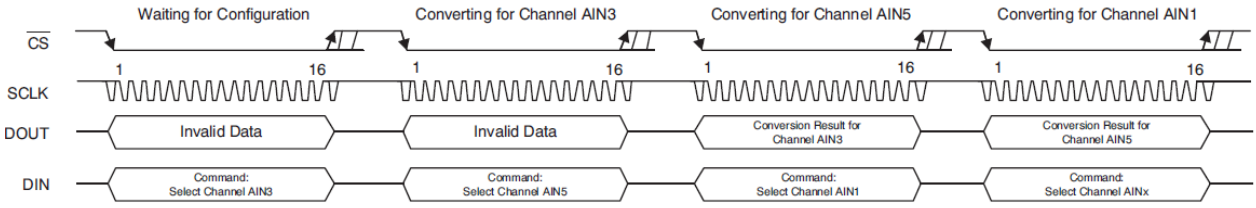


Figure 5 : Lecture du convertisseur AD en SPI [6]

Il est important de remarquer que les données converties pour un certain canal sont renvoyées deux trames plus tard que l'arrivée de la requête. L'annexe 1 contient une description du registre à configurer pour initialiser une conversion sur le convertisseur.

Le dernier bloc Code/Decode Trame est comme son nom l'indique utilisé pour déchiffrer les commandes provenant du PC et de piloter le reste du système en conséquence. Deux types de commandes sont prévus : une pour le pilotage du Magnéto-Torquer et une autre pour la lecture des valeurs du Sun-Sensor. Les trames pour ces commandes sont détaillées sur la figure 6.

Command	Frame 1	Frame 2	Frame 3		Frame 4
Write PWM	StartFrame	Address of the PWM bloc	Pos/Neg	DutyCycle Value	-
	10101010	10000000	x	xxxxxxx	
AD Conversion	Startframe	AD conversion channel	-	-	-
	10101010	0000xxxx			
Result AD Conversion	StartFrame	AD conversion channel	stuffing bits	AD MSB bits	AD LSB bits
	10101010	0000xxxx	0000	xxxx	xxxxxxxx

Figure 6 : Liste commandes UART du précédent protocole

Malgré que la structure générale puisse être réutilisée, plusieurs parties de ce système devront être modifiées, notamment le protocole UART, afin de permettre le pilotage des capteurs et actuateurs supplémentaires. Les modifications du code VHDL seront effectuées grâce au logiciel HDL Designer.

4.1.3 Application QT

Comme cité précédemment, le système est contrôlé par un ordinateur connecté à la carte de base par UART. Une application de démonstration (figure 7) a donc été développée sur QT par Stephane Lovejoy [2] pour permettre l'affichage des valeurs du Sun-Sensor et le contrôle du Magnéto-Torquer. Une simulation du satellite a aussi été intégrée à l'interface graphique et permet de visualiser les changements d'orientation du cube. Cette application devra, comme le reste, être modifiée afin d'y ajouter l'ensemble des valeurs des nouveaux capteurs/actuateurs.

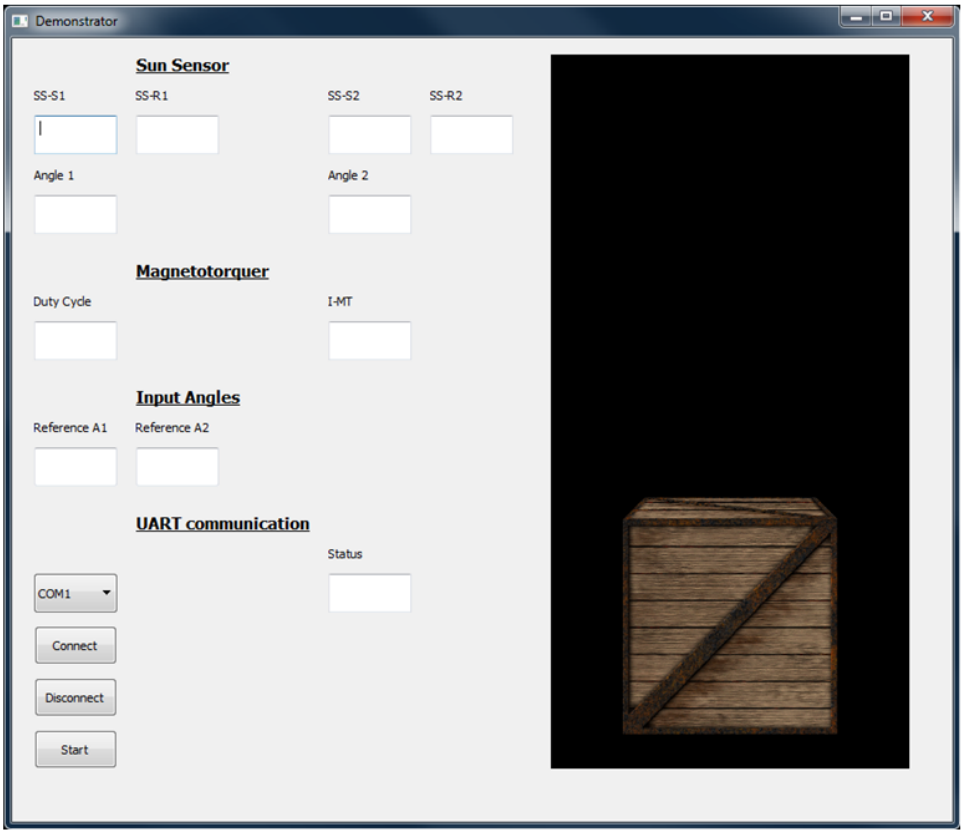


Figure 7 : Application de démonstration [2]

4.2 Analyse des besoins de l'EPFL

Suites aux discussions avec les responsables du projet cubETH à l'EPFL, les objectifs ont été modifiés pour correspondre d'avantage à leurs besoins. Ainsi, le nombre de chaque capteur et actuateur ont été revus et de nouveaux capteurs seront rajoutés comme par exemple un module GPS. Cette nouvelle configuration est prévue pour des futurs projets utilisant des 6units CubSats, ces derniers étant simplement des satellites d'une taille six fois plus grande qu'un CubSat ordinaire, ce qui permet d'avoir six fois plus d'espace pour embarquer des systèmes de mesures ou autre. Ainsi le système qui sera développé ne sera rattaché ni au projet cubETH ni au projet CleanSpaceOne, qui ont des exigences très pointues. La décision a donc été prise de développer un système FPGA avec un nombre « standard » de capteurs/actuateurs pour en prouver la viabilité et ainsi être utilisé dans de futurs projets.

En ce qui concerne la partie algorithmes de stabilisation, l'idée a été revue car ces derniers sont très complexes, nécessitent beaucoup de temps de mise en œuvre et sont très dépendants de la mission attribuée au satellite. Ainsi à la place de les intégrer directement dans la FPGA ils seront implémentés dans une autre unité de commande, que ça soit un microcontrôleur ou une autre FPGA. Leur implémentation ne sera donc pas traitée durant ce travail de Bachelor, ce dernier se focalisera plutôt sur le contrôle pur des actuateurs ainsi que la fusion des données provenant des différents capteurs. Dans cette optique une interface de commande de la FPGA devra être mise en place et permettra un choix indépendant des algorithmes de stabilisation. Des exemples de ces algorithmes sont disponibles en annexe 2.

Cahier des charges modifié :

- Implémenter une interface FPGA pour une configuration réaliste d'un ADCS :
    - 6 Sun-Sensors,
    - 2 Magneto-Meters
    - 2 Gyroscopes,
    - 1 GPS
    - 1 Star-Tracker
    - 3 Magneto-Torquers,
    - 3 Reaction-Wheels,
- Implémenter un code VHDL pour l'acquisition des données des capteurs et le contrôle des actuateurs, le tout piloté par un système de commande externe.
- Effectuer une comparaison de performances avec les solutions basées sur microcontrôleur.

5 SPECIFICATIONS CAPTEURS/ACTUATEURS

Avant de pouvoir se lancer dans la schématique de la nouvelle carte d'interface, une recherche sur chaque capteurs/actuateurs a été nécessaire pour, d'une part, comprendre leur mode de fonctionnement et d'une autre connaitre les composants nécessaire à leur intégration dans le circuit. Les chapitres suivant contiennent un descriptif des différents périphériques. Un résumé de leurs caractéristiques se trouve en annexe 3.

5.1 Sun-Sensors

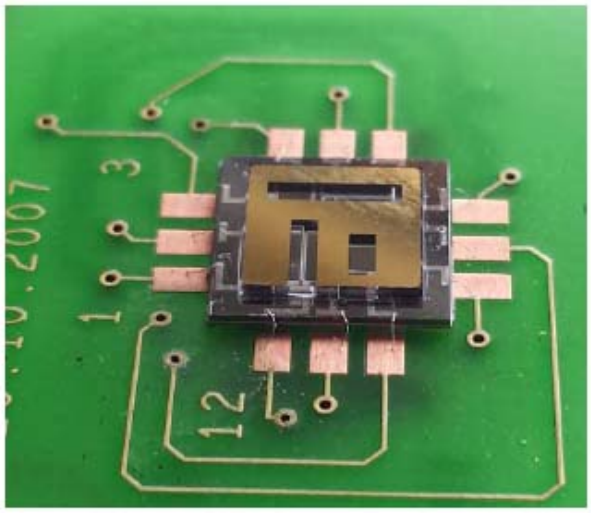


Figure 8 : Sun-Sensor [2]

Les capteurs solaires sont des composants indiquant la direction de la source de lumière par rapport à leur position en fournissant deux angles : dans l'axe X et dans l'axe Y, comme indiqué sur la figure 9.

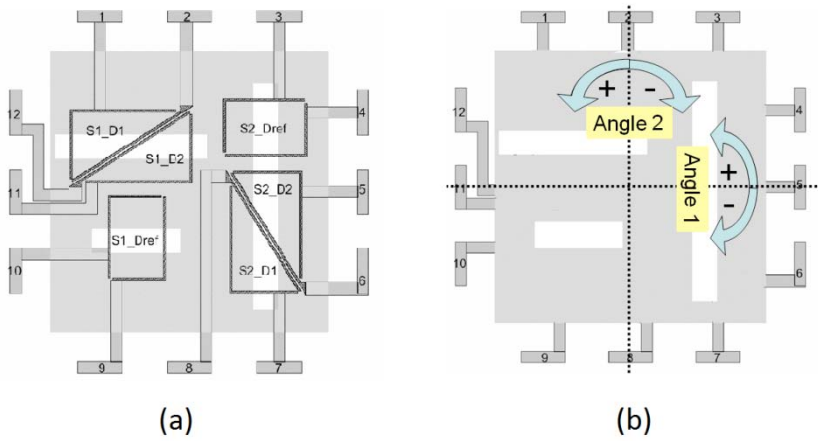


Figure 9 : Référence d'angles pour les Sun-Sensors [2]

La source de lumière sous-entendue ici est bien entendue le Soleil, et donc ces capteurs permettront d'orienter le satellite par rapport à notre étoile. Comme le satellite a une forme

cubique, il faudra déployer six capteurs (un sur chaque face) pour couvrir toutes les orientations possibles.

Les capteurs utilisés ici ont été récupérés du travail de Bachelor de Stéphane Lovejoy [2], ce qui implique que la schématique pour leur implémentation sera identique. En réalité ces capteurs fournissent quatre valeurs analogiques, deux pour chaque angle : une valeur de référence et la mesure en elle-même. Ces quatre signaux doivent être filtrés puis convertis en valeur digitale avant de pouvoir être utilisables. Une fois cela fait une petite opération doit être exécutée pour obtenir les véritables valeurs d'angles, cette opération sera faite numériquement dans la FPGA et est donc traités dans le chapitre correspondant. La figure 10 montre un graphe des valeurs de sorties du Sun-Sensor en fonction de l'angle de la source lumineuse.

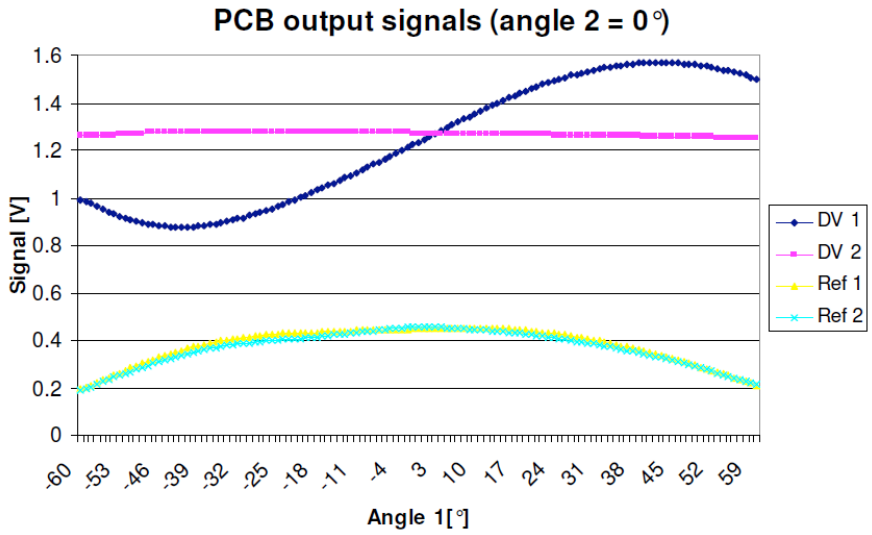


Figure 10 : Graphe de tension de sortie du Sun-Sensor en fonction de l'angle de la source lumineuse [2]

5.2 Magneto-Meters

Les Magnétomètres permettent de mesurer le champ magnétique présent aux alentours du capteur. Dans notre cas le champ magnétique mesuré sera celui de la terre, ce qui nous permettra d'orienter le satellite en fonction de notre planète. Deux capteurs seront présent dans le système, pour des raisons historiques, le choix s'est porté sur des magnétomètres qui ont déjà étés utilisés par l'EPFL: le MAG3110 et le HMC5883L. Bien qu'un seul magnétomètre suffise pour couvrir la totalité des mesures (Magnétomètre trois axes) l'utilisation de deux permet de garantir la fiabilité des mesures, et aussi d'avoir une sécurité supplémentaire dans le cas d'un défaut sur un ou l'autre des magnétomètres.

Un des avantages de ces capteurs est qu'ils sont pilotables par le protocole I2C, ce qui permettra de diminuer le nombre de pistes lors de la conception de la carte. L'utilisation de ce protocole sous-entend aussi que les valeurs mesurées sont directement disponible en digital, ce qui diminuera un peu la charge du convertisseur AC/DC de la carte de base. Six registres doivent être lus pour obtenir les valeurs du champ magnétique sur les trois axes, étant donné que ces valeurs sont codées sur 16bits, donc deux registres par axes. Plus de détails sont fournis dans le chapitre 8.4 consacré à l'implémentation du code de lecture de ces capteurs.

5.3 Gyroscopes

Les gyroscopes permettent de mesurer la vitesse angulaire d'un objet en utilisant l'effet des forces de Coriolis [4]. Ainsi il est possible de mesurer la rotation du satellite dans les trois axes et de pouvoir réagir en conséquence. Tout comme les magnétomètres, les gyroscopes choisis ont déjà étés utilisés par l'EPFL, seront aussi présent au nombre de deux et utilisent le protocole I2C. Les valeurs d'accélérations sont de nouveau codées sur 16bits, dans six registres (voir chapitre 8.4).

Ainsi la carte comprendra un ITG-3200 et un L3G4200D.

5.4 GPS

Le GPS ou Global Positioning System est un système qui permet d'obtenir sa position sur terre par des valeurs de longitude et latitude fournies grâce à un réseau de satellites stationnaires. Bien qu'il existe d'autres systèmes similaires, comme GLONASS (équivalent du GPS contrôlé par la fédération de Russie), le nom GPS est attribué le plus souvent pour les capteurs eux-mêmes.

Comme cité précédemment, le système utilise des satellites présents en orbite terrestre, ainsi le système est utilisable seulement sur terre et pour des satellites présents en orbite terrestre basse. Ce qui peut poser des problèmes si la mission du satellite utilisant le GPS exige de dépasser les 2000km d'altitude.

La position est donnée généralement en degré, minute et secondes. En sachant que 60 secondes font évidemment 1 minute et 60 minutes font 1 degré. La figure 11 offre une vue des références pour ces valeurs de longitude et latitude.

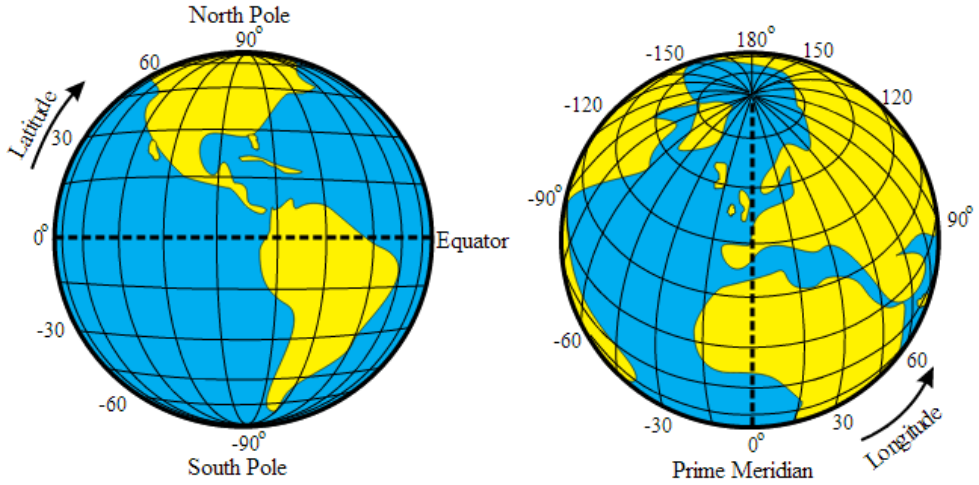


Figure 11 : Référence de longitude et latitude, source : Wikipédia

Le choix du GPS s'est porté sur un kit de MikroElektronika : le Mikroe-1032, visible en figure 12. Ce dernier dispose directement d'une interface SPI ou I2C, ainsi qu'un connecteur pour l'antenne, ce qui permet une implémentation rapide au détriment du prix.

Figure 12 : Kit GPS Mikoe-1032, source : www.mikroe.com/clicks/gps/

Le datasheet de la puce (LEA-6S) intégrée à cette carte indique que ce système fournit en sortie des messages ayant pour format le protocole NMEA. Un exemple d'un de ces messages est disponible en figure 13.

NMEA Protocol Frame

Checksum range

$	<Address>	{,<value>}	*<checksum>	<CR><LF>
Start character	Address field.	Data field(s)	Checksum field	End sequence
Always '$'	Only digits and uppercase letters, cannot be null. This field is subdivided into 2 fields: <XX> Talker Identifier, always **GP** for a GPS receiver, **P** for proprietary Messages; <XXX> Sentence Formatter Defines the message content	Delimited by a ','. Length can vary, even for a certain field.	Starts with a '*' and consists of 2 characters representing a hex number. The checksum is the exclusive OR of all characters between '$' and '*'.	Always <CR><LF>

Example:

$	GP	ZDA	,141644.00,22,03,2002,00,00	*67	<CR><LF>

Figure 13 : Format des messages NMEA [7]

Pour lire ces messages, il suffit de lire le registre à l'adresse 0xFF sur le bus I2C, malheureusement la façon dont sont débités ces messages sur le bus n'est pas précisé. En conséquence de cela, des tests de lecture du GPS devront être effectués avant de pouvoir implémenter le décodage des trames reçues et le formatage des valeurs de longitude et latitude. Une liste des messages NMEA supporté par ce GPS est disponible en annexe 4. La figure 14 illustre en détail le contenu du message GLL, message contenant les valeurs de longitude et latitude.

Position data: position fix, time of position fix, and status

An example of the GLL message string is:

$GPGSA,A,3,3,6,27,19,9,14,21,22,18,15,,,2.1,1.0,1.8*03

GLL message fields

Field	Meaning
0	Message ID $GPGLL
1	Latitude in dd mm,mmmm format (0-7 decimal places)
2	Direction of latitude N: North S: South
3	Longitude in ddd mm,mmmm format (0-7 decimal places)
4	Direction of longitude E: East W: West
5	UTC of position in hhmmss.ss format
6	Fixed text "A" shows that data is valid
7	The checksum data, always begins with *

Figure 14 : Détail du message GLL [8]

5.5 Star-Tracker

Cet appareil permet de se situer dans l'espace en fonction des étoiles visibles. En effet ce système photographie les constellations et compare avec une base de données ce qui permet de connaître la position du satellite. Comme ces appareils sont relativement cher, le choix pour le développement s'est porté sur une simple caméra qui prendra uniquement des photos. La détermination de la position ne se fera donc pas sur ce circuit, les photos seront simplement stockées et non comparées.

La HES a déjà développé une petite carte contenant la caméra MT9V114 de chez Aptina, cette carte, visible en figure 15 possède un microcontrôleur pour l'initialisation des différents registres ainsi qu'une interface I2C pour le pilotage de la caméra. Ce qui rendra aisé son ajout au système car il ne nécessitera qu'un connecteur au BUS I2C.

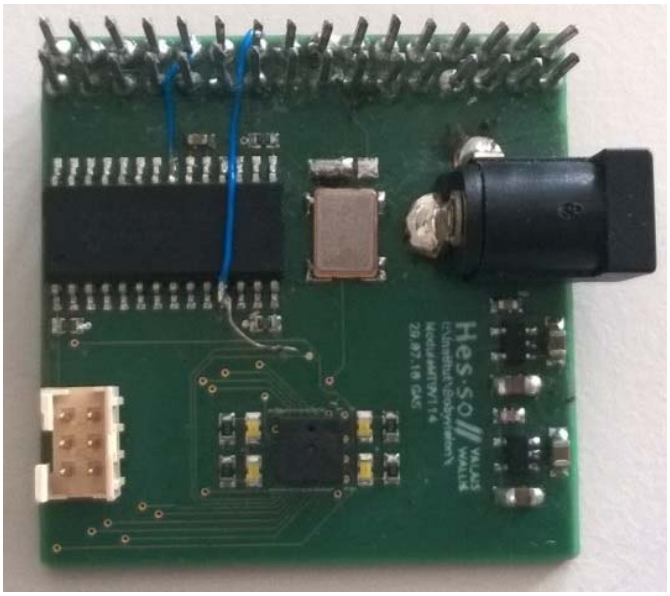


Figure 15 : Circuit caméra MT9V114

En ce qui concerne les données fournies par cette caméra, celles-ci pourront être sauvées dans la RAM présente sur la carte FPGA. La figure 16 montre comment ces données sont envoyées depuis la caméra.

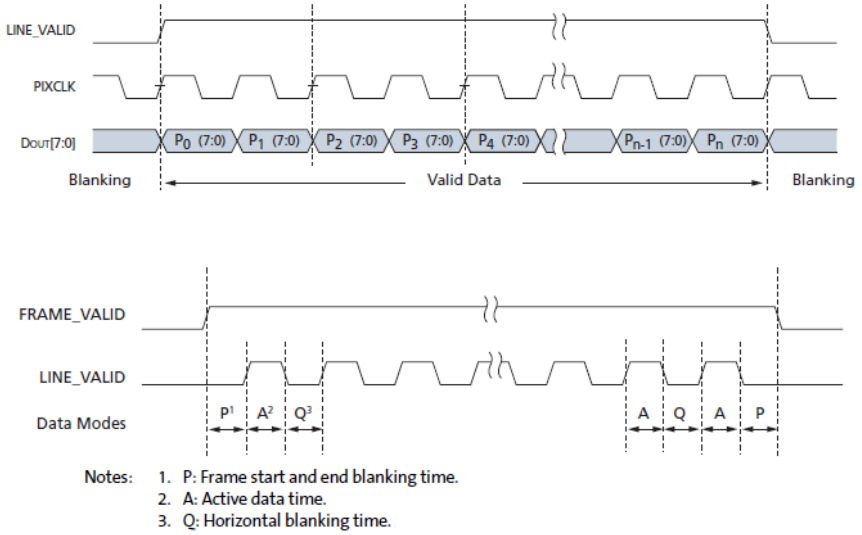


Figure 16 : Format des données fournies en sortie de la caméra [9]

5.6 Magneto-Torquers

Les magnétos coupleurs permettent à un satellite de modifier son orientation grâces aux effets des champs magnétiques. Ces actuateurs se composent uniquement d'une simple bobine où circule un courant électrique, ce dernier va générer un champ magnétique à travers la bobine qui va ensuite interagir avec le champ magnétique terrestre pour créer un couple, ce qui permet finalement de contrôler l'orientation du satellite.

Pour un contrôle total du satellite, trois magnétos coupleurs sont nécessaires, un pour chaque axe de rotation. Afin de pouvoir piloter la puissance des champs magnétiques générés, un signal PWM sera utilisé dans les bobines.

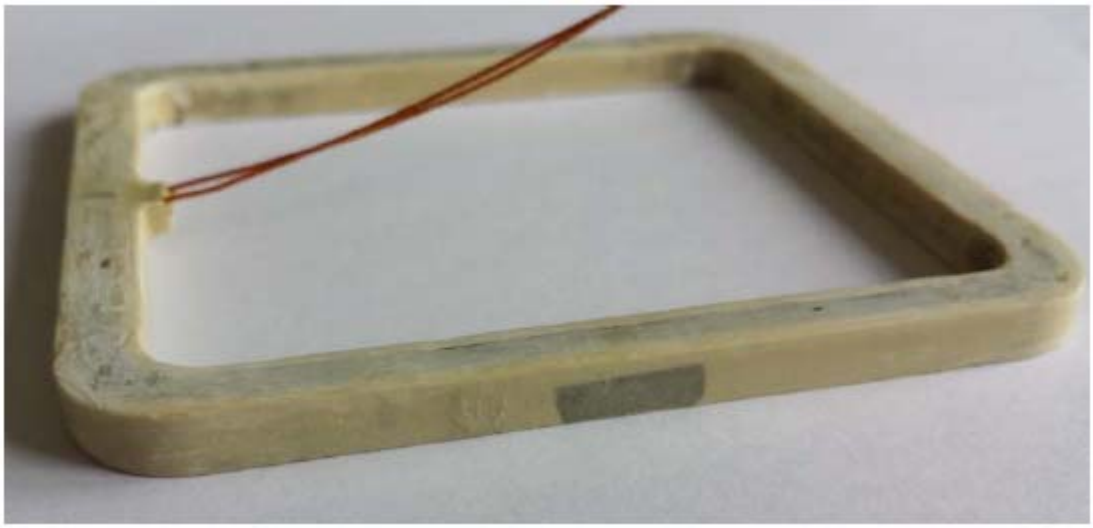


Figure 17 : Magneto-Torquer [2]

5.7 Reaction-Wheels

Les roues de réaction sont utilisées pour orienter le satellite de manière similaire aux magnéto-coupleur à la différence qu'ils ne nécessitent pas de champ magnétique externe. Ce sont de simples masses couplées à des moteurs électriques, lorsque le moteur est mis sous tension la masse va commencer à tourner, et par conservation du moment angulaire, le satellite va se mettre à tourner dans le sens opposé. Il en faut de nouveau trois pour pouvoir orienter le satellite dans les trois axes. La figure 18 montre un de ces systèmes qui a été développé au Space Center à l'EPFL. Dans ce projet de simples moteurs à courant continu seront utilisés pour simuler ces roues de réactions.

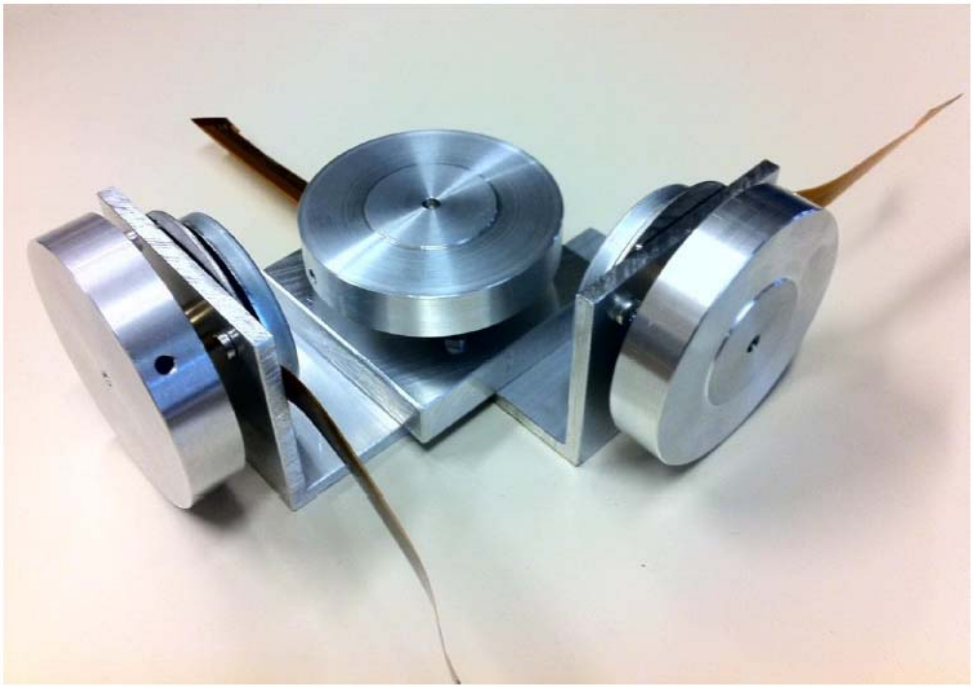


Figure 18 : Reaction Wheels [5]

6 CARTE D'INTERFACE

6.1 Schéma Bloc interface

Suite aux recherches et clarifications des différents capteurs, un schéma bloc de la carte d'interface a pu être réalisé : voir annexe 5.

On y retrouve la carte de base, contenant la FPGA et le convertisseur AC/DC, cette fois ci contrôlée par un ordinateur mais aussi par un microcontrôleur. En effet le PC n'est utilisé que pour la démonstration, lors de l'intégration du système dans un satellite, cette carte devra être pilotée par un autre microcontrôleur. C'est ce dernier qui implémentera les algorithmes de stabilisation propre à la mission du satellite.

Les différents capteurs et actuateurs sont présents, soit sur la carte d'interface, soit en dehors. La différence provient du fait que certains périphériques seront connectés par des câbles à la carte, car ils devront être placés à des endroits bien spécifiques sur le satellite. Comme par exemple la caméra qui sera sans doute sur une face du satellite, ou bien les Magneto Torquers qui doivent être placé sur chacun des trois axes du satellite. Les

autres périphériques englobés par la carte mezzanine sont directement des composants soudés sur le PCB.

En ce qui concerne les moteurs-DC (Reaction Wheels) et les Magneto Torquers, ceux-ci nécessitent des ponts H pour leur fonctionnement, étant donnés qu'ils sont pilotés par signaux PWM. De même, les Magneto Torquers disposent de capteurs de courant, ce qui permettra d'effectuer une régulation de ces actuateurs, plutôt qu'une simple commande.

Pour la partie Sun-Sensor, les signaux de ces derniers passent tout d'abord par des filtres, puis par des multiplexeurs. Ces derniers sont nécessaires car malheureusement le convertisseur ne dispose que de sept canaux d'entrée, et comme cité précédemment les Sun-Sensors procurent quatre signaux chacun. Ceci implique que la mesure de ces six capteurs seront faite séquentiellement et non en parallèle.

6.2 Schématique carte d'interface

La première chose qui a été effectuée avant de commencer la schématique a été de déterminer les différentes interfaces disponibles de la carte FPGA, soit par quel connecteurs transitent les différents signaux. Pour cela la schématique de la carte de base a été récupérée et analysée. Voir annexe 6

Une fois cela connu il a été possible de se pencher sur le problème de l'ajout des différents périphériques.

Pour l'ajout des Sun-Sensors, la schématique de la carte d'interface précédente était déjà munie de tous les éléments nécessaires. Les multiplexeurs présents permettent de multiplexer jusqu'à huit signaux. Ce qui est suffisant vu que seulement six Sun-Sensors sont utilisés. La schématique du traitement et filtrages des signaux provenant de ces capteurs a été simplement récupérée et multipliée. Ces filtres sont de type passe bas 1er ordre et ont une fréquence de coupure de 10.6kHz, ils ne servent qu'à réduire le bruit.

De manière similaire la schématique pour les Magneto Torquers, ayant déjà été développée, a été récupérée et multipliée. Pareil pour les Reaction Wheels étant donné qu'ils fonctionnent aussi avec des signaux PWM. La schématique est donc la même, si ce n'est que les capteurs de courant ont été supprimés. Pour le reste, cette partie du circuit se compose de ponts H suivi par des filtres afin de supprimer les commutations et de n'obtenir qu'une tension continue. Les commutations se font à 200khz et le filtre qui avait été choisi est de type passe bas 1er ordre avec une fréquence de coupure à 15.9kHz, ce qui est largement suffisant pour couper les hautes fréquences.

Tous les autres périphériques fonctionnent avec le protocole I2C, ainsi ils sont tous connectés au même bus. Pour la connexion avec la carte de la caméra, la schématique de cette dernière a été récupérée et se situe en annexe 7. De manière similaire, la schématique pour le kit GPS se trouve en annexe 8.

Finalement, avec toutes ces informations la schématique complète a pu être réalisée et se trouve en annexe 9.

6.3 Routage carte d'interface

Avant de démarrer le routage, il a fallu déterminer l'emploi par l'EPFL de cette carte, en effet selon son but certaines contraintes pourront rentrer en vigueur pour la taille et la forme du PCB. Il se trouve que cette carte devra pouvoir être intégrée dans un satellite utilisant la structure de six cubeSats, la figure 19 procure un exemple d'un de ces systèmes.

Figure 19 : 6 Units cubeSat, Source : http://www.cubesatshop.com

Ainsi la taille maximale pour les PCB est de 94*94mm, ce qui correspond avec le système précédemment réalisé qui a une taille d'environ 86*92mm, représenté en figure 20.

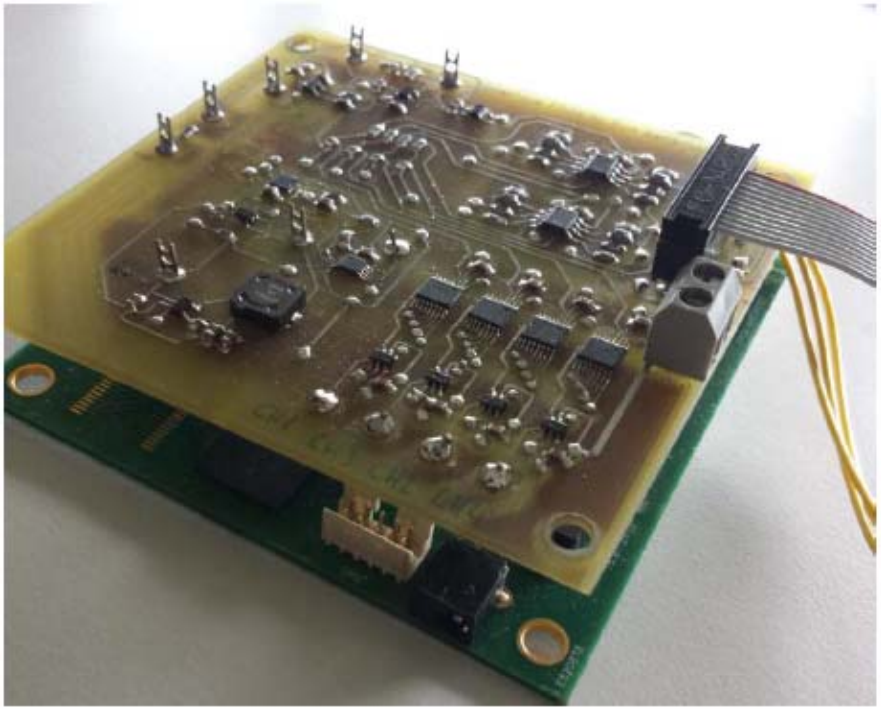


Figure 20 : Système complet précédent. [2]

Comme la nouvelle schématique est beaucoup plus volumineuse que pour le système précédent, un calcul a été effectué pour se rendre compte si son implémentation était possible sur une seule carte d'interface. Le tableau regroupant toutes les valeurs de surface de chaque partie de la schématique est présent en annexe 10. D'après ce tableau, la

surface d'un PCB est suffisamment grande pour accueillir l'ensemble des composants avec un facteur de routage de 2.4.

La figure 21 représente la nouvelle carte d'interface, ce PCB est routé sur six couches et les composants sont pratiquement tous monté sur le dessus de la carte, pour éviter qu'ils se heurtent à d'autres composants de la carte de base. Seul le connecteur de la mezzanine et quelques composants SMD passifs tels les résistances ou condensateurs sont soudés sur le dessous du PCB.

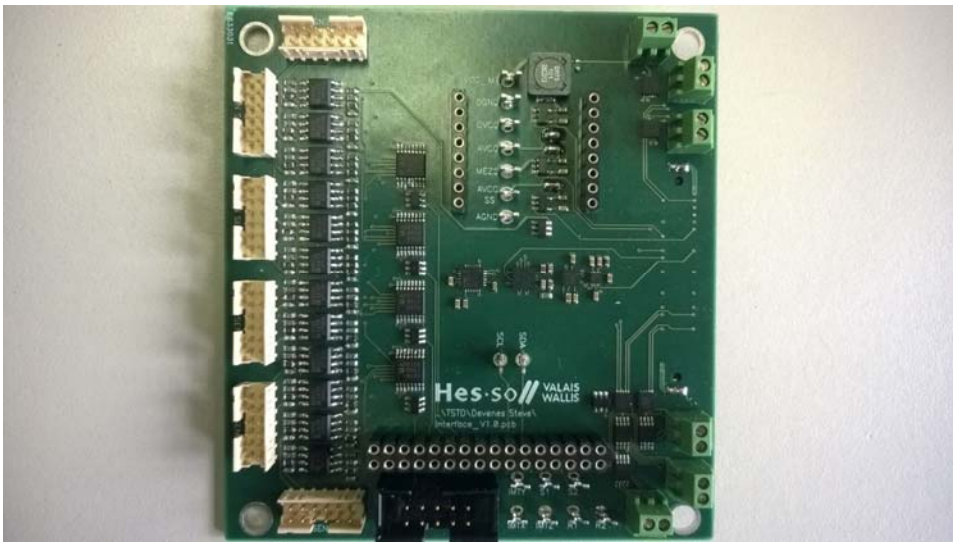


Figure 21 : Carte d'interface

La liste des composants commandés se trouve en annexe 11.

7 PROTOCOLE DE COMMUNICATION

7.1 Description

Etant donné que le protocole de communication UART initial n'ait été prévu que pour piloter un Sun-Sensor et un Magnéto-Torquer, il a été revu quelque peu pour augmenter sa flexibilité et ainsi permettre de piloter tous les nouveaux périphériques. Ce nouveau protocole s'inspire du protocole I2C et se présente comme à la figure 22. Les caractéristiques prévues sont les suivantes :

- Vitesse : 38400 baud
- Nombre de bit de données : 8
- Nombre de bit de fin : 1

Start Packet	Register Address	Register Value
10101010	xxxxxxxx	xxxxxxxx

Figure 22 : Format d'une commande UART

Les commandes commencent toutes par un premier paquet de start, que l'on pourrait comparer à l'adresse du périphérique avec qui l'on souhaite communiquer par I2C. Suivi par un paquet contenant l'adresse du registre que l'on souhaite lire ou écrire. Et finalement terminé par la valeur que l'on souhaite écrire. La figure 23 représente une table de toutes les commandes qui ont été prévues.

