Implémentation du monitoring, de la journalisation et de l'acquisition des mesures au sein de l'ordinateur de bord du nanosatellite OUFTI-1

Thomas Langohr

2011

REMERCIEMENTS

Je tiens tout d'abord à remercier mon superviseur V.Broun pour m'avoir permis d'effectuer mon travail de fin d'études sur ce projet extrêmement motivant et ô combien intéressant.

Je suis également très reconnaissant du soutiens, de l'encadrement et des conseils avisés de N.Crosset, mon maitre de stage à l'université de Liège.

Mes remerciements vont ensuite aux autres membres de l'équipe système d'OUFTI-1 : N.Marchal, J. Wertz, J. Pisane, L.Chiarello et plus particulièrement à A.Denis pour la gestion du projet.

Un grand merci à tous les étudiants du projet pour leur sympathie et le partage des connaissances propres à leur domaine.

Merci également à P. Parisis, directeur commercial chez Spacebel, pour m'avoir fait profiter de son expérience dans le domaine du logiciel embarqué des satellites.

Je terminerai par remercier mon collègue et camarade Sebastien De Dijcker pour m'avoir accompagné lors de cette expérience enrichissante.

TABLE DES MATIÈRES

LISTE DES ACRONYMES

µC:	Microcontroller
ACK:	Acknowledgement
ACLK:	Auxiliary clock
BCN:	Beacon
CPU:	Central Processing Unit
DMA:	Direct Memory Access
ESA:	European Space Agency
HKP:	Housekeeping Parameter
I/O:	Input / Output
KSPS:	Kilo Samples-Per-Second
LSB:	Least Significant Bit
MCLK:	Main Clock
MSB:	Most Significant Bit
OBC:	On Board Computer
OBSW:	On Board Software
PL:	Payload
RAM:	Random Access Memory
RTOS:	Real Time Operating System
SEL:	Single Event Latch-up
SEU:	Single Event Upset
SMCLK:	Sub-Main clock
SP:	Science Parameter
S-S:	Sub-System
TBD:	To Be Determined
TID:	Total Ionizing Dose
TOC:	Type Of Communication

UART:	Universal Asynchronous Receiver Transmitter
USART:	Universal Synchronous & Asynchronous Receiver Transmitter
WDT:	Watchdog Timer

GLOSSAIRE

Housekeeping Parameters (HKP) : mesures effectuées à bord du satellite qui subissent une analyse au sein de l'ordinateur de bord. La sortie de cette mesure hors de sa plage nominale provoque une action de la part de l'ordinateur de bord qui ne nécessite pas d'intervention au sol.

Payload : Les payloads sont les sous-systèmes qui constituent la charge utile du satellite (xEPS, D-STAR, cellules photovoltaïques). Par opposition à la plateforme du satellite qui permet aux payloads de remplir leurs objectifs.

Science Parameters (SP) : mesures effectuées a bord du satellite qui n'influencent pas directement son fonctionnement. Ces mesures sont échantillonnées et enregistrées en vue d'être rapatriées au sol.

Télécommandes (TC) : Ensemble de tous les ordres envoyés au satellite.

Télémétries (TM) : Ensemble des données collectées par le satellite, que ce soit pour sa mission ou des informations relatives à son état de fonctionnement.

Chapitre I - INTRODUCTION

1 OBJECTIFS DE CE TRAVAIL

L'objectif de ce travail est la conception du logiciel de bord du nanosatellite de l'université de Liège : OUFTI-1. La tâche à été divisée en deux grandes parties distinctes :

- L'implémentation de la gestion des télécommandes et des télémétries a été confiée à S.De Dijcker.
- L'implémentation du monitoring, de la journalisation et de l'acquisition des mesures est de ma responsabilité.

Mon travail est donc divisé en trois modules qui en pratique représentent chacune une tâche (un thread) du logiciel de bord(OBSW) ainsi que ses moyens de communication avec les autres tâches.

Voici les objectifs à remplir pour chaque module :

A. Monitor :
   - ✓ Activer ou désactiver les sous-systèmes et payloads du satellite en fonction des certaines mesures et certaines télécommandes.
   - ✓ Gestion de la payload xEPS et activation / désactivation du D-STAR.
   - ✓ Gestion de la redondance des OBC.
   - ✓ Reset des sous-systèmes en surconsommation de courant.

B. Log : Il récence et enregistre les événements se produisant dans le satellite.
   - ✓ Consigner les événements dans un journal de bord.
   - ✓ Rapatriement d'une partie ou de l'entièreté du journal de bord.

C. Measurement : Il sert à recueillir et enregistrer les mesures du satellite à envoyer sur terre.
   - ✓ Configuration de la liste des mesures à rapatrier.
   - ✓ Echantillonnage et enregistrement des mesures
   - ✓ Rapatriement des mesures.

Il est également demandé d'écrire les drivers utiles à ces modules, ces drivers doivent être le plus génériques possibles afin de pouvoir les utiliser pour remplir plusieurs fonctions.

2 ETAT DES LIEUX

La partie hardware de l'OBC est finalisée et ne requiert aucune modification.

La partie software a été entamée les années précédentes mais une partie de ce qui a été fait ne correspond plus à l'architecture actuelle de l'OBSW. Certains modules doivent être totalement ou partiellement réécrits. Quand aux drivers, ceux qui ont déjà été écrits sont fonctionnels mais certains doivent encore être conçus (ADCs et UART OBC-COM notamment).

Chapitre II - OUFTI-1

1 Le standard CubeSat

Tout d'abord, il faut savoir qu'en raison du nombre croissant de satellites de petite taille en développement, un classement en fonction de la masse existe :

- Minisatellite : Satellite de masse comprise entre 100 et 500 kg. Il utilise souvent le même équipement que les gros satellites, et peut se permettre d'être équipé de propulseurs pour le contrôle d'attitude ;

- Microsatellite : Satellite de masse comprise entre 10 et 99,99 kg. La miniaturisation se fait déjà ressentir, néanmoins des propulseurs sont encore utilisés ;

- Nanosatellite : Satellite de masse comprise entre 1 et 9,99 kg. Tous les composants se doivent d'être réduits en termes de masse et de volume, les propulseurs sont donc souvent une option non envisageable. Ce type de satellite peut être lancé en tant que passager secondaire d'un lanceur destiné à un satellite commercial plus important ;

- Picosatellite : Satellite de masse comprise entre 0,1 et 0,99 kg. Le processus de miniaturisation est au maximum, par conséquent, des composants nouvelle technologie doivent souvent être utilisés. Les picosatellites sont également lancés à bord d'un lanceur destiné à un satellite commercial plus important.

Défini par la Cal Poly (California Polytechnic State University) et l'université de Stanford, le standard CubeSat 1 est un exemple de nanosatellite utilisant typiquement des composants commerciaux, et pas nécessairement qualifiés pour le spatial. Pour répondre au standard CubeSat, certains critères doivent être respectés, dont voici les plus importants :

- Chaque CubeSat ne peut excéder une masse d'un kg ;
- Les dimensions sont de 100 mm ± 0,1 selon les axes X et Y et de 113,5 mm ± 0,1 selon l'axe Z (cf. figure 1) ;
- Le centre de gravité doit se trouver à maximum 2 cm du centre géométrique du CubeSat ;

- Le CubeSat ne peut pas être un danger pour les CubeSats avoisinants, pour le lanceur ou pour le satellite principal ;
- Aucune partie du CubeSat ne peut se détacher de celui-ci, ni pendant le lancement, ni pendant l'éjection, ni dans sa phase orbitale ;
- Aucun système pyrotechnique ne peut se trouver à bord du CubeSat.

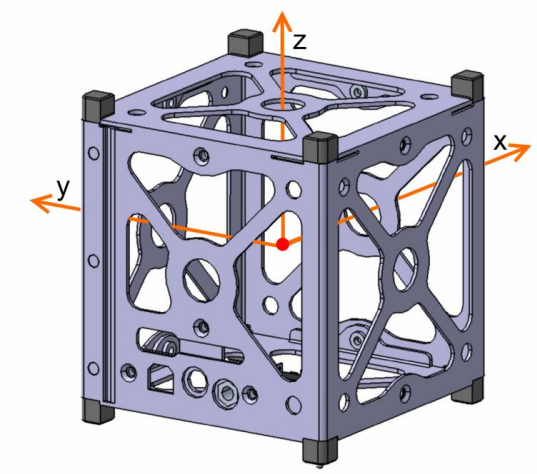

FIGURE 1 - STRUCTURE ET AXES D'UN CUBESAT

Pour permettre aux CubeSats d'être passagers secondaires lors du lancement d'un satellite plus important, l'université de Cal Poly a également développé un standard de déployeur : le Poly Picosatellite Orbital Deployer (P-POD) (cf. figure 2).

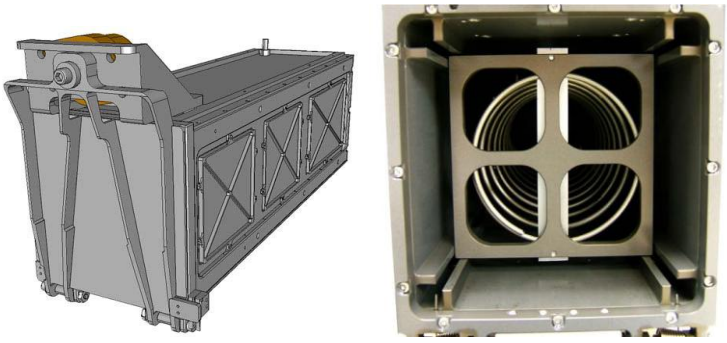

FIGURE 2 - LE DÉPLOYEUR P-POD

Ce dernier a la possibilité de contenir 3 CubeSats d'une unité (1U) et permet d'assurer la sécurité du satellite principal. Du point de vue du fonctionnement, un ressort est placé dans le fond du PPOD, les 3 CubeSats sont ensuite insérés, et la porte du P-POD refermée. Lorsque le

lanceur a atteint l'altitude et l'emplacement désirés, la porte du P-POD s'ouvre, et les trois CubeSats sont éjectés par la poussée du ressort. Etant donné qu'il possède assez de place pour contenir trois CubeSats, cela permet de développer des CubeSats de deux ou trois unités (2U ou 3U). Ainsi un CubeSat peut avoir une masse allant jusqu'à 2 kg pour le 2U ou 3 kg pour le 3U. La section du CubeSat doit évidemment rester la même, tandis que la hauteur est alors multipliée par deux ou par trois.

2 LE PROJET OUFTI-1

OUFTI-1 (Orbital Utility For Telecommunication Innovations – 1) est le premier projet étudiant de CubeSat belge. OUFTI-1 est développé à l'ULg (Université de Liège) dans l'optique d'un projet éducatif à long terme : l'initiative Léodium. Le but principal de cette initiative est de fournir une expérience pratique dans le domaine de la conception de satellites, de la phase de design au contrôle en orbite.

FIGURE 3 - LE LOGO OUFTI-1

OUFTI-1 est le premier satellite de cette initiative et servira donc de base à la conception des futurs satellites expérimentaux de l'ULg. À l'heure actuelle, un instrument pour une potentielle mission OUFTI-2 est d'ailleurs en phase d'analyse de faisabilité.

3 LES SOUS-SYSTÈMES D'OUFTI-1

Comme tout satellite, la plateforme d'OUFTI-1 est divisée en différents sous-systèmes, selon les fonctionnalités nécessaires au bon fonctionnement de la charge utile et du satellite dans son ensemble. Cette section est dédiée à l'explication des différents rôles de chaque sous-système.

a) STRU

La Structure mécanique (STRU) décrit la façon dont sont agencés les différents composants du satellite (PCBs, connectique, éléments mécaniques, etc.) et assure leur maintien sur l'ossature du CubeSat.

b) MECH

Le sous-système Mechanic (MECH) comprend le mécanisme de rétention et de déploiement des antennes. En effet, la communication entre le satellite et la station sol nécessite deux antennes de respectivement 50 et 17cm. Ces antennes ne peuvent pas se déployer avant les trente minutes qui suivent l'éjection du CubeSat afin d'éviter d'endommager les autres satellites présents dans le module P-POD. La figure 4 représente le support de déploiement des antennes qui sera collé sur une des 6 faces du CubeSat.

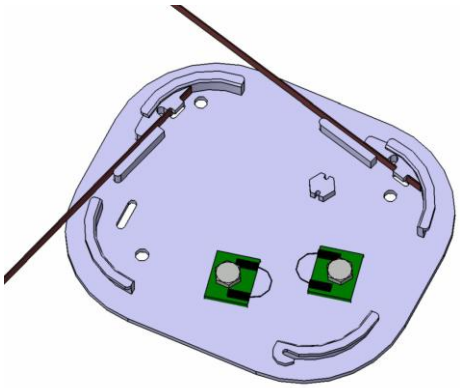

FIGURE 4 - SUPPORT DE DÉPLOIEMENT DES ANTENNES.

c) THER

Le sous-système de Thermique (THER) doit s'assurer que tous les composants du satellite restent dans une plage de température acceptable durant la mission, les capteurs de

température doivent également être positionnés correctement afin de permettre une précision de mesure optimale. Finalement, un système d'asservissement comportant un capteur de température et une chaufferette permet de réguler la température des batteries afin de les protéger du froid lorsque le satellite est en éclipse.

d) VIB

Le sous-système de Vibrations (VIB) est l'étude de la résistance d'OUFTI-1 aux chocs et vibrations durant tout son cycle de vie : de l'assemblage du modèle de vol à sa vie orbitale, en passant par la phase de lancement, qui est la phase la plus contraignante.

e) EPS

Le sous-système Electrical Power Supply (EPS) inclut les panneaux solaires, les batteries et le Printed Circuit Board (PCB) de l'EPS. Les différentes fonctions de ce sous-système sont :

- la distribution de trois tensions électriques différentes (3.3V, 5V, 7.2V) à la totalité du satellite via trois bus d'alimentation distincts ;
- la génération de puissance électrique grâce aux panneaux solaires ;
- le stockage de surplus d'énergie dans les batteries afin de pouvoir restituer cette énergie ultérieurement, notamment lorsque le satellite est en éclipse.

f) COM

Le sous-système de Télécommunication (COM) est composé d'un récepteur qui permet de recevoir des télécommandes et d'un transmetteur afin de transmettre des télémétries. Ce canal de communication avec le sol utilise le protocole Amateur X.25 (AX.25). Le sous-système COM gère également le répéteur Digital Smart Technologies for Amateur Radio (D-STAR) en mode voix pour les télécommunications radioamateur. Il est à noter que l'encodage et le décodage des trames AX.25 est la responsabilité du sous-système On-Board Computer (OBC).