Register Name	Read/Write	RegAddress	Register Value							
			B7	B6	B5	B4	B3	B2	B1	B0
SS_Enable	Write	0x01	-	-	SS6	SS5	SS4	SS3	SS2	SS1
Gyro_Enable	Write	0x02	-	-	Gyro2	Gyro1	MM2	MM1	GPS	ST
MT/RW_Enable	Write	0x03	-	-	MT3	MT2	MT1	RW3	RW2	RW1
MT1_DutyCycle	Write	0x04	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
MT2_DutyCycle	Write	0x05	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
MT3_DutyCycle	Write	0x06	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
RW1_DutyCycle	Write	0x07	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
RW2_DutyCycle	Write	0x08	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
RW3_DutyCycle	Write	0x09	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0

Register Name	Read/Write	Register Address	Returned Trame						Nb Data Packet
			P1	P2	P3	P4	P5	P6	
SSAngle_Value	Read	0x0A	SSid	A1 MSB	A1 LSB	A2 MSB	A2 LSB		5
Gyro_Value	Read	0x0B	Gyro1 X H	Gyro1 X L	Gyro1 Y H	Gyro1 Y L	Gyro1 Z H	Gyro1 Z L	12
			Gyro2 X H	Gyro2 X L	Gyro2 Y H	Gyro2 Y L	Gyro2 Z H	Gyro2 Z L	
MM_Value	Read	0x0C	MM1 X H	MM1 X L	MM1 Y H	MM1 Y L	MM1 Z H	MM1 Z L	12
			MM2 X H	MM2 X L	MM2 Y H	MM2 Y L	MM2 Z H	MM2 Z L	
GPS_Value	Read	0x0D	Lg degré	Lg min H	Lg min L	Lat degré	Lat min H	Lat Min L	6
ST_Value	Read	0x0E	pixel0	pixel1	pixel2	pixel3	...	...	307200

Figure 23 : Listes des commandes UART

Les neufs premiers registres ne sont accessibles qu'en écriture, alors que les cinq derniers ne sont accessibles qu'en lecture. En ce qui concerne les trames de réponses retournées par la FPGA suite aux commandes de lecture, elles souscrivent aux mêmes règles que les trames de commande, c'est-à-dire un paquet de start, suivi par l'adresse du registre, et terminée par les paquets de données. Le nombre de paquets de données diffère selon les capteurs, par exemple la lecture des deux gyroscopes demande 12 paquets : deux paquets pour chaque axes vu que les données sont codées sur 16 bits, trois axes par gyroscopes, donc 2*3*2 = 12.

L'avantage de ce protocole est non seulement sa simplicité mais aussi sa grande flexibilité, en effet il est aisé d'ajouter ou supprimer des commandes, dans le cas par exemple ou l'on voudrait ajouter des capteurs ou d'autres fonctionnalités au circuit.

7.2 Détails des commandes

7.2.1 SS_Enable

Ce registre permet de contrôler l'activation des Sun-Sensors. Chaque bit du registre correspond à un Sun-Sensor, ce qui permet de les activer individuellement des autres. L'écriture d'un '1' logique active le capteur correspondant tandis qu'un zéro le désactive. Comme seulement six Sun-Sensors sont utilisé dans ce circuit, seul les six bits de poids faible sont utilisés.

7.2.2 Gyro_Enable

Ce registre permet de contrôler l'activation des périphériques qui sont connectés au bus I2C du circuit, c'est-à-dire non seulement les gyroscopes, mais aussi les magnétomètres, le GPS et la caméra. De nouveau, chaque bit correspond à un capteur.

7.2.3 MT/RW_Enable

Tout comme les deux registres précédents, celui-ci permet d'activer les Magnéto Torquers et les Reaction Wheels. Rendant ainsi possible le choix de n'utiliser qu'un seul, deux ou les trois actuateurs de chaque type.

7.2.4 MTx_Enable RWx_Enable

Ces six registres contiennent la valeur des Duty-Cycles en % pour chaque actuateur. Ces valeurs sont codées en complément à deux sur huit bits, permettant théoriquement une plage allant de -128 à 127 mais seules des valeurs comprises entre -100 et 100 sont autorisée pour le contrôle des PWM.

7.2.5 SSAngle_Value

La lecture de ce registre permet d'obtenir la valeur d'angle que forme le soleil avec le satellite. Le premier paquet SSid contient le numéro du Sun-Sensors qui détecte le plus de luminosité, donc théoriquement le Sun-Sensor qui se trouve face au soleil. Les 4 paquets suivant contiennent les valeurs des deux angles du Sun-Sensors, codées en complément à deux sur 16 bits : les MSB contenants les bits d'entier, précédent la virgule, et les LSB les bits de fraction, suivant la virgule. Ce qui donne une plage de valeur allant environ de -128 à 127.996 avec une résolution de 0.00390625.

7.2.6 Gyro_Value

Ce registre permet d'obtenir les valeurs des deux gyroscopes. Les six premiers paquets contiennent les valeurs du premier gyroscope, en l'occurrence le ITG-3200 tandis que les six derniers contiennent les valeurs du L3G4200D. Ces valeurs sont codées en compléments à deux sur 16 bits, c'est pourquoi chaque axe nécessite deux paquets.

7.2.7 MM_Value

Ce registre fonctionne exactement de la même manière que le précédent, le premier magnétomètre étant le MAG3110 et le second le HMC5883L.

7.2.8 GPS_Value

Ce registre permet la lecture des valeurs du GPS Mikroe-1032. Six paquets sont prévus pour envoyer les coordonnées de longitudes et latitude. Les trois premiers contenant les valeurs de longitude et les trois derniers contenant les valeurs de latitude, selon le format suivant : un paquet pour la valeur des degrés, un autre pour la valeur entière des minutes et le dernier pour la valeur décimale des minutes.

7.2.9 ST_Value

Ce dernier registre permet de lire les valeurs des pixels de la caméra MT9V114. Comme cette caméra possède une résolution de 640*480 pixels la lecture du tout nécessite 307200 paquets. Selon la vitesse de transmission de la communication UART, ce transfert peut nécessiter énormément de temps, et donc empêcher la lecture des autres registres.

8 DEVELOPPEMENT FPGA

8.1 Bloc de génération PWM

8.1.1 Description

Ce nouveau bloc de génération de PWM reprend les caractéristiques du précédent, à la différence qu'il permet de contrôler trois signaux PWM à la place d'un seul, permettant ainsi le pilotage simultané des trois Magnéto Torquers. De manière similaire, l'ajout d'un deuxième bloc de génération PWM permettra le pilotage des Reaction Wheels, étant donné qu'ils fonctionnent sur le même principe.

La figure 24 décrit l'entité de ce bloc.

	A	B	C	D	E
M	Group	Name	Mode	Type	Bounds
1		clock	IN	std_ulogic	
2		reset	IN	std_ulogic	
3		Duty_Cycle	IN	DutyCycleBus	
4		Duty_Cycle_valid	IN	std_ulogic_vector	(PWMOutputNb-1:0)
5		E_PWM	OUT	std_ulogic	
6		PWM1_pos	OUT	std_ulogic	
7		PWM1_neg	OUT	std_ulogic	
8		PWM2_pos	OUT	std_ulogic	
9		PWM2_neg	OUT	std_ulogic	
10		PWM3_pos	OUT	std_ulogic	
11		PWM3_neg	OUT	std_ulogic	

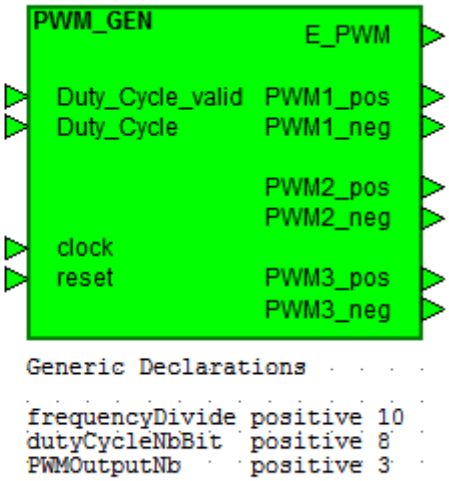


Figure 24 : Entité du bloc de génération de PWM

Le bus d'entrée Duty_Cycle se compose de trois signaux de huit bits contenant les valeurs de Duty cycle de chaque sortie PWM, codées en compléments à deux. Le signal d'entrée Duty_Cycle_valid permet quand à lui d'activer ou non les sorties, chaque bit correspondant à un signal PWM.

La sortie E_PWM permet d'activer l'alimentation pour les ponts-H tandis que les autres sorties sont connectées aux entrées de ces ponts-H.

8.1.2 Simulation

La figure 25 montre le circuit de test du bloc PWM _Gen, tandis que la figure 26 montre le résultat de la simulation. La fréquence de sortie a été configurée sur 200kHz.

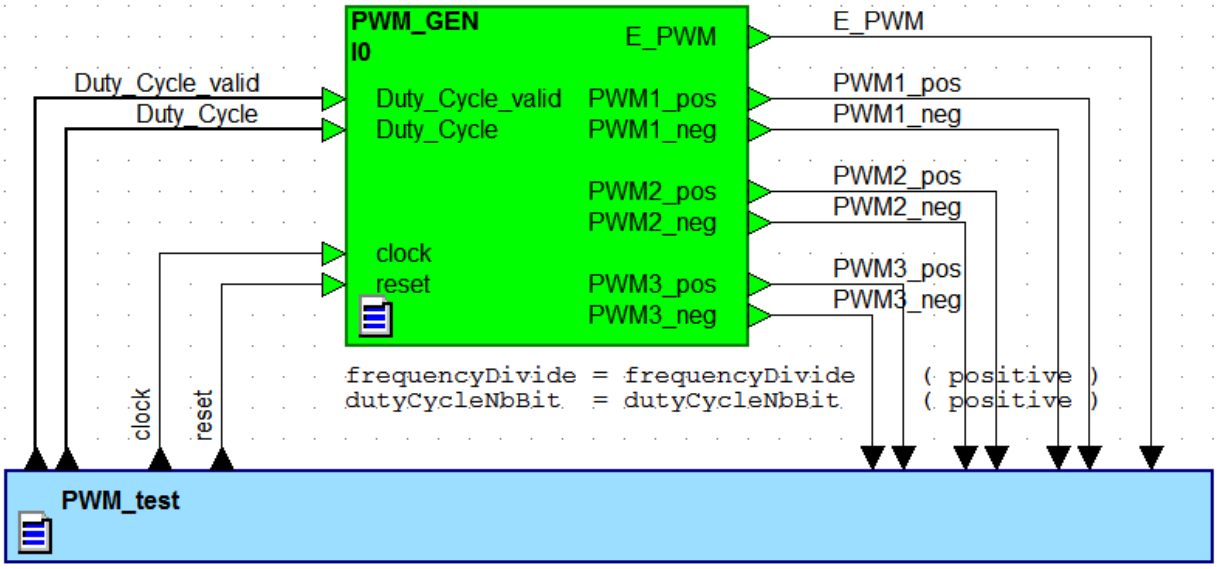


Figure 25 : Configuration de test du bloc de génération de PWM

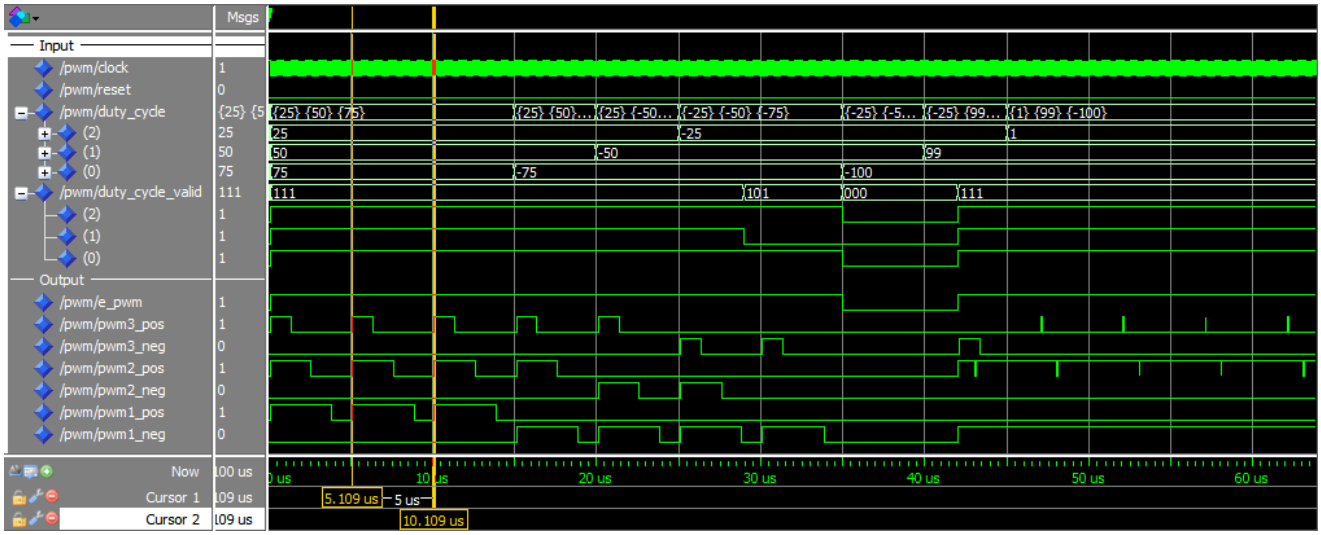


Figure 26 : Simulation 1 du bloc de génération de PWM

La simulation prouve que le bloc est fonctionnel, lorsque des valeurs positives sont écrites sur les entrées ce sont bien les sorties positives qui sont activées et inversement. Les duty cycles correspondent bien aux valeurs inscrites et la fréquence des PWM est correcte. La différence des deux curseurs donne la période qui est de 5us, ce qui donne bien une fréquence de 200kHz. Il est possible de remarquer que la sortie e_pwm est mise à zéro lorsque qu'aucun des PWM n'est activé, afin de déclencher l'alimentation lorsque cela n'est pas nécessaire.

8.2 Bloc de contrôle de l'AD

8.2.1 Description

Dans le travail précédent, un bloc de contrôle de l'AD a déjà été développé, celui-ci permet de former des trames SPI pour convertir un signal sur un des canaux du convertisseur dont la valeur doit être passée en entrée du bloc.

Comme chaque Sun-Sensor possède quatre signaux qui doivent être convertis, donc quatre canaux sur le convertisseur, l'idée est de développer un nouveau bloc qui va s'occuper de la gestion des multiplexeurs et du choix du canal à convertir. La figure 27 montre l'entité de ce bloc.

	A	B	C	D	E
M	Group	Name	Mode	Type	Bounds
1		clock	IN	std_ulogic	
2		reset	IN	std_ulogic	
3		SS_Enable	IN	std_ulogic_vector	(NbSS-1:0)
4		AD_Convert	OUT	std_ulogic	
5		AD_Channel	OUT	std_ulogic_vector	(ChannelNbBit-1:0)
6		AD_Converted_Data	IN	std_ulogic_vector	(adDataNbBit-1:0)
7		AD_Convert_done	IN	std_ulogic	
8		E_AVCC	OUT	std_ulogic	
9		E_SS_Alim	OUT	std_ulogic	
10		SS_MUX_A0	OUT	std_ulogic	
11		SS_MUX_A1	OUT	std_ulogic	
12		SS_MUX_A2	OUT	std_ulogic	
13		EnProcessing	OUT	std_ulogic	
14		S1	OUT	unsigned	(SSValueNbBit-1:0)
15		S2	OUT	unsigned	(SSValueNbBit-1:0)
16		R1	OUT	unsigned	(SSValueNbBit-1:0)
17		R2	OUT	unsigned	(SSValueNbBit-1:0)
18		SSNum	OUT	unsigned	(NbSS-1:0)

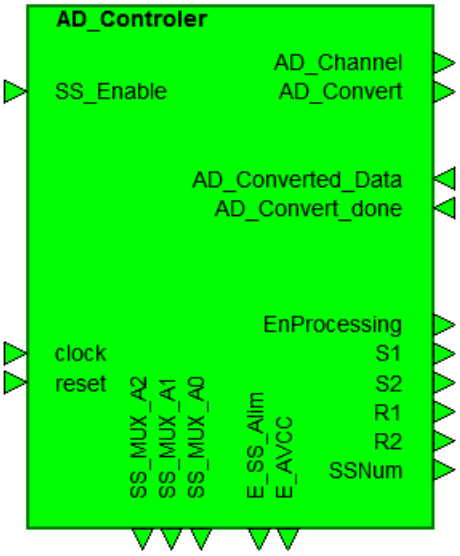


Figure 27 : Entité du bloc de contrôle du convertisseur AD

Il suffit de lui indiquer en entrée quels Sun-Sensors doivent être pris en compte pour les conversions et le bloc s'occupe de configurer les multiplexeurs en conséquence et de demander des conversions sur les quatre canaux pour obtenir respectivement les valeurs S1, R1, S2 et R2.

Pour sélectionner les Sun-Sensors qui doivent être activés, il suffit de mettre au niveau logique '1' les bits correspondant sur le signal d'entrée SS_Enable, pour exemple le bit 0 correspond au premier Sun-Sensors, le bit 1 au deuxième etc...

La sortie AD_Convert permet de lancer une conversion sur l'AD sur le canal dont la valeur est contenue sur le signal AD_Channel. Lorsque la conversion est terminée, le résultat est délivré sur le signal AD_Converted_Data, et le signal AD_Convert_done est mis à '1'. Les sorties SS_MUX_Ax pilotent les multiplexeurs, permettant de choisir les signaux qui seront sur le convertisseur parmi les six Sun-Sensors. Les sorties E_AVCC et E_SS_Alim sont connectées sur les entrées d'activation des alimentations pour les Sun-Sensors, et permettent ainsi de couper l'alimentation pour cette partie du circuit lorsqu'elle n'est pas utilisée.

Les autres sorties sont connectées au bloc de calcul des angles. En effet les valeurs reçues du convertisseur doivent être traitées pour obtenir les valeurs d'angles que forme le soleil avec les capteurs et cela est calculé dans un bloc prévu spécifiquement pour cette tâche. Cependant, pour éviter de devoir calculer inutilement pour chaque capteur des valeurs d'angles, seules les valeurs pour le Sun-Sensors qui sera le plus éclairé seront envoyées plus loin.

Comme certaines actions dans ce bloc doivent être effectuées successivement, il a été décidé d'utiliser une machine d'état, celle-ci est visible en figure 28. Dans le premier état sWaitEnable, les bits du signal d'entrée SS_Enable sont parcouru pour savoir si un des Sun-Sensors est activé. Dès que c'est le cas, la machine passe dans l'état sChannelSet. Dans celui-ci est effectué le choix du canal à convertir, pour cela un compteur allant de 0 à 3 est utilisé. D'une fois que le choix est fait l'état sStartAdConvert est atteint et comme son nom l'indique permet simplement d'activer la conversion. Ensuite la fin de la conversion est attendue dans l'état sAdWaitResponse, le résultat est stoqué et si les quatre canaux ont été convertis alors la machine repasse dans l'état initial, sinon la machine continue dans l'état sChannelSet où le compteur de canaux est incrémenté et ainsi de suite.

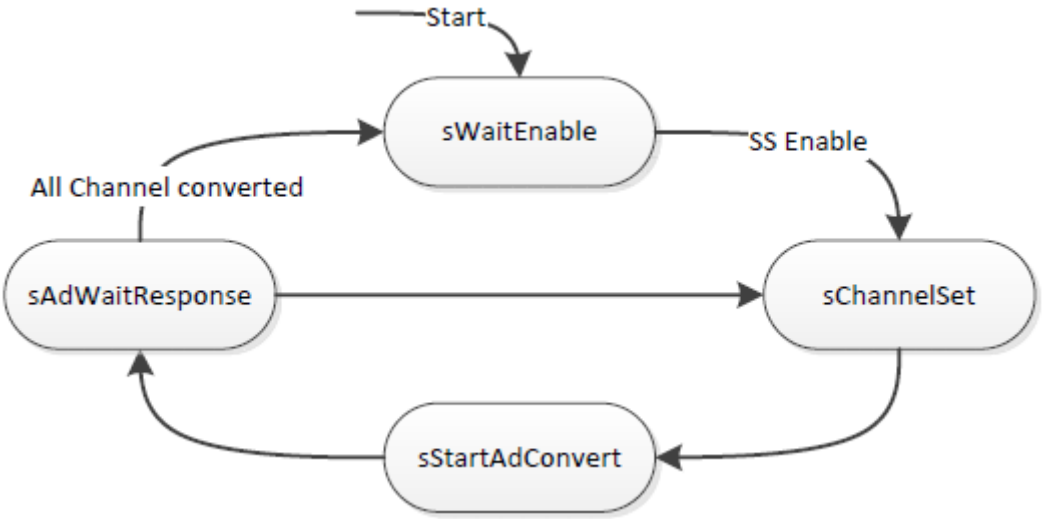


Figure 28 : Machine d'état du bloc de contrôle du convertisseur AD

En plus de cette machine d'état, un autre processus du bloc s'occupe de parcourir les valeurs précédemment stoquée afin de déterminer laquelle est la plus haute, et ainsi de déterminer quel Sun-Sensor est le plus proche de la source lumineuse, ou du soleil. Ce processus est démarré uniquement lorsque la machine est dans l'état sWaitEnable et que le compteur des Sun-Sensors atteint la valeur 0, indiquant ainsi que tous les capteurs ont été mesurés. D'une fois que la plus grande valeur est déterminée, les quatre signaux du Sun-Sensors correspondant sont envoyés au bloc de calcul des angles.

8.2.2 Simulation

La figure 29 montre le circuit de test pour le bloc AD_Controler, il est directement connecté sur le bloc adCom précédemment développé qui pour rappel permet de faire l'interface entre le convertisseur et le reste du circuit.

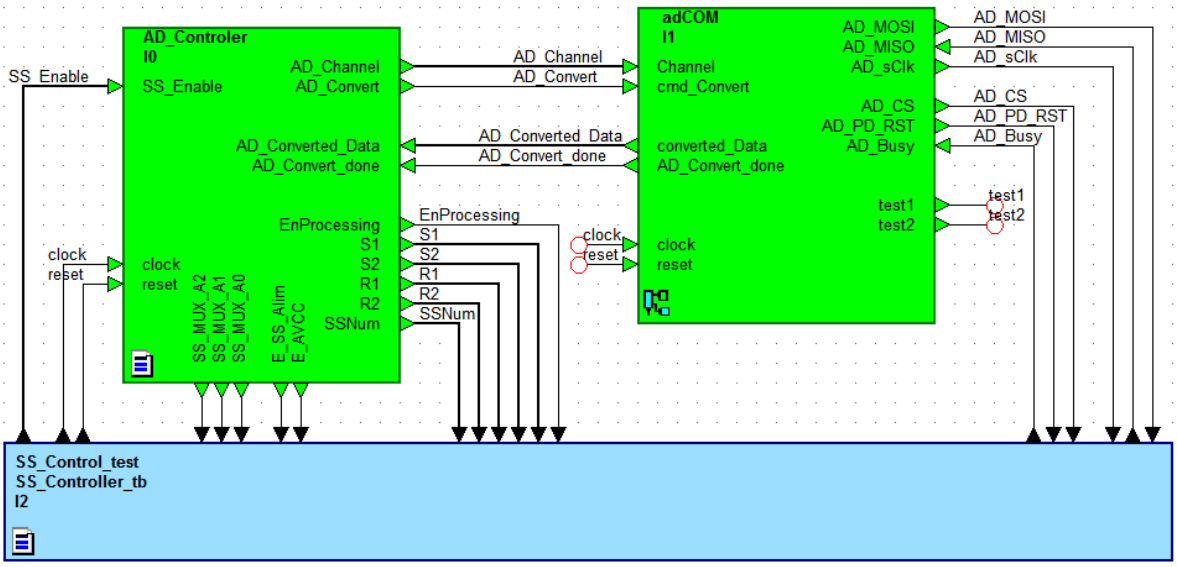


Figure 29 : Configuration de test pour le bloc de contrôle du convertisseur AD

La figure 30 montre une première simulation effectuée sur ce banc de test. Sur celle-ci, un seul Sun-Sensor est activé en entrée. Le signal CntSS permet de passer en revue les différents bits du signal d'entrée jusqu'à trouver un état haut. Dès que cela est fait le numéro du Sun-Sensor est stoqué dans currentSS, le canal à convertir est déterminé (cntChannel) et la conversion est démarrée (ad_convert). Etant donné que les Sun-Sensors sont connectés sur les canaux 3 à 6 du convertisseur, la valeur de cntChannel est simplement incrémentée de 3 avant de la passer en paramètre du bloc AdCom (ad_channel). A la fin de la première trame les valeurs des multiplexeurs sont modifiées, étant donné que la conversion va se faire réellement pendant l'envoi de la deuxième trame, le canal est incrémenté et le signal previousSS prend la valeur de currentSS. A la fin de la deuxième trame le compteur de canal est de nouveau incrémenté et previousSS2 prend la valeur de previousSS. Ce signal permettra de connaitre le numéro du Sun-Sensor dont les valeurs ont été converties et renvoyées durant la troisième trame.

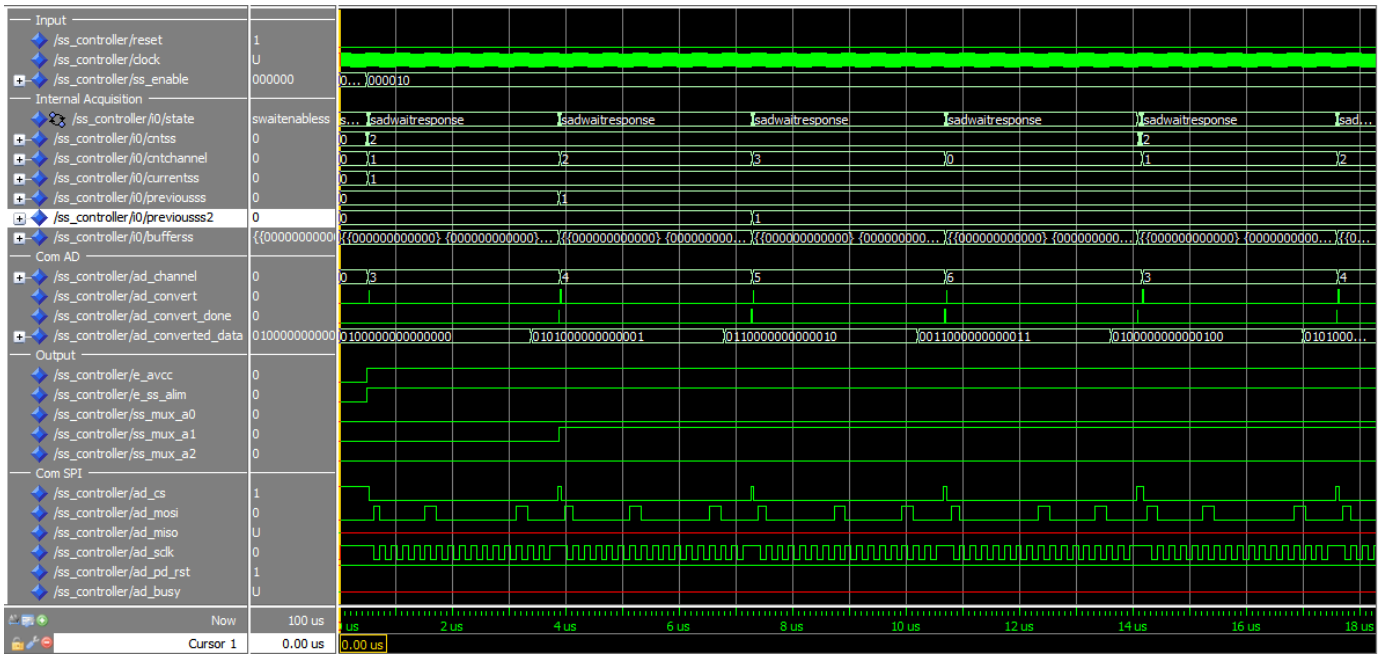


Figure 30 : Simulation 1 du bloc de contrôle du convertisseur AD

La figure 31 contient une deuxième simulation et permet de démontrer le processus de sauvegarde des données renvoyées par l'AD. Ces dernières sont stoquées dans un buffer à deux dimensions nommé bufferSS. Celui-ci est divisé en six parties pour chaque capteur et subdivisé en quatre pour les quatre valeurs de chaque capteur : S1, R1, S2 et R2. Pour ce test les valeurs du signal ad_converted_data ont été générées étant donné que l'on ne dispose pas des véritables valeurs du convertisseur. Les quatre MSB de ce signal contiennent la valeur du canal converti tandis que les 12 suivants la valeur en elle-même. Il est possible de remarquer que les valeurs sont correctement stoquées aux bons endroits dans le buffer, selon le numéro du canal et du capteur.

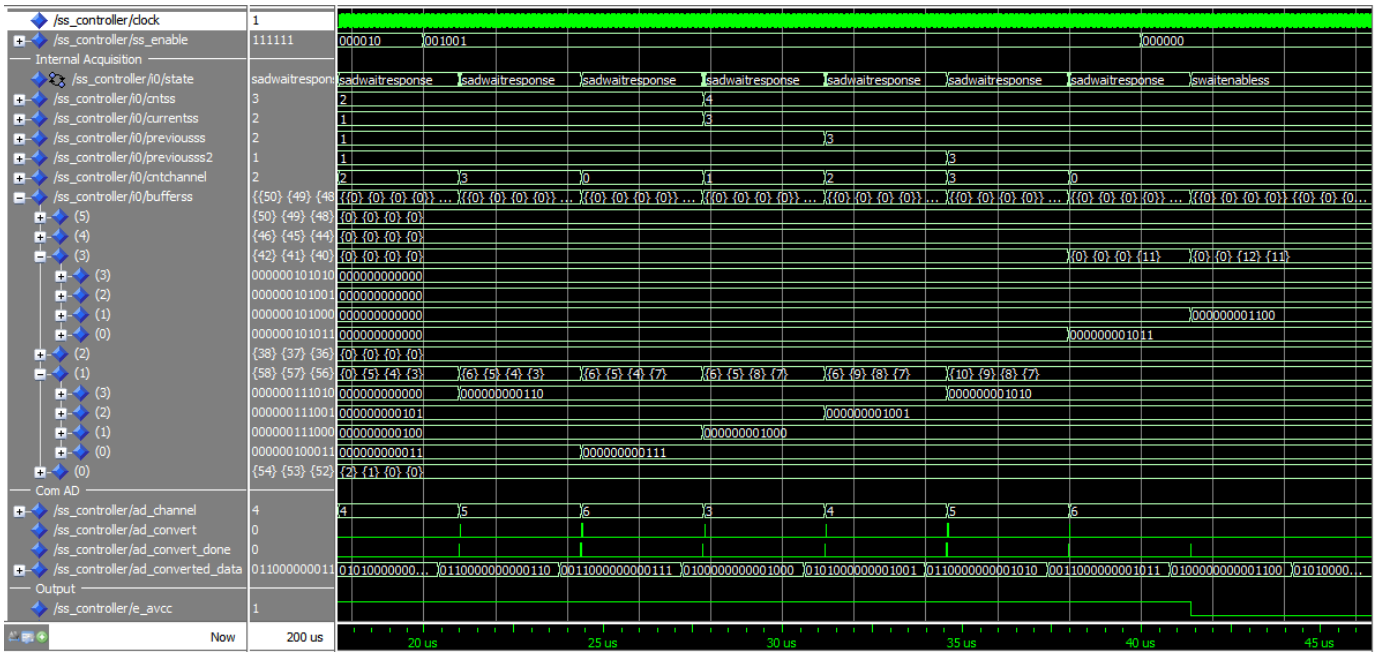


Figure 31 : Simulation 2 du bloc de contrôle du convertisseur AD

Enfin, une dernière simulation visible en figure 32 permet de valider le contrôle du bloc de calcul des angles ainsi que le fonctionnement global des conversions et des multiplexeurs. En effet le signal d'activation du calcul des angles (enProcessing) ne s'active que lorsque cntSS prend la valeur '0', c'est-à-dire chaque fois que tous les Sun-Sensors actifs ont été mesurés.

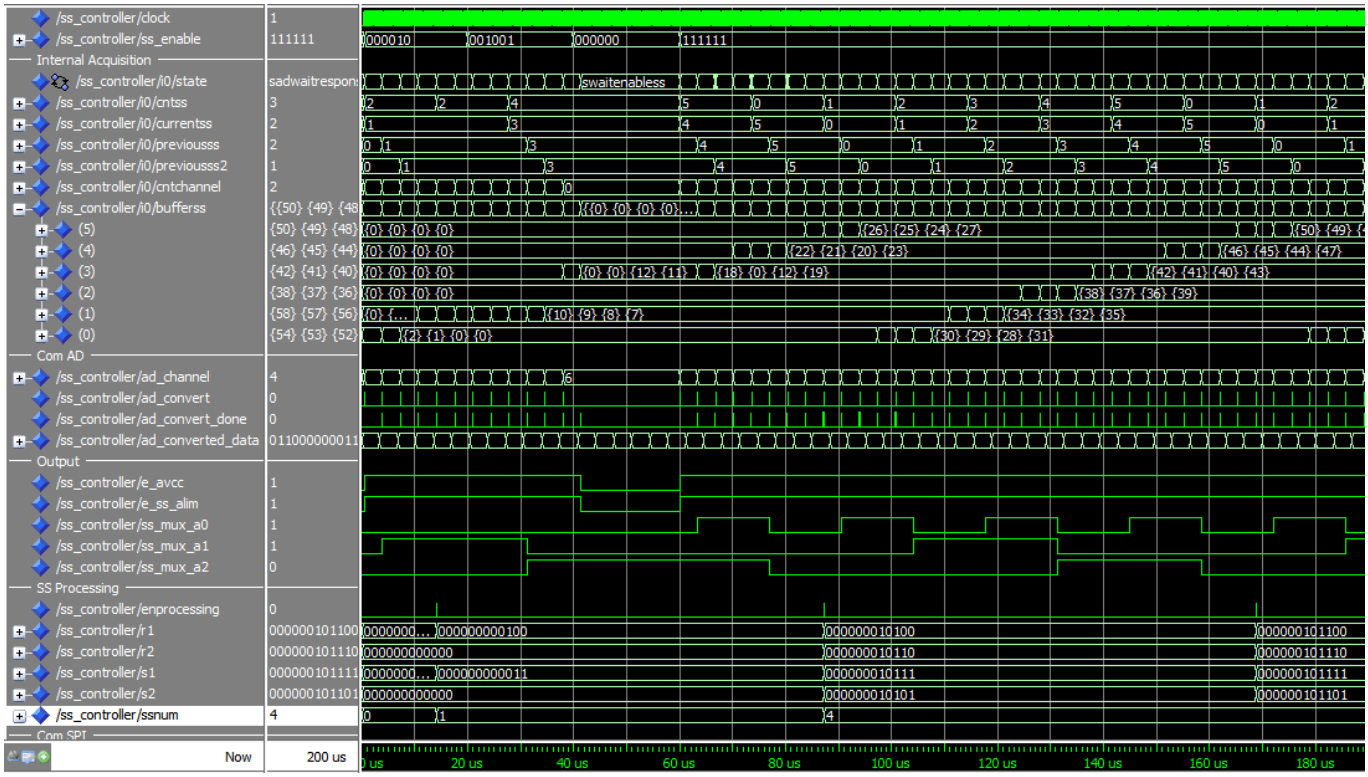


Figure 32 : Simulation 3 du bloc de contrôle du convertisseur AD

8.3 Bloc de calcul des angles des Sun-Sensors

8.3.1 Description

Ce bloc permet de calculer les valeurs d'angles que fait la source lumineuse avec le Sun-Sensors en utilisant les valeurs fournies par ce dernier, soit S1, R1, S2 et R2. L'entité de ce bloc est reprise sur la figure 33.

	A	B	C	D	E
	Group	Name	Mode	Type	Bounds
1		EnIn	IN	std_ulogic	
2		S1	IN	unsigned	(SSValueNbBit-1:0)
3		S2	IN	unsigned	(SSValueNbBit-1:0)
4		R1	IN	unsigned	(SSValueNbBit-1:0)
5		R2	IN	unsigned	(SSValueNbBit-1:0)
6		NumSSIn	IN	unsigned	(NbSS-1:0)
7		EnOut	OUT	std_ulogic	
8		Angle1	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
9		Angle2	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
10		NumSSOut	OUT	unsigned	(NbSS-1:0)
11		clock	IN	std_ulogic	
12		reset	IN	std_ulogic	

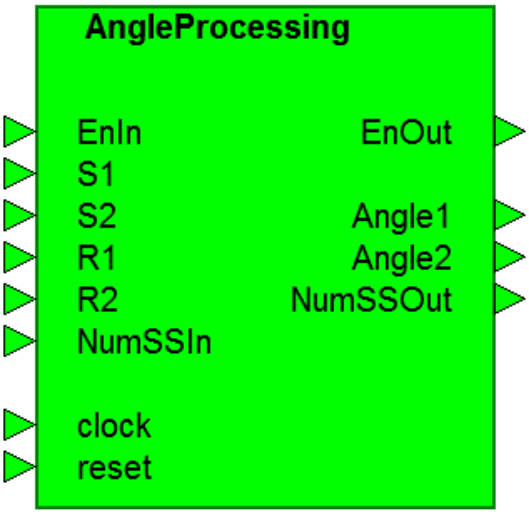


Figure 33 : Entité du bloc de calcul d'angle

Mise à pars le signal d'activation du bloc (enIn) et les quatre valeurs des Sun-Sensors, ce bloc comprend aussi une entrée pour le numéro du capteur, ce qui permet de connaitre la face du cube selon laquelle les angles seront mesurés.

Les diverses opérations que doivent subir ces valeurs d'entrées pour en retrancher les angles sont visibles sur la figure 34.

```
Out1 = (S1-1.25/R1)
Out2 = (S2-1.25/R2)
Angle1 = c0 + Out1*c1 + Out1^2*c2 + Out1^3*c3 + Out1^4*c4
Angle2 = d0 + Out2*d1 + Out2^2*d2 + Out2^3*d3 + Out2^4*d4
```

Figure 34 : Calcul des angles des Sun-Sensors [2], [10]

Comme les deux angles ont une influence mutuelle l'une sur l'autre, la mesure peut être faussée, pour palier à cela un dernier contrôle est nécessaire. Il consiste à regarder si l'angle 2 est compris entre -20° et -40° ou entre -40° et -60°, si cela est le cas, il faut modifier les coefficients 'c' pour l'angle 1 et re-effectuer le calcul de l'angle 1. De manière similaire pour l'angle 1, si celui-ci est compris entre -20° et -40° ou entre -40° et -60°, il faut modifier les coefficients 'd' cette fois ci et re effectuer le calcul de l'angle 2. Un tableau de ces coefficients est disponible en annexe 12. Pour l'instant, seuls les coefficients pour le Sun-Sensor R10 sont utilisés, vu que ces valeurs doivent toutes être converties et remplies à la main dans le code, cela demande passablement de temps. A long terme les coefficients pour les autres capteurs devraient être aussi utilisés.

Etant donné que cet algorithme utilise des nombres à virgule, il a fallut dans un premier temps créer des blocs pour effectuer les opérations de base nécessaires, soit l'addition, la soustraction, la multiplication et la division. Au préalable il a fallut déterminer sous quel format allait être stoquées les différentes valeurs. Pour cela la résolution du convertisseur AD à été calculée : les valeurs sont codés sur 12 bits, pour une valeur maximale de 3.3V, ce qui fait 3.3/4095 = 0.00080586. Cette valeur demande une infinité de bits pour être convertie exactement en binaire, mais 32 bits de décimale suffisent déjà pour obtenir une erreur de moins de 1/10^7. Afin d'obtenir une précision encore plus grande, notamment pour les très petites valeurs qui seront mise à la puissance 4, il a été décidé d'utiliser 64 bits de décimale, et 16 bits d'entier afin de pouvoir y stoquer les valeurs fournies par le convertisseur et notamment pouvoir les coder en complément à deux. Une autre solution aurait pu être d'utiliser des valeurs codées en virgule flottante, mais les opérations aurait été plus compliquées et vu que les valeurs dans cette algorithme sont toutes centrées autour d'une certaine plage, un codage en virgule flottante n'est vraiment pas nécessaire.

Une fois cela connu, il a été possible de développer ces différents blocs d'opérations de base. L'entité du premier de ces blocs est visible à la figure 35 et permet d'effectuer les opérations d'addition et de soustraction. Cinq entrées de valeurs signées sont disponibles, ce qui permet de calculer un polynôme à cinq termes, étant donné que l'algorithme a besoin de cela.

	A	B	C	D	E
M	Group	Name	Mode	Type	Bounds
1		clock	IN	std_ulogic	
2		reset	IN	std_ulogic	
3		EnIn	IN	std_ulogic	
4		Value1	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
5		Value2	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
6		Value3	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
7		Value4	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
8		Value5	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
9		EnOut	OUT	std_ulogic	
10		Result	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)

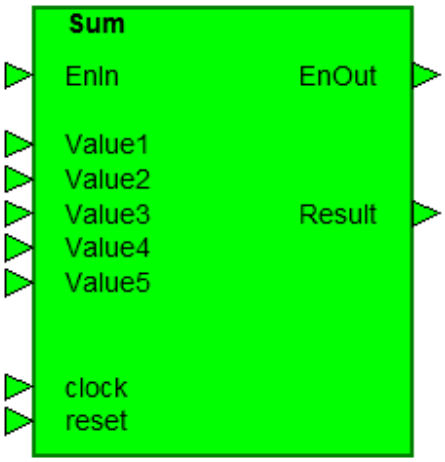


Figure 35 : Entité du bloc de sommation/soustraction

Le deuxième bloc présent à la figure 36 permet de multiplier deux valeurs signées. Il sera utilisé aussi pour effectuer les opérations de mise à la puissance.

	A	B	C	D	E
M	Group	Name	Mode	Type	Bounds
1		clock	IN	std_ulogic	
2		reset	IN	std_ulogic	
3		Value1	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
4		Value2	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
5		Result	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
6		EnIn	IN	std_ulogic	
7		EnOut	OUT	std_ulogic	

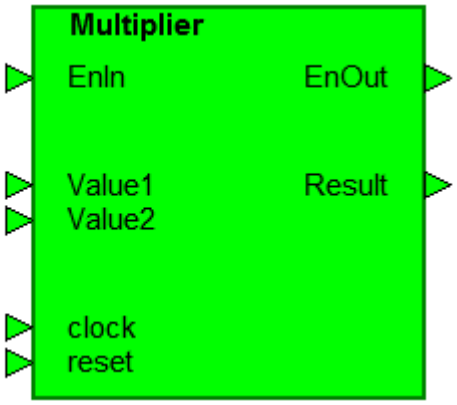


Figure 36 . Entité du bloc de multiplication

Enfin le dernier bloc permet d'effectuer l'opération de division d'un nombre non signé par un autre nombre non signé. La gestion du signe devra se faire en dehors du bloc.

	A	B	C	D	E
M	Group	Name	Mode	Type	Bounds
1		clock	IN	std_ulogic	
2		reset	IN	std_ulogic	
3		Dividend	IN	unsigned	(IntegerBitNb+FractionalBitNb-1:0)
4		Divisor	IN	unsigned	(IntegerBitNb+FractionalBitNb-1:0)
5		Quotient	OUT	unsigned	(IntegerBitNb+FractionalBitNb-1:0)
6		Rest	OUT	unsigned	(IntegerBitNb+FractionalBitNb-1:0)
7		Start	IN	std_ulogic	
8		Done	OUT	std_ulogic	

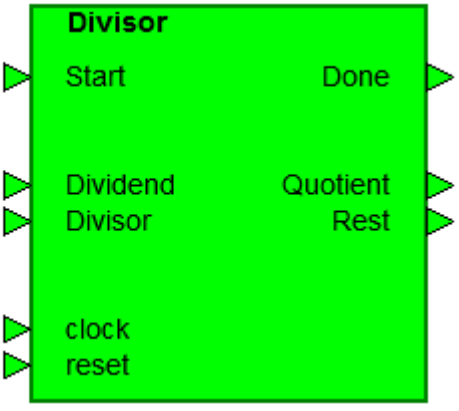


Figure 37 : Entité du bloc de division

D'une fois que ces blocs de bases ont été développés, l'algorithme à pu être mis en place. Pour cela deux méthodes sont possibles, la première consiste à placer un certain nombre de ces blocs à la chaine pour effectuer l'opération totale. La deuxième consiste à

n'utiliser qu'une fois chaque bloc dans le système et de développer un nouveau bloc qui fera office d'organe de pilotage. Ce dernier devra s'occuper de gérer les blocs de bases de telle manière à reproduire lui-même l'algorithme.

La deuxième méthode a été retenue, étant donné qu'il fallait de toute manière faire un nouveau bloc pour s'occuper de la gestion du signe du bloc de division, et que dans la première méthode la synchronisation entre les différents blocs aurait posé un problème. En effet chaque opération de base demande plus ou moins de temps de calcul, ainsi les entrées du bloc de sommation finale du polynôme à cinq terme doivent être prêtes toutes au même instant, et pourtant ces termes ne demandent pas le même nombre d'opérations.

Ainsi un nouveau bloc à été développé et est visible sur la figure 38. Celui-ci prend en entrées les signaux fournis par le bloc AD_Controler, dont les quatres valeurs pour le calcul d'angle, et donne en sortie les valeurs de ces angles ainsi que le numéro du capteur. Le reste des signaux ne sont que les connections aux blocs d'opérations.

	A	B	C	D	E
M	Group	Name	Mode	Type	Bounds
1		clock	IN	std_ulogic	
2		reset	IN	std_ulogic	
3		EnIn	IN	std_ulogic	
4		S1	IN	unsigned	(SSValueNbBit-1:0)
5		S2	IN	unsigned	(SSValueNbBit-1:0)
6		R1	IN	unsigned	(SSValueNbBit-1:0)
7		R2	IN	unsigned	(SSValueNbBit-1:0)
8		NumSSIn	IN	unsigned	(NbSS-1:0)
9		EnOut	OUT	std_ulogic	
10		Angle1	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
11		Angle2	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
12		NumSSOut	OUT	unsigned	(NbSS-1:0)
13		EnMulti	OUT	std_ulogic	
14		Val1Multi	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
15		Val2Multi	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
16		DoneMulti	IN	std_ulogic	
17		ResultMulti	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
18		EnSum	OUT	std_ulogic	
19		Val1Sum	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
20		Val2Sum	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
21		Val3Sum	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
22		Val4Sum	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
23		Val5Sum	OUT	signed	(IntegerBitNb+FractionalBitNb-1:0)
24		DoneSum	IN	std_ulogic	
25		ResultSum	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
26		EnDiv	OUT	std_ulogic	
27		Val1Div	OUT	unsigned	(IntegerBitNb+FractionalBitNb-1:0)
28		Val2Div	OUT	unsigned	(IntegerBitNb+FractionalBitNb-1:0)
29		DoneDiv	IN	std_ulogic	
30		ResultDiv	IN	unsigned	(IntegerBitNb+FractionalBitNb-1:0)

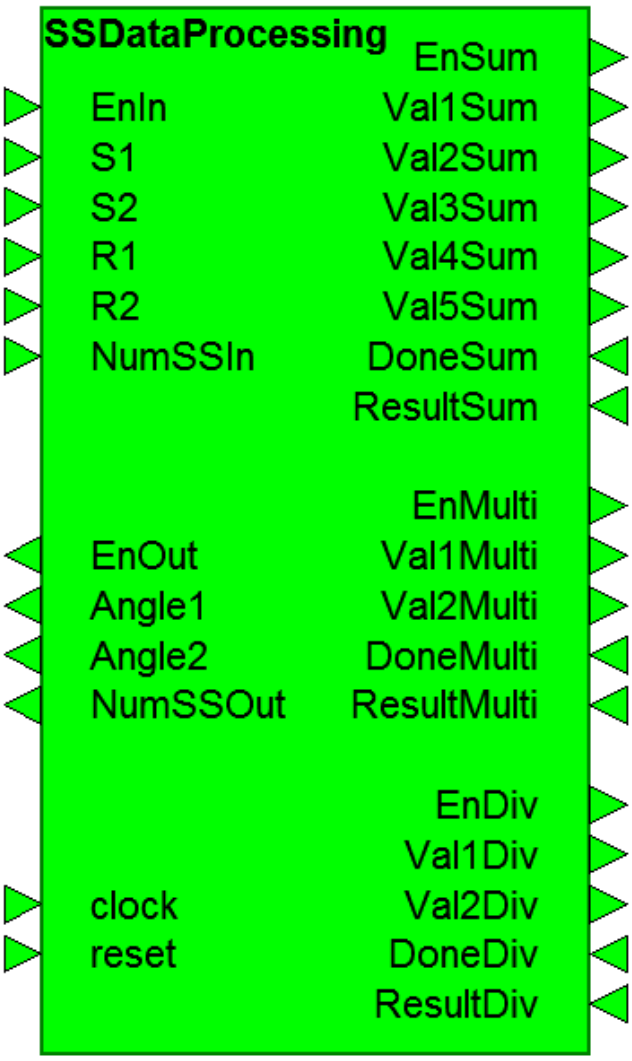


Figure 38 : Entité du bloc de gestion des opérations de base

Pour gérer la suite des opérations nécessaire à l'algorithme, une machine d'état a été développée dans ce bloc et est visible en figure 39.

La machine commence donc dans l'état sWaitNewData et y reste tant que l'ordre d'effectuer le calcul n'est pas reçu. Une fois ce dernier reçu, le système lance la première opération qui est de calculer la véritable valeur de S1 soit : valeur du convertisseur S1 * 0.00080586 et passe dans le second état sS1Process. Une fois cela fait le système re-lance la même opération mais pour R1 cette fois ci. En parallèle, et étant donné que le bloc d'addition n'est pas utilisé, le système lance aussi l'opération pour retrancher les 1.25 à S1. Une fois ces deux opérations terminées, le système peut passer dans l'état sOutProcess et ainsi lancer l'opération de division : (S1-1.25) / R1. La suite est facilement compréhensible au vue du nom des différents états. La particularité de cette machine est qu'elle parcourt deux fois la boucle avant de revenir dans l'état initiale, ceci permettant de calculer lors du premier passage, la valeur de l'angle 1 et lors du deuxième, la valeur de l'angle 2.

Hélas cette machine d'état est incomplète dans le sens où la vérification des angles et la rectification de ceux-ci n'est pas faite. Cette étape était prévue mais le temps n'a pas permis son développement et son intégration au reste du bloc.

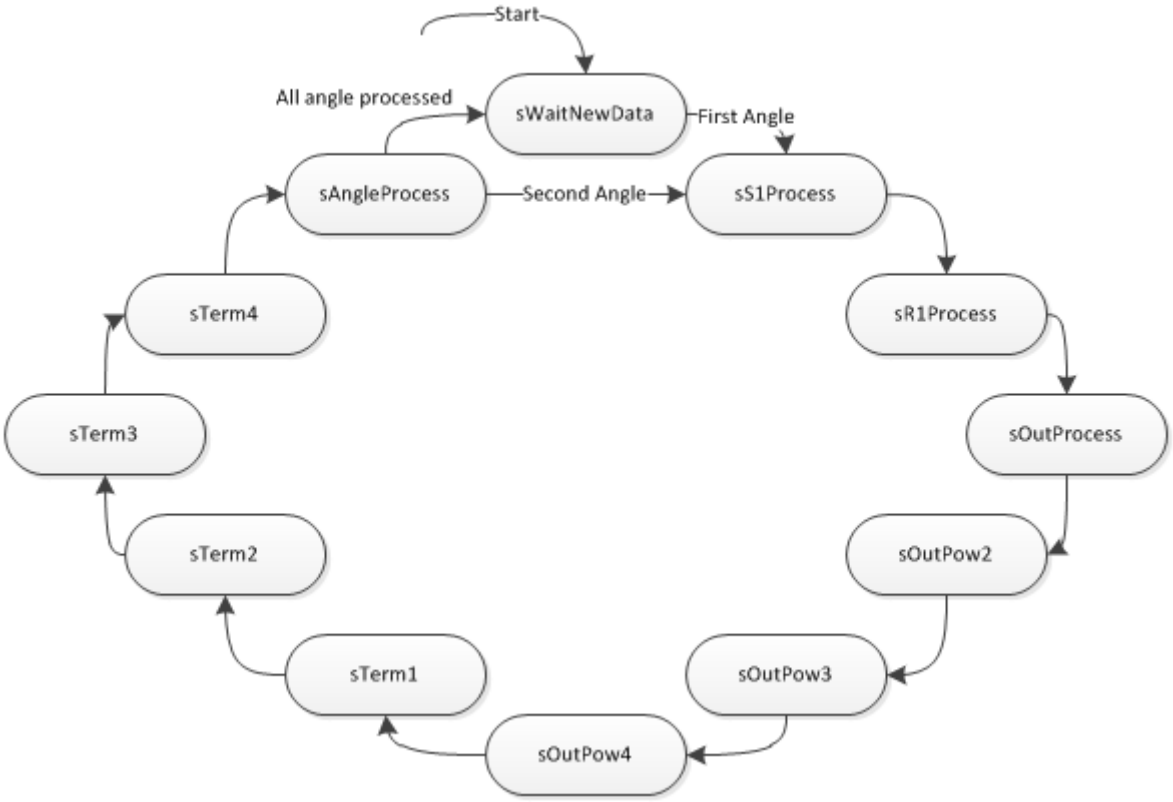


Figure 39 : Machine d'état du bloc de gestion des opérations de base

8.3.2 Simulation

Le circuit de la figure 40 permet de tester le bon fonctionnement du bloc de sommation.

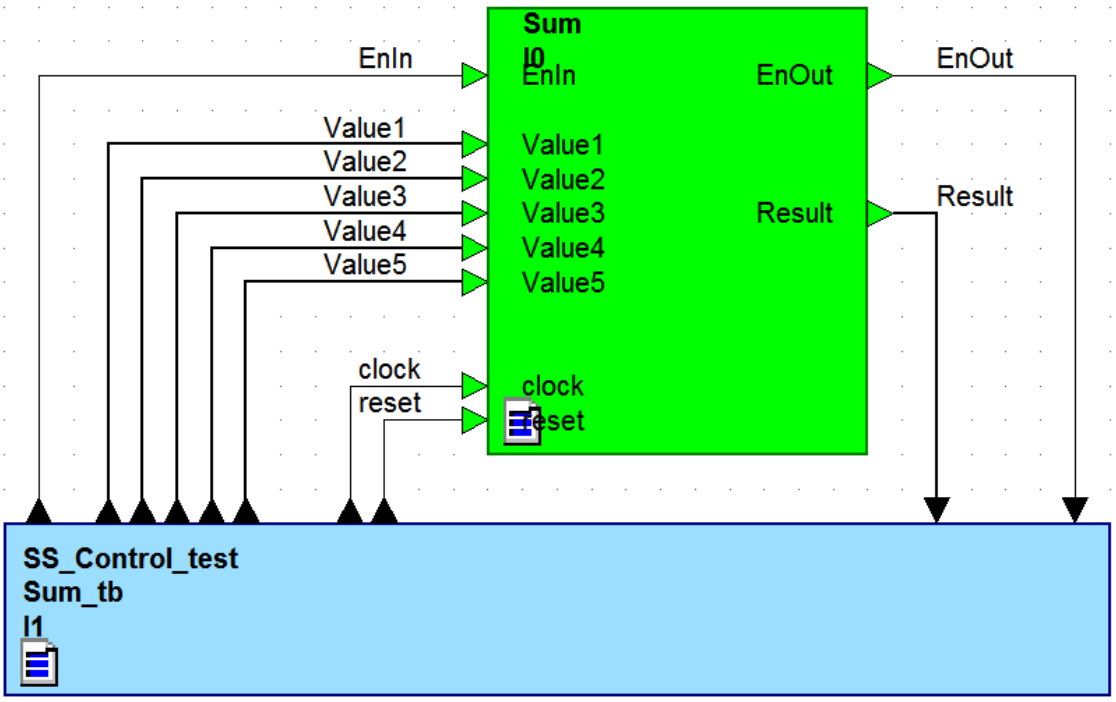


Figure 40 : Configuration de test du bloc de sommation/soustraction

La simulation correspondante est visible sur la figure 41. Deux des opérations qui ont été testées y sont montrées : -1 -1 -1 -1 -1 = ? et 0.5 – 1 = ?. Force est de constater que les résultats pour ces deux opérations sont correct, le premier donnant bien -5 en complément à deux et le deuxième -0.5.

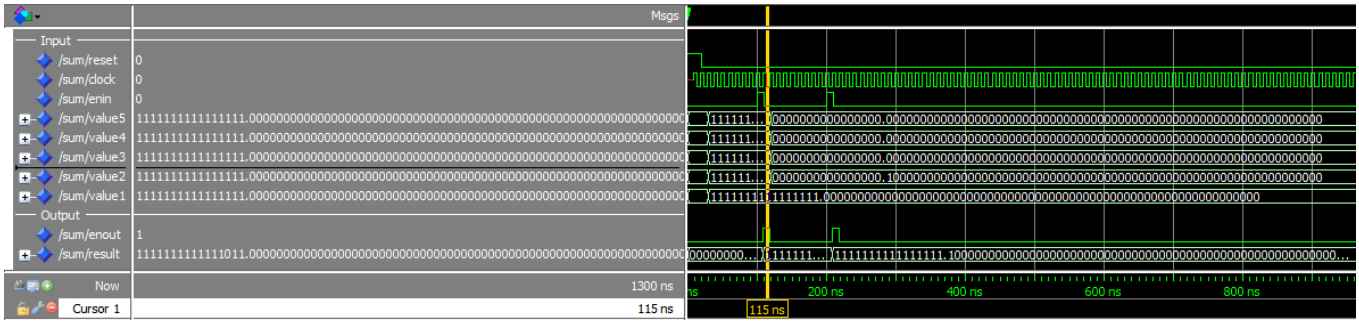


Figure 41 : Simulation 1 du bloc de sommation/soustraction

La configuration du test pour le bloc de multiplication est visible en figure 42.

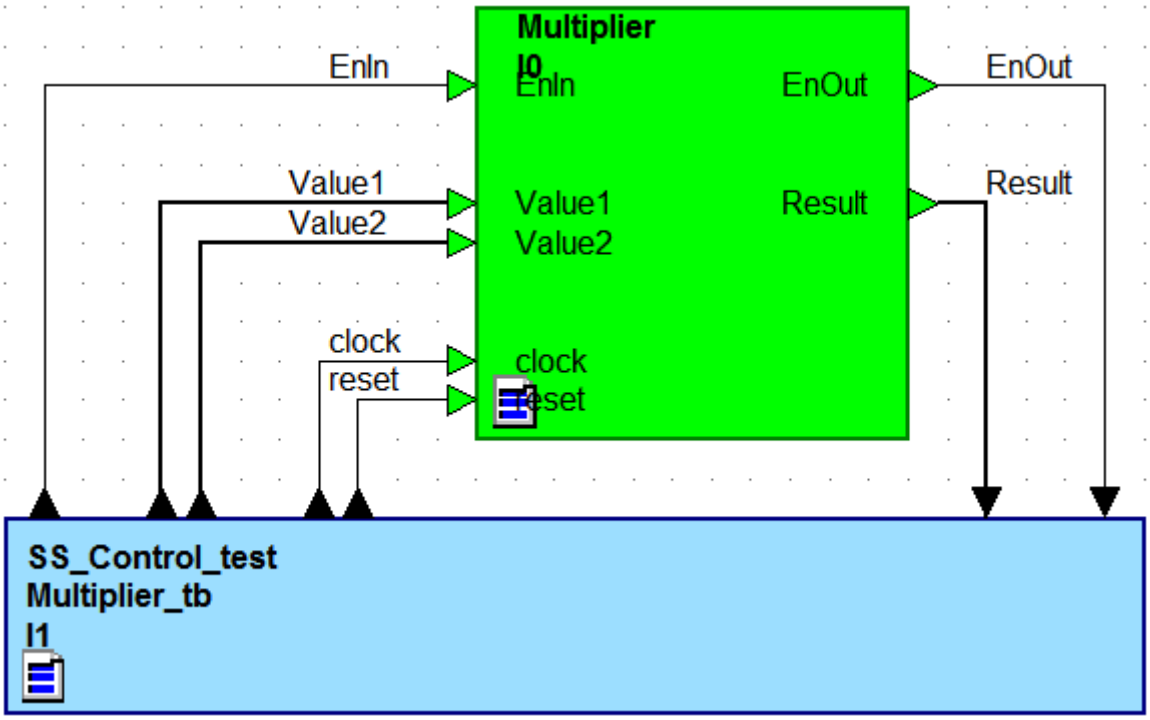


Figure 42 : Configuration de test du bloc de multiplication

Le résultat de la simulation en figure 43 montre trois fois la même fenêtre, mais avec le curseur placé sur des endroits différents à chaque fois afin d'y voir les valeurs des différents signaux. La première opération est la suivante : 0.000805 * 1, afin de vérifier qu'une valeur *1 reste inchangée. La deuxième consiste à tester le cas où le convertisseur renverrait sa valeur maximale soit : 4095 * 0.000805. Le résultat correct et retourné est bien de 3.296475. Le dernier test permet d'effectuer une opération avec un nombre négatif soit : 0.000805*(-1), le résultat est la aussi correct.

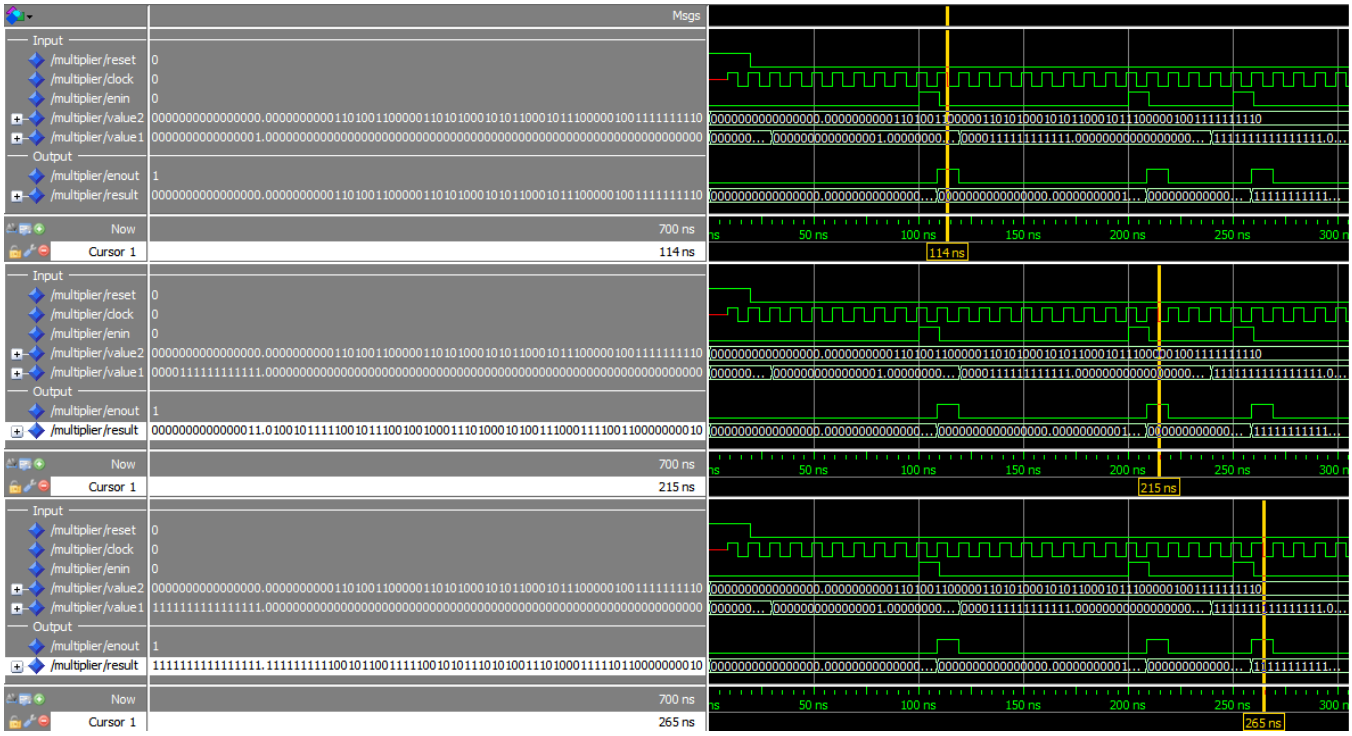


Figure 43 : Simulation 1 du bloc de multiplication

Le dernier bloc d'opération a été testé selon la figure 44.

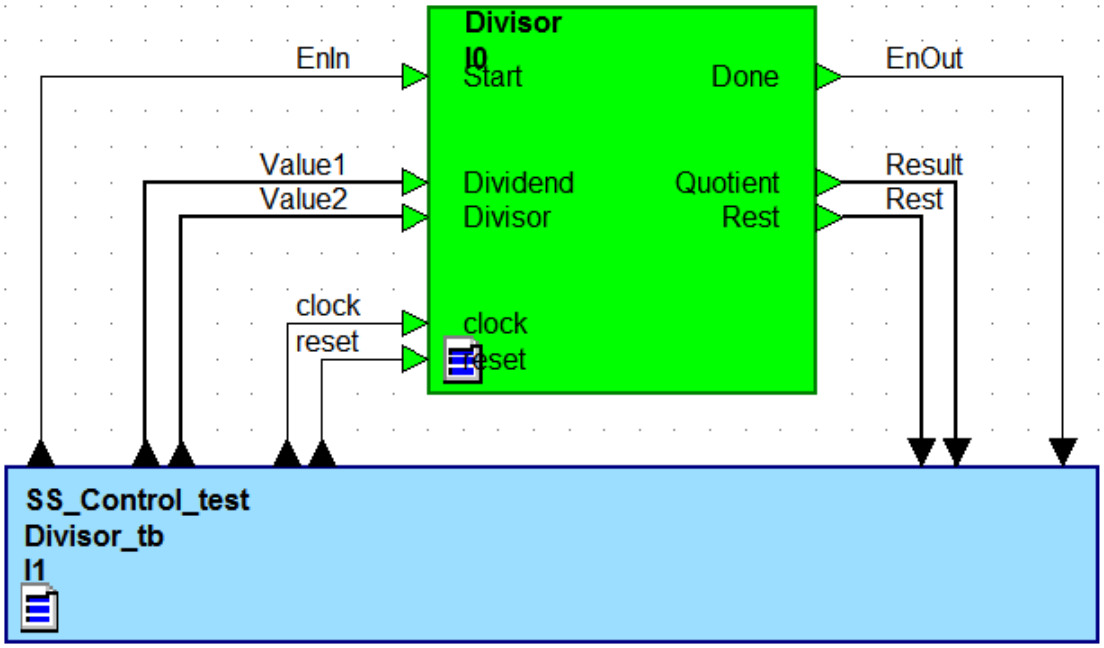


Figure 44 : Configuration de test du bloc de division

Sur la figure 45, seulement deux opérations ont été effectuées. La première est la suivante : 10 / 3. On remarque que le quotient retourné est correct, mais pas mis en forme selon les autres valeurs, cela devra être fait dans le bloc de commande, au même titre que la gestion des nombres négatifs. Etant donné que le résultat est toujours donné sous la forme d'un quotient entier et d'un reste, l'astuce pour obtenir un quotient à virgule est de décaler vers la gauche du nombre de décimale souhaitée la valeur du dividende.

Ce qui est fait pour le deuxième test, le dividende vaut toujours 10 mais cette fois ci décalé de 10 bits vers la gauche. Le résultat contiendra donc 10 bits de décimale, ce qui donne selon la simulation 3.3330078125.

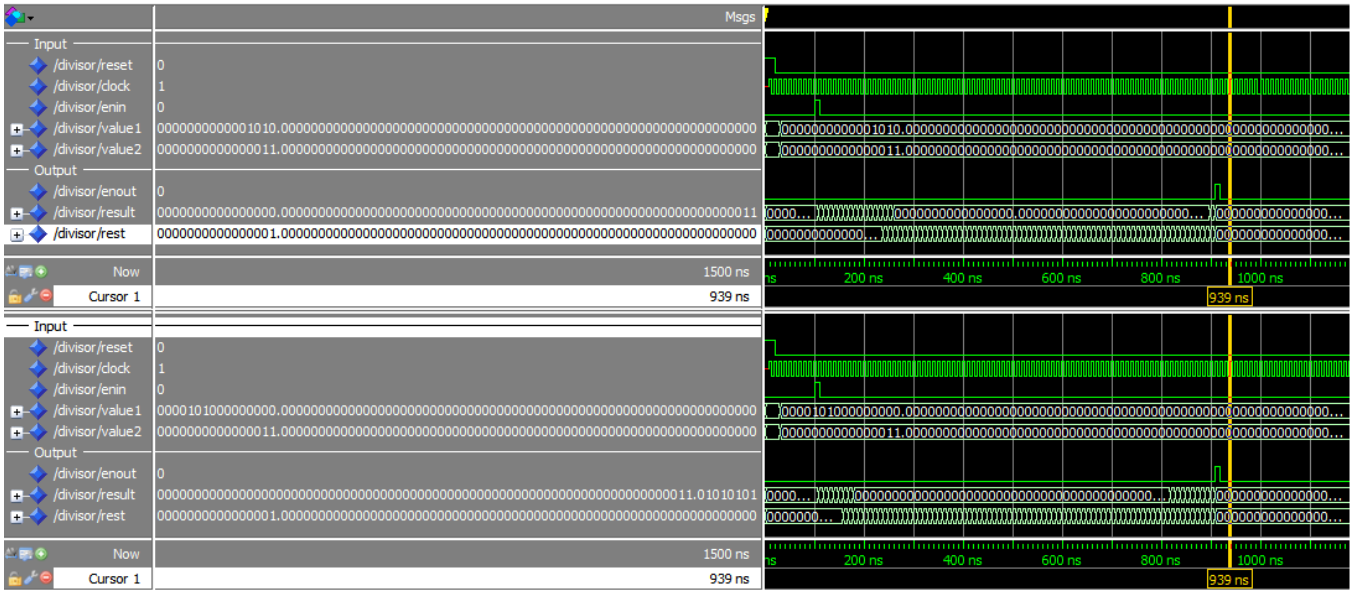


Figure 45 . Simulation 1 du bloc de division

D'une fois que ces trois blocs ont été testés, il a été possible de pouvoir tester le circuit de calcul d'angle complet, visible en figure 46.

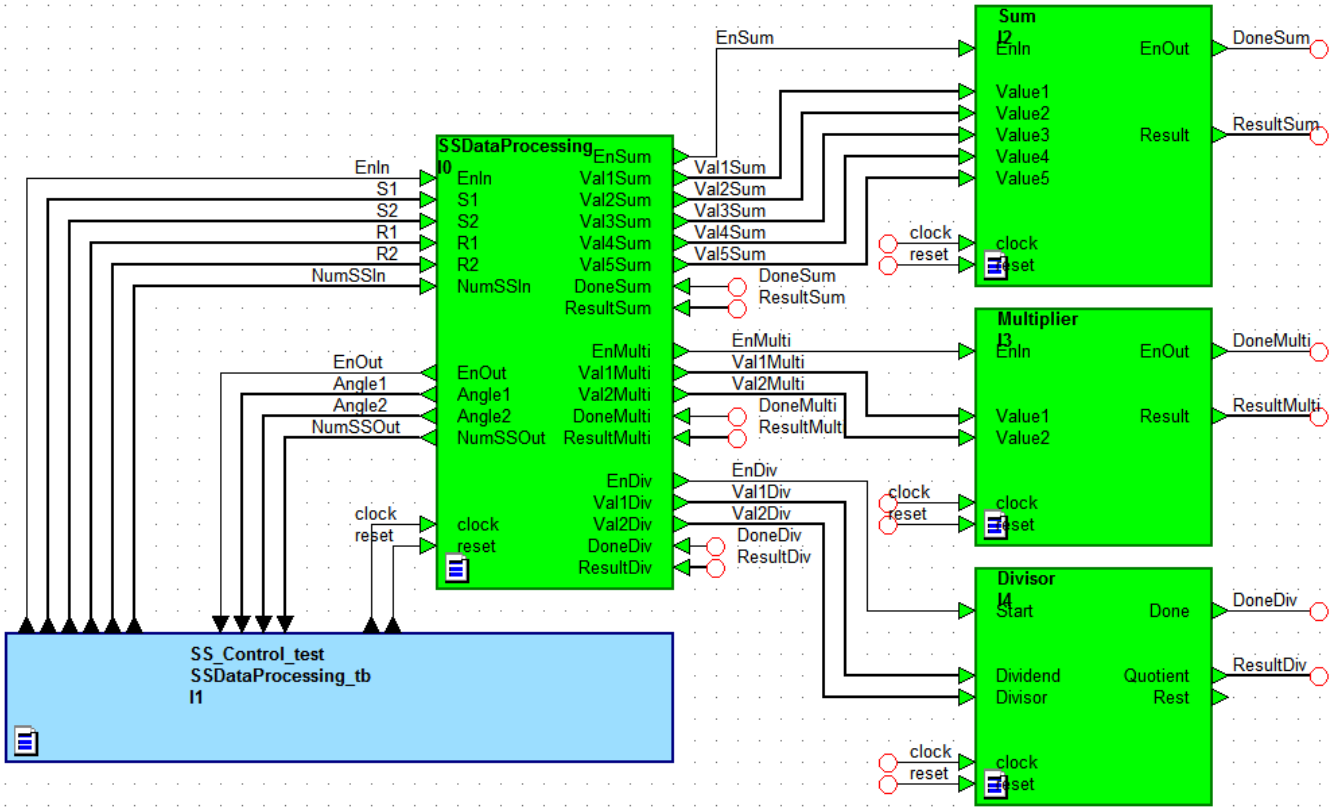


Figure 46 : Configuration de test pour le calcul des angles

La simulation de ce dernier est visible en figure 47. Quatre valeurs plausibles ont été entrée manuellement soit : S1 = 0x6C9 = 1.4V, R1 = 0x215 = 0.43, S2 = 0x45D = 0.9 et R2 = 0x1B2 = 0.35. Comme vu précédemment sur la figure 10, ces valeurs devraient donner pour l'angle 1 environ 15° et pour l'angle 2 environ -39°. Sur la simulation, les valeurs retournées valent pour l'angle 1 : 13.9877° et pour l'angle 2 : -44.9687°. Ce qui est plutôt proche du résultat escompté, validant de ce fait ce bloc.

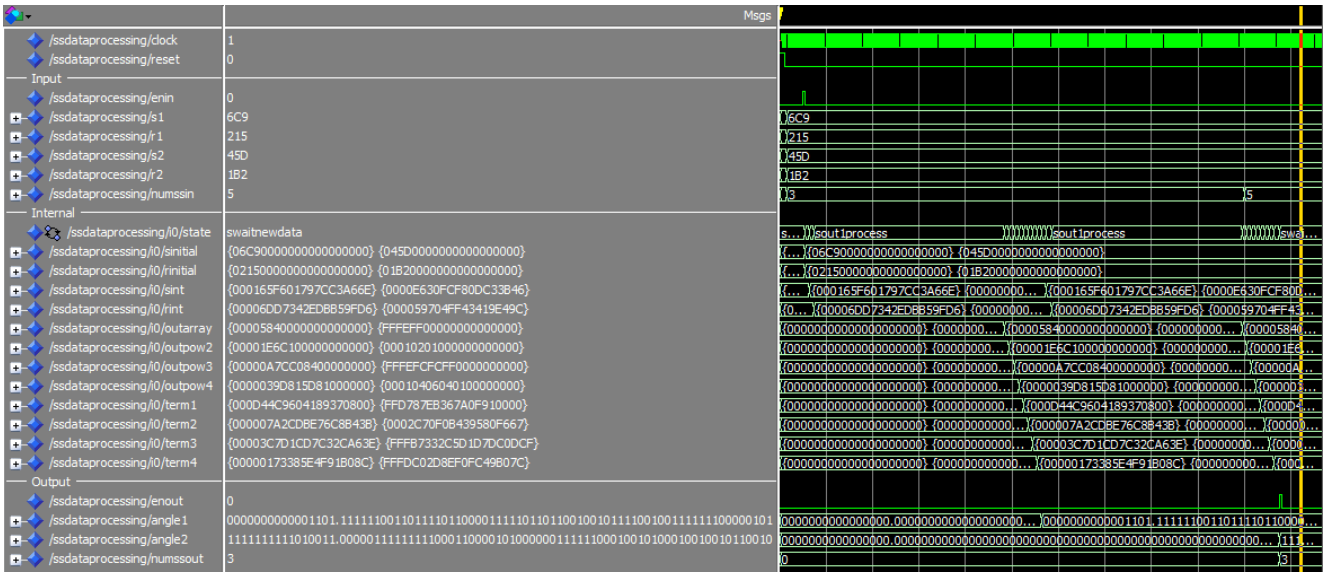


Figure 47 : Simulation 1 du calcul des angles

8.4 Bloc de contrôle du bus I2C

8.4.1 Description

Plusieurs périphériques sont connectés au bus I2C. Pour les piloter, un bloc de contrôle a donc été développé et est visible en figure 48. La HES disposait déjà d'un bloc contenant une fifo pour l'I2C, donc le nouveau bloc devra pouvoir s'y connecter pour simplifier le développement.

	A	B	C	D	E
M	Group	Name	Mode	Type	Bounds
1		clock	IN	std_ulogic	
2		reset	IN	std_ulogic	
3		DataOut	OUT	std_ulogic_vector	(DataFifoBitNb-1:0)
4		DataIn	IN	std_ulogic_vector	(DataFifoBitNb-1:0)
5		FifoFull	IN	std_ulogic	
6		FifoEmpty	IN	std_ulogic	
7		FifoWr	OUT	std_ulogic	
8		FifoRd	OUT	std_ulogic	
9		Gyro_Enable	IN	std_ulogic_vector	(NbGyro-1:0)
10		MM_Enable	IN	std_ulogic_vector	(NbMM-1:0)
11		GPS_Enable	IN	std_ulogic	
12		ST_Enable	IN	std_ulogic	
13		Gyro_Value	OUT	GyroArrayValue	
14		MM_Value	OUT	MMArrayValue	
15		GPS_Value	OUT	std_logic_vector	(15:0)
16		GPS_Reset_N	OUT	std_ulogic	

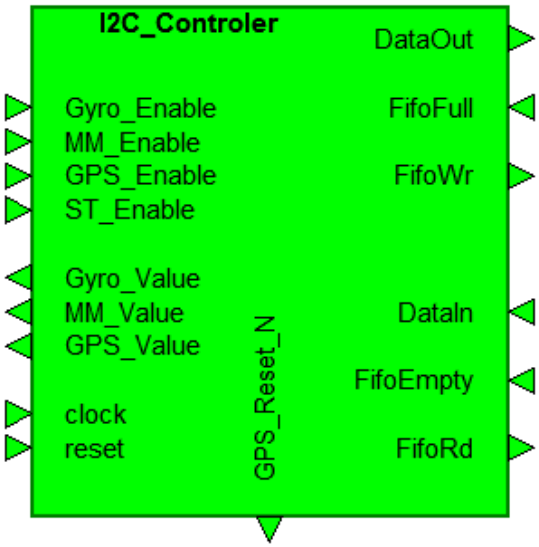


Figure 48 : Entité du bloc de gestion des périphériques I2C

Ce nouveau bloc utilise quatre entrées, qui permettent de déterminer quels capteurs doivent être interrogés, ainsi que trois sorties contenant les différentes valeurs pour ces capteurs. Il n'y a pas de signal de sortie pour les valeurs de la caméra, car celle-ci débite ces données non pas sur le bus I2C, mais sur d'autre broche du périphérique. Ces données seront récupérées directement dans le bloc principale de la FPGA soit le bloc de Code/Decode Trame. Ainsi, la caméra est simplement configurée ici par le bus I2C.

Comme précédemment, une machine d'état a été implémentée dans ce bloc et est représentée en figure 49. Celle-ci démarre dans l'état sInitPeriph, dans cet état les différentes trames de configuration pour initialiser certains des capteurs sont envoyées. En effet les capteurs comme le L3G4200D sont configurés pour démarrer en mode idle, ces trames sont destinées à les passer en mode normal. Une fois cela fait, la machine passe dans l'état sWaitEnable et passe en revue les entrées d'enable jusqu'à trouver un périphérique activé.

A partir de la, deux choix sont possibles, si c'est la caméra qui demande à être activée, alors la machine d'état va suivre le chemin vert, si c'est un autre périphérique, c'est le chemin rouge qui va être emprunté. Dans ce dernier cas, la machine passe dans l'état sSendTrame. Là, les valeurs pour les écritures dans les registres sont envoyées dans la Fifo, puis la machine d'état passe dans l'état sWaitFifo. Comme son nom l'indique, cet état sert à attendre que la fifo envoie la trame et renvoie les valeurs fournies par le capteur. Lorsque tous les paquets de données ont été reçus, la machine repasse dans l'état sWaitEnable.

Pour initialiser la caméra, de nombreux registres doivent être configuré. Cela est normalement effectué au démarrage par le microcontrôleur intégré à la carte de la caméra mais pour des raisons inconnues il ne le fait pas. Ainsi la séquence d'initialisation est faite par la FPGA. Pour cela un tableau à deux colonnes a été utilisé, la première colonne contenant l'adresse du registre à écrire tandis que la deuxième contenant la valeur à écrire. L'état sConfigTrameCam sert simplement à parcourir ce tableau pour préparer les trames à envoyer. Ensuite les états sSendTrame et sWaitFifo sont parcouru jusqu'à ce que tous les paquets soient envoyés. Parfois il est nécessaire d'attendre quelque temps entre deux trames de configuration pour la caméra, cela est fait dans l'état sDelay. Ces delays sont représentés dans le tableau par une ligne de valeur 0x00.