g) BCN

Le sous-système Beacon (BCN) est la balise morse de secours du satellite. Il transmet en continu douze mesures caractéristiques du satellite et cela indépendamment des autres sous-systèmes. En effet il n'a pas besoin du sous-système COM pour transmettre son message et peut s'affranchir du sous-système OBC car il dispose de sa propre chaîne de mesure.

h) OBC

L'On-Board Computer (OBC) est chargé du contrôle de la totalité du satellite. Il interprète les ordres venant du sol, les traites, et rapatrie les résultats. Il surveille et entretient le bon fonctionnement du système. Voici les différents rôles qui lui sont attribués :

- déploiement des antennes et activation des autres sous-systèmes ;
- acquisition, enregistrement et rapatriement de mesures ;
- transmission d'une référence temporelle à la totalité du système ;
- activation / désactivation des autres sous-systèmes en fonction de certaines conditions;
- redémarrage d'un sous-système en cas de court-circuit ;
- gestion et configuration des payloads (xEPS et D-STAR) ;
- configuration du récepteur et de l'émetteur ;
- décodage des télécommandes ;
- ordonnancement des télécommandes afin de les exécuter ;
- encodage des télémétries.

i) GND

Le sous-système Ground Station (GND) est le seul à ne pas faire partie du satellite car il se trouve dans le segment sol et non dans le segment espace. Son rôle est de permettre la communication avec le satellite OUFTI-1 une fois qu'il sera en orbite et donc de le commander en envoyant des télécommandes et en réceptionnant les télémétries. Il doit ensuite être capable d'écouter et de décoder le signal morse provenant de la balise à bord du satellite

et être en mesure d'effectuer des communications D-STAR afin de pouvoir tester la payload principale d'OUFTI-1. La figure 5 schématise l'implémentation du sous-système GND.

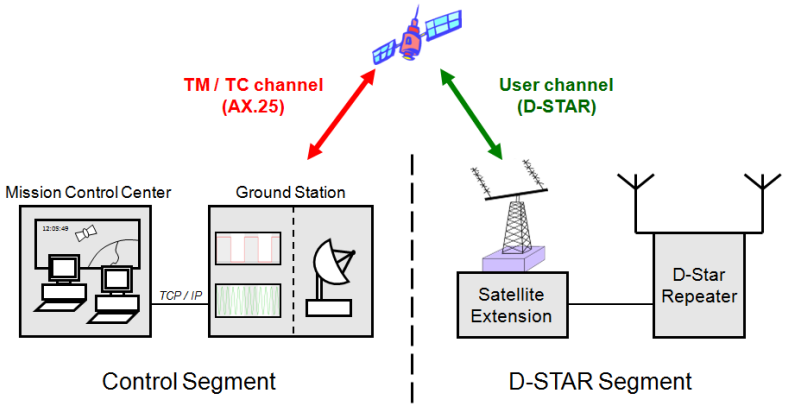

FIGURE 5 - LE SOUS-SYSTÈME GND.

4 PAYLOADS

a) D-STAR

Le D-STAR est un protocole de communication radioamateur numérique permettant de transporter simultanément voix et données. Il est utilisable sur les bandes radioamateur classiques (VHF, UHF et L) et propose également une méthode de connexion aux réseaux informatiques (dont internet) via TCP/IP. OUFTI-1 sera le premier satellite équipé d'un relais D-STAR permettant de relier deux opérateurs au sol avec une couverture bien plus large qu'un relais terrestre traditionnel.

Il faudra cependant tenir compte de l'effet Doppler induit par la vitesse de révolution du satellite par rapport aux utilisateurs au sol. Les émetteurs/ récepteurs D-STAR actuels ne permettant pas de corriger l'effet Doppler de façon suffisamment fine, les compensations doivent donc être effectuées à bord du satellite. Afin de simplifier l'utilisation du relais à bord, les données permettant de calibrer la compensation Doppler pour deux zones distinctes seront envoyées à OUFTI-1 par le canal de télécommunication classique (AX.25) avant chaque établissement d'une communication D-STAR.

b) xEPS

L'eXperimental Electrical Power Supply (xEPS) constitue une expérience technologique à bord d'OUFTI-1. La caractéristique principale de cette alimentation électrique expérimentale est son convertisseur de tension numérique chargé de réguler du 3,3 V. L'intérêt de l'expérimentation est que ce type de convertisseur n'a jamais été testé dans l'espace, contrairement aux alimentations à convertisseurs analogiques. La figure 6 représente la carte xEPS.

FIGURE 6 - LA CARTE DE L'XEPS

c) CELLULES SOLAIRES

Pour pouvoir produire sa propre énergie électrique, OUFTI-1 utilise des cellules solaires à haut rendement fournies par la société AZUR SPACE Solar Power. Ces cellules solaires à triple jonction sont capables d'atteindre les 30% de rendement. AZUR SPACE fournit gracieusement ces cellules solaires en contrepartie des données recueillies sur celles-ci à bord d'OUFTI-1. La figure 7 représente une des dix cellules présentes sur OUFTI-1.

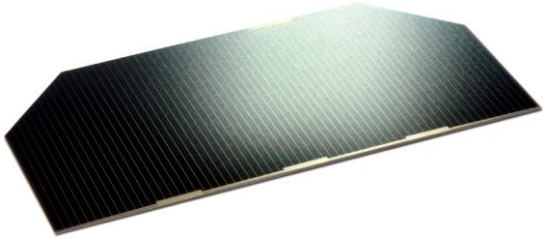

FIGURE 7 - CELLULES SOLAIRES D'OUFTI-1. SOURCE : AZURSPACE

5 LES OBJECTIFS DU PROJET

La mission d'OUFTI-1 est définie de façon graduelle et comporte donc différents objectifs à remplir pour la mener à bien. Chaque objectif comporte un ou plusieurs critères de succès qui permettent de déterminer si l'objectif est atteint ou non. Voici cette liste par ordre d'importance :

1. **Amusement et éducation**. Le premier objectif concerne les différentes équipes d'étudiants travaillant sur le projet. Chaque étudiant doit prendre du plaisir à travailler sur son sous-système. L'objectif principal est en effet de leur apporter de l'expérience dans le domaine spatial et donc de mener à leur développement personnel. Les trois critères de réussite de cet objectif sont les suivants :
    a. Les étudiants devraient obtenir de bonnes notes pour leur travail sur le projet ;
    b. Un nombre significatif d'étudiants devrait avoir envie de travailler sur le projet OUFTI-1 chaque année ;
    c. Le succès d'une année de travail peut être mesuré par la quantité d'étudiants diplômés qui continuent à participer activement ou non au projet.

2. **Design du système OUFTI-1.** Après l'amusement et l'éducation, le but principal et de construire un satellite, ce qui implique le design et l'implémentation d'un système spatial et d'un système sol. Le critère de succès de cet objectif est d'avoir un satellite fonctionnel et son système sol correspondant fonctionnant correctement.

3. **Lancement d'OUFTI-1.** Une fois le satellite construit, la prochaine étape est son lancement. Le critère de succès est un lancement effectué correctement.

4. **OUFTI-1 fonctionnant dans l'espace.** Après le lancement, la vie du satellite démarre. Le critère de succès est de recevoir un signal du satellite, indiquant qu'il est en vie.

5. **Commander le satellite.** Le critère de succès est de pouvoir envoyer une télémétrie, et de recevoir une télécommande.

6. **Avoir un système D-STAR fonctionnel.** La payload D-STAR sera la première à être activée car il s'agit de la payload principale. Le but est de prouver que le protocole D-STAR fonctionne dans l'espace. Un unique contact D-STAR via le satellite est suffisant pour considérer la payload D-STAR comme un succès.

7. **Faire fonctionner les payloads secondaires.** Les deux payloads secondaires, les cellules solaires et l'xEPS, sont finalement prises en compte. Le critère de succès est de recevoir des télémétries de ces deux payloads.

Chapitre III - HARDWARE

1 ORDINATEUR DE BORD REDONDANT

L'ordinateur de bord d'OUFTI-1 est composé de deux cartes, la première a été achetée par l'université de Liège auprès de la société Pumkin™ avec le reste du kit CubeSat : le FM430 (cf. figure 8). La seconde a été créée par les étudiants en s'inspirant du FM430 (cf figure 8). La motivation première concernant la carte faite maison était d'envoyer dans l'espace du matériel conçu par l'équipe d'OUFTI-1. La seconde raison était qu'un bon nombre de composants sur le FM430 ne sont pas utilisés. Créer une copie du FM430 sans les composants inutiles constituerait donc un gros gain d'espace, et on pourrait utiliser l'espace disponible sur la carte afin d'y mettre un autre sous-système (la balise morse de secours par exemple).
Mais le fait d'envoyer une carte qui n'a jamais été dans l'espace comporte des risques, c'est pourquoi il a été décidé d'utiliser la carte FM430 comme OBC de secours au cas où la carte « homemade » viendrait à tomber en panne.

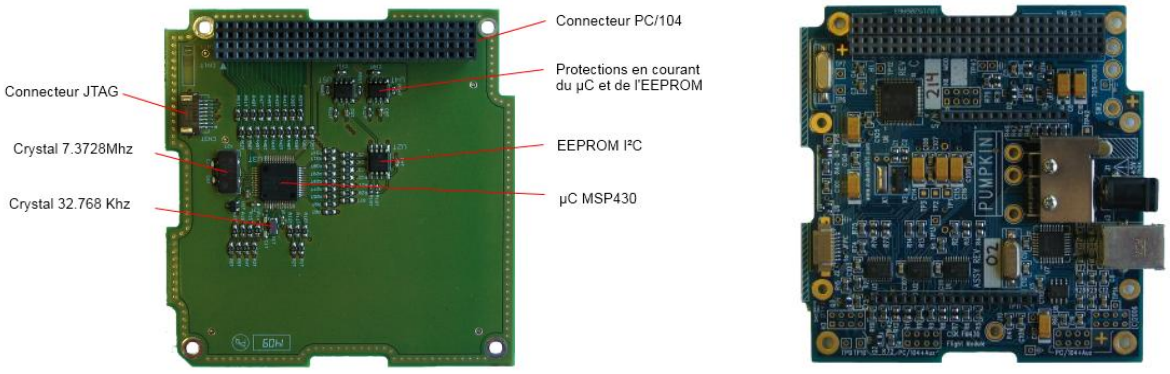

FIGURE 8 - A GAUCHE : DEFAULT OBC, CRÉE PAR LES ÉTUDIANTS DE L'ULG. A DROITE LE BACKUP OBC DE CHEZ PUMKIN

L'ordinateur de bord fait maison sera désormais désigné sous le nom de : « Default OBC ».
L'ordinateur de bord FM430 sera désormais désigné sous le nom de : « Backup OBC ».

Si le Default OBC présentait une défaillance, le Backup OBC prendra le contrôle des opérations. L'implémentation détaillée de la redondance des ordinateurs de bord est expliquée dans le chapitre 4.5 section « d ».

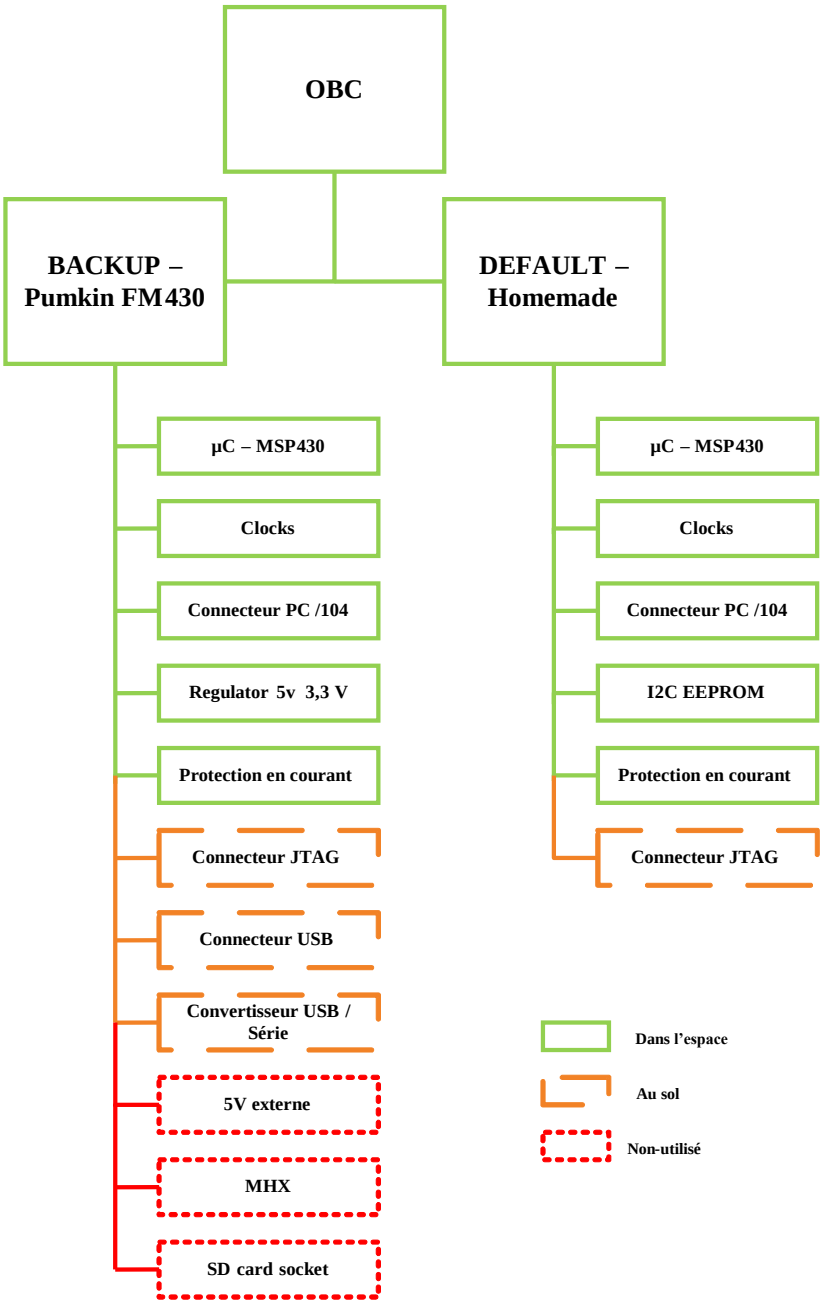

FIGURE 9 - PRODUCT TREE DE L'HARDWARE DE L'OBC

a) LE MSP430

Le MSP430F1612 de Texas Instruments™ est un microcontrôleur disposant d'un CPU RISC 16 bits programmable en langage C ou en assembleur. Il dispose de 55 Ko de mémoire flash pour le segment de code et le vecteur d'interruptions ainsi que de 5 Ko de RAM et est cadencé à 7.3728 Mhz via un oscillateur externe. Ce microcontrôleur a été choisi entre autre pour sa faible consommation mais surtout car il est identique à celui présent sur la carte FM430 de Pumkin™, facilitant le passage d'un OBC à l'autre en cas de défaillance.