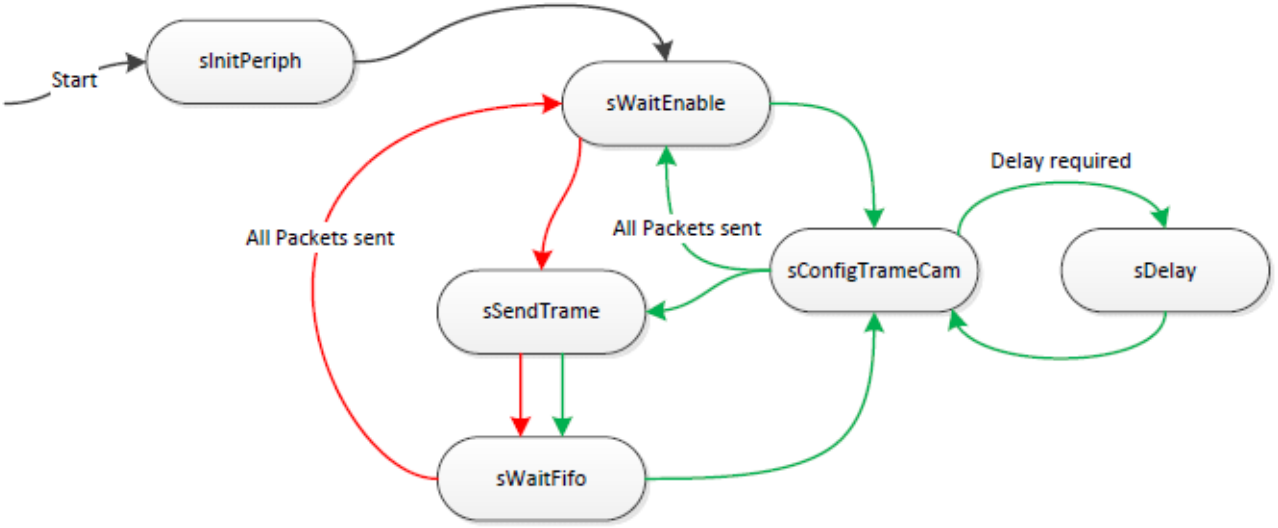


Figure 49 : Machine d'état du bloc de gestion des périphériques I2C

La liste des différents registres pour chaque capteur est disponible en figure 50. Pour rappel, le bit de poids faible de l'adresse signifie une lecture lorsqu'il vaut '1' et une écriture lorsqu'il vaut '0'.

Gyroscope ITG3200	Address	Xout H	Xout L	Yout H	Yout L	Zout H	Zout L
	1101001x	0x1D	0x1E	0x1F	0x20	0x21	0x22
Gyroscope L3G4200D	Address	Xout L	Xout H	Yout L	Yout H	Zout L	Zout H
	1101000x	0x28	0x29	0x2A	0x2B	0x2C	0x2D
MagnetoMeter MAG3110	Address	Xout H	Xout L	Yout H	Yout L	Zout H	Zout L
	0001110x	0x01	0x02	0x03	0x04	0x05	0x06
MagnetoMeter HMC5883L	Address	Xout H	Xout L	Zout H	Zout L	Yout H	Yout L
	0011110x	0x03	0x04	0x05	0x06	0x07	0x08
GPS Mikroe 1032	Address	Data					
	1000010x	0xFF					
Camera	Address						
	0111100x						

Figure 50 : Liste d'adresses et de registres des capteurs

La figure 51 représente le format d'une trame de lecture en I2C.

Format Read Sequence																			
Master		Start	Addr +Wr		Reg Addr		Start	Addr +Rd			Ack		Ack		...		NAck	Stop
Slave				Ack		Ack			Ack	DATA		DATA		...		DATA		

Figure 51 : format d'une trame I2C de lecture

Cette trame de lecture fonctionne pour les gyroscopes, les magnétomètres et le GPS. Ce dernier ne contient qu'une adresse utilisable. Il suffit donc de lire plusieurs fois le registre 0xFF pour lire les valeurs du GPS, qui sont fournies selon le format du protocole NMEA. Il y a cependant une petite particularité pour la lecture du L3G4200D : pour lire plusieurs registres à la suite, il faut mettre le MSB de l'adresse du premier registre à '1'. Donc dans ce cas ci : 0xA8 à la place de 0x28.

8.4.2 Simulation

La figure 52 reprend la configuration de test utilisée pour les blocs de pilotage de l'I2C.

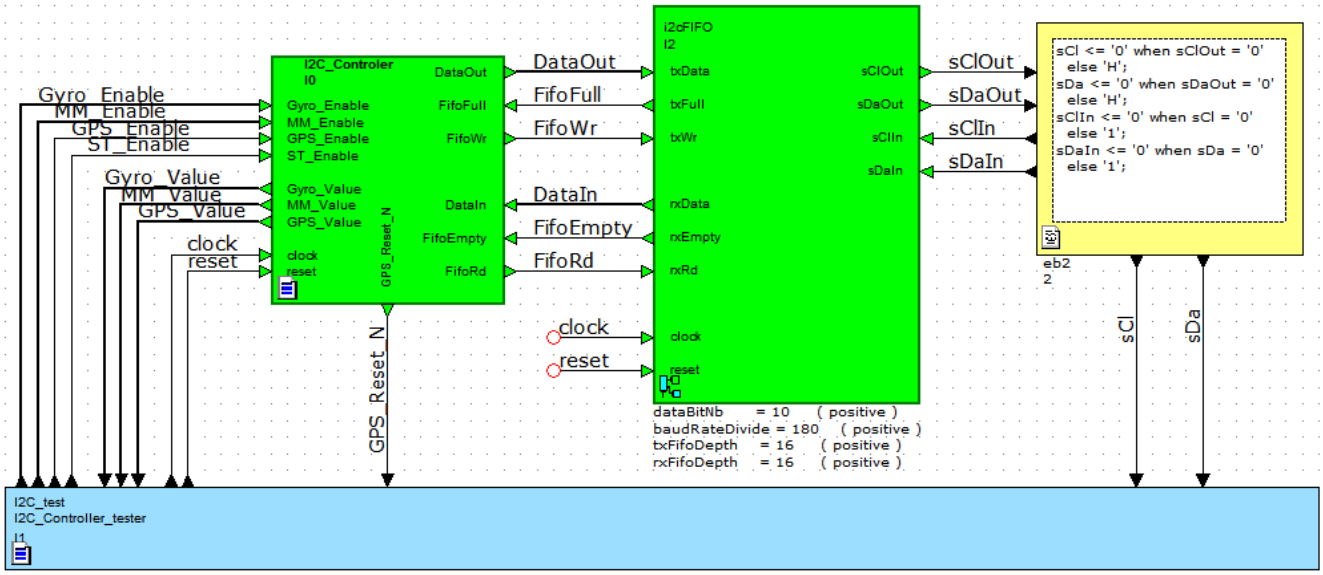


Figure 52 : Configuration de test du bloc de gestion des périphériques I2C

La simulation en figure 53 montre le fonctionnement en début de séquence. Tout d'abord, les premières trames d'initialisation sont envoyées sur le bus I2C. Ensuite le système attend dans l'état sWaitEnable qu'un des périphériques soit activé. Pendant cet état, il est possible de voir le signal cntPeriph qui varie pour passer en revue toutes les entrées d'enable. Dès qu'une entrée est activée, ici un des gyroscopes est enclenché, le système commence à envoyer la trame de lecture correspondante. Après chaque paquet envoyé, le signal cntPacket s'incrémente, jusqu'à atteindre le nombre de paquet requis, qui est de 12 pour les gyroscopes. A partir du cinquième paquet, les valeurs de l'I2C sont enregistrées dans un buffer nommé Gyro_value_temp. Celui-ci est divisé en deux pour chacun des gyroscopes, et subdivisé en trois pour chacun des axes à mesurer. D'une fois que la trame est complètement reçue, les valeurs du buffer temporaire sont envoyées en sortie, ici nommé gyro_value et la recherche de périphériques activés reprend.

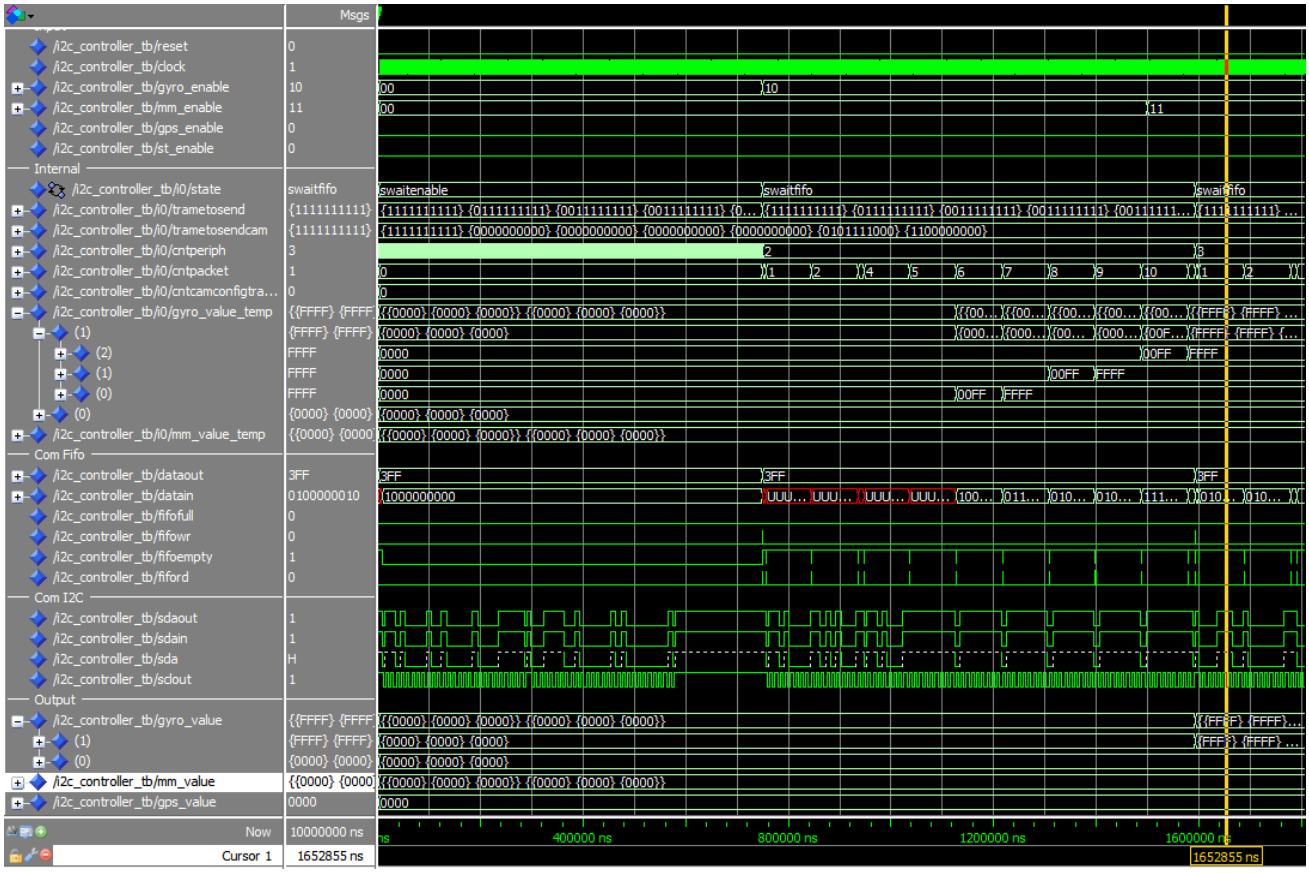


Figure 53 : Simulation 1 du bloc de gestion des périphériques I2C

La figure 54 est un zoom de la simulation précédente, à l'endroit où la trame à envoyer est écrite dans la fifo (signal FifoWr à '1'). Il est possible d'y voir les 12 paquets de la trame en détail, ainsi que le vidage de la Fifo avant l'envoi effectif des données (FifoRd).

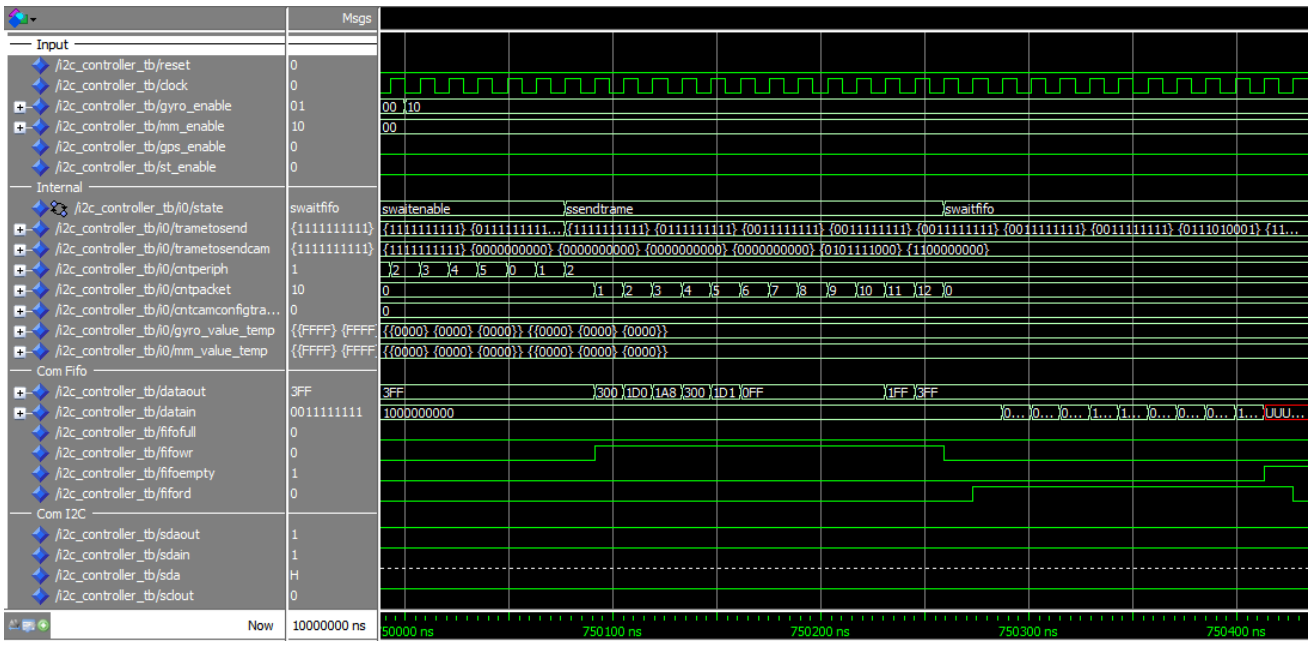


Figure 54 : Simulation 2 du bloc de gestion des périphériques I2C

Cette dernière simulation en figure 55 montre la séquence d'écriture pour la configuration de la caméra. Les deux curseurs délimite le début et la fin de la séquence, celle-ci est relativement longue par rapport aux autres étant donné qu'elle dure environ 85ms.

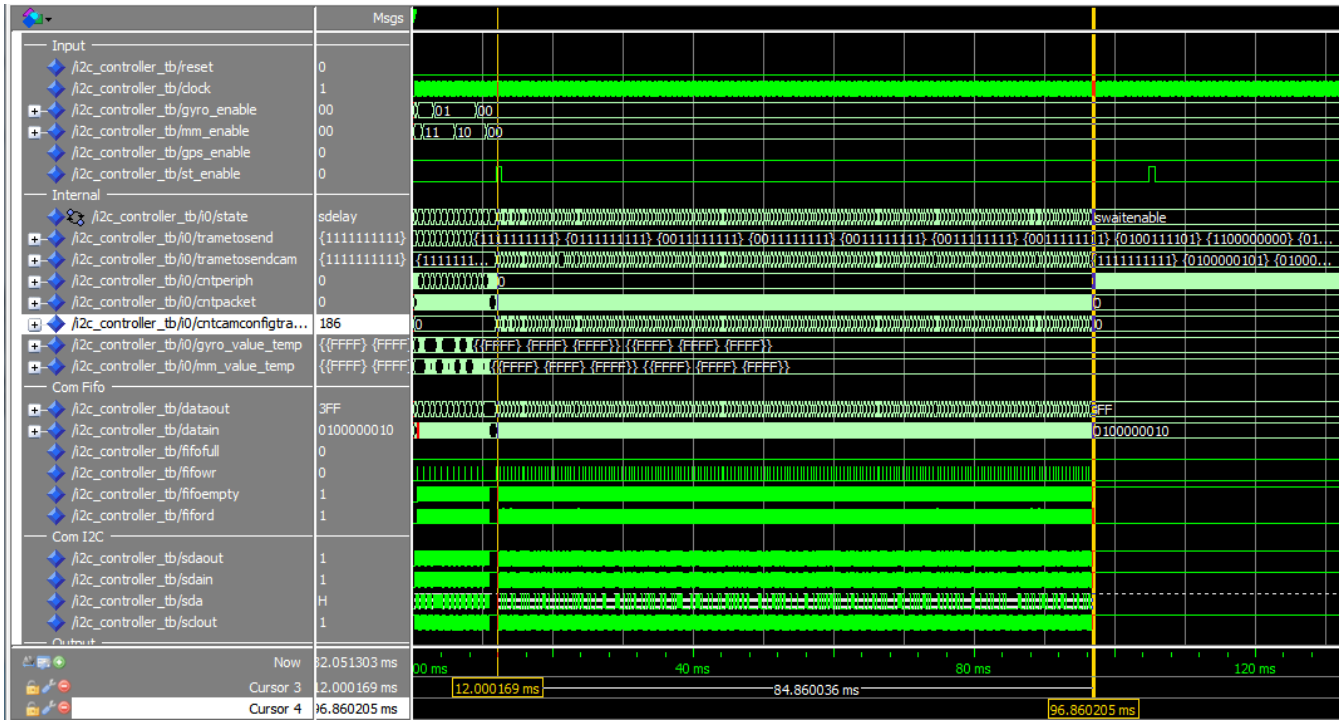


Figure 55 : Simulation 3 du bloc de gestion des périphériques I2C

8.5 Bloc de codage/décodage des trames UART

8.5.1 Description

Le dernier bloc qui a été modifié est celui qui fait office de cœur au système, la figure 56 offre une vue de son entité.

Sur la gauche du bloc se tiennent les entrées sorties dédiées à la communication UART, sur le côté gauche viennent se connecter les différents blocs de mesures et contrôle des capteurs, actuateurs qui ont été décris précedemment. Au fond à gauche se tiennent les entrées pour les signaux fournis par la caméra, tandis qu'au centre se tiennent les signaux pour le pilotage de la SDRam. Cela servira à stoquer les images procurées par la caméra. De nouveau, un bloc implémentant une fifo pour la ram a déjà été développé par la HES et est donc simplement repris ici.

	A	B	C	D	E
	Group	Name	Mode	Type	Bounds
2		reset	IN	std_ulogic	
3		In_Stream	IN	std_ulogic_vector	(dataStreamBitNb-1:0)
4		In_Stream_valid	IN	std_ulogic	
5		Out_Stream	OUT	std_ulogic_vector	(dataStreamBitNb-1:0)
6		Out_Stream_valid	OUT	std_ulogic	
7		OK_TO_TX	IN	std_ulogic	
8		MT_Enable	OUT	std_ulogic_vector	(2:0)
9		MT_PWM_DutyCycle	OUT	DutyCycleBus	
10		RW_Enable	OUT	std_ulogic_vector	(2:0)
11		RW_PWM_DutyCycle	OUT	DutyCycleBus	
12		SS_Enable	OUT	std_ulogic_vector	(NbSS-1:0)
13		EnAngle	IN	std_ulogic	
14		Angle1	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
15		Angle2	IN	signed	(IntegerBitNb+FractionalBitNb-1:0)
16		NumSS	IN	unsigned	(NbSS-1:0)
17		Gyro_Enable	OUT	std_ulogic_vector	(NbGyro-1:0)
18		MM_Enable	OUT	std_ulogic_vector	(NbMM-1:0)
19		GPS_Enable	OUT	std_ulogic	
20		ST_Enable	OUT	std_ulogic	
21		Gyro_Value	IN	GyroArrayValue	
22		MM_Value	IN	MMArrayValue	
23		GPS_Value	IN	std_logic_vector	(15:0)
24		Cam_FV	IN	std_ulogic	
25		Cam_LV	IN	std_ulogic	
26		Cam_Clk	IN	std_ulogic	
27		Cam_Data	IN	std_ulogic_vector	(dataCamBitNb-1:0)
28		ramEn	OUT	std_ulogic	
29		ramDataValid	IN	std_ulogic	
30		ramWr	OUT	std_ulogic	
31		ramRd	OUT	std_ulogic	
32		ramDataOut	OUT	std_ulogic_vector	(dataRamBitNb-1:0)
33		ramDataIn	IN	std_ulogic_vector	(dataRamBitNb-1:0)
34		ramAddr	OUT	unsigned	(AddressRamBitNb-1:0)

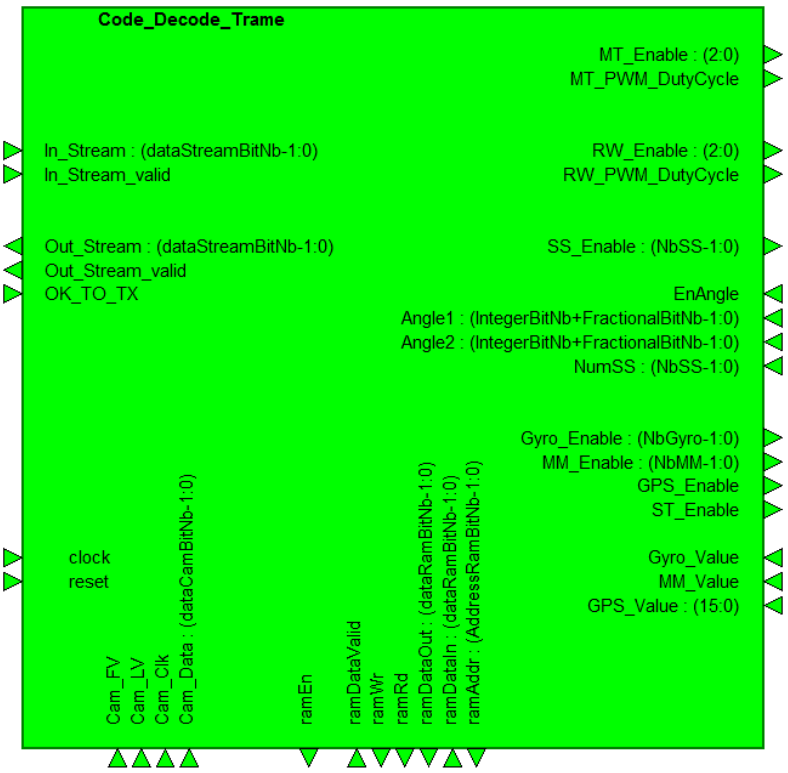


Figure 56 : Entité du bloc de codage/décodage des commandes UART

Pour gérer le tout, une machine d'état est implémentée, celle-ci est visible en figure 57.

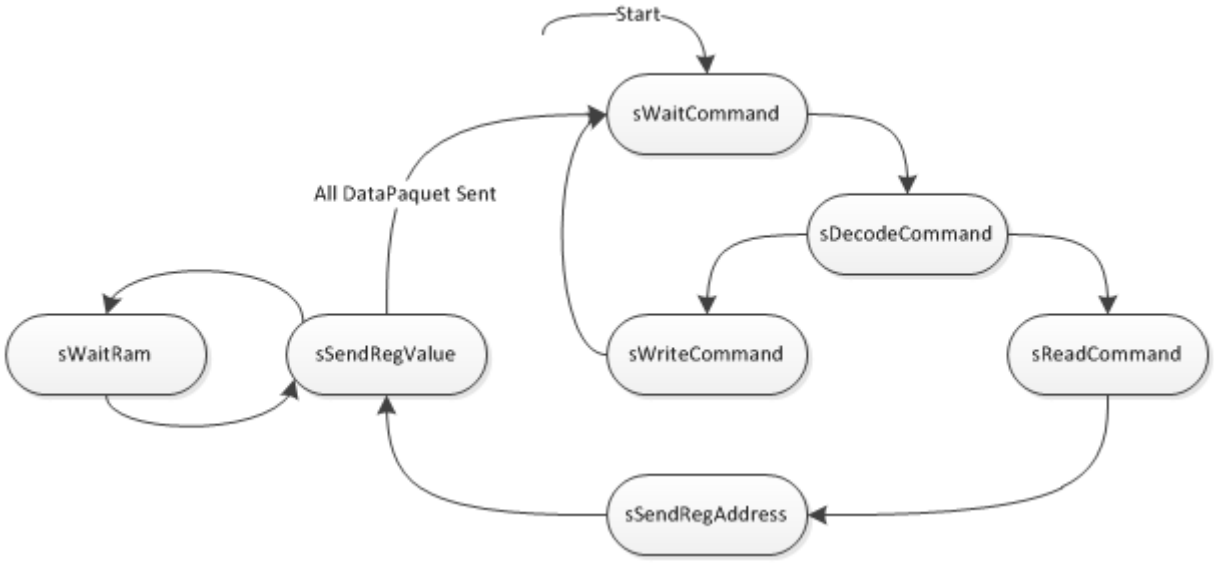


Figure 57 : Machine d'état du bloc de codage/décodage des commandes UART

Le système démarre dans l'état sWaitCommand et y reste jusqu'à l'arrivée d'une commande par l'UART. Dès qu'un paquet est reçu, celui-ci est analysé, si c'est un paquet start, donc 10101010 selon le protocole décrit dans le chapitre correspondant, le système passe dans l'état sDecodeCommand. A l'arrivée du prochain paquet, le système décode la commande et diverge dans l'état sWriteCommand ou sReadCommand selon si c'est une demande d'écriture ou de lecture. Dans le premier cas, la valeur du prochain paquet est simplement écrite dans le registre correspondant. Dans le deuxième cas, le système initialise une trame de réponse en envoyant sur l'UART un paquet de start et continue dans l'état sSendRegAddress. Dans ce dernier, l'adresse du registre à lire est simplement envoyée, puis l'état sSendRegValue est atteint. A partir de la, deux possibilités existent. Soit c'est la caméra que l'on souhaite lire, soit un des autres capteurs. Dans la première possibilité la machine va alterner entre les états sWaitRam pour lire la valeur des pixels de l'image et sSendRegValue pour les envoyer sur l'UART. Si au contraire c'est un autre registre que l'on souhaite lire, la valeur de ce dernier est simplement envoyée sur l'UART et le système reprend son état initial qui est sWaitCommand.

En parallèle à cela et lorsque une lecture de la caméra n'est pas demandée, les valeurs des pixels de l'image sont stoquées en continu dans la RAM, dans deux plages d'adresses différentes. Ceci permettant d'avoir toujours une image complète lors d'une demande de lecture alors qu'une écriture est en cours.

8.5.2 Simulation

Le test de ce bloc se présente comme à la figure 58.

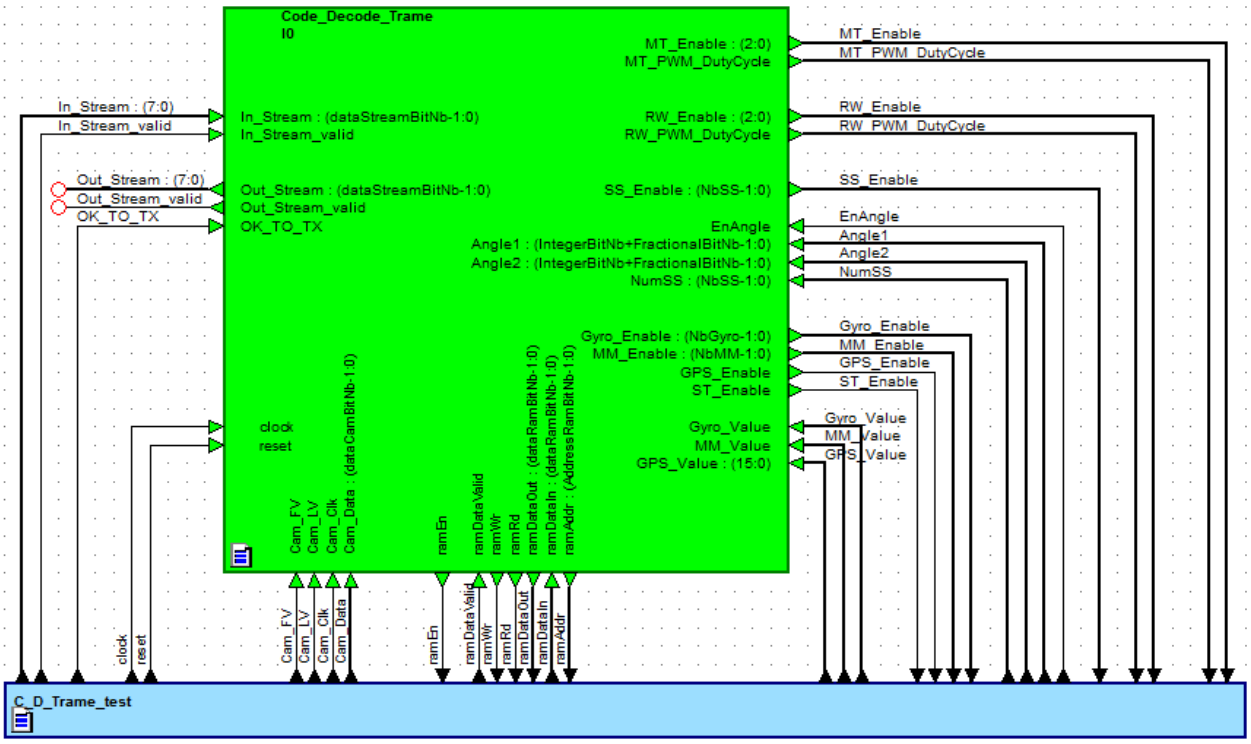


Figure 58 : Configuration de test du bloc de codage/décodage des trames UART

Dans la simulation présentée à la figure 59, les commandes d'activation des périphériques sont testées. Le test commence par activer cinq Sun-Sensors, ensuite les deux gyroscopes et la caméra, puis ces derniers sont désactivés tandis que les Magnétomètres et le GPS sont activés et finalement le Magnéto-Torquer 2 et les Reaction-Wheels 1 et 3 sont activés.

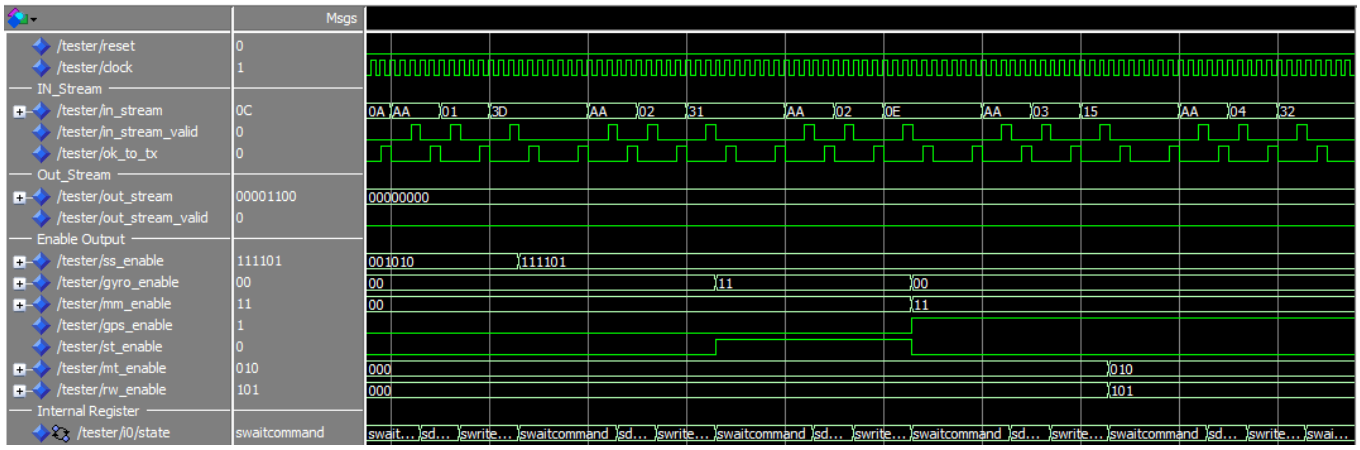


Figure 59 : Simulation 1 du bloc de codage/décodage des trames UART

La simulation en figure 60 présente la suite des opérations d'écriture, en l'occurrence des valeurs des PWMs. Il est possible d'y apercevoir ces valeurs qui sont correctement mise en sortie sur les signaux mt_pwm_dutycycle et rw_pwm_dutycycle.

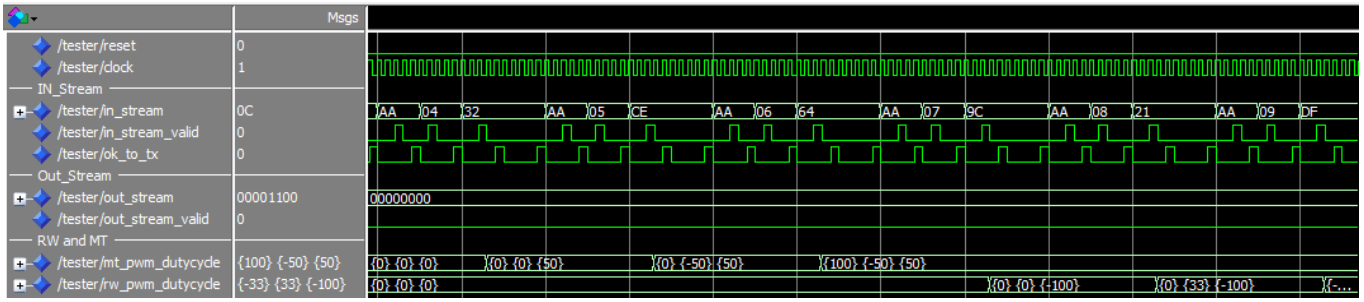


Figure 60 : Simulation 2 du bloc de codage/décodage des trames UART

Afin de tester les commandes de lecture, la simulation en figure 61 a été effectuée. Celle-ci comprend la lecture des valeurs d'angles de la source lumineuse, ainsi que la lecture des valeurs des gyroscopes. Bien sur les valeurs sur les entrées correspondantes ont été générées dans le banc de test, mais ces valeurs sont tout de même bien reprises et envoyées dans le bon format et le bon ordre sur l'UART.

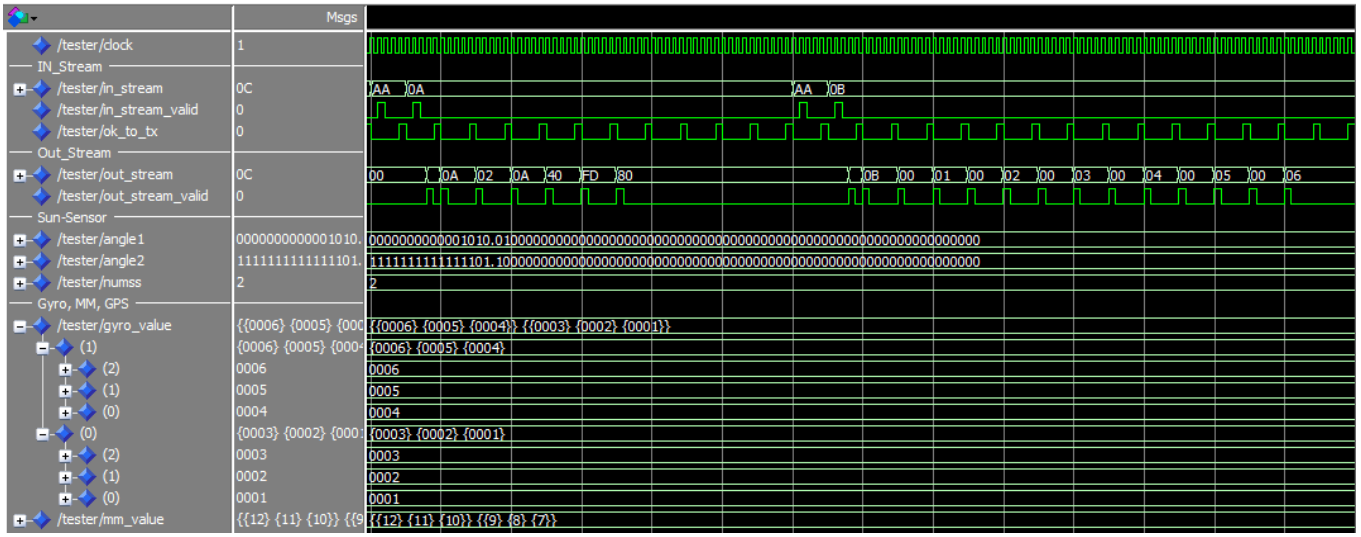


Figure 61 : Simulation 3 du bloc de codage/décodage des trames UART

La prochaine simulation, visible en figure 62, permet d'avoir un aperçu de la sauvegarde des données de la caméra dans la mémoire RAM.

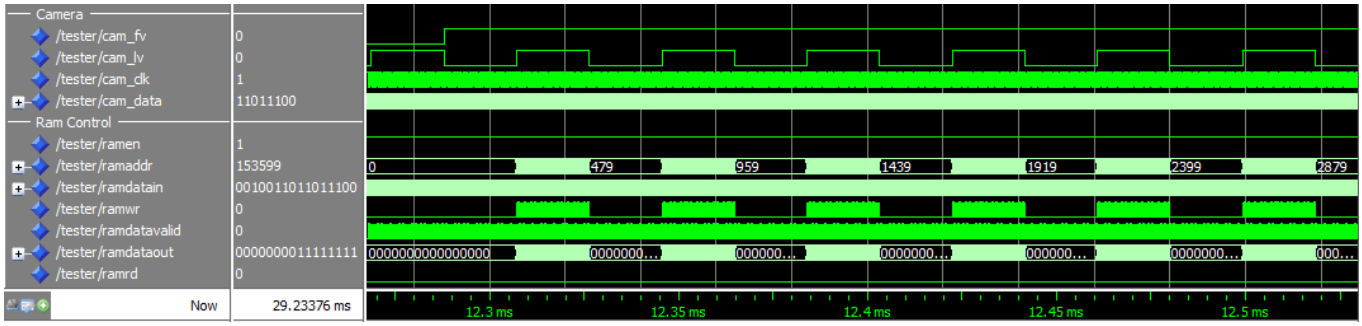


Figure 62 : Simulation 4 du bloc de codage/décodage des trames UART

Lorsque les signaux cam_fv et cam_lv sont à l'état haut, l'adresse de la ram est incrémentée, et les données reçues de cam_data sont transférées sur ramDataOut. Un zoom sur le flanc montant du signal cam_lv disponible en figure 63 permet de voir en détail les différents signaux.

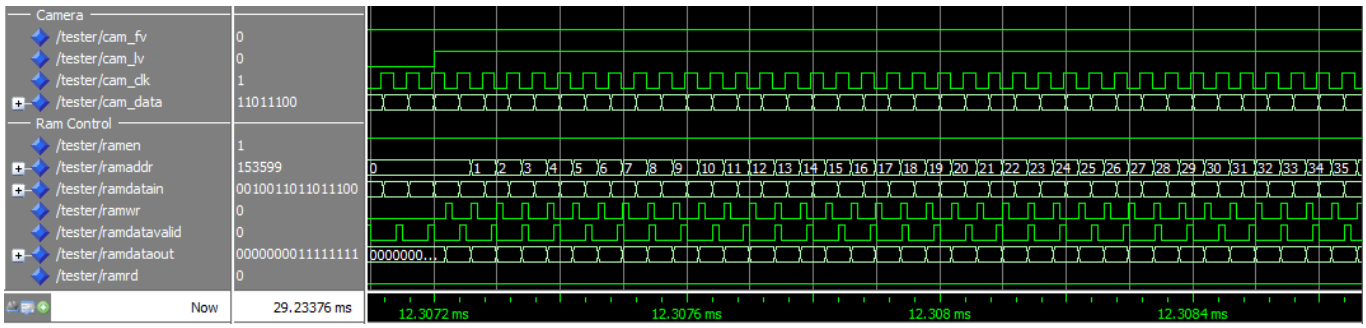


Figure 63 : Simulation 5 du bloc de codage/décodage des trames UART

8.6 Système complet

8.6.1 Description

D'une fois que toutes les parties du circuit ont été testées individuellement, elles ont été connectées ensemble pour former le circuit final, ce circuit est visible en annexe 13 et la figure 64 reprend juste l'allure du bloc final.