Liste non-exhaustive des caractéristiques et périphériques du MSP430F1612:

- 5 modes d'économie d'énergie
- DMA
- ADC 12bits interne
- 2 timers 12bits
- 2 USART dont un I²C™-capable
- 6 ports d'I/O de 8 bits (Port1 & Port2 interruptibles sur front montant ou descendant)
- 3 clocks : ACLK (32.768 KHz), MCLK (7.3728 Mhz, utilisée par le CPU), SMCLK (utilisée par les modules périphériques)
- Watchdog Timer (WDT)

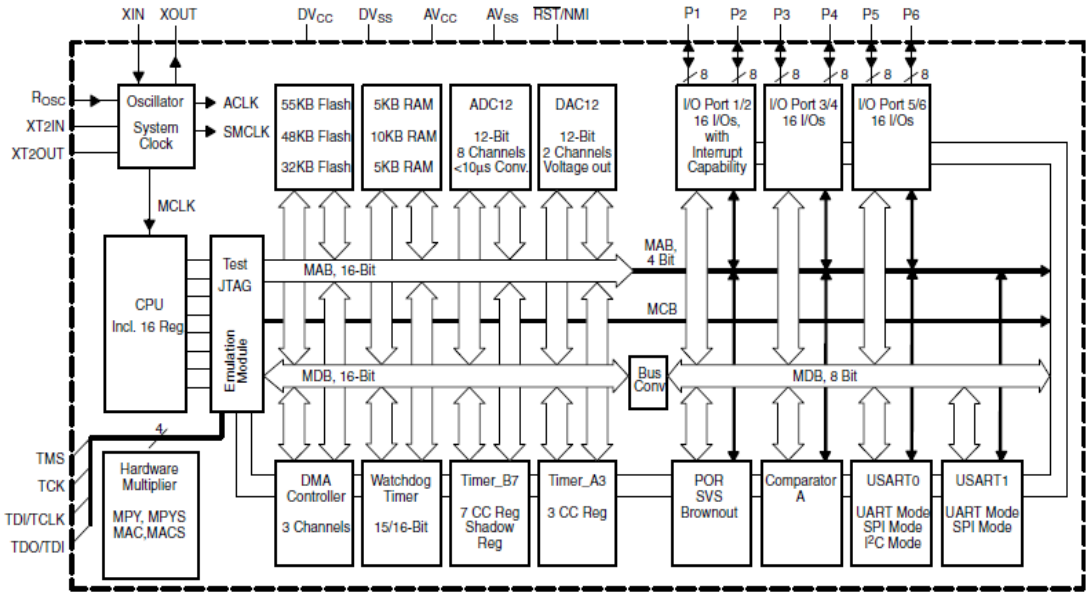

FIGURE 10 - ARCHITECTURE DU MSP430

b) LE BUS I²C

Le bus I²C, pour “Inter Integrated Circuit” est un bus série qui sert, comme son nom l’indique à faire communiquer les circuits intégrés. Il est facile à implémenter et ne requiert que deux lignes pour fonctionner : La ligne « DATA » transporte les données alors que la ligne « CLOCK » cadence les échanges. Le bus fonctionne en mode maître-esclave, c’est-à-dire qu’un périphérique du bus qui est configuré comme maître (un µC) orchestre les échanges sur celui-ci. C’est le maître du bus qui le cadence la ligne CLK, dans OUFTI-1 le bus est cadencé à 400 Khz (fastmode) par le MSP430 de l’OBC grâce à une division de l’oscillateur SMCLK. Chaque périphérique possède une adresse I²C sur 7 bits qui permet au maître de communiquer avec lui, pour plus d’information concernant le fonctionnement du bus I²C, consulter le chapitre 4.4 section « a » à propos de l’échantillonnage des mesures au sein du satellite.

La figure ci-dessous montre les circuits intègres connectés au bus I²C d’OUFTI-1.

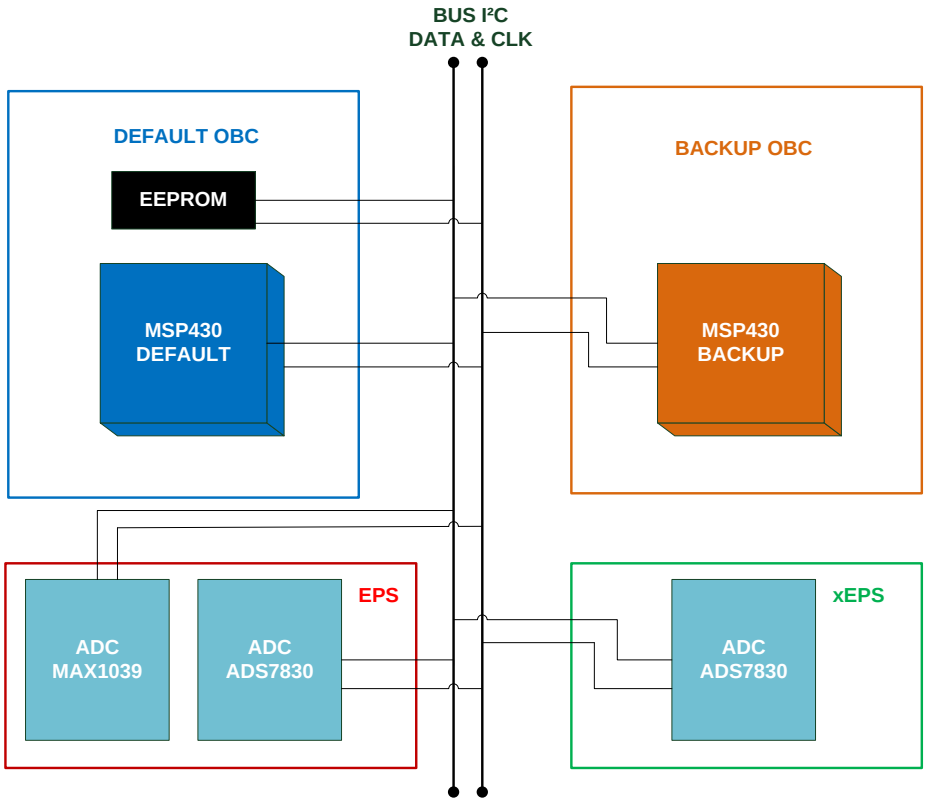

FIGURE 11- INTERCONNETION DU BUS I2C

c) L'EEPROM I²C

L'EEPROM située sur le Default OBC est le modèle 24xx1025 de chez MAXIM™. Elle est utilisée pour stocker les mesures prises en différents points du satellite, les événements du journal de bord et un flag signalant si les antennes ont été déployées ou non. Les deux OBCs peuvent effectuer des échanges avec l'EEPROM car elles communiquent grâce au bus I²C passant par les connecteurs PC/104 des cartes. Voici une liste de ses principales caractéristiques :

- Basse consommation (5mA en écriture / 500 µA en lecture / 100 nA au repos)
- I²C compatible.
- Écriture octet par octet ou en mode page (128 octets max.).
- Temps d'écriture de maximum 5ms.
- Capacité : 1Mo (deux blocs de 512 Mo).

Le fonctionnement détaillé de l'EEPROM 24xx1025 est illustré le chapitre 4.4 section « b » concernant l'enregistrement des mesures et le chapitre 4.4 section « c » à propos du rapatriement de ces mesures.

Chapitre IV - SOFTWARE

1 FREERTOS

freeRTOS est le système d'exploitation choisis pour l'ordinateur de bord d'OUFTI-1. C'est un système d'exploitation temps-réel spécialement conçu pour les systèmes embarqués de petite taille offrant peu de ressources matérielles. freeRTOS est gratuit et open-source, libre à qui le souhaite de consulter et/ou modifier son code source pour l'adapter à une application spécifique. Le noyau (kernel) de freeRTOS est très léger (seulement 3 fichiers) et occupe donc peu de place en ROM et en RAM. freeRTOS supporte un grand nombre de microprocesseurs, et est disponible préconfiguré pour chaque type d'architecture dans un package désigné sous le nom de « port ». Sa faible consommation de ressource due à sa petite taille n'est pas sans conséquences, en effet un certain nombre de fonctionnalités présentes dans d'autres RTOS ne sont pas disponibles (mailboxes, events, POSIX signals, ect.).

[www.freertos.org]

2 CONTRAINTES LIÉES AU MILIEU SPATIAL

L'espace constitue un milieu particulièrement rude pour le matériel électronique, on y trouve entre autres : des particules chargés, des champs électriques, des champs magnétiques, des radiations solaires ou encore des débris. OUFTI-1 est particulièrement vulnérable et cella pour deux raisons :

1. Son orbite est particulièrement agressive du point de vue des radiations [14].

2. L'ensemble des composants électroniques sont de type commercial, ils ne sont donc pas qualifiés pour l'espace. La différence de prix entre les composants commerciaux et composants qualifiés spatial est de l'ordre du ratio de mille. Ce qui est difficilement envisageable pour un projet étudiant.

Les effets des radiations peuvent être repartis en trois catégories:

- Effet de dose (TID, Total Ionizing Dose).

  Cet effet est irréversible, il est causé par l'accumulation de charges électriques sur un semi-conducteur qui une fois arrivé à saturation cessera de fontionner.

- Effet permanent (SEL, Single Event LatchUp).

  Causé par une particule fortement chargé, le Single Event Latchup provoque un court-circuit dans le composant affecté. Le composant court-circuité devient inutilisable tant qu'il n'a pas été reseté. Cet effet peut être détecté grâce aux protections de surconsommation MAX890 qui équipent les sous-systèmes. Pour plus d'informations concernant la correction de ce problème, consulter le chapitre 4.5 section « b ».

- Effet transitoire (SEU, Single Event Upset).

  Le single event upset est causé par l'impact une particule chargée avec une cellule mémoire pouvant modifier la charge de celle-ci. Ainsi, dans la l'EEPROM, la RAM ou même la ROM un « 1 » peut devenir un « 0 » et vice-versa. Les conséquences de cette altération peuvent être le plantage du programme si une donnée critique de la RAM est touchée, ou l'altération des données enregistrées en EEPROM. Un moyen de contrer cet effet est la redondance des données critique suivie d'un vote à la majorité. Cette précaution particulière est utilisée pour le flag signalant le déploiement des antennes (voir chapitre 4.5 section « a »).

3 ORGANISATION DU SOFTWARE

L'OBSW (On Board Software) est divisé en deux grandes parties : les modules et les drivers. Les modules représentent la partie applicative de l'OBSW, chaque module est construit autour d'une tâche (équivalent des threads dans freeRTOS) qui remplis un ou plusieurs rôles (cf. figure 12). Autour de chaque tâche gravite toute une série de fonctions et de variables qui lui permettent d'interagir avec l'extérieur (les autres tâches et les drivers). Les drivers quand à eux servent de couche d'abstraction entre les modules et le matériel, ils permettent de ne se soucier qu'une seule fois du fonctionnement bas niveau du matériel.

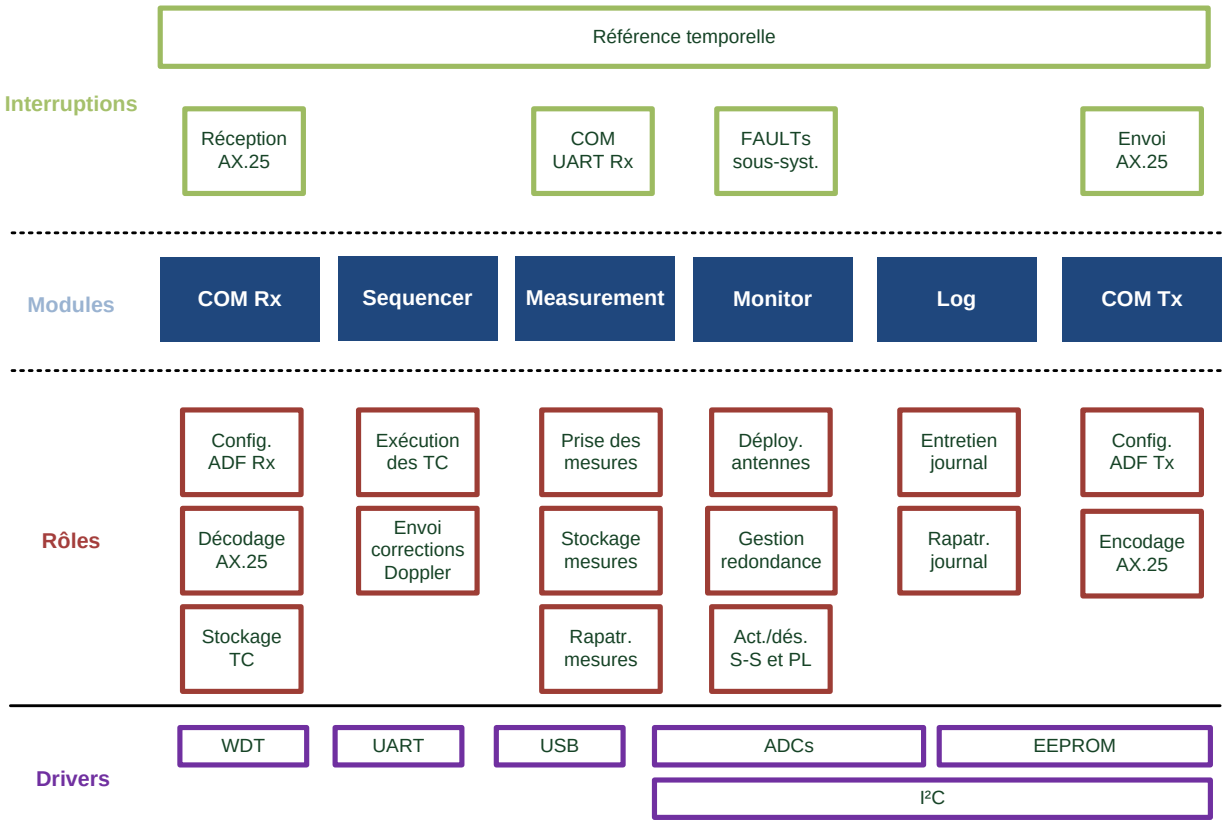

FIGURE 12 - ORGANISATION EN COUCHE DU LOGICIEL DE BORD

Chaque tâche de l'OBSW est crée par la fonction « main » qui démarre ensuite de scheduler de freeRTOS afin partager le temps processeur entre celles-ci.

.

- Toutes les tâches sont cycliques et ponctuelles, elles s'exécutent à une fréquence prédéterminée
- Chaque tâche a sa propre priorité, elle est déterminée par son importance et sa charge de travail.

- Le rôle du module COM Rx est de recevoir les télécommandes codées en AX.25, les décoder et les stocker dans le séquenceur. Il doit aussi ponctuellement reconfigurer l'ADF Rx (démodulateur) de la carte COM [1].