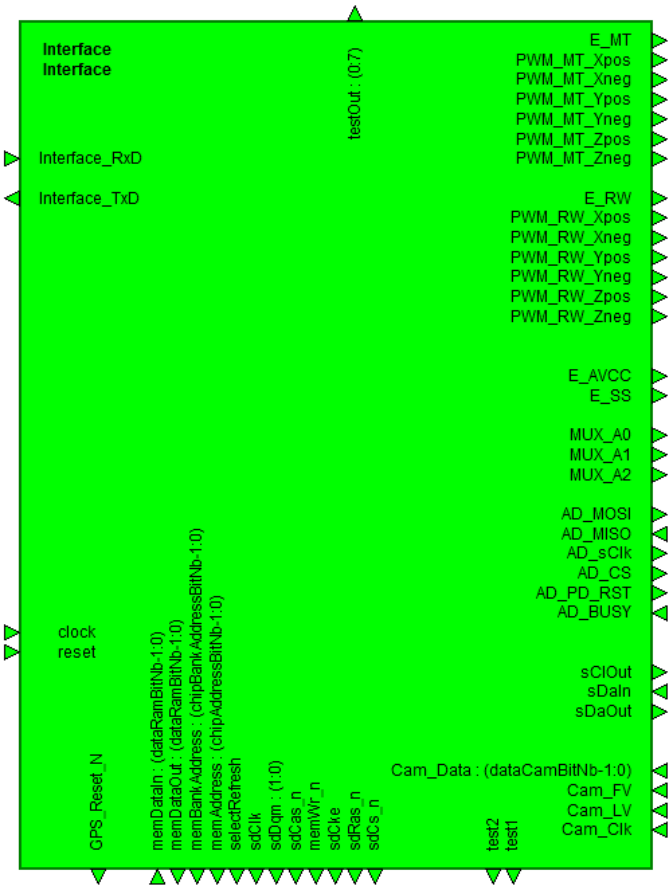


Figure 64 : Entité du bloc d'interface final

8.6.2 Simulation

Pour la simulation, tous les signaux du bloc ont simplement été connectés à un banc de test.

La simulation présentée en figure 65 montre globalement que les sorties réagissent en fonction des commandes UART. La première commande active tous les Sun-Sensors, ainsi les signaux sur le convertisseur AD et sur les multiplexeurs commencent à varier. La deuxième commande active les deux gyroscopes, donc le bus I2C se met en activité. La troisième commande désactive certains des Sun-Sensors de manière à laisser le numéro 1 et 3 activés. Etant donné qu'ils sont connectés aux entrées 0 et 2 des multiplexeurs, seul le signal mux_A1 continue de varier. La cinquième englobe les commandes d'activation et de configuration des PWMs pour les magnéto-torquer 1 et 2. Ensuite ce se succèdent des commandes de lectures et finalement des commandes pour désactiver les périphériques.

Figure 65 : Simulation 1 du circuit final

La figure 66 est simplement un zoom de la simulation précédente sur une commande d'écriture et une de lecture. Comme les entrées des valeurs des capteurs n'ont pas été générées par le banc de test, la commande de lecture renvoie zéro pour ce registre.

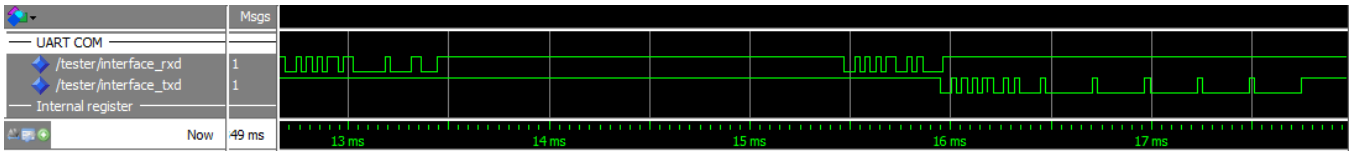


Figure 66 : Simulation 2 du circuit final

8.6.3 Synthèse

Comme les simulations ont prouvé le bon fonctionnement du circuit, la synthèse a pu être faite. Pour cela une nouvelle librairie nommée board a été créée. Dans celle-ci se trouve le bloc du système précédemment décris, plus quelques autres notamment pour la synchronisation des signaux d'entrées. Ces blocs supplémentaires sont présentés en figure 67.

Le premier bloc jaune agit comme interface sur le bus I2C, si le signal demande d'être à l'état bas, la sortie sera tirée à l'état bas, par contre si le signal est à '1' cela signifie qu'on ne souhaite rien envoyer, et donc laisser la ligne en haute impédance pour que les autres périphériques connectés puissent eux même y placer un état bas ou autre.

Le deuxième bloc jaune agit comme un buffer tri-states sur le bus de données de la RAM. Si le signal memWr_n vaut '0' cela signifie que l'on souhaite écrire dans la RAM, et donc memData prendra la valeur de memDataOut. Au contraire si memWr_n vaut '1', cela si-

gnifie que l'on doit laisser memData en haute impédance au cas où la RAM voudrait envoyer des données.

La porte D est simplement un exemple de comment synchroniser une entrée avec le clock, tandis que le dernier bloc permet d'activer l'alimentation des Reaction-Wheels et des Magnéto-Torquer si l'un ou l'autre de ces actuateurs est activés.

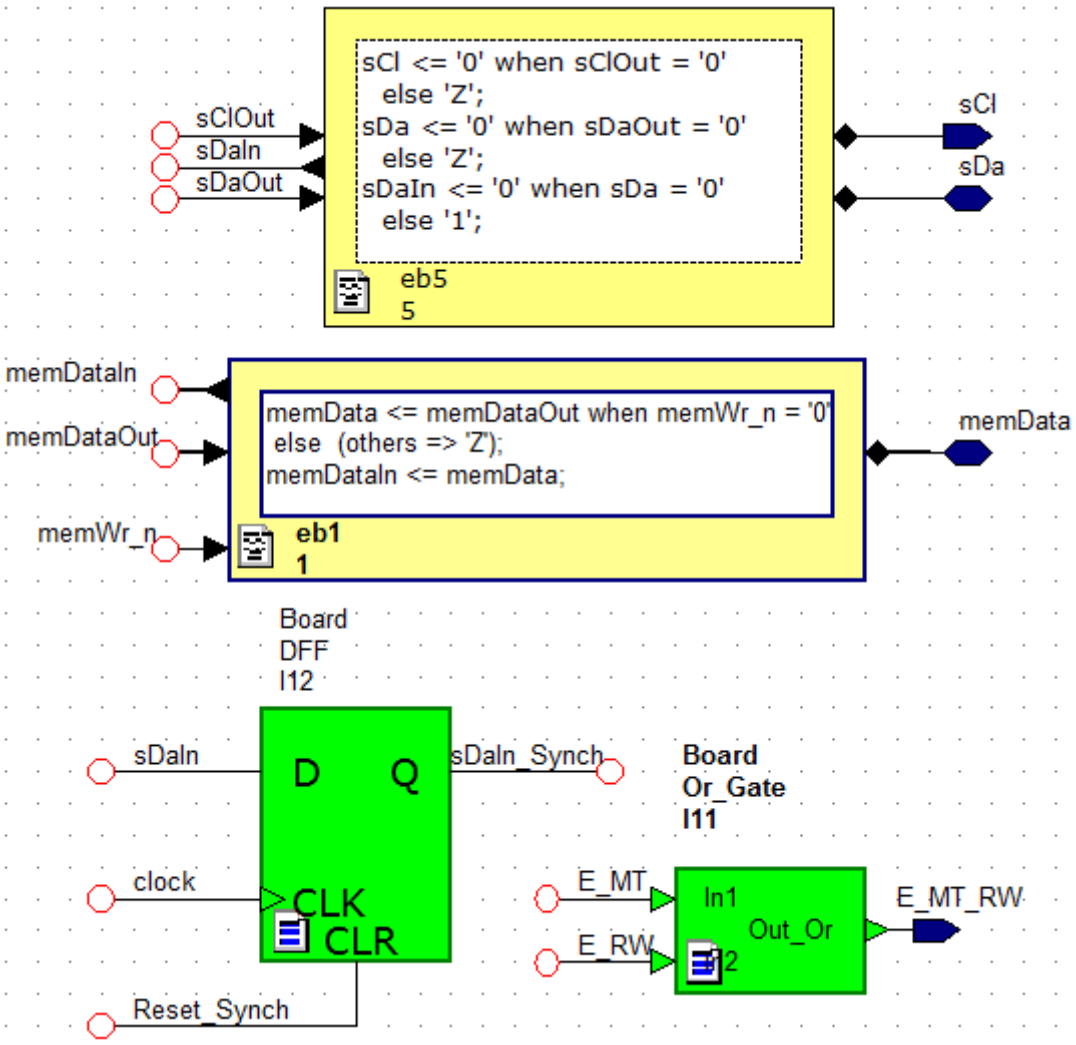


Figure 67 : Bloc d'ajout pour le portage du système sur la FPGA

Une fois ce circuit achevé, il est possible de lancer réellement la synthèse. Pour cela il suffit d'aller dans la librairie board, puis d'appuyer sur « Prepare for synthesis » suivi par « Xilinx Project navigator », ce qui devrait faire démarrer le programme de synthèse.

Dans l'onglet en haut à gauche se trouve une série de fichier, il est possible d'éditer le fichier ayant l'extension .ucf pour configurer les différentes broches de la FPGA sur lesquels devront être connectés les différents signaux du circuit. Lorsque cela est fait, un double clic sur « Configure Target Device » permet de lancer le processus de synthèse.

S'il n'y a pas d'erreur, un fichier .bit sera généré et le programme IMPACT sera lancé. A partir de là, la suite des manipulations est la suivante : double clic sur Boundary Scan, clic droit sur la fenêtre au centre, Initialize Chain, puis sélectionner le fichier .bit précédemment généré. Finalement clic droit sur la FPGA-> Program.

9 APPLICATION QT

9.1 Description de l'interface graphique

La figure 68 offre une vue de l'application de démonstration développée sur QT Creator 2.7.1.

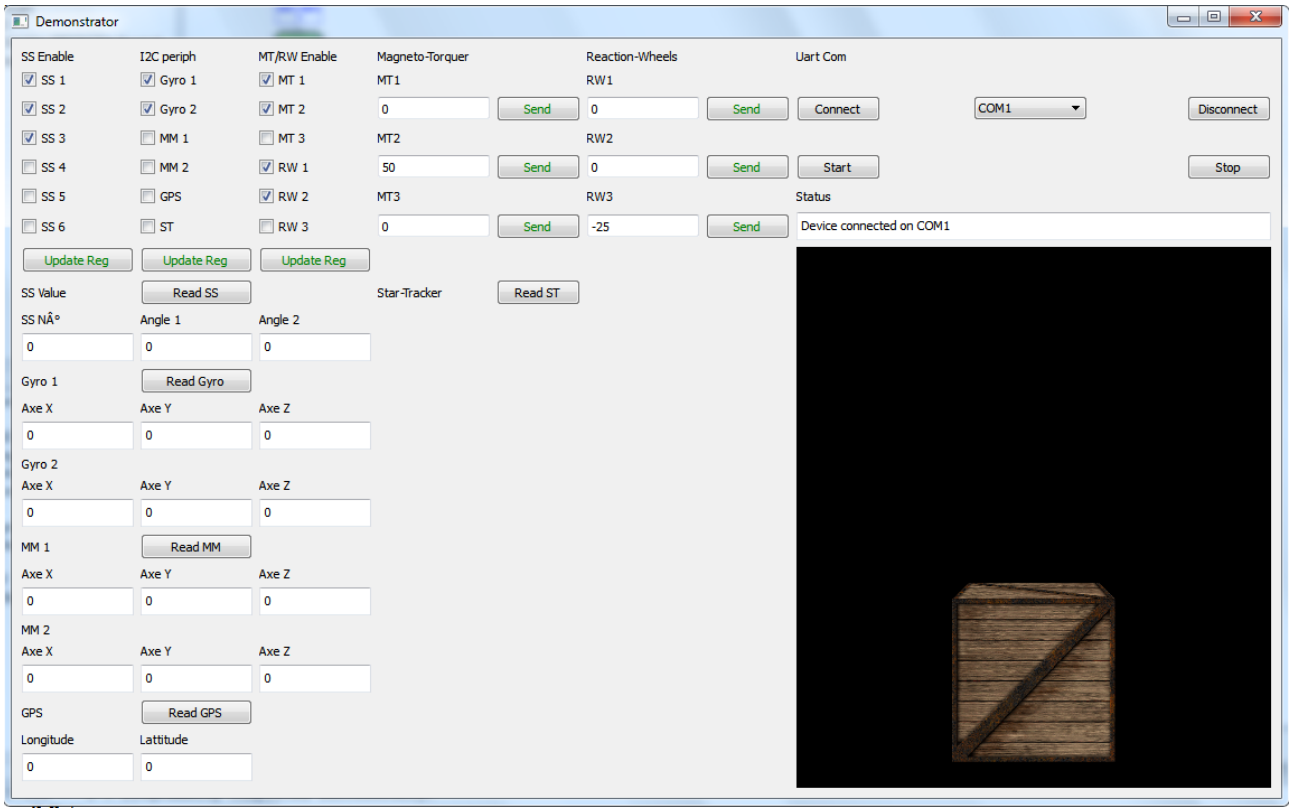


Figure 68 : Application de démonstration

Lors du démarrage de l'application, une majorité de boutons et de champs de texte sont grisés, jusqu'à l'activation de la communication UART. La partie en haut à droite permet de faire cela, une liste des différents ports est disponible, et lorsque le port désiré est sélectionné, l'appui sur le bouton « connect » permet de s'y connecter. A partir de là les boutons deviennent accessibles. En haut à gauche se situe la zone de gestion de l'activation des capteurs et des actuateurs. Lorsque les différents périphériques désirés sont sélectionnés à travers les checkboxs, un appui sur les boutons « Update Reg » permet d'envoyer la trame UART de configuration correspondante au système. La zone en haut au centre contient la partie pour le pilotage des actuateurs, les champs de texte permettent d'inscrire les valeurs de dutyCycles en % pour chaque Magneto-Torquer et chaque Reaction-Wheel. L'appui sur les boutons « Send » permet d'envoyer ces valeurs sur l'UART, mais seules les valeurs comprises entre -100 et 100 seront effectivement envoyées.

Toute la partie gauche est destinée à l'affichage des valeurs mesurées provenant des différents capteurs. Les boutons « Read » permettent d'envoyer les trames UART pour la récupération de ces données. Au centre un espace est réservé pour l'affichage de l'image prise par la caméra, tandis que tout à droite se tient la partie de visualisation du satellite. Les boutons « Read » ne sont présents ici que pour une question de débugage de l'application, car en effet un appui sur le bouton « Start » situé dans la partie UART permet de lancer l'acquisition automatique des valeurs des capteurs, selon s'ils sont activés

ou non. La fréquence de rafraichissement de ces valeurs peut être très rapide, et donc ces dernières peuvent changer tellement vite que leur lecture devient difficile. Pour une meilleure visualisation il serait possible d'ajouter des ProgressBars sur la fenètre.

Finalement sur le côté droit se situe une fenêtre de visualisation de l'orientation du satellite qui a été simplement récupérée mais pas implémentée par manque de temps. Du coup le cube reste statique, peu importe les valeurs lues.

9.2 Description des classes

La figure 69 montre le diagramme de classe de l'application. Comme cette dernière a été reprise et modifiée suite aux nouveaux besoins, certaines classes ne sont plus utilisées mais ont été laissées de côté dans le cas où leur utilisation serait de nouveau nécessaire dans un stade future de développement.

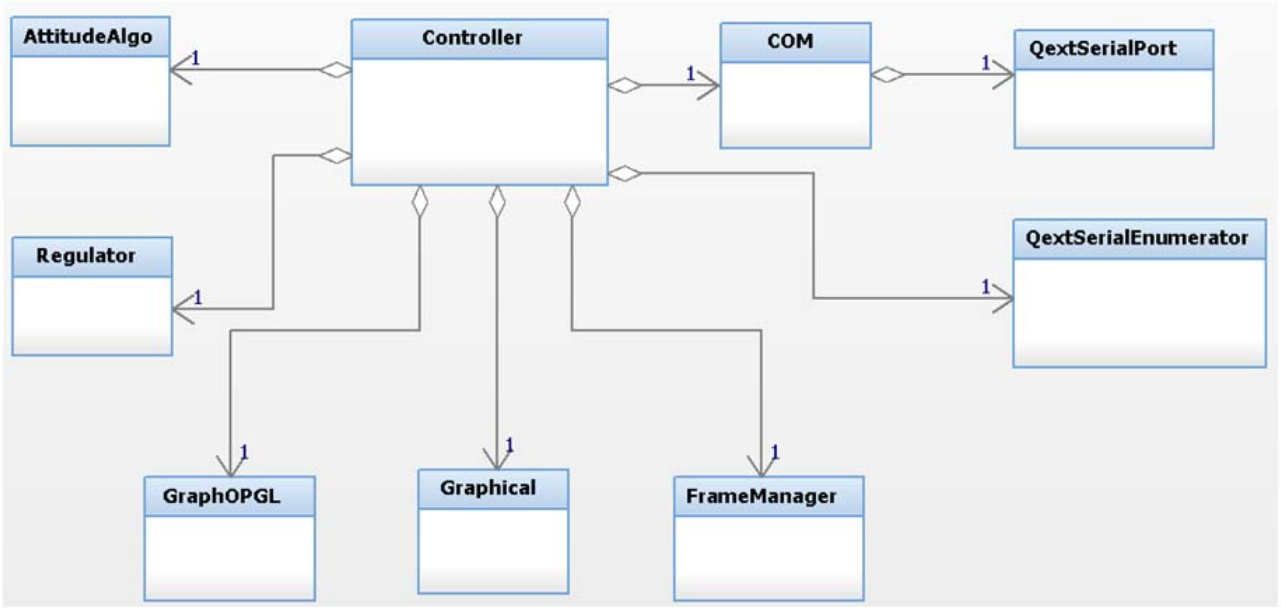


Figure 69 : Diagramme de classe de l'application [2]

La classe « controller » fait office de cœur du système, c'est elle qui prend toute les décisions et pilote les autres classes.

Les classes « QextSerialEnumerator » et « QextSerialPort » sont deux classes API consacrés à la communication UART, elles gèrent l'envoi et la réception des différentes trames.

La classe « Com » instancie simplement la classe « QextSerialPort », permettant l'utilisation des méthodes de cette dernière par le Controller.

La classe FrameManager est comme son nom l'indique, utilisé pour créer les différentes trames à envoyer sur l'UART.

L'interface graphique est gérée par la classe « Graphical » et la simulation du cube est gérée par la classe « GraphOPGL »

Finalement, les classes « Regulator » et « attitudeAlgo » étaient dédiées respectivement à la régulation du magnéto-torquer en fonction des valeurs d'angles du Sun-Sensor, et au calcul de ces angles en fonction des valeurs S1, R1, S2 et R2 du capteur. Etant donné que ces fonctions doivent être codées directement dans la FPGA, elles n'ont plus d'utilité ici.

9.3 Diagramme de séquence

La figure 70 représente la séquence qui est effectuée lorsque les valeurs des capteurs doivent être lues suite à l'appui sur le bouton « Start ».

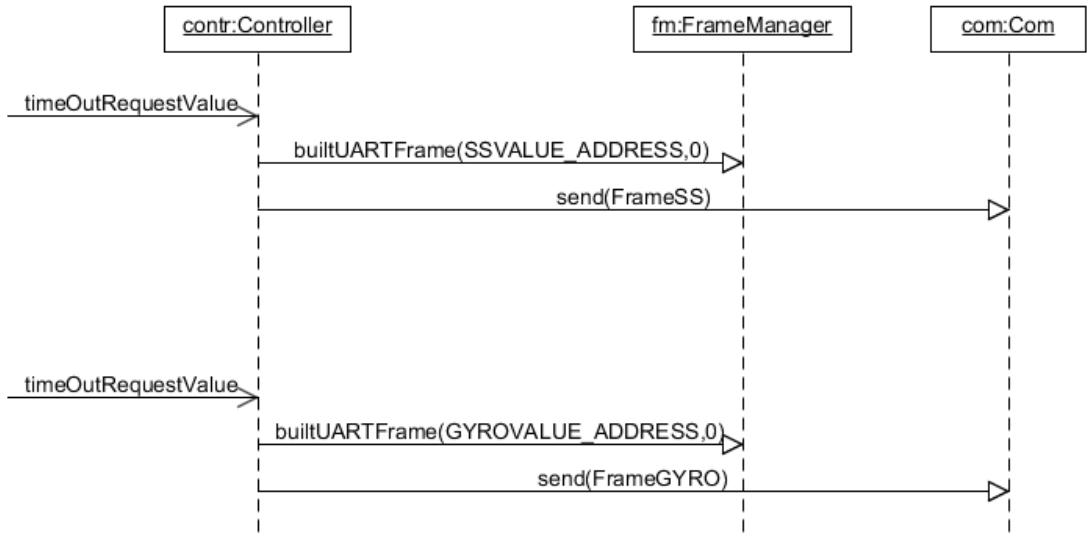


Figure 70 : Diagramme de séquence du rafraichissement des valeurs mesurées

L'appui sur ce dernier enclenche un timer, qui va appeler le slot timeOutRequestValue à chaque échéance. Dans cette fonction, un type de périphérique est analysé, si celui-ci est activé, alors la fonction builtUARTFrame() du FrameManager est appelée, avec comme paramètre l'adresse du périphérique. La deuxième valeur du paramètre vaut '0' lorsque l'on souhaite effectuer une lecture, autrement elle contient la valeur à écrire dans le registre. D'une fois que la trame a été préparée, la fonction send() de la classe Com est appelée, cette dernière envoie simplement la trame sur l'UART.

Si au contraire le périphérique n'est pas activé, le code sort de la fonction jusqu'à la prochaine échance du timer, où un autre type de périphérique sera pris en compte. De cette manière seuls les capteurs activés sont interrogés, augmentant ainsi la fréquence de rafraichissement des mesures.

La caméra n'est pas pris en compte dans cette fonction étant donné qu'il y énormément de données à transférer, et que le baudrate de l'UART est relativement faible, chaque transmission d'image prend environ 30 secondes.

10 TESTS ET RESULTATS

10.1 Caractéristiques

Caractéristiques de la carte électronique :

- Taille : 87x93x48 mm (Largeur x longueur x hauteur)
- Poids : 119 gr
- Consommation électrique : ~170mA / 200mA (aucun / tous les capteurs activés)
- Coûts total : 260.- + 254.- (Coûts de fabrication 2 PCB + composants)

Caractéristiques de la partie FPGA :

- Taux d'occupation de la FPGA : 2% / 5% (Slice Registers / Slice LUTs)
- Fréquence maximale d'acquisition des valeurs des capteurs[1] : 236.9Hz

Caractéristiques de l'application QT

- Fréquence maximale de rafraichissement des valeurs des capteurs[2] : 55.6Hz

Aucune contrainte n'a été donnée pour ce projet, sauf pour la taille du circuit et cette dernière a été respectée.

[1] Fréquence mesurée pour acquérir les valeurs de l'ensemble des gyroscopes, magnétomètres et GPS sur le bus I2C, dont la vitesse de ce dernier est réglée sur 100kHz. Cette vitesse pourrait être augmentée jusqu'à 400kHz si les valeurs du GPS ne sont pas nécessaires, étant donné que c'est le seul périphérique qui supporte un clock maximum de 100kHz.

[2] Fréquence d'acquisition des valeurs pour les Sun-Sensors, les gyroscopes et les magnétomètres à travers la communication UART.

10.2 Mesure de la position du soleil

Setup

Ce test a pour but de valider le fonctionnement de la partie touchant aux capteurs solaires. Pour effectuer celui ci, une alimentation a été branchée sur les canaux du convertisseur AD pour simuler un Sun-Sensor avec les valeurs suivantes :

- Canal 3 de l'AD = S1 = 1.4V
- Canal 4 de l'AD = R1 = 0.42V
- Canal 5 de l'AD = S2 = 0V
- Canal 6 de l'AD = R2 = 0V

Ensuite, un seul des capteurs solaire a été activé sur l'application de démonstration. Finalement des trames de lecture des valeurs d'angles des Sun-Sensors ont été envoyées depuis l'application à l'aide du bouton « Read SS».

Résultat attendu

L'application devrait afficher sous SS N° le numéro du capteur qui a été activé, sous Angle 1 une valeur se rapprochant de 17° et 0° sous Angle 2.

Résultat obtenu

Le numéro sous SS N° correspond bien à celui qui est activé, par contre les deux autres valeurs varient à chaque fois et ne sont même pas centrées autour d'un certain nombre.

Ainsi des tests supplémentaires ont été effectués pour déterminer l'origine de l'erreur. Les signaux des blocs de la FPGA ont été mesurés en partant de la communication SPI, puis en remontant progressivement bloc par bloc jusqu'à celui de la communication UART. Il se trouve que les signaux sont tout à fait correct jusqu'aux entrées du bloc de calcul des angles. Les valeurs de sorties de ce bloc ne correspondent pas avec ce qui a été simulé, sauf pour le numéro du capteur qui est correct.

Cela permet néanmoins de valider la chaîne de communication, depuis la lecture de l'AD jusqu'à l'application. Pour la suite, et d'une fois que le problème du bloc de calcul d'angle

sera résolu, d'autres tests pourront être effectués pour valider le circuit électronique des Sun-Sensors. Pour cela une simple lampe pourrait être utilisée dans une pièce sombre afin d'éclairer tour à tour les capteurs et de voir si les valeurs correctes sont affichées sur l'application.

10.3 Mesure de l'accélération angulaire

Setup

Ce test a pour but de valider le fonctionnement des gyroscopes. Pour cela, le circuit a été connecté normalement à l'application de démonstration, puis les gyroscopes ont été activés. Finalement, les valeurs ont été lues automatiquement suite à l'appui sur le bouton « start » ou alors manuellement avec le bouton « Read Gyro »

Résultat attendu

Les valeurs des différents axes devraient se centrer autour de 0 lorsque le circuit reste statique, et varier selon les mouvements sur chaque axe, comme imagé sur la figure 71. Les flèchent indiquent le sens positif.

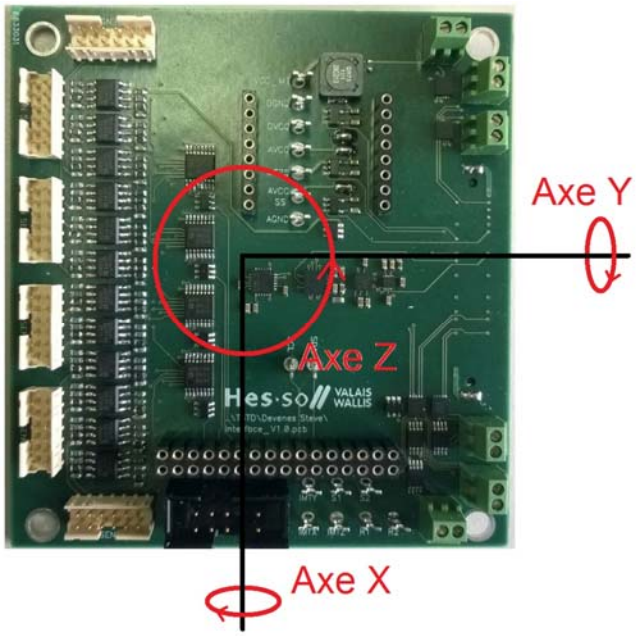


Figure 71 : Référence pour les gyroscopes

Résultat obtenu

À l'état statique, les valeurs varient autour de zéro, sauf pour l'axe X du gyroscope 1 (ITG-3200) qui a un offset d'environ 700 et l'axe Z du gyroscope 2 (L3G4200D) qui a un offset d'environ -100. Pour l'heure, la source de ce problème reste inconnue.

En dépit de cela, les accélérations subies par le circuit sur les trois axes lorsqu'il est mis en mouvement sont quand même détectées correctement par les gyroscopes, et affichées sur l'application.

10.4 Mesure du champ magnétique

Setup

Ce test a pour but de valider le fonctionnement des magnétomètres. De manière similaire à la mesure de l'accélération angulaire, il a suffit pour cela d'activer les deux magnétomètres depuis l'application et de presser sur le bouton « start », ou de manière répétée sur le bouton « Read MM », pour y lire les valeurs mesurées. Ensuite à l'aide d'un des magnéto-coupleur branché à une alimentation, un champ magnétique a été généré et approché des capteurs sur les trois axes.

Résultat attendu

Les valeurs du champ magnétique devraient varier selon l'axe approché par le magnéto-coupleur, une référence de ces axes se trouve en figure 72.

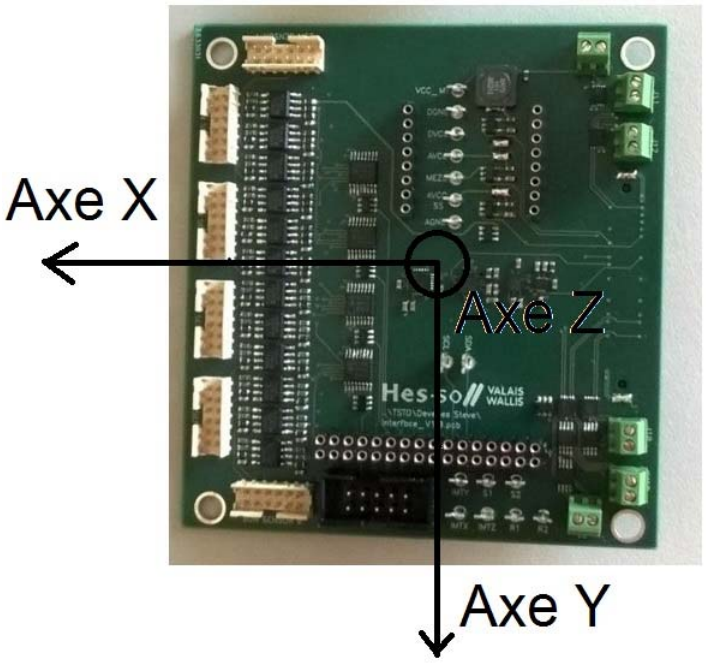


Figure 72 : Références pour les magnétomètres

Résultat obtenu

Le magnétomètre 2 (HMC5883L) détecte correctement les changements du champ magnétique, et ces valeurs sont affichées sur l'application. Par contre le magnétomètre 1 (MAG3110) ne renvoie que la valeur -1 pour chacun des axes.

Une mesure sur le bus I2C à l'aide d'un oscilloscope a permis de se rendre compte que le magnétomètre ne répondait pas quand il était interrogé (voir figure 73).

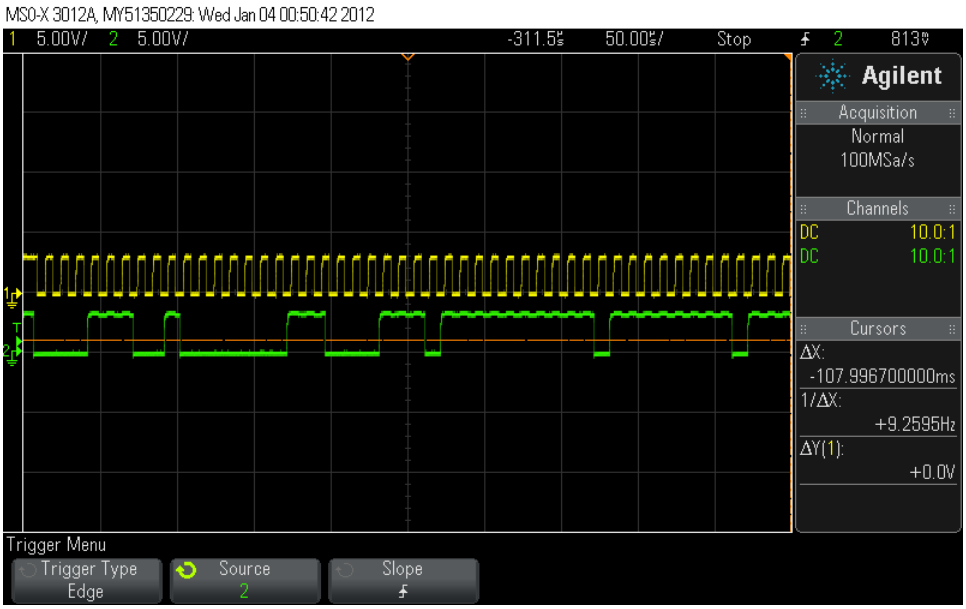


Figure 73 : Mesure à l'oscilloscope du bus I2C

En effet le bit d'acknowledge ne passe pas à l'état bas lors de l'écriture de l'adresse du périphérique. Plusieurs hypothèses furent envisagées pour expliquer ce problème. L'adresse du périphérique est fausse, le composant est mal soudé, une piste du PCB est interrompue ou bien le composant est tout simplement défectueux.

La première hypothèse fut écartée, d'après le datasheet du composant, son adresse est 0x0E, ce qui correspond bien à ce que l'on peut apercevoir sur la simulation. Des mesures à l'ohm-mètre sur les pistes du capteur on permit d'écarter l'hypothèse des pistes coupées, l'idéal serait de mesurer à l'oscilloscope directement sur les broches du capteur les différents signaux, mais ces broches sont tellement petites que ces mesures sont difficiles. Au final le capteur pourrait être soit mal soudé, soit défectueux.

10.5 Mesure de la position GPS

Setup

Pour ce test, le kit GPS mikroe-1032 a été connecté au circuit, ensuite le GPS a été activé depuis l'application et les broches de l'I2C ont été mesurées à l'aide d'un oscilloscope.

Résultat attendu

Ce test avait pour but de comprendre comment les trames du protocole NMEA sont envoyées en I2C, dans le but de pouvoir décoder ces messages et y retrancher les valeurs voulues de longitude et latitude.

Résultat obtenu

La figure 74 montre la trame I2C de la lecture du GPS.

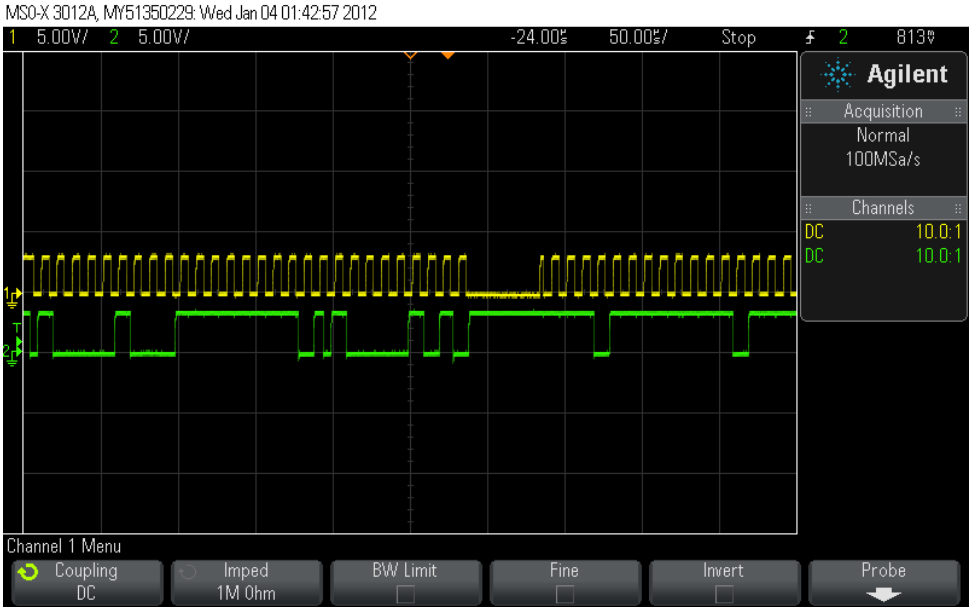


Figure 74 : Trame I2C de lecture du GPS

Il est possible d'y apercevoir que le GPS envoie bien des bits d'acknowledge après chaque bytes transmis, indiquant que la communication est correctement effectuée. Par contre le signal SCL est mis à zéro pendant un bref instant, ce qui perturbe la suite de la transmission. Après quelques recherches, il s'avère que c'est le GPS qui met la ligne à l'état bas lorsque la vitesse du clock est trop élevée, demandant ainsi au master la main sur la transmission. Dans un tel cas, le master est censé relâcher la ligne SCL pour permettre au slave de générer lui-même le clock. Ce procédé est nommé Clockstretching et malheureusement le système n'a pas été prévu pour cela.

Deux possibilités existent pour palier à cela, soit modifier le code de la FPGA pour prendre en compte ce procédé, soit réduire la vitesse de l'I2C, ce qui réduirait la vitesse de lecture des autres périphériques, et donc les performances du système entier.

10.6 Prise de vue du Star-Tracker

Setup

Ce test a pour but de valider le fonctionnement de la caméra. Pour cela, la caméra a été connectée au circuit, puis activée à l'aide de l'application. Le bouton « Read ST » permet de lire et d'afficher l'image la plus récente.

Résultat attendu

L'image doit s'afficher sur l'application.

Résultat obtenu

L'application crash lorsque le transfert de l'image est terminé et que cette dernière essaie d'être affichée. Néanmoins des données sont bien reçues dans la classe « contrôler » de l'application même si celles-ci ne sont pas forcément correctes. La figure 75 montre le début de la communication UART pour obtenir ces données. Le problème provient donc simplement du code de l'affichage de l'image, mais par faute de temps celle ci n'a pu être résolue, et donc il est impossible de savoir si l'image transmise est correct ou non.

Figure 75 : Trame UART de lecture de la caméra

10.7 Contrôle de l'orientation

Setup

Ce test a pour but de valider le fonctionnement des magnéto-coupleurs et des roues de réactions (moteurs DC). Pour cela, ces actuateurs ont été connectés aux bornes prévues à cet effet sur le circuit, puis ont été activés à travers l'application. Ensuite diverses valeurs de duty cycle ont été entrées et envoyées au système. Pour les moteurs, un simple test visuel permet de constater si le système fonctionne ou non. Pour les magnéto-coupleurs, la valeur de leur champ magnétique a été mesurée en utilisant les magnétomètres présents sur le circuit, ceux-ci ont donc été activés aussi.

Résultat attendu

Les moteurs DC doivent modifier leur sens et leur vitesse en fonction des valeurs des duty cycles.

La valeur du champ magnétique mesuré doit varier aussi en fonction du duty cycle.

Résultat obtenu

Les moteurs fonctionnent selon le résultat escompté, à quelques exceptions prêt. Ils ont de la peine à démarrer lorsque des duty cycles faibles sont entrés. De plus ils perdent en vitesse lorsque plusieurs actuateurs sont activés simultanément. Cela provient certainement du fait que les abaisseurs de tension faisant office d'alimentation pour ces actuateurs n'arrivent pas à fournir suffisamment de courant. Plusieurs solutions sont possibles : choisir des composants pouvant fournir plus de courant, choisir des moteurs moins gourmant en courant ou n'utiliser qu'un seul actuateur à la fois.

En ce qui concerne les magnéto-coupleurs, les mesures du magnétomètre ont montrés que ceux-ci fonctionnent correctement. Les valeurs de champ magnétique varient bien en fonction du duty cycle.

11 CONCLUSION

D'après le cahier des charges de ce projet de diplôme qui était :

- Analyser le système existant et identifier les modifications à y effectuer.
- Modifier le hardware/software existant pour une configuration réaliste d'un ADCS.
- Proposer et implémenter dans la FPGA de simple algorithme de stabilisation.

Deux objectifs sur trois ont été réalisés complètement, mais tous ont été traités. En effet l'objectif d'implémenter des algorithmes simples a été mis de côté suites aux discussions avec les responsable de l'EPFL, pour rappel ces algorithmes sont très dépendant de la mission du satellite et cette dernière n'a pas été déterminée. De plus ils peuvent vite s'avérer très complexes à implémenter et donc exiger un temps considérable. Etant donné la difficulté rencontrée pour implémenter juste le « petit » calcul des angles des Sun-Sensors, il est facile de s'imaginer la complexité à implémenter des calculs matriciels. L'idée ici a donc été de considérer un système de récolte/fusion de données provenant des capteurs et contrôle simple des actuateurs plutôt qu'un système plus sophistiqué utilisant des algorithmes pour piloter les actuateurs.