- Le module séquenceur reçoit les télécommandes du module COM Rx et est responsable de leur exécution. Il stock tout les jobs de l'OBSW, les organise et les exécutent au bon moment [1].

- Le module measurement est responsable de l'acquisition, l'enregistrement, et le rapatriement des SP au sein d'OUFTI-1 (voir chapitre 4.4).

- Le monitor est un module relativement multifonctions, il se charge entre autres du déploiement des antennes, de la surveillance des paramètres vitaux du satellite ou du changement de configuration (mode) de celui-ci (voir chapitre 4.5).

- Le module log est chargé de la journalisation des événements importants de la vie du satellite, il doit pouvoir les enregistrer et les rapatrier (voir chapitre 4.6).

- COM Tx est le module de transmission des télémétries, ils les encodent au format AX.25 et les transmet au système d'émission radio de la carte COM. Tout comme le module COM Tx, il doit ponctuellement reconfigurer le modulateur d'émission.

4 MODULE MEASUREMENT

Les fonctions d'acquisition et d'enregistrement des « Science Parameters » d'OUFTI-1 sont effectuées par la tâche « measurement », ces mesures qui sont pour la plupart d'origine matérielle sont enregistrées dans l'EEPROM I²C. Les mesures échantillonnées à un moment donné et la fréquence de celles-ci sont des paramètres configurables par télécommande. Lorsqu'une télécommande ordonne le rapatriement des mesures stockées, l'OBC doit récupérer celle-ci dans l'EEPROM et les insérer dans une télémétrie. La figure 13 illustre le fonctionnement de la tâche avec ces interfaces en entrée et en sortie.

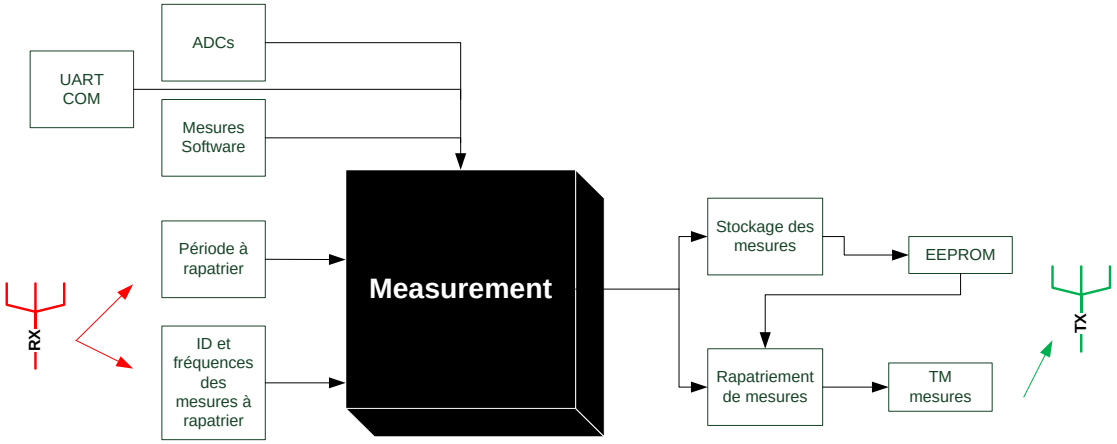

FIGURE 13 - BLOCK DIAGRAM DU MODULE MEASUREMENT

La structure en RAM utilisé pour configurer la boucle d'échantillonnage des mesures est présentée dans la figure 14.

mID	Frequency	Last value	Timestamp	Get it?
Identifiant unique pour cette mesure	Fréquence d'acquisition pour ce type. (un multiple de la fréquence de la tâche)	Dernier valeur échantillonnée.	Emprunte temporelle du dernier échantillonnage pour ce type	Ce type doit-il être échantillonné et enregistré actuellement.

FIGURE 14 - STRUCTURE DE CONFIGURATION DE LA BOUCLE D'ÉCHANTILLONNAGE DES MESURES

Même si le fait de conserver les dernières acquisitions et leur timestamp peut paraitre couteux en mémoire vive, cella facilite et accélère le rapatriement du dernier échantillonnage.

La tâche « measurements » :

Comme toutes les tâches du logiciel de bord, la tâche measurements est cyclique et se répète à intervalles réguliers, c'est cette propriété qui va lui permettre d'échantillonner les mesures à des fréquences bien précises. La fréquence d'occurrence de la tâche à été fixée à 1Hz, c'est cette fréquence qui fixe la fréquence maximale d'acquisition.

Avant d'entrer dans sa boucle, la tâche va lire en EEPROM l'index des mesures permettant de trouver l'adresse à laquelle il faut enregistrer les mesures en EEPROM. Au premier démarrage cet index doit valoir 0. Ensuite, à chaque tour de boucle, la tâche passe sur chaque mesure pour voir si celles-ci doivent être échantillonnées actuellement. Il faut pour cella que :

1 – Le flag « Get It ? » de cette mesure soit positionné.

2 – Que ce tour de boucle soit celui qui corresponde à la fréquence d'acquissions de la mesure. On peut déterminer cela en regardant si le reste de la division de la valeur actuelle de la référence temporelle (en secondes) par la fréquence d'échantillonnage de la mesure est nul.

Si ces deux critères sont satisfaits on peut alors faire l'acquisition de la mesure (voir chapitre suivant), son enregistrement dans l'EEPROM et passer à la mesure suivante.

Une fois toutes les mesures nécessaires acquises on enregistre une copie de l'index EEPROM et on recommence depuis la première mesure.

Toutes ces étapes sont visibles dans l'ordinogramme de la figure 15.

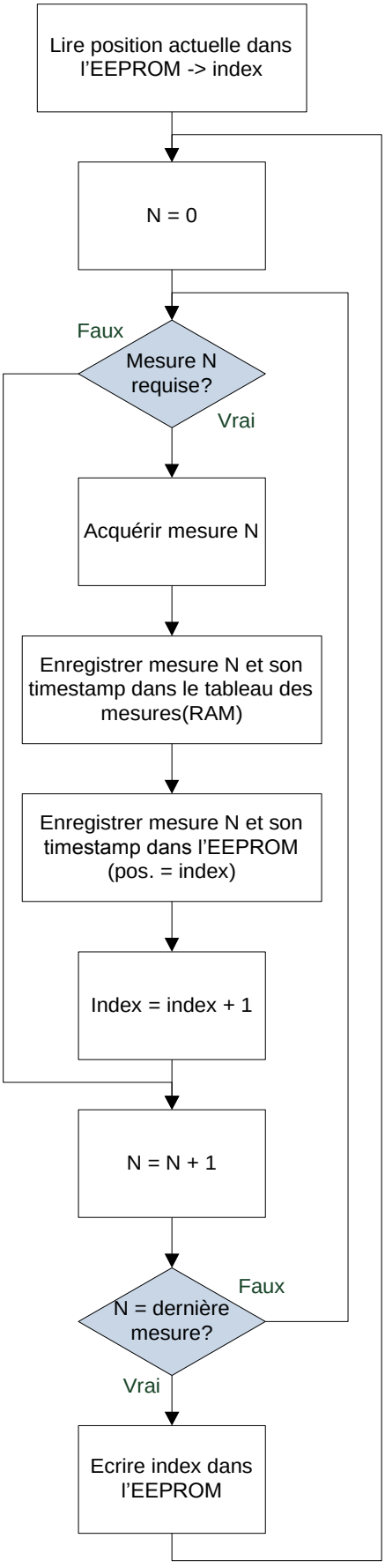

FIGURE 15 - ORDINOGRAME DE LA TÂCHE MEASUREMENT

a) ACQUISITION DES MESURES

Les mesures à effectuer à bord sont de 4 types :

- Mesures de tensions : Utilisation de ponts diviseurs pour adapter la tension à mesurer à la plage d'entrée du convertisseur analogique/digital.
- Mesures de courant : le circuit intégré MAX9938 en série donne une tension proportionnelle au courant mesuré.
- Mesures de températures : Le circuit intégré LM94022 fournis une tension inversement proportionnelle à la température mesurée.
- Mesures logicielles : contenu de registres de différents µC.