Ce qui en revient à l'objectif d'identifier les modifications à effectuer sur le système existant, qui a été accompli complètement. Les modifications se résument aux points suivants :

- Carte d'interface
    - o Ajout de cinq Sun-Sensors supplémentaires sur la carte d'interface
    - o Ajout de deux Magneto Torquers supplémentaires
    - o Ajout de trois Reaction Wheels supplémentaires
    - o Ajout de deux gyroscopes
    - o Ajout de deux magnétomètres
    - o Ajout d'un GPS
    - o Ajout d'une caméra faisant office de Star Tracker
- FPGA
    - o Modification du code VHDL suites aux ajouts des capteurs/actuateurs
- Application QT
    - o Modification du programme de démonstration pour l'affichage et le pilotage des nouveaux capteurs/actuateurs
- Autre
    - o Modification du protocole de communication UART entre la FPGA et le PC

En ce qui concerne le dernier objectif, celui-ci a été atteint étant donné qu'il consistait à effectuer les modifications précédemment déterminées. Un nouveau circuit a été monté comportant tous les capteurs et actuateurs souhaités et le protocole de communication ainsi que le code de la FPGA et l'application de démonstration ont subis les modifications nécessaires pour piloter ces nouveaux périphériques.

Quelques problèmes subsistent pourtant et empêchent un fonctionnement total du système :

- Les valeurs des Sun-Sensors sont correctement récupérées, mais le calcul pour en déduire les angles n'est pas correctement réalisé.
- Les valeurs des gyroscopes et des magnétomètres sont correctement récupérées et affichées, cependant un des magnétomètres ne fonctionne pas.
- La communication avec le GPS n'a pu être aboutie suite au problème du Clockstretching.
- Des valeurs pour les images de la caméra sont bien récupérées sur l'application mais un problème d'affichage fait planter cette dernière.
- Les PWMs pour les Magneto-Torquers et les Reactions-Wheels sont correctement générées selon les valeurs de Duty-Cycle entrées. Ces actuateurs fonctionnent correctement malgré les soucis observés lors des tests.

12 ORIENTATION FUTURE

Avant de pouvoir continuer plus loin, les différents problèmes des capteurs et actuateurs devraient être réglés, pour cela des pistes ont été présentées dans le chapitre de test. Une fois ces problèmes résolus, il serait possible de commencer à intégrer un algorithme dans la FPGA, ainsi qu'une interface en SPI ou autre pour permettre le pilotage du système par le reste du satellite. Point qui était prévu mais non réalisé par manque de temps.

13 BIBLIOGRAPHIE

[1] EPFL, « The first swiss satellite », [En ligne], http://swisscube.epfl.ch/, 03.05.14.

[2] Lovejoy Stéphane, « cubETH Command and Data Management System », Bachelor HES-SO Infotronic, 12.07.13.

[3] Praplan Bastien, « Carte processeur pour satellite », Master HES-SO TIN, 2013.

[4] Kozlovsky, « Principe d'un capteur gyroscopique », [En ligne], http://kozlovsky.pagesperso-orange.fr/cours/1_capteur_gyro.pdf, 03.05.14.

[5] Achouri Karim, « ADCS Simulation Platform: Reaction Wheels and Power System » Space Center EPFL, 03.02.12.

[6] Texas-Instruments, "datasheet ADS8028", mai 2011, révisé en mars 2012

[7] ublox, « u-blox 6 Receiver Description including Protocol Specification », [En ligne], http://www.u-blox.com/images/downloads/Product_Docs/u-blox6_ReceiverDescriptionProtocolSpec_%28GPS.G6-SW-10018%29.pdf, 09.07.14

[8] « NMEA-0183 messages : Overview », [En ligne], http://www.trimble.com/OEM_ReceiverHelp/V4.44/en/NMEA-0183messages_MessageOverview.html, 09.07.14

[9] Aptina, « datasheet caméra MT9V114 », 2006, révisé en juillet 2008

[10] EPFL, « Characterization and calibration of dual axis sun sensors for ADCS », 15.01.08

14 ANNEXES

Annexe 1 : Registre de configuration du convertisseur ADS8028

Annexe 2 : Algorithmes de stabilisation

Annexe 3 : Tableau récapitulatif des capteurs et actuateurs

Annexe 4 : Listes des messages du protocole NMEA

Annexe 5 : Schéma bloc du système

Annexe 6 : Schématique carte FPGA

Annexe 7 : Schématique carte Caméra

Annexe 8 : Schématique kit GPS

Annexe 9 : Schématique nouvelle carte d'interface

Annexe 10 : Calcul de surface nécessaire pour l'implémentation sur PCB

Annexe 11 : Listes des composants commandés

Annexe 12 : Tableau des coefficients des Sun-Sensors

Annexe 13 : Circuit FPGA complet

MSB

15	14	13	12	11	10	9	8
WRITE	REPEAT	AIN0	AIN1	AIN2	AIN3	AIN4	AIN5

LSB

7	6	5	4	3	2	1	0
AIN6	AIN7	T_{SENSE}	X	X	EXT_REF	TMP_AVG	STANDBY

Bit 15 **WRITE: Write to Control Register**

Enable write operation.
0 = Write disabled; Control Register is not updated and the next 15 bits are ignored (default)
1 = Write enabled; the next 15 bits update the Control Register

Bit 14 **REPEAT: Repeat conversion mode**

Enable conversion repeat mode (refer to the *Modes of Operation* section).
0 = Disable repeat conversion mode (default)
1 = Enable repeat conversion mode

Bits[13:6] **AIN[0:7]: Analog input channel selection**

Each AINx bit corresponds to the associated analog input channel, AIN0 to AIN7.
0 = AINx channel is not selected for conversion (default)
1 = AINx channel is selected for conversion

Bit 5 **T_{SENSE}: Internal temperature sensor selection**

Internal temperature sensor selection for conversion in subsequent cycles.
0 = Internal temperature sensor output is not selected for conversion (default)
1 = Internal temperature sensor output is selected for conversion

Bits[4:3] **X: Don't care**

Bit 2 **EXT_REF: Reference source selection**

This bit selects the reference source for the next conversion.
0 = Internal reference is used for the next conversion (default)
1 = External reference is used for the next conversion

Bit 1 **TMP_AVG: Temperature sensor averaging selection**

This bit selects the mode of operation for the temperature sensor channel; this bit is ignored if bit 5 is set to '0'.
0 = Averaging is disabled on the temperature sensor result (default)
1 = Averaging is enabled on the temperature sensor result

Bit 0 **STANDBY: STANDBY mode selection**

This bit sets the mode (normal or standby) for the ADS8028.
0 = The ADS8028 operates in normal mode (default)
1 = The ADS8028 goes to standby mode in the next cycle

B-Dot Algorithm

Tiré du document : Brady W. Young, “Design and Specification of an Attitude Control System for the DANDE Mission”, MIT, 2006

The B-dot control law is an active control algorithm to stabilize a randomly tumbling spacecraft and bring it to rest with respect to the local magnetic field vector. The algorithm monitors the Earth's local magnetic field B in the direction of one magnetic torque rod fixed to the body frame, takes the body-fixed first derivative $\dot{B}$ (hence the name “B-dot"), and creates a magnetic dipole in the opposite direction. This acts to create a magnetic torque opposing the current rotation of the spacecraft. Conversely, creating a dipole in the same direction as $\dot{B}$ acts to accelerate a body.

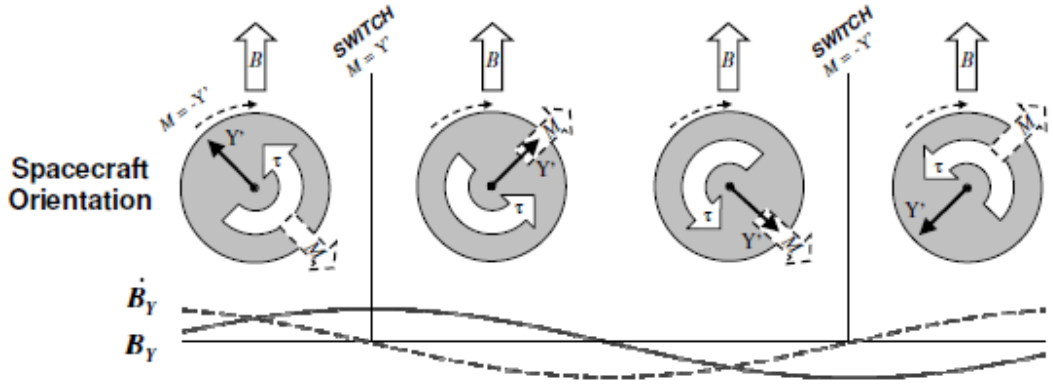


Figure 1 : B-dot law concept

The B-dot control law concept: As the spacecraft rotates so that the y-axis more closely aligns with the Earth's B field, the first derivative of the B-field in the y direction is positive. The y-axis torque rod creates a magnetic dipole in the -y direction. By _ = M _ B, a torque opposing the spin rate is generated. The dipole direction is reversed when the y-axis passes the B field direction and $\dot{B}_Y$ becomes negative.

TRIAD Algorithm

Tiré de : Wikipédia, « Triad Method »,[En ligne], http://en.wikipedia.org/wiki/Triad_method, 10.07.14

We consider the linearly independent reference vectors $\vec{R}_1$ and $\vec{R}_2$. Let $\vec{r}_1, \vec{r}_2$ be the corresponding measured directions of the reference unit vectors as resolved in a body fixed frame of reference. Then they are related by the equations,

$$\vec{R}_i = A\vec{r}_i$$

for $i = 1, 2$, where A is a rotation matrix (sometimes also known as a proper orthogonal matrix, i.e., $A^T A = I, det(A) = +1$). A transforms vectors in the body fixed frame into the frame of the reference vectors. Among other properties, rotational matrices preserve the length of the vector they operate on. Note that the direction cosine matrix A also transforms the cross product vector, written as,

$$\vec{R}_1 \times \vec{R}_2 = A\left(\vec{r}_1 \times \vec{r}_2\right)$$

Triad proposes an estimate of the direction cosine matrix A as a solution to the linear system equations given by

$$\left[\vec{R}_1 \vdots \vec{R}_2 \vdots \left(\vec{R}_1 \times \vec{R}_2\right)\right] = A\left[\vec{r}_1 \vdots \vec{r}_2 \vdots \left(\vec{r}_1 \times \vec{r}_2\right)\right]$$

where $\vdots$ have been used to separate different column vectors.

The solution presented above works well in the noise-free case. However, in practice, $\vec{r}_1, \vec{r}_2$ are noisy and the orthogonality condition of the attitude matrix (or the direction cosine matrix) is not preserved by the above procedure. Triad incorporates the following elegant procedure to redress this problem. To this end, we define unit vectors

$$\hat{S} = \frac{\vec{R}_1}{||\vec{R}_1||}$$

$$\hat{s} = \frac{\vec{r}_1}{||\vec{r}_1||}$$

And

$$\hat{M} = \frac{\vec{R}_1 \times \vec{R}_2}{||\vec{R}_1 \times \vec{R}_2||}$$

$$\hat{m} = \frac{\vec{r}_1 \times \vec{r}_2}{||\vec{r}_1 \times \vec{r}_2||}$$

to be used in place of the first two columns of (**3**). Their cross product is used as the third column in the linear system of equations obtaining a proper orthogonal matrix for the spacecraft attitude given by

$$\left[\hat{S} \vdots \hat{M} \vdots \hat{S} \times \hat{M}\right] = A\left[\hat{s} \vdots \hat{m} \vdots \hat{s} \times \hat{m}\right]$$

While the normalizations of Equations (**4**) - (**7**) are not necessary, they have been carried out to achieve a computational advantage in solving the linear system of equations in (**8**). Thus an estimate of the spacecraft attitude is given by the proper orthogonal matrix as

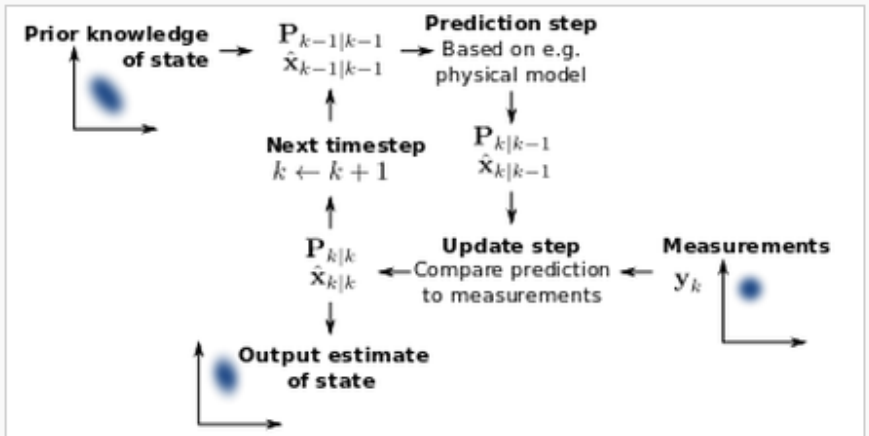
$$\hat{A} = \left[\hat{S} \vdots \hat{M} \vdots \hat{S} \times \hat{M}\right]\left[\hat{s} \vdots \hat{m} \vdots \hat{s} \times \hat{m}\right]^T.$$

Note that computational efficiency has been achieved in this procedure by replacing the matrix inverse with a transpose. Equation(**8**) shows that the matrices used for computing attitude are each composed of an orthogonal triad of basis vectors. "TRIAD" derives its name from this observation.

Kalman Filter

Tiré de : Wikipédia, « Kalman Filter », [En ligne], http://en.wikipedia.org/wiki/Kalman_filter, 10.07.14

The Kalman filter, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing noise (random variations) and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. More formally, the Kalman filter operates recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state. The filter is named forRudolf (Rudy) E. Kálmán, one of the primary developers of its theory.



The Kalman filter keeps track of the estimated state of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements. $\hat{x}_{k|k-1}$ denotes the estimate of the system's state at time step *k* before the *k*-th measurement y_k has been taken into account; $P_{k|k-1}$ is the corresponding uncertainty.

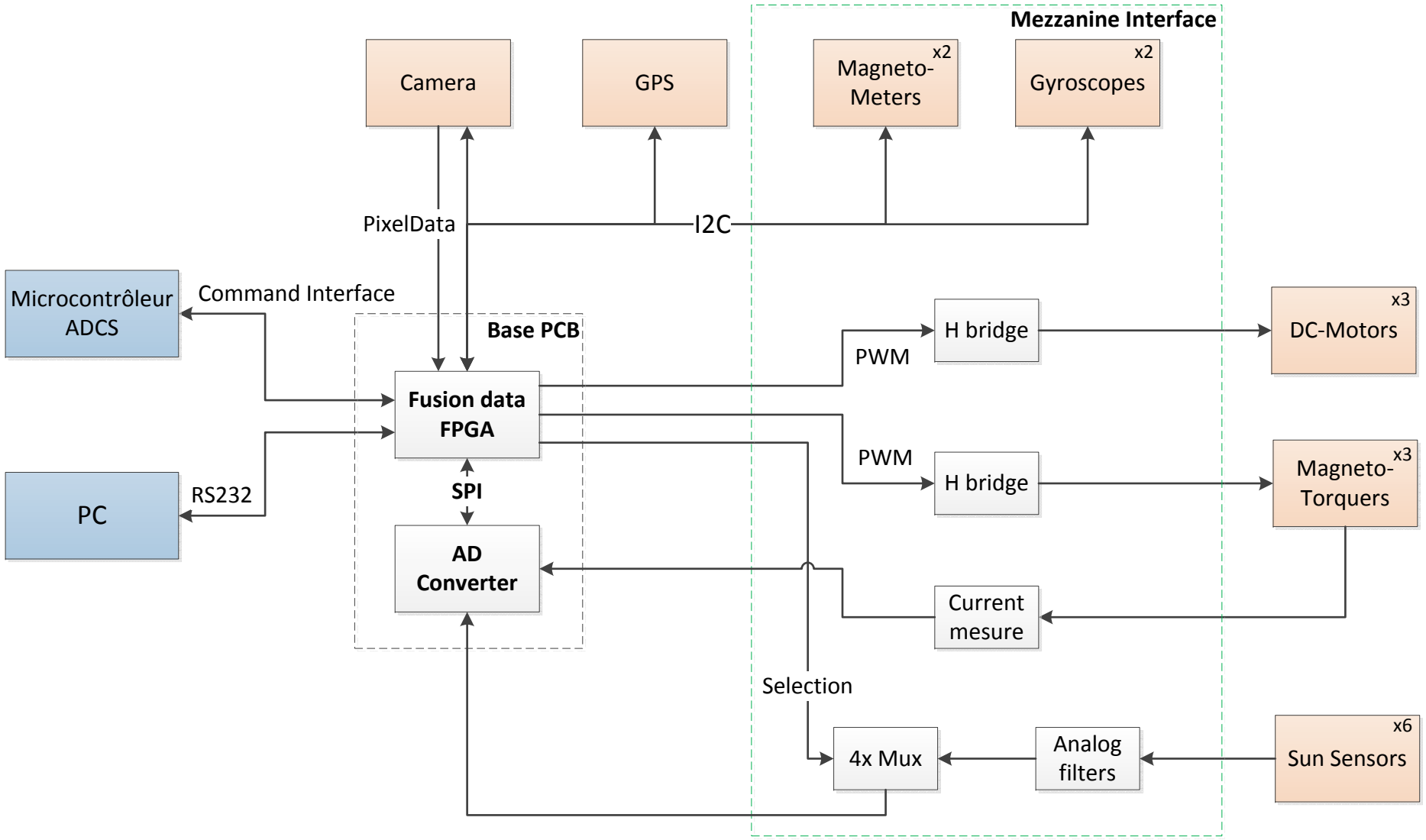
Figure 2 : Kalman Filter concept

List of sensors-actuators

Sensors	Specifications	Reference	Interfaces(Data,Power)	Particular processing	Availability	Costs, delivery
Sun-Sensor	Permet d'obtenir l'angle sur deux axes par rapport à la source lumineuse	Space Center Sun Sensor	Out : 4 values, 2 for each axis / Power: 3V	Low-Pass filter, must be digitalised	Yes	-
Gyroscope	Permet d'obtenir l'angle de rotation sur les axes X, Y et Z	ITG-3200 InvenSense	I2C, Power :2.1V to 3.6V	Digitally-programmable low-pass filter	Sparkfun, Farnell	21.98 SFr, 31.10 SFr
		L3G4200D Stmicroelectronics	I2C/SPI, power : 2.4 V to 3.6 V	Programmable FS range (250,500,2000dps) Digitally-programmable low-pass filter On-chip temperature sensor	Sparkfun, Mouser	44.01 SFr, 6.50 SFr
Magneto-meter	Permet de mesure le champ magnétique selon les axes X, Y et Z	MAG3110, Freescale Semiconductor	I2C, Power :1.95V to 3.6V	-	Sparkfun, Mouser	13.17 SFr, 1.05 SFr
		HMC5883L, Honeywell	I2C, Power : 2.16 to 3.6VDC	-	Sparkfun, Farnell	13.17 SFr, 7.- SFr

Actuators	Specifications	Reference	Interfaces(Data,Power)	Particular processing	Availability	Costs, delivery
Magneto-torquer	Permet d'effectuer des rotations du satellite grâce au champ magnétique de la terre et celui du magnetorquer	Space Center Magneto-Torquer	In : PWM 32.5Khz / Power : 3.0V	H-Bridge required	2(engineering models)+2(dummy models) from the space center	-
Reaction-Wheel	Permet d'effectuer des rotations plus précises que les magnetotorquers grâce aux rotations d'une petite masse	DC Motors	PWM	H-Bridge required	To be checked at the SwissSpace Center	DC : < 20 SFr Brushless : > 100 SFr
GPS	Permet d'obtenir les coordonées de longitude et lattitude de la positon actuelle	Ublox / MIKROE 1032	I2C, UART	Antenna required	Farnell	93.75 SFr
Star-Tracker, Camera	Permet de calculer la position actuelle grâce à de simples photographies des étoiles	MT9V114	I2C, parallel datas, 640x480pixels	RAM required for data	Available at school	-

Message	Function
ADV	Position and satellite information for RTK network operations
DTM	Datum reference information
GBS	GNSS satellite fault detection (RAIM support)
GGA	Time, position, and fix related data
GLL	Position data: position fix, time of position fix, and status
GNS	GNS Fix data
GRS	GRS range residuals
GSA	GPS DOP and active satellites
GST	Position error statistics
GSV	Number of SVs in view, PRN, elevation, azimuth, and SNR
HDT	Heading from True North
LLQ	Leica local position and quality
PFUGDP	A proprietary message containing information about the type of positioning system, position, number of satellites and position statistics
PTNL,AVR	Time, yaw, tilt, range, mode, PDOP, and number of SVs for Moving Baseline RTK
PTNL,BPQ	Base station position and position quality indicator
PTNL,DG	L-band corrections and beacon signal strength and related information
PTNL,GGK	Time, position, position type, and DOP values
PTNL,PJK	Time, position, position type, and DOP values
PTNL,PJT	Projection type
PTNL,VGK	Time, locator vector, type, and DOP values
PTNL,VHD	Heading Information
RMC	Position, Velocity, and Time
ROT	Rate of turn
VTG	Actual track made good and speed over ground
ZDA	UTC day, month, and year, and local time zone offset


Mezzanine Interface
Camera
GPS
Magneto-Meters
x2
Gyroscopes
x2
PixelData
I2C
Microcontrôleur ADCS
Command Interface
Base PCB
H bridge
PWM
DC-Motors
x3
Fusion data FPGA
PWM
H bridge
Magneto-Torquers
x3
PC
RS232
SPI
AD Converter
Current mesure
Selection
4x Mux
Analog filters
Sun Sensors
x6

HEV–SO \\ Valais Wallis
Route du Rawyl 47
1950 Sion
www.hevs.ch

Designer:
Praplan Bastien
bastien.praplan@hevs.ch

Sat_FPGA Board: Board satelite with FPGA

Page	Title	Description
1	Cover	This first page
2	Power	3.3V 1.8V 1.2V Regulation
3	FPGA Power	Xilinx Spartan 6 Power and Decoupling
4	FPGA 1	Xilinx Spartan 6 Bank 0 and 2
5	FPGA 2	Xilinx Spartan 6 Bank 1 and 3
6	FPGA Config	Oscillator / JTAG Connector / Reset Circuit / Done
7	Memory Flash	FLASH
8	Memory RAM	SDRAM
9	Mezzanine	Mezza Connectors / CAN tranceiver
10	AOCS & PL	AOCS & Payload Connectors / ADC
11	UART	UART tranceiver / UART Connector
12		

Board satelite with FPGA Sat_FPGA Cover		DES	{Date} prb
		REV	V1.0
HAUTE ECOLE VALAISANNE		1/25	{Path} Board_FPGA.sch

Power supply

3.3V –> From LTM4615 (1)
1.2V –> From LTM4615 (2)
1.8V –> From LTM4615 LDO

Layout Hints
– THERMAL_SW1 and THERMAL_SW2 Floating on seperate copper planes
– Decoupling capacitors close to pins

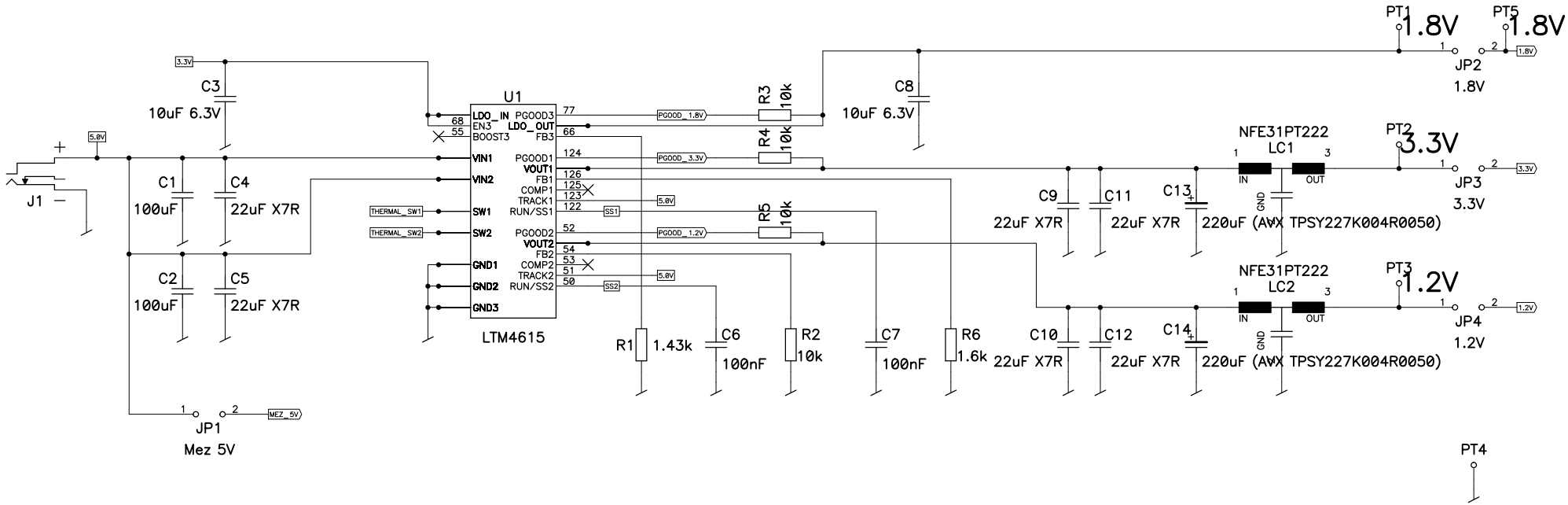
LDO VOUT
VLDO_OUT = 0.4V*(4.99k+RFB_LDO)/RFB_LDO
–> RFB_LDO = 1.996k / (VLDO_OUT – 0.4V)
RFB_LDO (Vout = 1.8V) = 1.996k / (1.8 – 0.4) = 1.42571k –> 1.43k 1% –> 1.80563V

VOUT = 0.8V * (4.99k + RFB) / RFB
–> RFB = 3.992k / (Vout – 0.8)

RFB (Vout = 3.3V) = 3.992k / (3.3 – 0.8) = 1.5968k –> 1.6k 1% –> 3.295 V

RFB (Vout = 1.2V) = 3.992k / (1.2 – 0.8) = 9.98k –> 10k 1%–> 1.1992 V

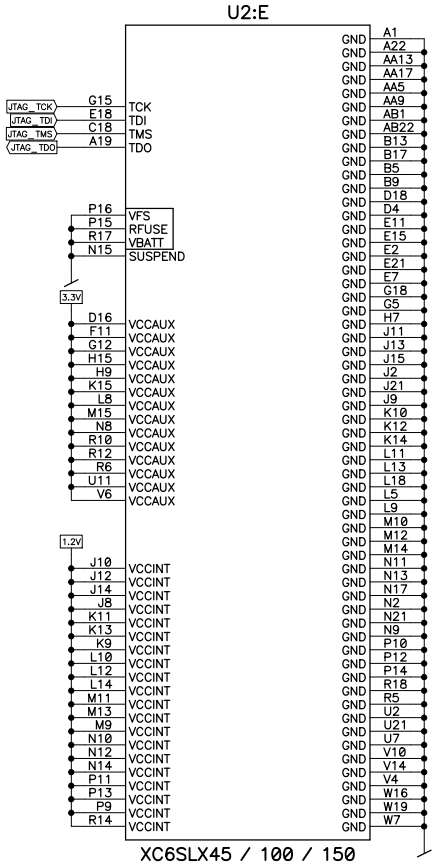
3.3V @ 4A / 1.2V @ 4A / 1.8V @ 1.5A



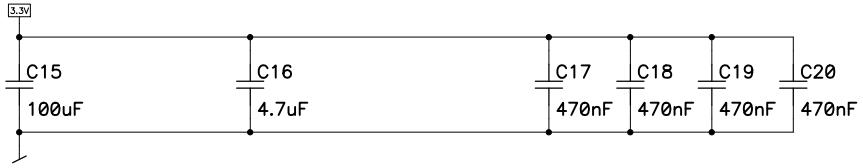
Board satelite with FPGA		DES	{Date} prb
Sat_FPGA	Power	REV	V1.0
HAUTE ECOLE VALAISANNE		2/25	{Path} Board_FPGA.sch

FPGA Power

Pins NOT AVAILABLE ON LX45
ONLY AVAILABLE ON LX100 / 150
– VFS
– RFUSE
– VBATT

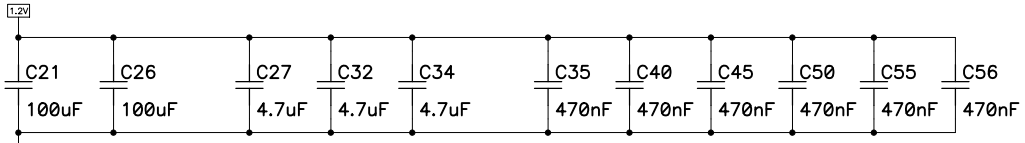


VCC0 Bank 3 Decoupling
LX45, LX100, LX150 –> 1x100uF / 1x4.7uF / 4x0.47uF

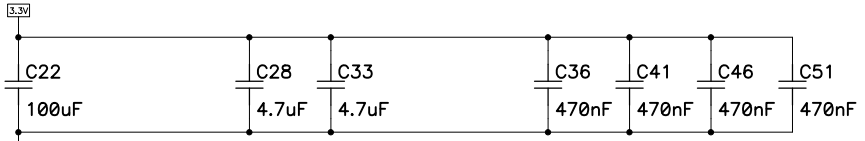


FPGA Decoupling

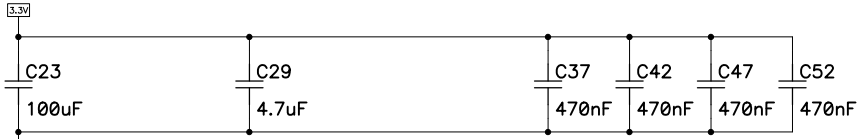
VCCINT Decoupling
LX45 –> 1x100uF / 1x4.7uF / 2x0.47uF
LX100 –> 1x100uF / 2x4.7uF / 4x0.47uF
LX150 –> 2x100uF / 3x4.7uF / 6x0.47uF



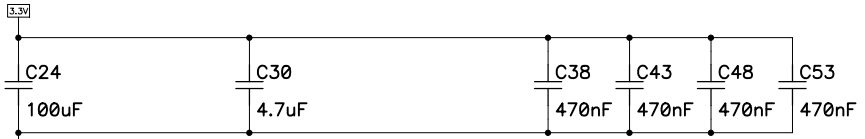
VCCAUX Decoupling
LX45, LX100, LX150 –> 1x100uF / 2x4.7uF / 4x0.47uF



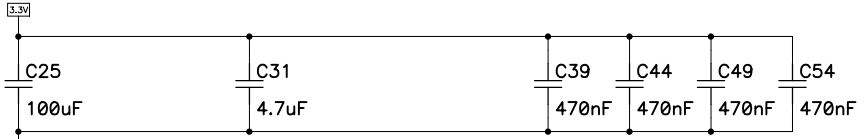
VCC0 Bank 0 Decoupling
LX45, LX100, LX150 –> 1x100uF / 1x4.7uF / 2x0.47uF



VCC0 Bank 1 Decoupling
LX45, LX100, LX150 –> 1x100uF / 1x4.7uF / 4x0.47uF



VCC0 Bank 2 Decoupling
LX45 –> 1x100uF / 1x4.7uF / 2x0.47uF
LX100, LX150 –> 1x100uF / 1x4.7uF / 2x0.47uF



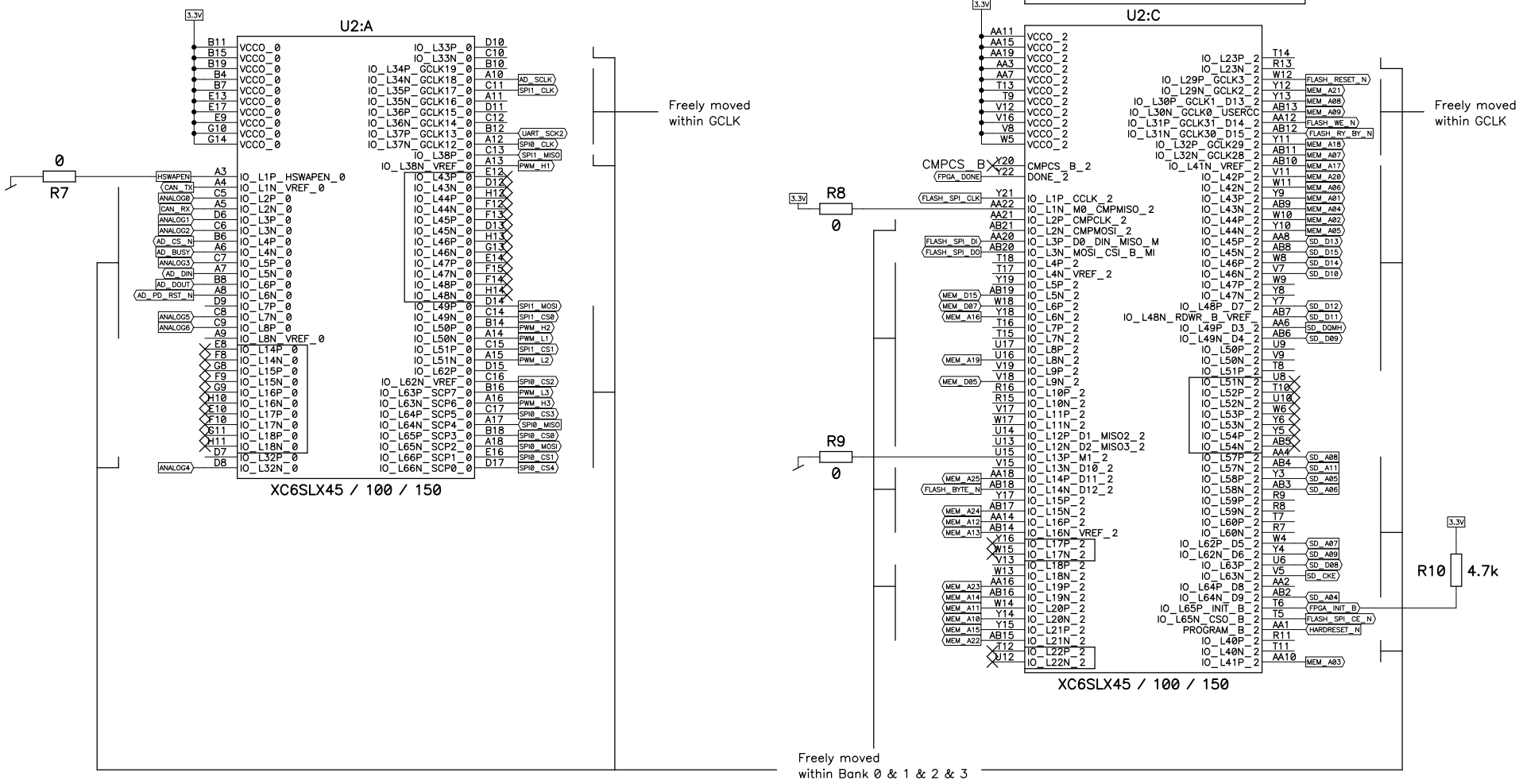
Board satelite with FPGA Sat_FPGA FPGA Power	DES	{Date} prb
	REV	V1.0
HAUTE ECOLE VALAISANNE	3/25	{Path} Board_FPGA.sch

FPGA Bank 0 & 2

Pins NOT AVAILABLE ON LX45
ONLY AVAILABLE ON LX100 / 150
- IO_L14P_0, IO_L14N_0, IO_L15P_0, IO_L15N_0, IO_L16P_0, IO_L16N_0, IO_L17P_0, IO_L17N_0, IO_L18P_0

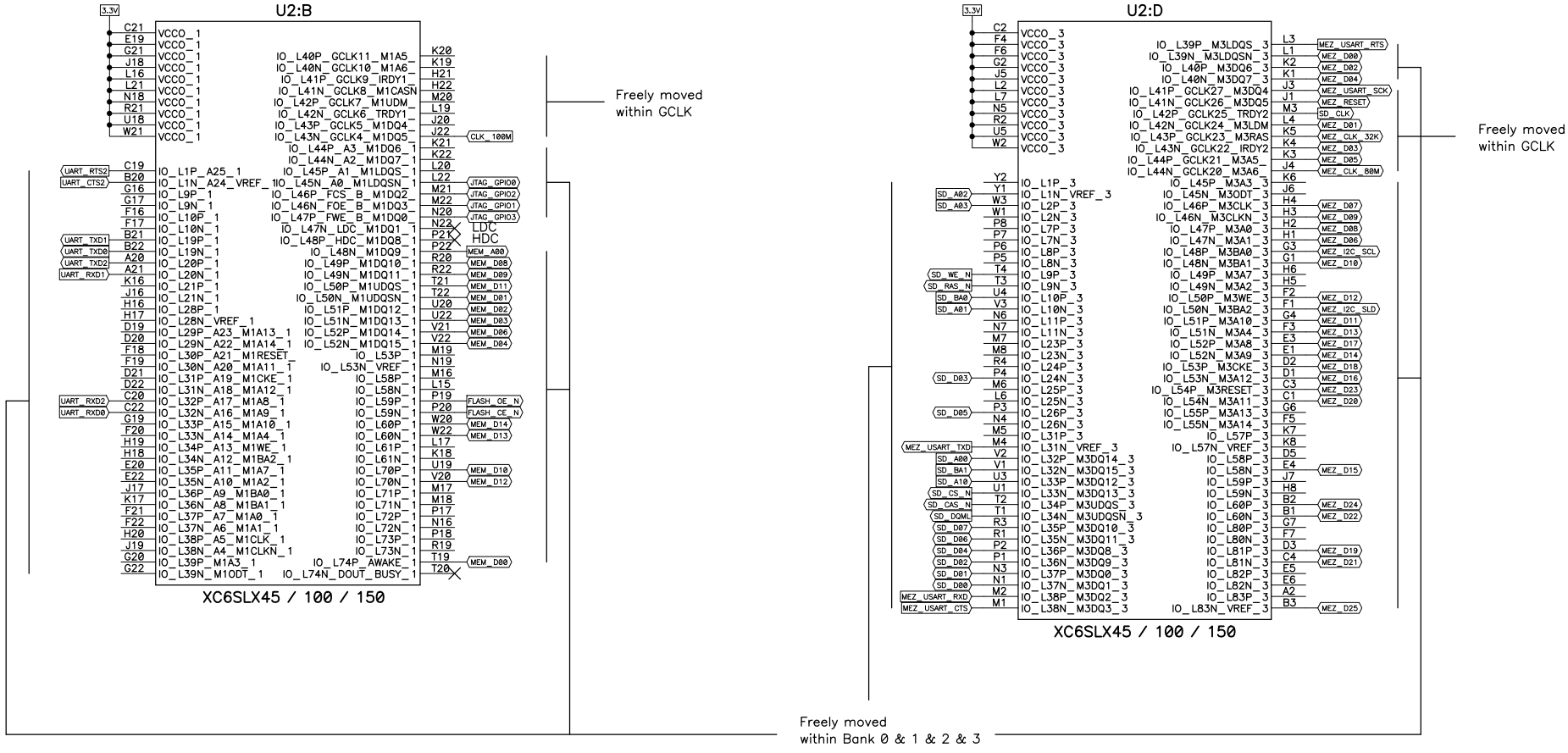
BPI Programming mode
M0 = 0
M1 = 0
SPI Programming mode
M0 = 1
M1 = 0

Pins NOT AVAILABLE ON LX100
ONLY AVAILABLE ON LX45 / 150
- IO_L17P_2, IO_L17N_2
- IO_L22P_2, IO_L22N_2
- IO_L51P_2, IO_L51N_2, IO_L52P_2, IO_L52N_2, IO_L53P_2, IO_L53N_2, IO_L54P_2, IO_L54N_2



Board satelite with FPGA Sat_FPGA FPGA 1	DES	{Date}	prb
	REV	V1.0	
HAUTE ECOLE VALAISANNE	4/25	{Path} Board_FPGA.sch	

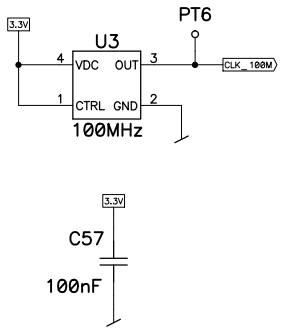
FPGA Bank 1 & 3



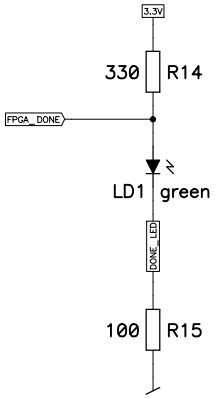
Board satelite with FPGA Sat_FPGA	FPGA 2	DES	{Date} prb
		REV	V1.0
HAUTE ECOLE VALAISANNE		5/25	{Path} Board_FPGA.sch

FPGA Config

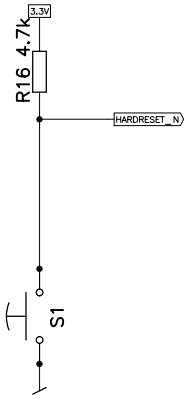
Main Oscillator



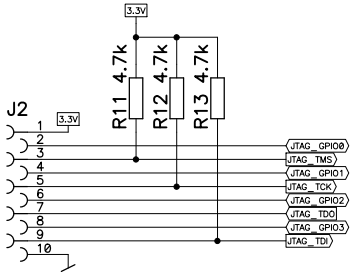
FPGA Done LED



FPGA Hardreset Reprogram



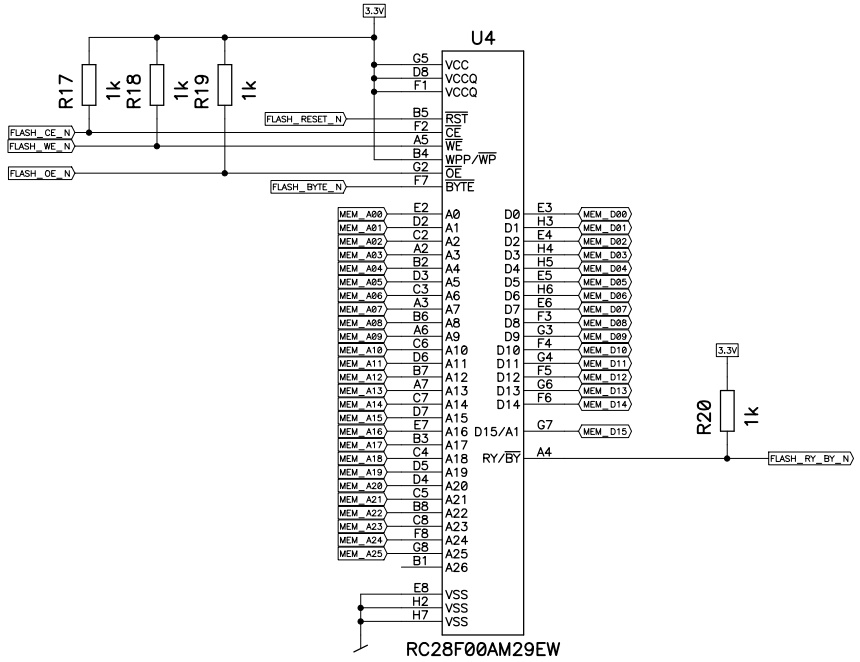
JTAG connector



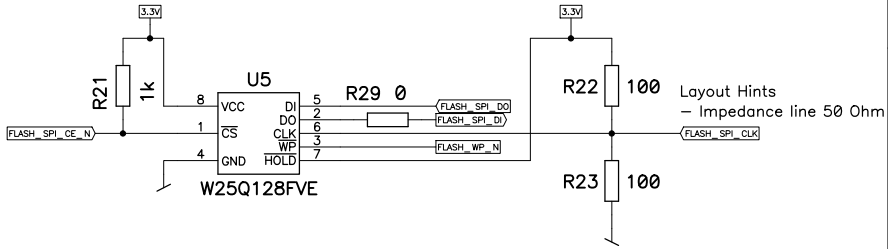
Board satelite with FPGA Sat_FPGA FPGA Config		DES	{Date} prb
		REV	V1.0
HAUTE ECOLE VALAISANNE		6/25	{Path} Board_FPGA.sch

Memory Flash

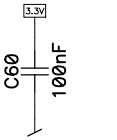
Flash 128MB

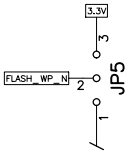


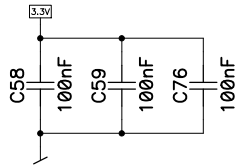
Flash SPI 16MB



Layout Hints
– Impedance line 50 Ohm



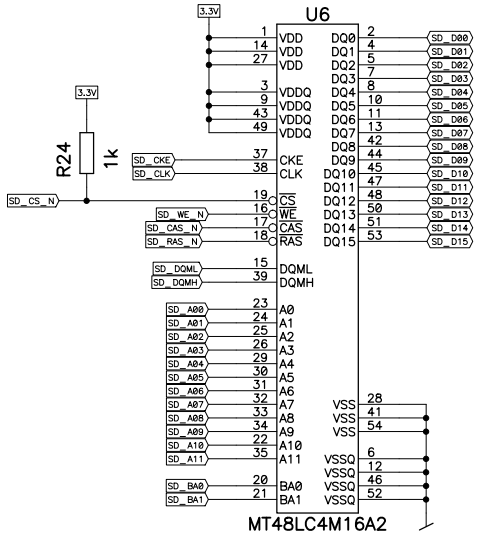


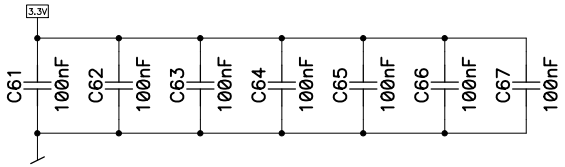


Board satelite with FPGA Sat_FPGA — Memory Flash	DES	{Date} prb
	REV	V1.0
HAUTE ECOLE VALAISANNE	7/25	{Path} Board_FPGA.sch

Memory RAM

SDRAM

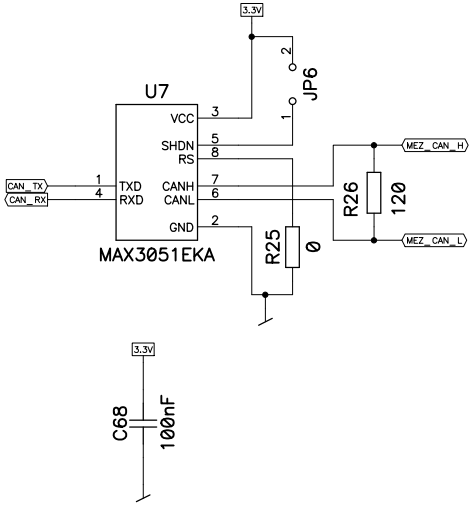




Board satelite with FPGA Sat_FPGA — Memory RAM		DES	{Date} prb
		REV	V1.0
HAUTE ECOLE VALAISANNE		8/25	{Path} Board_FPGA.sch

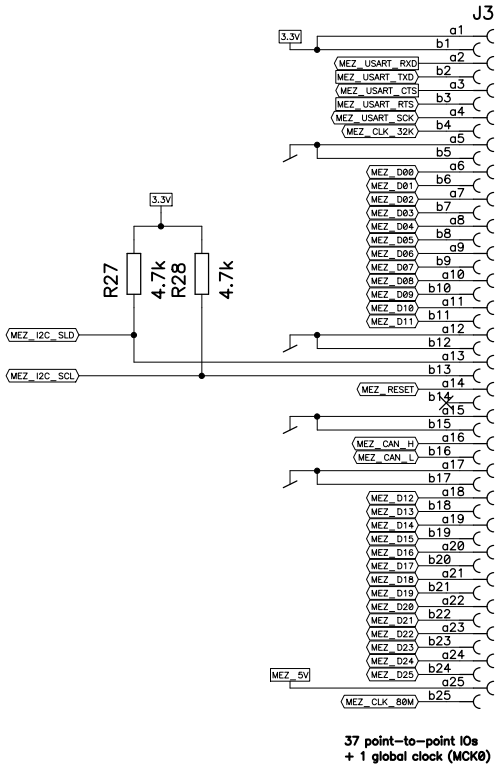
Mezzanine

CAN tranceiver



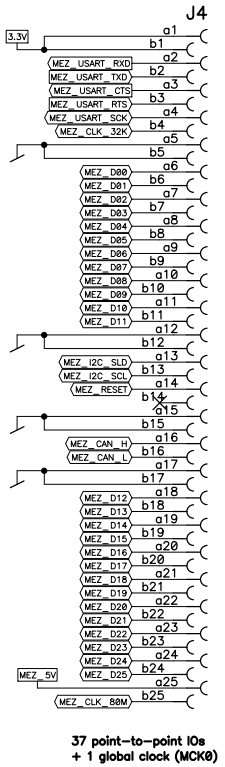
Mezza

TOP SIDE



37 point-to-point IOs
+ 1 global clock (MCK0)

BOTTOM SIDE

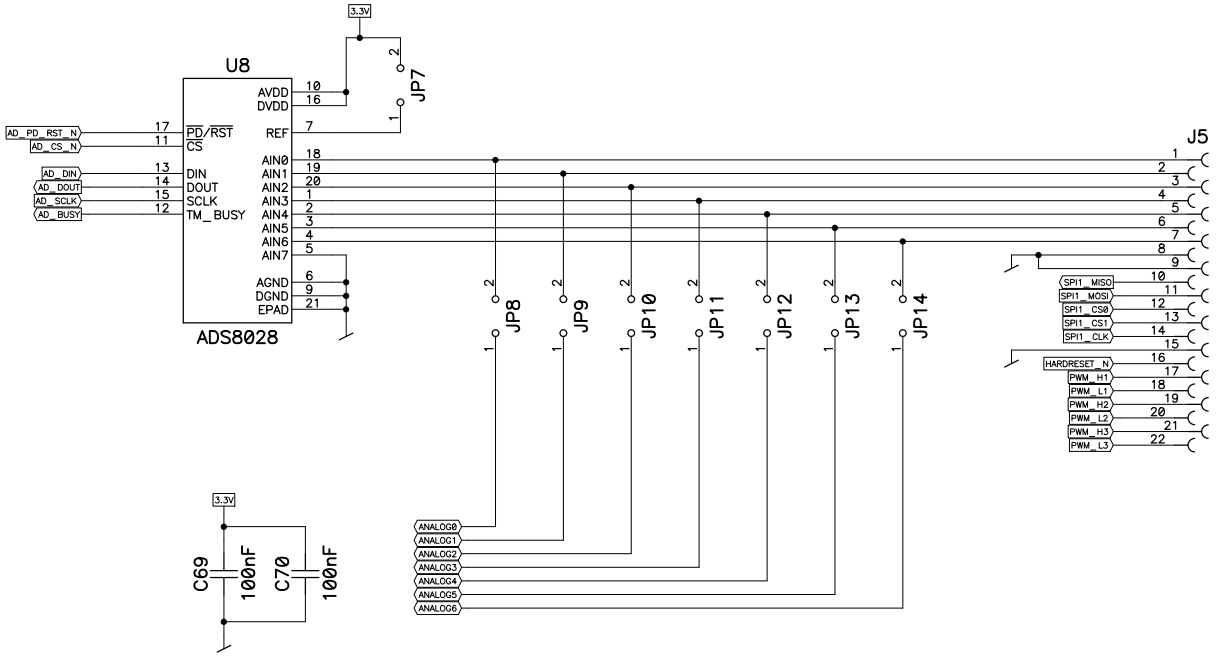


37 point-to-point IOs
+ 1 global clock (MCK0)

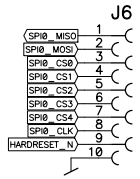
Board satelite with FPGA Sat_FPGA Mezzanine	DES	{Date} prb
	REV	V1.0
HAUTE ECOLE VALAISANNE	9/25	{Path} Board_FPGA.sch

AOCS & Payload

AOCS I/O



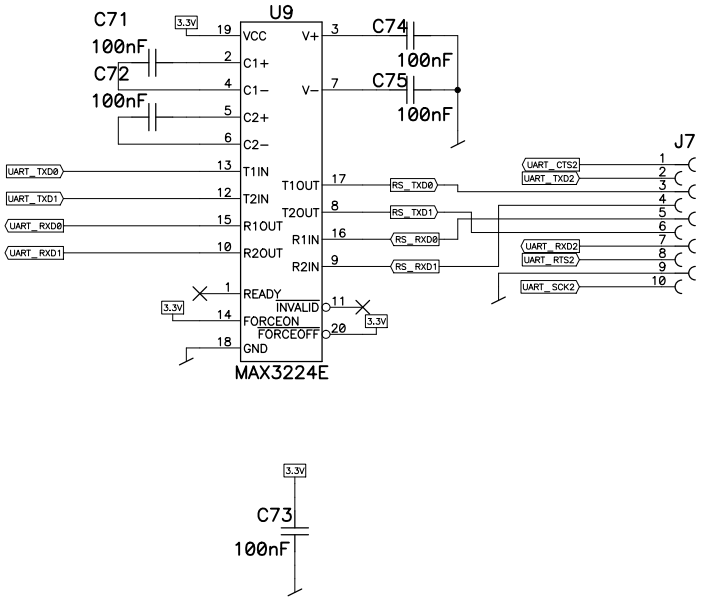
Payload I/O



Board satelite with FPGA		DES	{Date} prb
Sat_FPGA	AOCS & PL	REV	V1.0
HAUTE ECOLE VALAISANNE		10/25	{Path} Board_FPGA.sch

UART

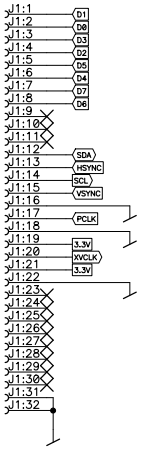
RS 232 Serial ports



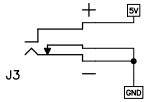
Board satelite with FPGA Sat_FPGA UART		DES	{Date} prb
		REV	V1.0
HAUTE ECOLE VALAISANNE		11/25	{Path} Board_FPGA.sch

Conntector with ov538 demo board

2x16 pins connector

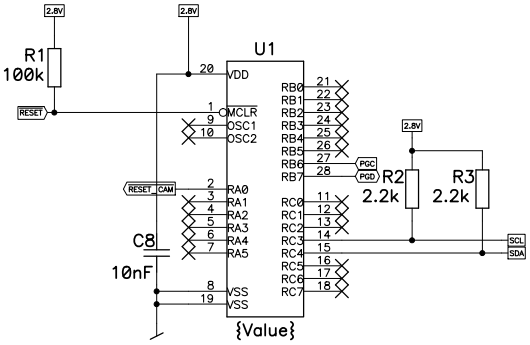


5V for LEDs

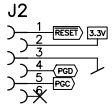


Microcontroller

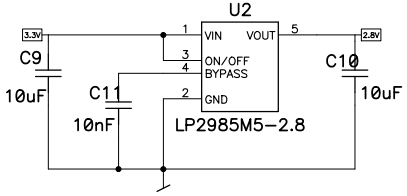
For doing i2c initialization



PIC programming



Power supply

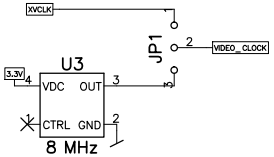


Clock (+selection)

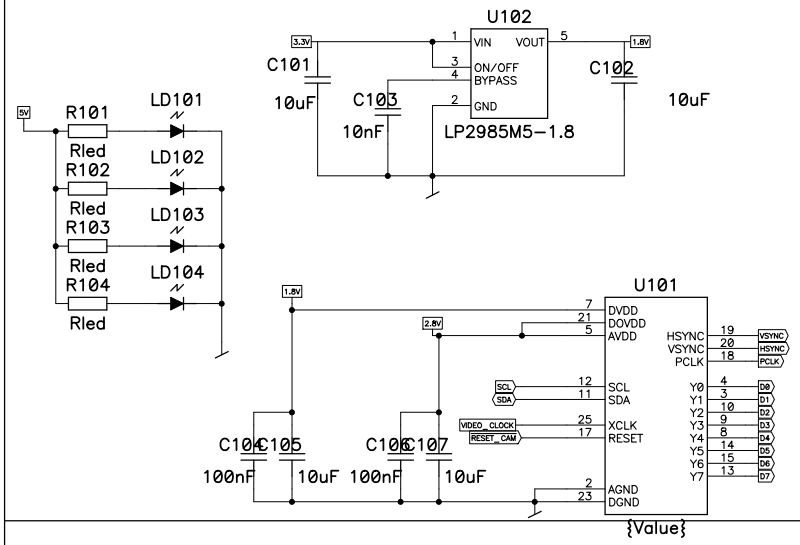
JP1 : clock select

from usb : 1–2

local : 2–3



Aptina MT9V114



HEB (Higher Educational Board) heb_camera : Interface	DES	2010-06-15pim
	REV	v1.0a
Copyright (C) 2003, Amontec & HEVs	3/5	{Path} moduleMT9V114.sch

GPS click™

1. Introduction

GPS Click is an accessory board in **mikroBus™** form factor. It's a compact solution for adding GPS functionality to your device. It features **LEA-6S,** high performance u-blox 6 positioning engine. Board can be interfaced with a microcontroller through UART or I²C connection, or data can be acquired using PC application through USB connection. Board features connector compatible with active and passive antennas. It can operate on 3.3V power supply only.

2. Soldering the headers

Before using your click board, make sure to solder the provided 1x8 male headers to both sides of the board. Two 1x8 male headers are included with the board in the package.

1

2

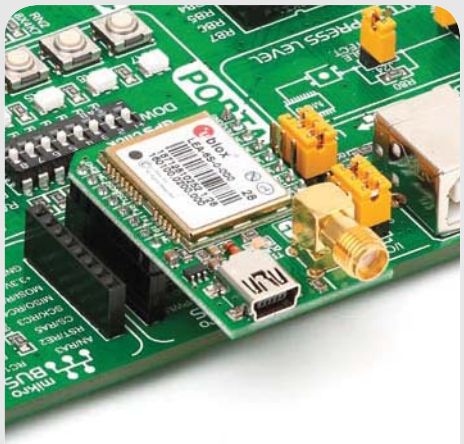
Turn the board upside down, so that bottom side is facing you upwards. Place shorter parts of the header pins in the both soldering pad locations.

3

Turn the board upward again. Make sure to align the headers so that they are perpendicular to the board, then solder the pins carefully.

3. Plugging the board in

Once you have soldered the headers your board is ready to be placed into desired mikroBUS™ socket. Make sure to align the cut in the lower-right part of the board with the markings on the silkscreen at the mikroBUS™ socket. If all the pins are aligned correctly, push the board all the way into the socket.



4. Features and applications

GPS Click board with it's **LEA-6S** module features acquisition down to 1s, -147dBm coldstart sensitivity and 5Hz update rate. You can achive low power consumption due to intelligent, user configurable power management. Mentioned features make this board ideal for asset tracking, road navigation devices, public transportation vehicle information systems and more.

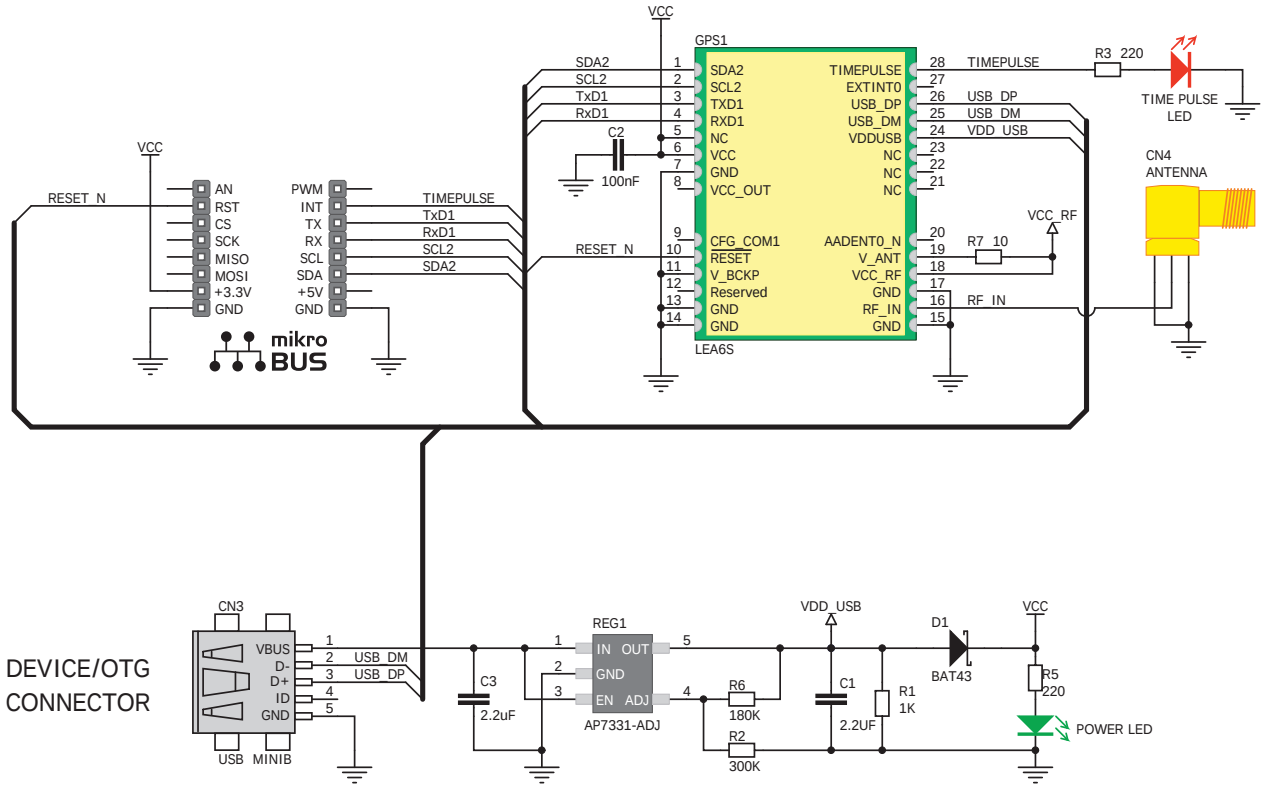
click™ BOARD

www.mikroe.com

GPS click Manual
ver. 1.00

0 100000 018637

5. GPS click Board Schematics



MikroElektronika assumes no responsibility or liability for any errors or inaccuracies that may appear in the present document.
Specification and information contained in the present schematic are subject to change at any time without notice. Copyright © 2012 MikroElektronika. All rights reserved.

6. Power supply - 3.3V only

Board is designed to use 3.3V power supply only. If you need to add GPS feature to your 5V prototype or development board, we recommend you to use other boards such as the SmartGPS Accessory Board:

http://www.mikroe.com/eng/products/view/122/smartgps-board/

7. Code Examples

Once you have done all the necessary preparations, it's time to get your click board up and running. We have provided the examples for mikroC, mikroBasic and mikroPascal compilers on our **Libstock** website. Just download them and you are ready to start.

8. Support

MikroElektronika offers **Free Tech Support (www.mikroe.com/esupport)** until the end of product lifetime, so if something goes wrong, we are ready and willing to help!

MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

GPS click™

1. Introduction

GPS Click is an accessory board in **mikroBus™** form factor. It's a compact solution for adding GPS functionality to your device. It features **LEA-6S,** high performance u-blox 6 positioning engine. Board can be interfaced with a microcontroller through UART or I²C connection, or data can be acquired using PC application through USB connection. Board features connector compatible with active and passive antennas. It can operate on 3.3V power supply only.

2. Soldering the headers

Before using your click board, make sure to solder the provided 1x8 male headers to both sides of the board. Two 1x8 male headers are included with the board in the package.

1

2

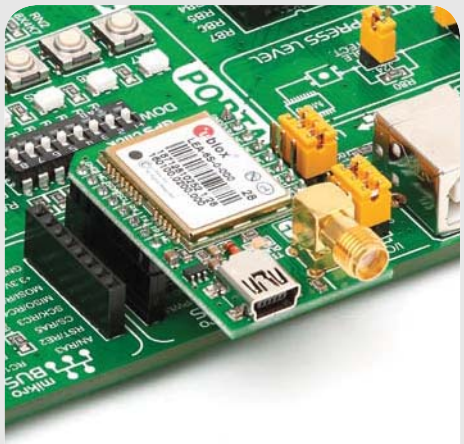
Turn the board upside down, so that bottom side is facing you upwards. Place shorter parts of the header pins in the both soldering pad locations.

3

Turn the board upward again. Make sure to align the headers so that they are perpendicular to the board, then solder the pins carefully.

3. Plugging the board in

Once you have soldered the headers your board is ready to be placed into desired mikroBUS™ socket. Make sure to align the cut in the lower-right part of the board with the markings on the silkscreen at the mikroBUS™ socket. If all the pins are aligned correctly, push the board all the way into the socket.



4. Features and applications

GPS Click board with it's **LEA-6S** module features acquisition down to 1s, -147dBm coldstart sensitivity and 5Hz update rate. You can achive low power consumption due to intelligent, user configurable power management. Mentioned features make this board ideal for asset tracking, road navigation devices, public transportation vehicle information systems and more.

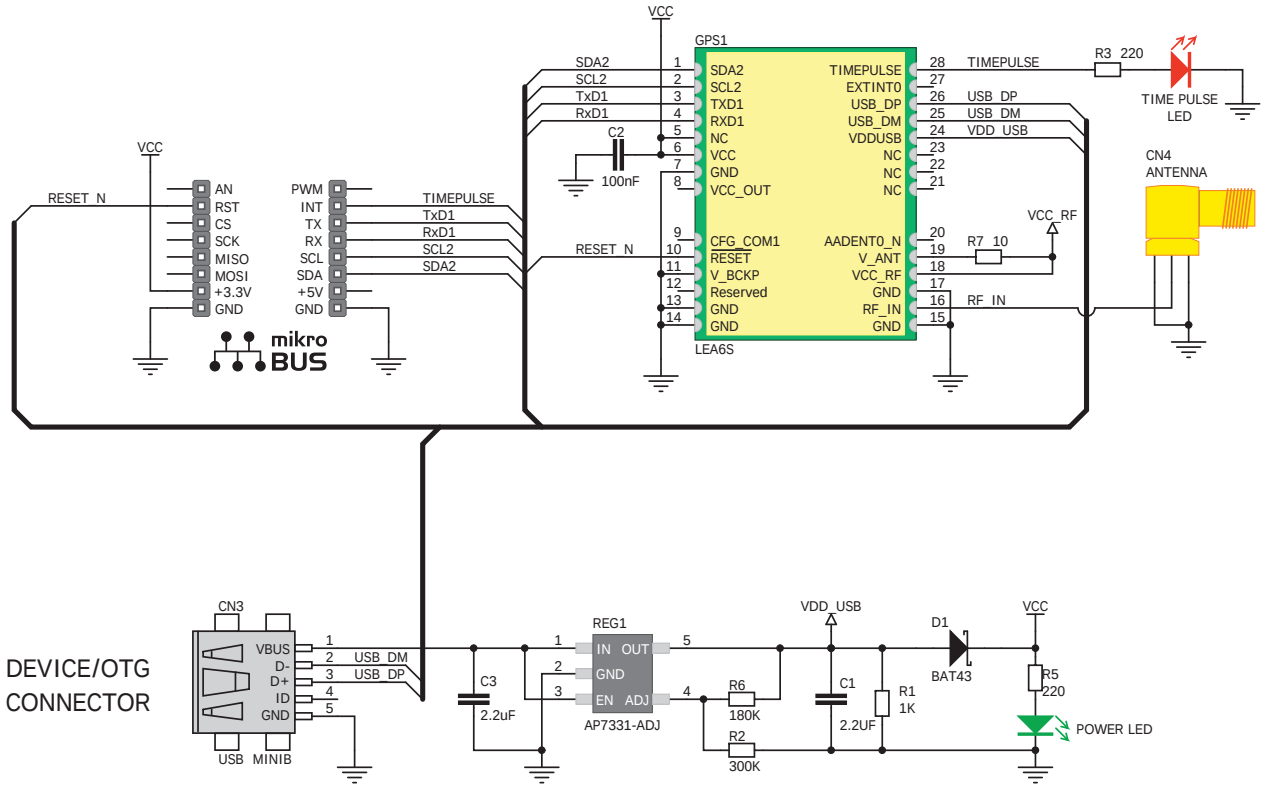
click™ BOARD

www.mikroe.com

GPS click Manual
ver. 1.00

0 100000 018637

5. GPS click Board Schematics



MikroElektronika assumes no responsibility or liability for any errors or inaccuracies that may appear in the present document.
Specification and information contained in the present schematic are subject to change at any time without notice. Copyright © 2012 MikroElektronika. All rights reserved.

6. Power supply - 3.3V only

Board is designed to use 3.3V power supply only. If you need to add GPS feature to your 5V prototype or development board, we recommend you to use other boards such as the SmartGPS Accessory Board:

http://www.mikroe.com/eng/products/view/122/smartgps-board/

7. Code Examples

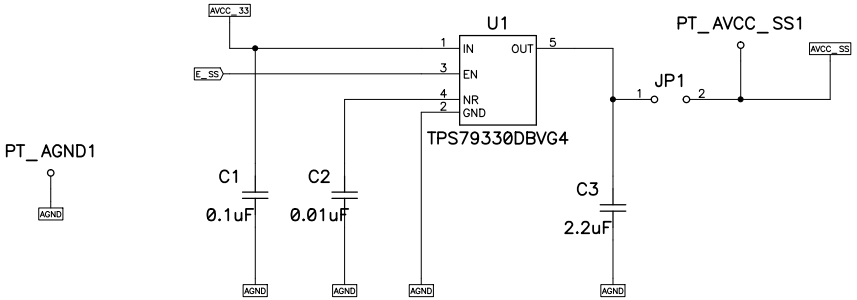
Once you have done all the necessary preparations, it's time to get your click board up and running. We have provided the examples for mikroC, mikroBasic and mikroPascal compilers on our **Libstock** website. Just download them and you are ready to start.

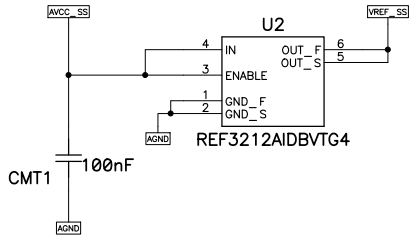
8. Support

MikroElektronika offers **Free Tech Support (www.mikroe.com/esupport)** until the end of product lifetime, so if something goes wrong, we are ready and willing to help!

MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

Alimentation for Sun Sensor

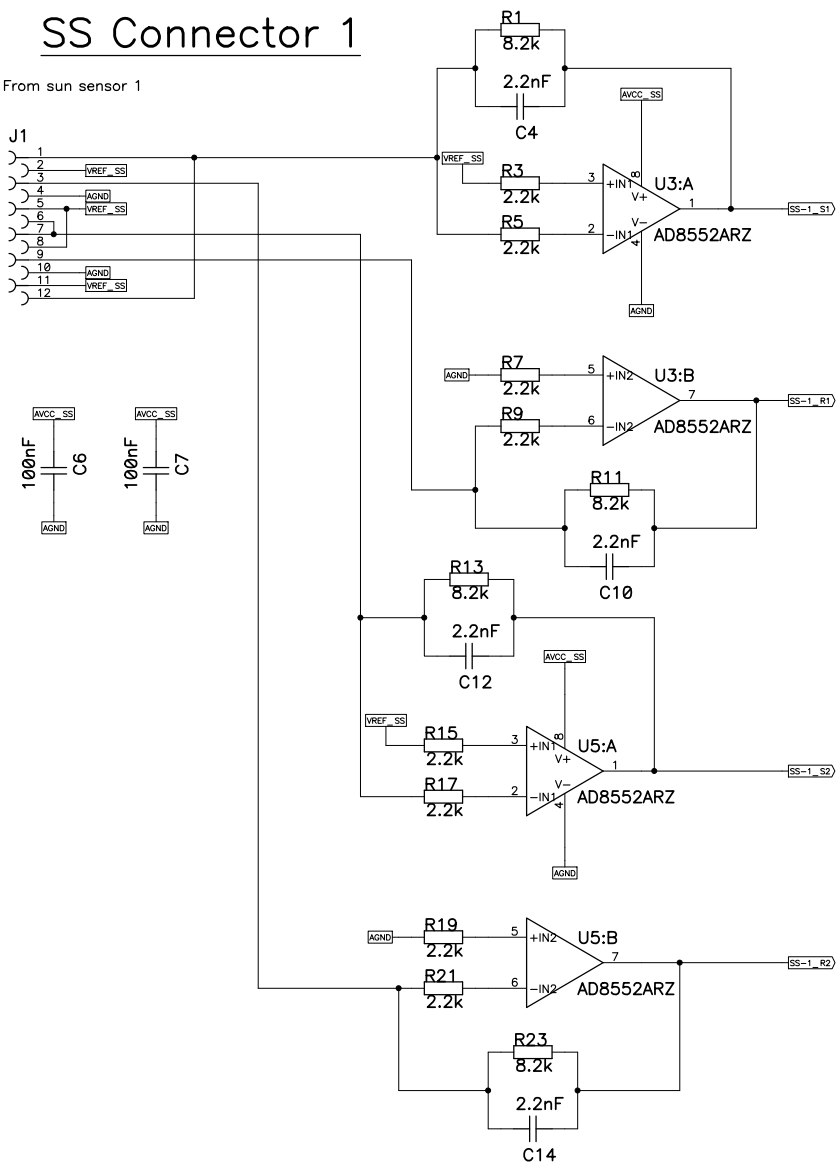




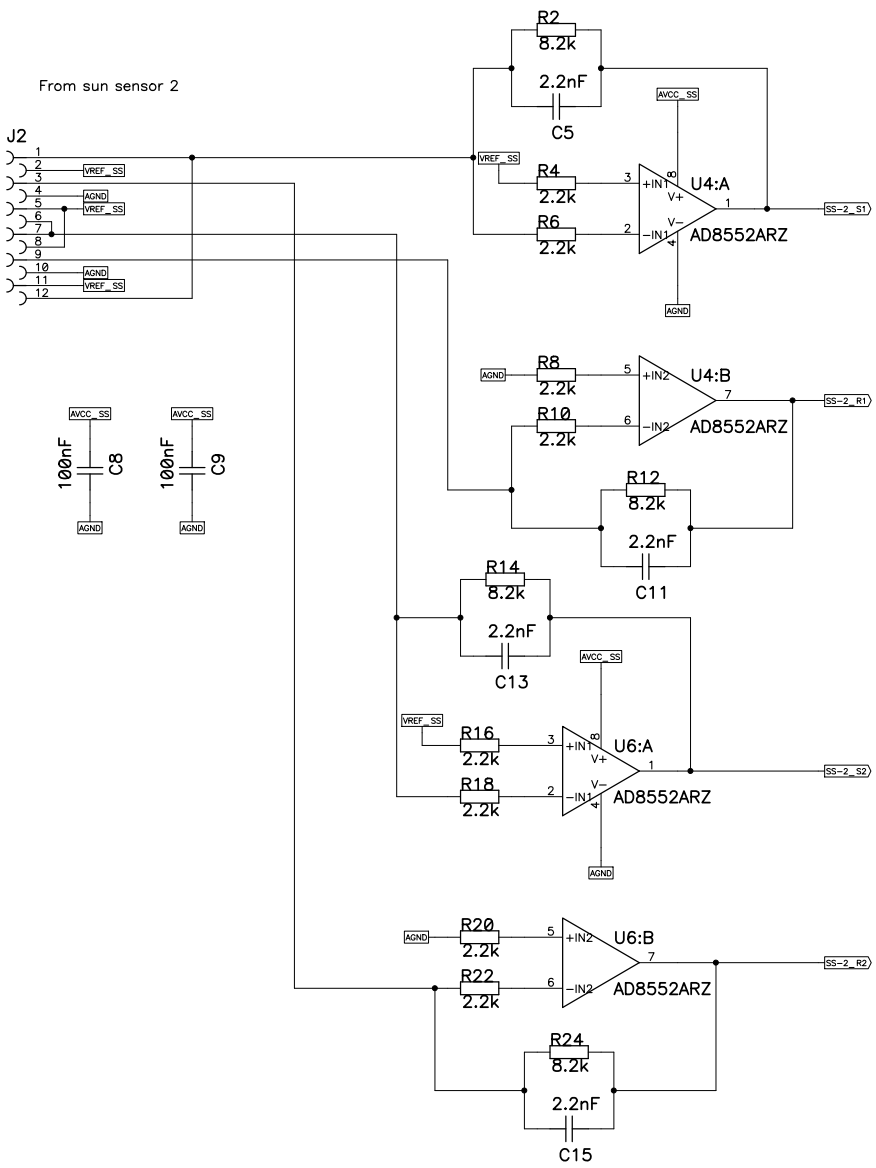
CubETH CDMS Interface FPGA to sensors	Sun Sens Alim	DES	{Date} Devs
		REV	V1.0
HAUTE ECOLE VALAISANNE		1/12	{Path} Interface_V1.0.sch

SS Connector 1

From sun sensor 1



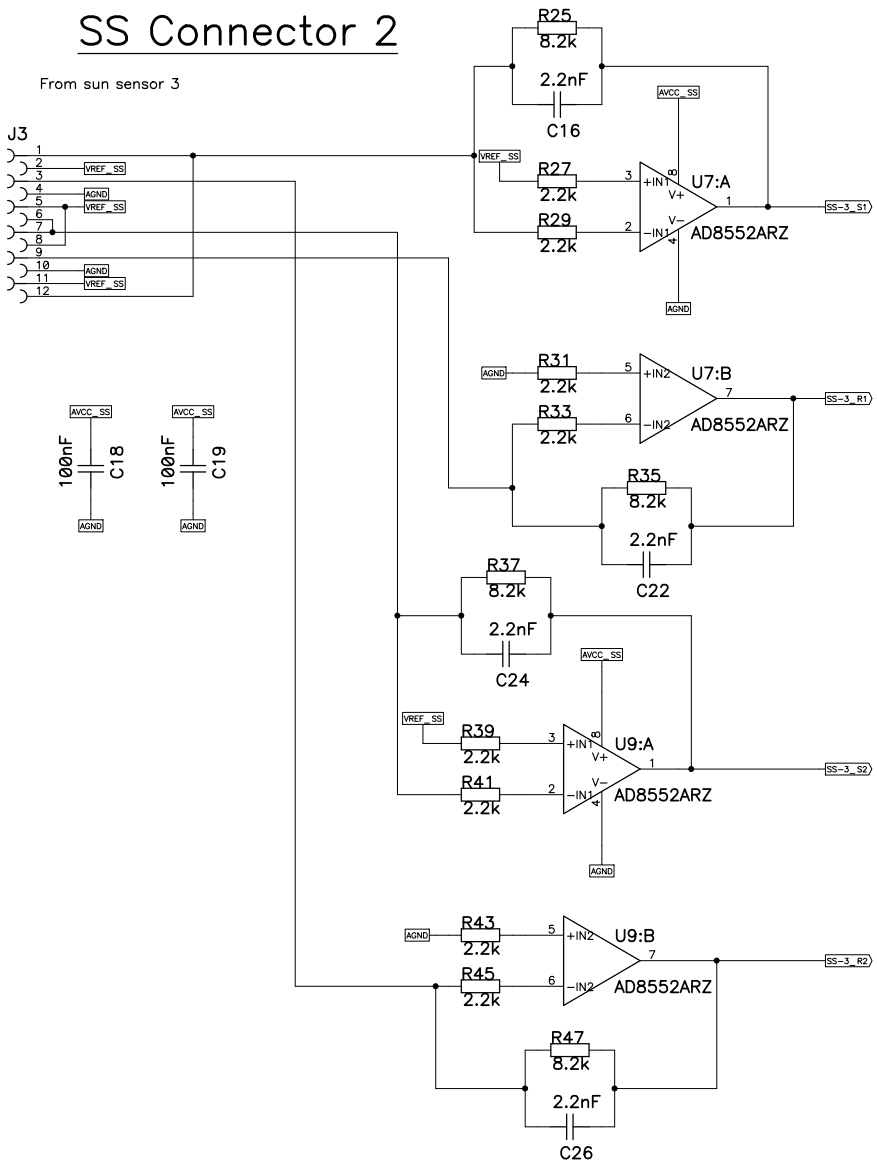
From sun sensor 2



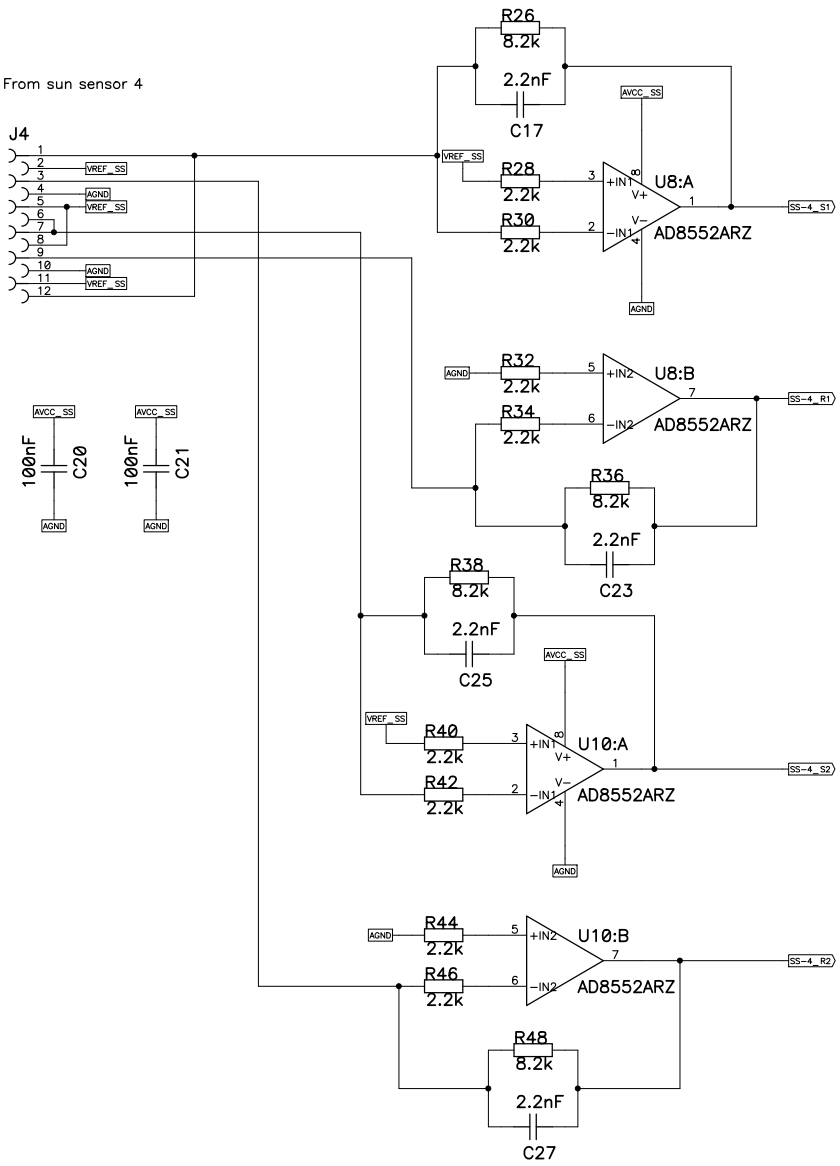
CubETH CDMS Interface FPGA to sensors Sun Sensor Conn 1	DES	{Date} Devs
	REV	V1.0
HAUTE ECOLE VALAISANNE	2/12	{Path} Interface_V1.0.sch

SS Connector 2

From sun sensor 3



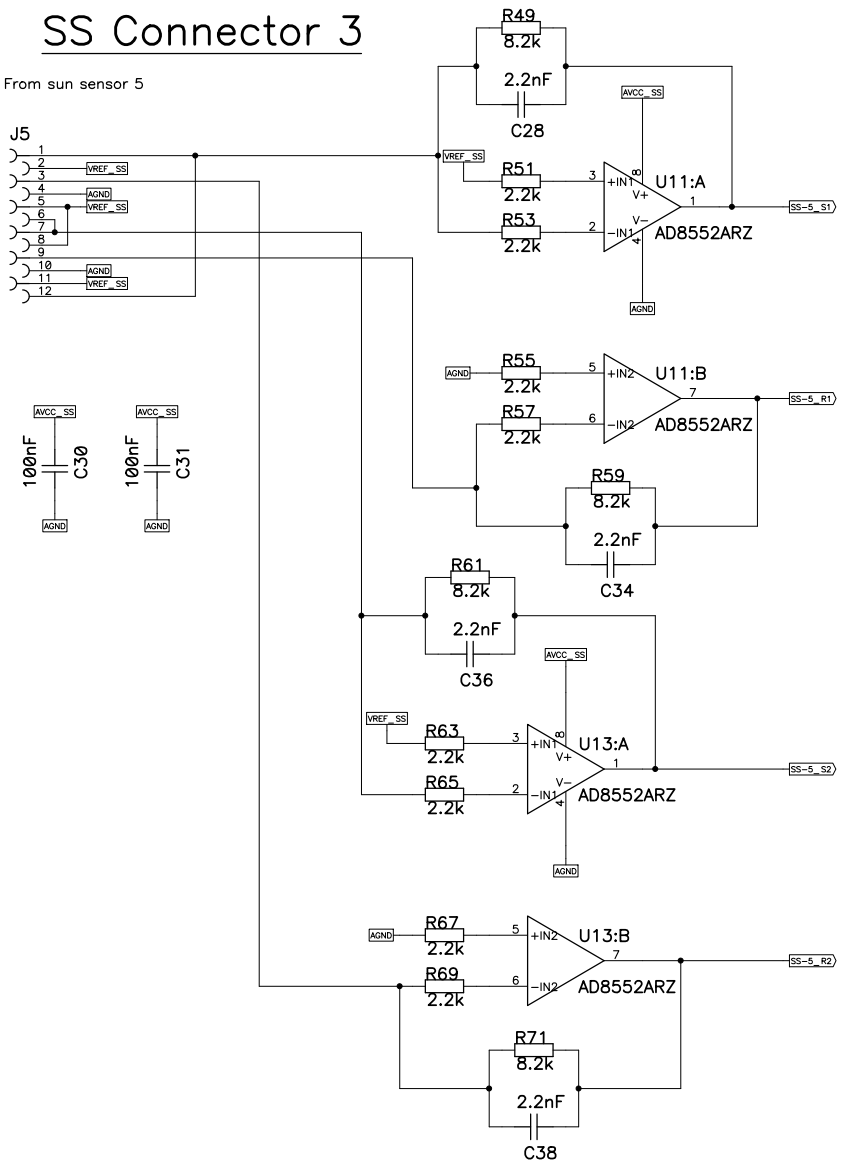
From sun sensor 4



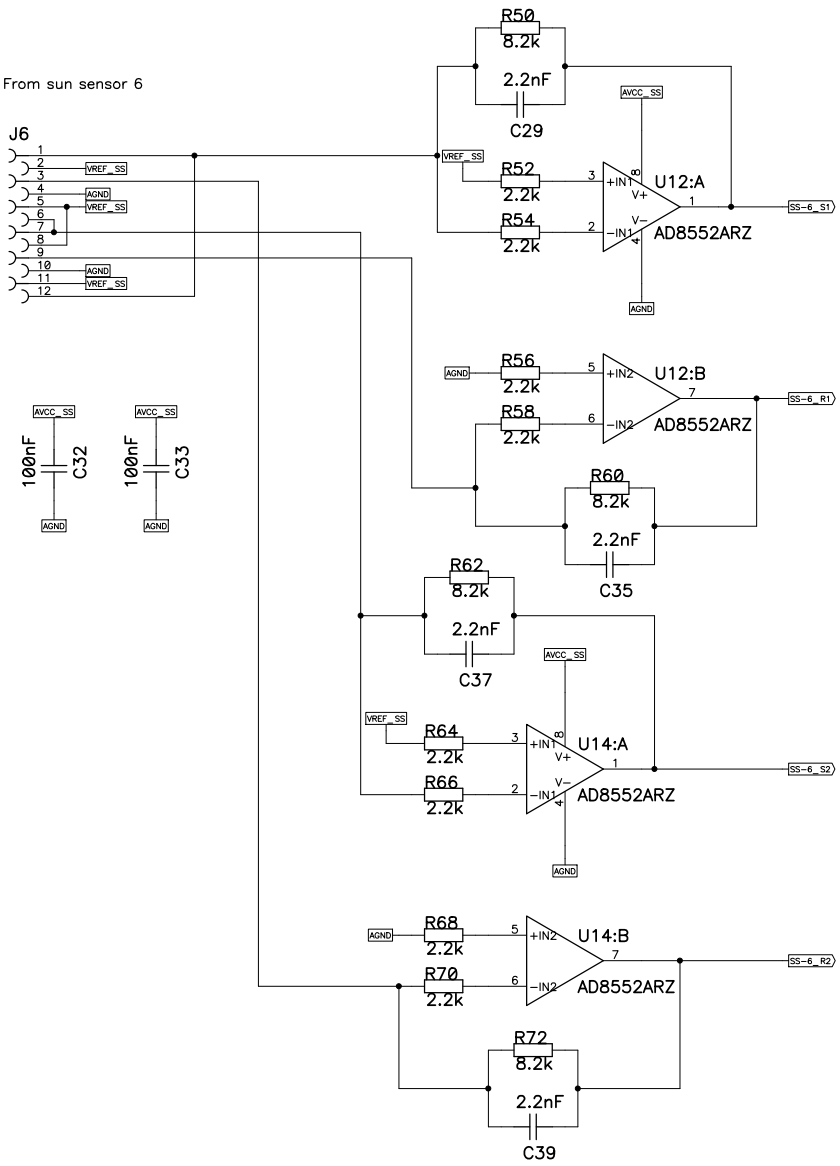
CubETH CDMS Interface FPGA to sensors Sun Sensor Conn 2	DES	{Date} Devs
	REV	V1.0
HAUTE ECOLE VALAISANNE	3/12	{Path} Interface_V1.0.sch

SS Connector 3

From sun sensor 5



From sun sensor 6

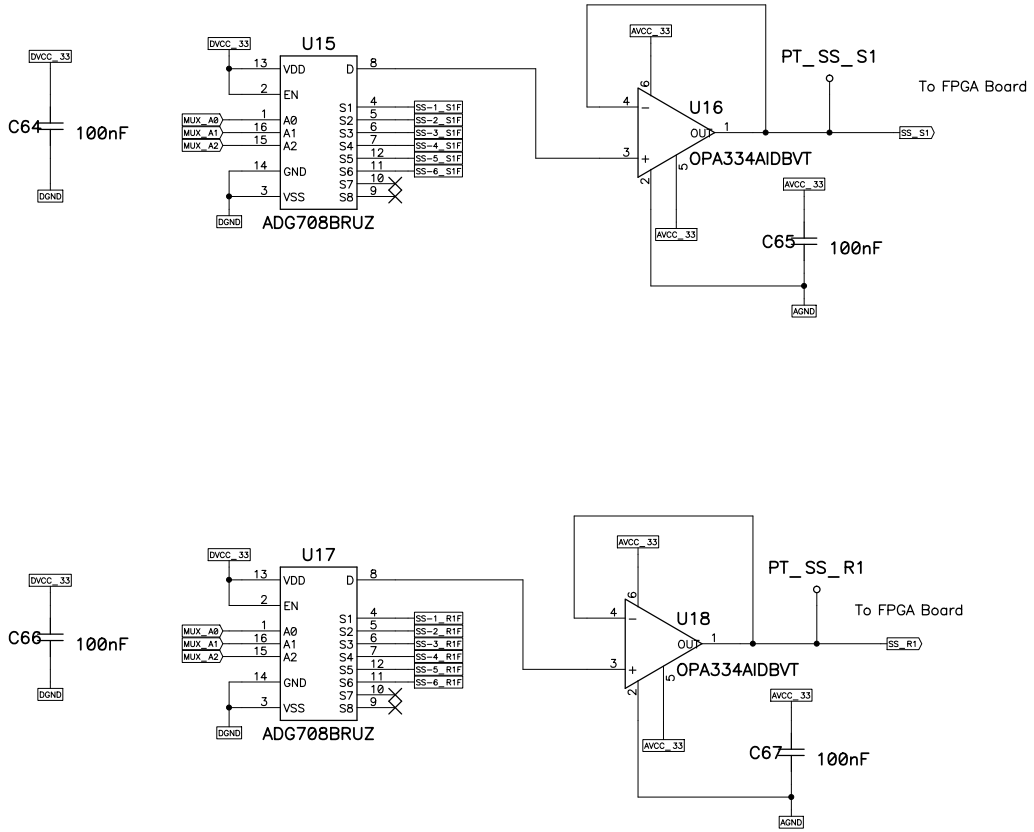


CubETH CDMS Interface FPGA to sensors Sun Sensor Conn 3	DES	{Date} Devs
	REV	V1.0
HAUTE ECOLE VALAISANNE	4/12	{Path} Interface_V1.0.sch

Sun Sensors Filters

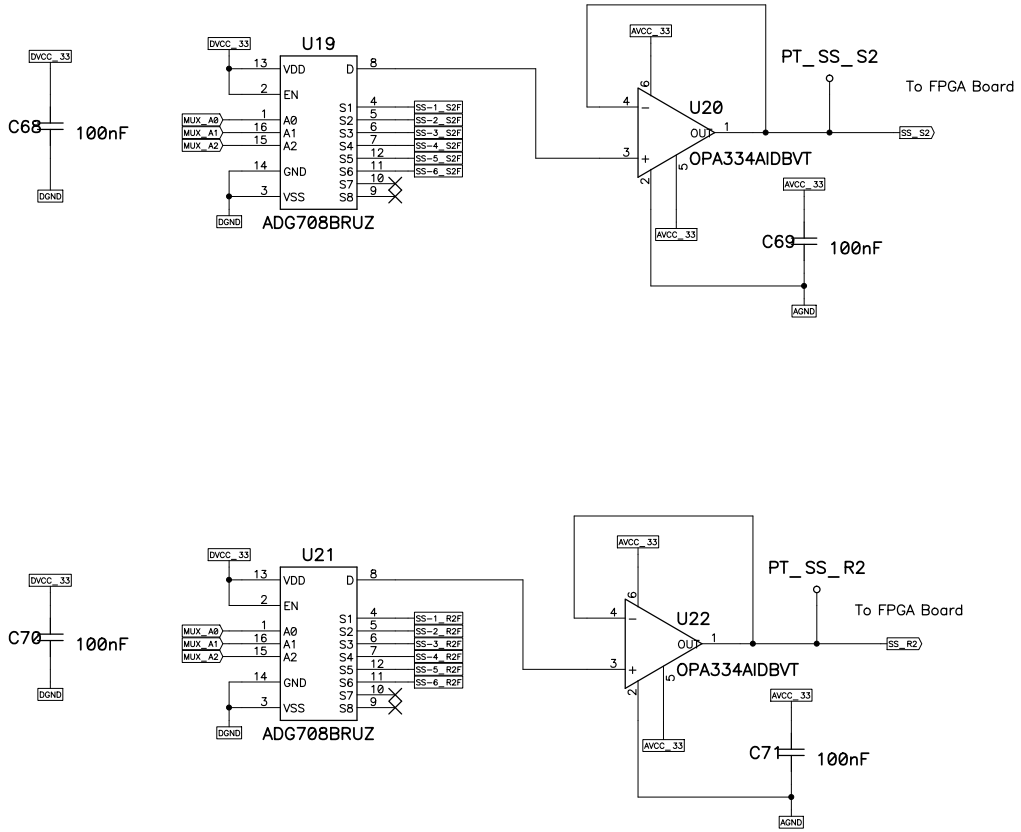
CubETH CDMS Interface FPGA to sensors Sun Sens Filters	DES	{Date} Devs
	REV	V1.0
HAUTE ECOLE VALAISANNE	5/12	{Path} Interface_V1.0.sch

Sun Sensors Multiplexer 1



CubETH CDMS		DES	{Date} Devs
Interface FPGA to sensors	Sun Sens Mux 1	REV	V1.0
HAUTE ECOLE VALAISANNE		6/12	{Path} Interface_V1.0.sch

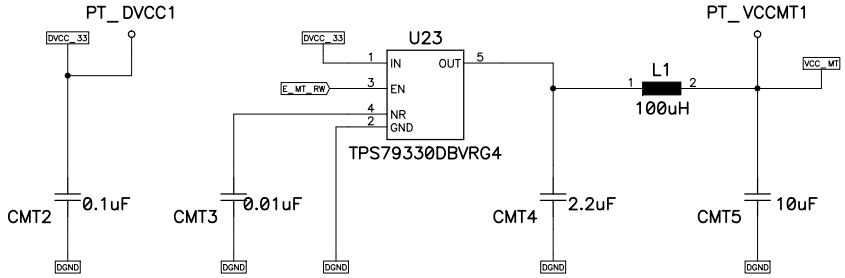
Sun Sensors Multiplexer 2



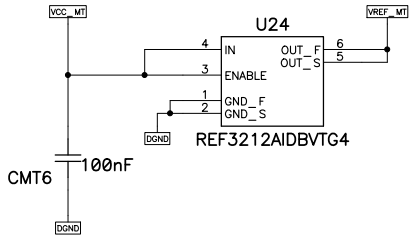
CubETH CDMS Interface FPGA to sensors Sun Sens Mux 2		DES	{Date} Devs
		REV	V1.0
HAUTE ECOLE VALAISANNE		7/12	{Path} Interface_V1.0.sch

Magneto Torquers Alimentation

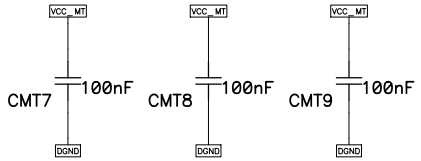
Alim for magnetotorquers



Reference voltage for current sensors

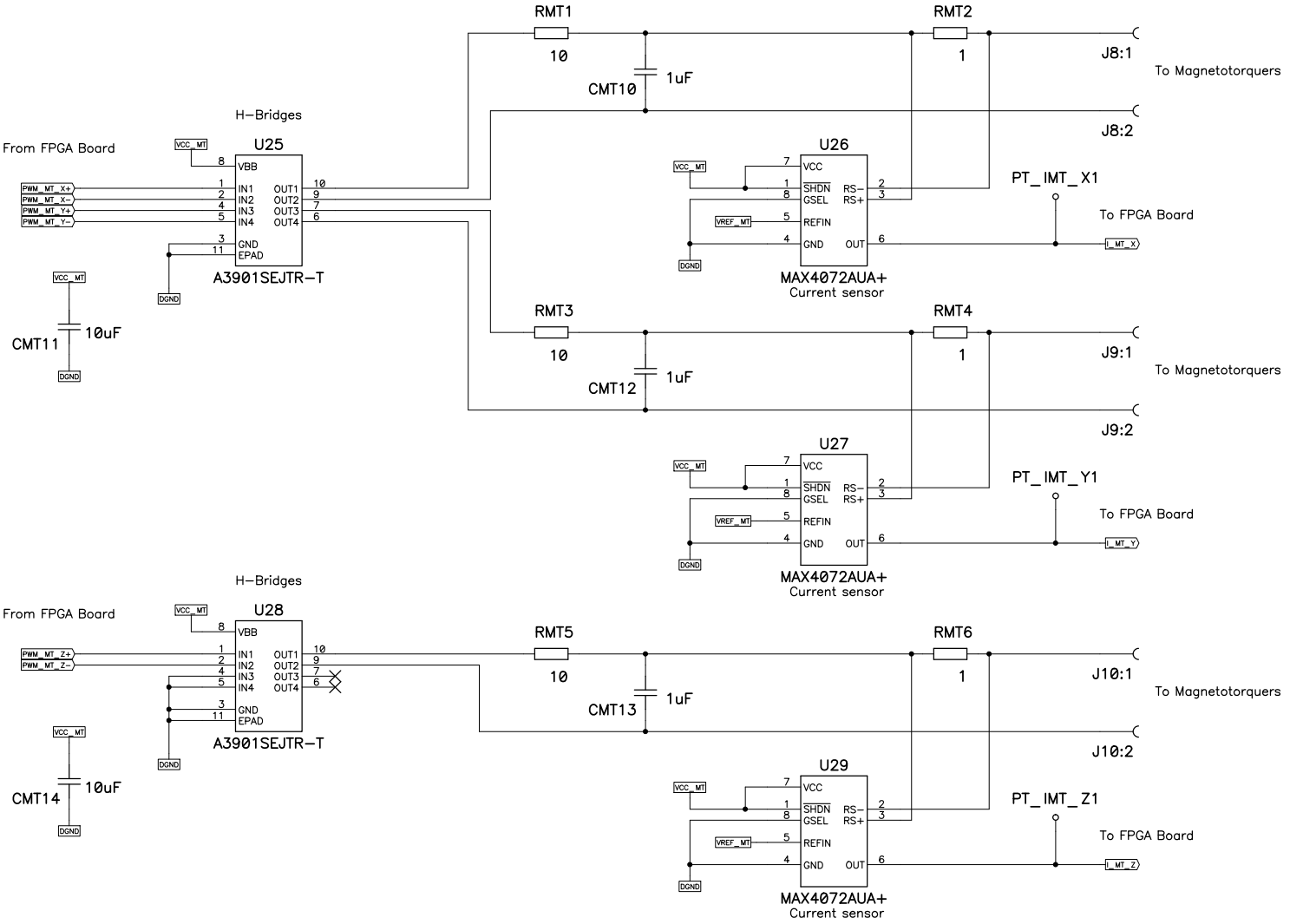


Capacitors for current sensors



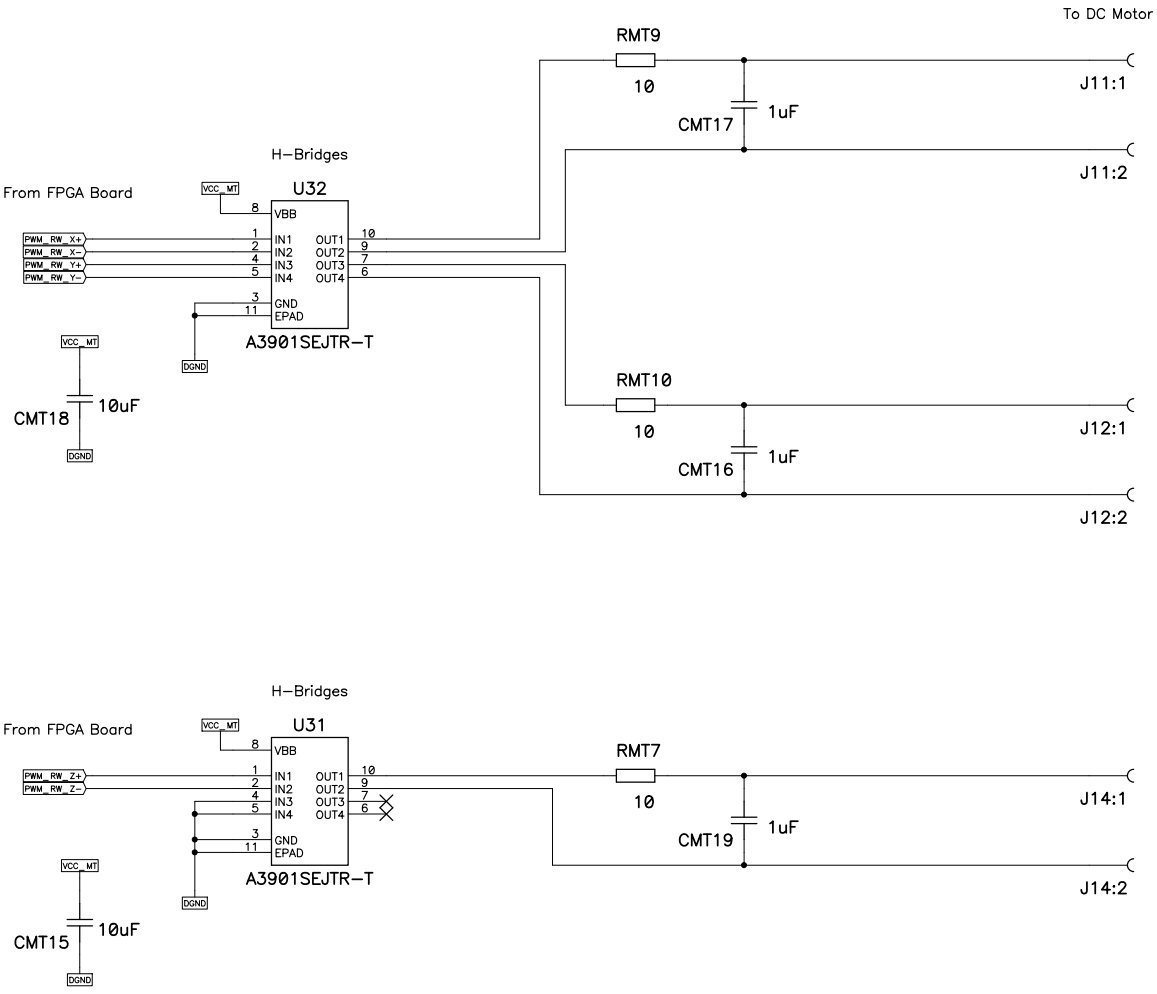
CubETH CDMS Interface FPGA to sensors Mag Torquer Alim	DES	{Date} Devs
	REV	V1.0
HAUTE ECOLE VALAISANNE	8/12	{Path} Interface_V1.0.sch

Magneto Torquers electronics



CubETH CDMS		DES	{Date} Devs
Interface FPGA to sensors	Mag Torquer	REV	V1.0
HAUTE ECOLE VALAISANNE		9/12	{Path} Interface_V1.0.sch

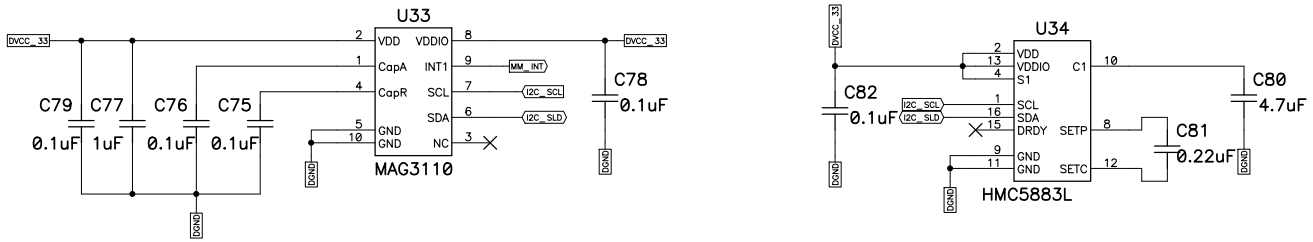
DC Motors electronics



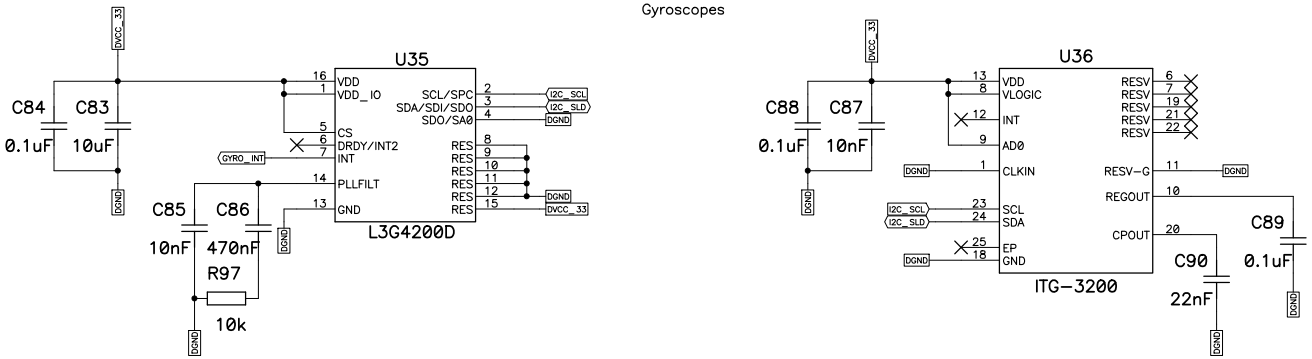
CubETH CDMS Interface FPGA to sensors	DC Motor	DES	{Date} Devs
		REV	V1.0
HAUTE ECOLE VALAISANNE		10/12	{Path} Interface_V1.0.sch

Gyros And MagnetoMeters

MagnetoMeters



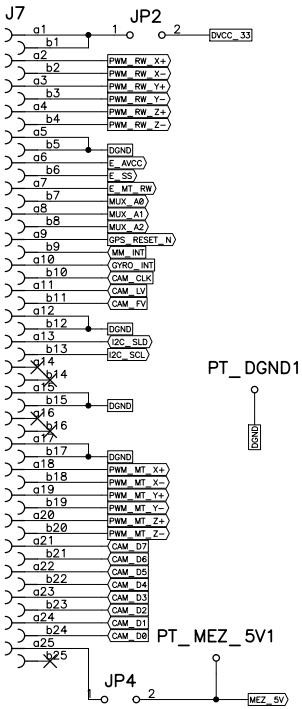
Gyroscopes

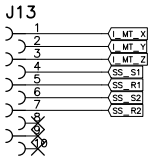


CubETH CDMS Interface FPGA to sensors Gyro and MM		DES	{Date} Devs
		REV	V1.0
HAUTE ECOLE VALAISANNE		11/12	{Path} Interface_V1.0.sch

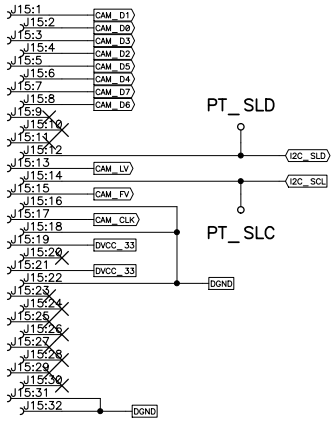
Connectors and Alims

From FPGA Board

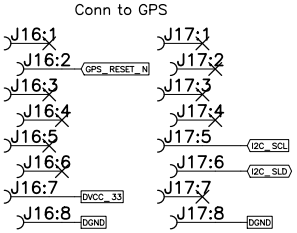




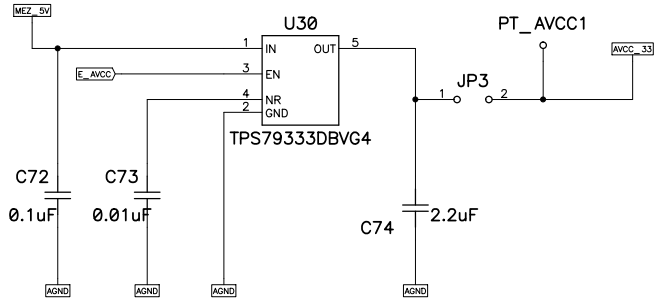
Conn to Camera



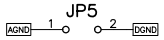
Conn to GPS



Analogique alimentation



Connection between AGND and DGND



CubETH CDMS Interface FPGA to sensors — Conn and Alims	DES	{Date} Devs
	REV	V1.0
HAUTE ECOLE VALAISANNE	12/12	{Path} Interface_V1.0.sch

Farnell			Monnaie	CHF
Quantity	**Reference**	**Designation**	**Unit Price**	
1	1858279RL	Gyroscope ITG-3200	31.10	31.10
1	1971743	Magnéto-mètre HMC5883L	7.00	7.00
3	599116	Moteur DC miniature	6.50	19.50
1	1287689	Regulateur LDO 3.3V	0.66	0.66
2	1287688RL	Regulateur LDO 3.0V	0.53	1.06
2	1180175	Référence de tension	4.25	8.50
4	1651947RL	Pont H	2.25	9.00
12	9425845	Ampli Op	5.05	60.60
15	1463473	Inductance DR73 100uH	2.15	32.25
				169.66

Mouser			Monnaie	CHF
Quantity	**Reference**	**Designation**	**Unit Price**	
1	511-L3G4200D	Gyroscope L3G4200D	6.53	6.53
1	841-MAG3110FCR1	Magnéto-mètre MAG3110	1.04	1.04
1	932-MIKROE-1032	Kit GPS Mikroe 1032	47.43	47.43
4	584-ADG708BRUZ	Multiplexeur	3.27	13.08
4	595-OPA334AIDBVT	Ampli op	2.95	11.80
3	700-MAX4072AUA	Ampli de détection de courant	1.38	4.14
				84.02

Sensor_axis_SA	c0	c1 (x)	c2(x^2)	c3 (x^3)	c4 (x^4)
F1_1_0	-1.7007	38.5654	1.2784	8.8279	-4.6980
F1_1_30	-3.3655	34.0001	0.0376	11.5380	-5.2253
F1_1_50	-5.6993	30.7440	-3.6047	16.2773	-7.1022
F1_2_0	-1.1433	40.7581	1.1222	4.0533	-1.5994
F1_2_30	-0.9661	35.7350	3.2890	6.9182	-3.2644
F1_2_50	-0.4579	32.2761	3.3798	8.1518	-3.5162
F2_1_0	1.6323	40.2953	1.9738	7.3122	-0.1997
F2_1_30	1.8304	35.3533	0.7600	9.4203	0.5360
F2_1_50	1.9159	31.4676	0.7582	11.3100	0.4661
F2_2_0	-1.4394	41.0591	1.6098	3.1546	-1.5529
F2_2_30	-0.8459	35.8139	3.2340	6.5413	-3.2526
F2_2_50	-0.4133	32.7163	4.2122	7.3352	-3.8587
F3_1_0	1.1739	40.8948	2.2660	7.0611	-0.8288
F3_1_30	1.7017	36.2704	0.3663	9.1624	0.2927
F3_1_50	1.6793	33.4806	1.0401	10.3074	-0.3751
F3_2_0	0.1842	41.5188	2.2955	2.5670	-1.2988
F3_2_30	0.7964	36.3613	4.1272	5.9094	-2.9541
F3_2_50	1.3745	33.7121	6.1317	6.3062	-4.1888
F4_1_0	1.8351	41.0068	1.6478	7.1043	-0.0762
F4_1_30	2.0353	35.8413	0.0547	9.7598	1.3964
F4_1_50	1.8202	32.7429	0.7167	11.3304	1.2301
F4_2_0	0.3194	41.2487	3.1254	2.7362	-1.7696
F4_2_30	-0.1995	36.3273	4.4321	5.7517	-2.9327
F4_2_50	1.0570	34.1733	6.3800	6.0295	-4.4082
F5_1_0	1.7947	40.2243	2.7159	7.4622	-0.3329
F5_1_30	2.5800	35.4101	0.4973	10.1190	1.2477
F5_1_50	2.9612	31.9628	1.0959	12.5386	1.5179
F5_2_0	-0.7551	40.9054	1.6337	2.9798	-1.5877
F5_2_30	-0.3905	35.3117	3.2762	6.6151	-3.4447
F5_2_50	0.0452	32.4135	4.3313	7.0619	-4.0813
F6_1_0	1.9716	41.3092	3.1408	7.9133	-0.3787
F6_1_30	2.2942	35.7413	1.3386	11.0138	1.1918
F6_1_50	2.4748	32.2441	0.7180	13.6846	2.5298
F6_2_0	-0.0984	41.2333	2.3302	2.5397	-1.6784
F6_2_30	0.4011	35.7295	4.2897	6.0962	-3.6639
F6_2_50	0.9048	32.8977	5.4461	6.5491	-4.3358

Sensor_axis_SA	c0	c1 (x)	c2(x^2)	c3 (x^3)	c4 (x^4)
R1_1_0	0.6747	40.2758	1.6722	6.0279	-0.0936
R1_1_30	0.8449	37.5229	0.1418	7.1735	1.3557
R1_1_50	0.7062	34.7060	0.8579	7.8495	0.3828
R1_2_0	-2.1732	39.7928	2.5577	2.9885	-1.6908
R1_2_30	-1.6091	37.4656	2.0721	4.3235	-1.7923
R1_2_50	-0.5657	33.0857	5.1055	6.1309	-4.0862
R2_1_0	0.3873	39.9837	1.4360	5.7308	-0.4921
R2_1_30	0.8049	37.2027	-0.6162	6.6838	1.0160
R2_1_50	0.8698	33.5884	-0.0837	8.4455	0.7364
R2_2_0	-0.4859	43.1979	0.8168	3.1893	-1.1537
R2_2_30	-0.1001	39.4618	2.5274	5.6244	-2.8857
R2_2_50	0.2058	36.1704	5.0643	5.6455	-4.4945
R3_1_0	1.8084	42.7628	2.8677	6.7305	-1.2169
R3_1_30	1.5014	39.3848	0.9146	9.2314	1.0453
R3_1_50	1.5070	35.5272	0.5083	12.0086	1.7343
R3_2_0	-0.8404	43.0828	2.9155	3.7116	-1.6055
R3_2_30	-0.0015	40.1593	2.0503	5.3546	-1.4209
R3_2_50	0.7381	37.2592	5.2714	6.1147	-4.2209
R4_1_0	1.5256	39.6241	0.9346	4.5337	-0.0163
R4_1_30	1.5118	36.9448	0.4518	6.0204	0.5268
R4_1_50	1.3653	33.0894	1.4204	8.5207	0.4809
R4_2_0	-0.1275	42.1509	3.2183	3.0760	-2.2492
R4_2_30	-0.3036	39.3584	2.5977	4.7262	-2.2756
R4_2_50	-0.4043	35.6318	5.3511	5.7539	-4.4001
R5_1_0	1.7331	39.2334	1.4573	6.1862	0.0665
R5_1_30	2.1304	36.4019	1.3437	7.9152	0.3659
R5_1_50	2.1134	32.3301	2.0205	10.8530	0.7416
R5_2_0	0.0729	40.1095	3.4872	2.6074	-1.8878
R5_2_30	1.6717	37.6126	1.5449	3.1682	-1.3542
R5_2_50	2.3681	33.7476	3.3179	4.1701	-2.1379
R6_1_0	1.5503	40.4976	0.0645	5.6286	0.1263
R6_1_30	1.3892	37.3308	0.0695	6.7836	0.3222
R6_1_50	0.9070	33.3015	0.7920	9.4484	-0.1242
R6_2_0	0.3140	40.8721	3.4512	3.1008	-2.0992
R6_2_30	0.4192	38.4538	2.9088	3.9350	-1.9656
R6_2_50	0.4531	35.5376	5.0224	4.6445	-3.6531

Sensor_axis_SA	c0	c1 (x)	c2(x^2)	c3 (x^3)	c4 (x^4)
R7_1_0	1.2729	38.1209	0.9767	6.2521	0.5125
R7_1_30	1.5612	35.2392	0.0294	7.3659	0.7483
R7_1_50	1.4191	31.7055	0.3473	9.1199	0.4863
R7_2_0	-2.1811	41.1652	2.0064	6.2269	-2.9827
R7_2_30	-1.7935	38.5903	2.6767	7.9847	-4.1172
R7_2_50	-0.9806	34.4297	2.6375	9.2408	-4.4289
R8_1_0	1.3102	38.9583	0.9977	5.9618	0.4693
R8_1_30	1.7111	36.4097	-0.3800	6.8934	1.1085
R8_1_50	1.7186	33.0025	0.5772	8.5990	0.4390
R8_2_0	-0.5275	40.4188	3.3234	4.4266	-2.2921
R8_2_30	-0.1255	38.0437	4.3159	5.4518	-3.3774
R8_2_50	0.5819	34.9930	4.1854	5.6109	-3.2966
R9_1_0	1.3328	39.5731	0.6794	6.4340	0.5849
R9_1_30	1.5741	36.9374	-0.7209	7.6472	1.4022
R9_1_50	1.7243	33.5797	0.1161	9.8162	0.9825
R9_2_0	-0.6079	40.6514	2.9880	4.1857	-2.3214
R9_2_30	-0.4335	38.3937	3.7913	5.4414	-3.1241
R9_2_50	0.1541	34.7233	3.5977	6.2403	-2.8653
R10_1_0	0.4473	38.4905	0.2510	5.7678	0.4011
R10_1_30	0.4113	36.1582	-1.0079	6.6204	0.9663
R10_1_50	0.0414	32.9083	0.1238	8.1030	0.0351
R10_2_0	-0.4780	40.3116	2.7560	4.4971	-2.2145
R10_2_30	-0.1571	38.1720	3.7378	5.4366	-3.1158
R10_2_50	0.5095	34.8288	3.2285	5.5979	-2.7331
R11_1_0	0.6602	38.3207	0.1112	5.7212	0.5155
R11_1_30	0.5559	36.0945	-0.7407	6.5427	0.8458
R11_1_50	0.2790	32.6554	-0.1255	8.1847	0.1948
R11_2_0	-0.9638	40.1665	2.8350	4.5734	-2.6067
R11_2_30	-0.8668	37.6898	3.3302	5.8926	-3.2674
R11_2_50	-0.6529	34.0105	2.8835	6.5363	-3.1846
R12_1_0	0.9105	37.5209	-0.1690	3.4904	0.4663
R12_1_30	1.2948	35.4954	-0.9597	3.7483	0.6669
R12_1_50	1.2155	32.4883	-0.1981	3.7778	0.0865
R12_2_0	-1.0746	40.5617	3.0114	4.4847	-2.5275
R12_2_30	-0.6148	38.2490	3.1034	5.6595	-3.0307
R12_2_50	-0.2010	34.9074	3.0265	5.8113	-3.0574