Les tensions sont donc recueillies par des convertisseurs A/D en différents points du satellite, on en retrouve deux sur la carte EPS, un sur la carte xEPS et un sur la carte COM.

Ces ADCs sont de trois types différents.

Le MAX1039AEEE de chez Maxim Integrated Products™ est un convertisseur A/D 8 bits, I²C-Compatible, onze ou douze canaux en mode unipolaire en fonction du choix de la référence (interne ou externe) ou alors cinq à six canaux en mode différentiel, 188Ksps (Kilo Samples Per Second) Maximum.

L'ADS7830 de chez Texas Instruments™ est un ADC 8 bits, I²C-Compatible, huit canaux en mode unipolaire ou quatre canaux en mode différentiel, référence interne de 2.5V, vitesse d'échantillonnage maximum : 70ksps.

Le dernier type d'ADC est en fait **le convertisseur ADC12 interne du MSP430**, il possède une résolution sur 12 bits, une vitesse de conversion jusqu'à 200ksps et une référence programmable a 1.5V ou 2.5V.

Les deux premiers ADC sont utilisé en mode esclave sur le bus I²C du satellite, ils nécessitent un driver afin d'être pilotés à partir du µC de l'OBC, ces drivers s'appuient sur le driver de bus I²C développé par D. Teney dans le cadre de son mémoire sur l'OBC d'OUFTI-1 [5]. Le troisième ADC est quand à lui utilisé pour les mesures sur la carte COM, c'est donc l'étudiant en charge de ce sous-système qui est responsable de l'écriture de ce driver. Néanmoins, la carte COM n'étant pas desservie par le bus I²C, un protocole de communication série « OBC–COM » doit être mis en place pour permettre le rapatriement des données à acquérir sur le

convertisseur A/D COM, ce cas spécifique sera abordé plus tard dans ce chapitre. Un diagramme récapitulatif des chaines de mesures du nanosatellite est présenté sur la figure 16, les mesures désignées par leur identifiant sur ce diagramme son détaillées dans l'annexe B.

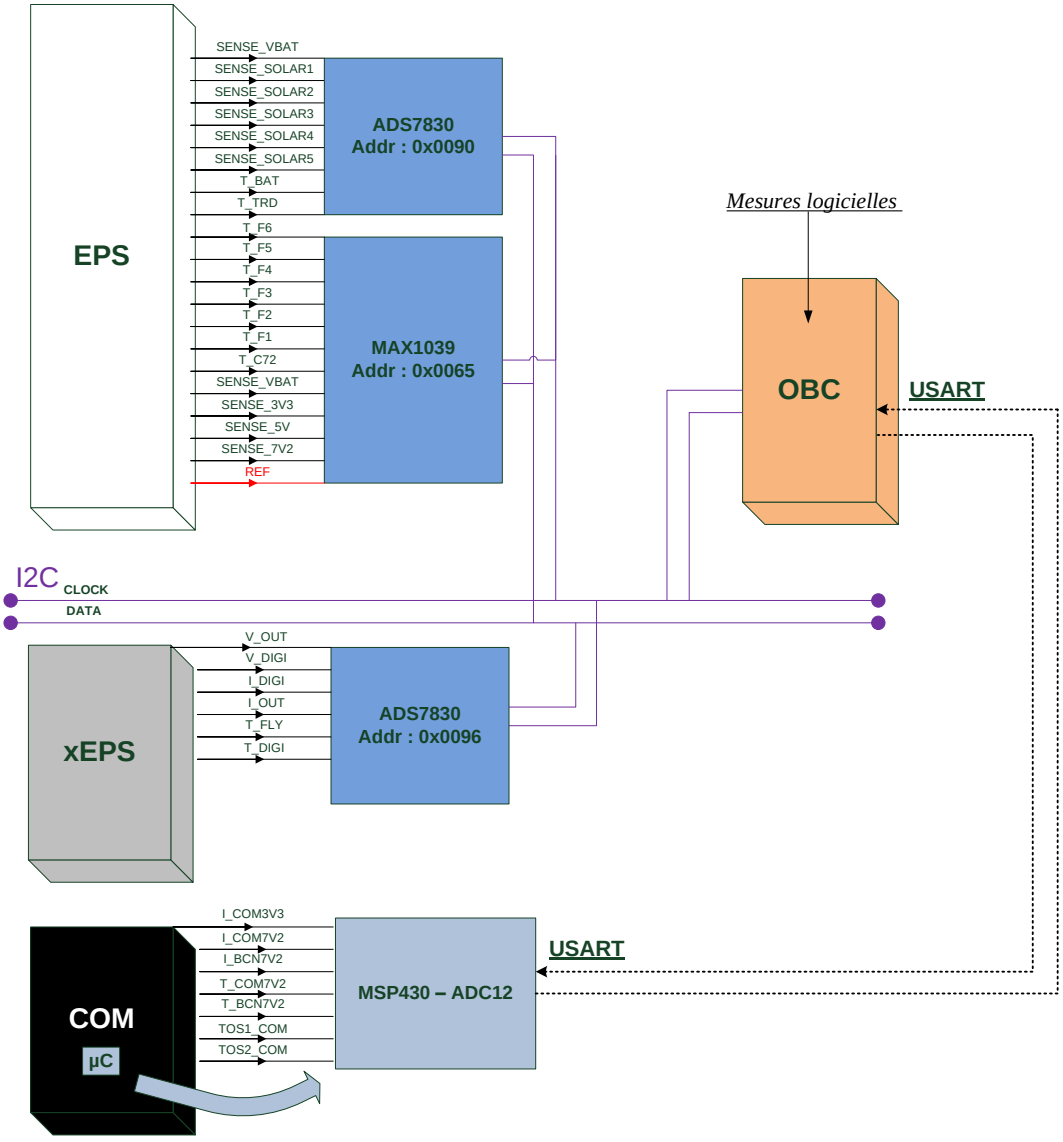

FIGURE 16 - CHAINE COMPLÈTE D'ACQUISITION DES MESURES AU SEIN D'OUFTI-1

Le MAX1039AEEE de l'EPS

Description du brochage du composant (cf figure 17) selon sa configuration sur l'EPS :

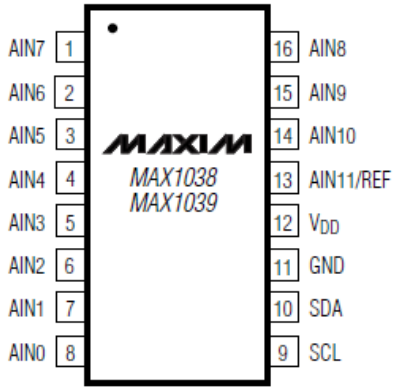

FIGURE 17 - BROCHAGE DU COMPOSANT MAX1039AEEE

AIN0 – AIN10: 11 entrées analogiques en mode unipolaire.

AIN11 / REF: Référence externe calibrée à 2.5V.

V_{DD} / GND: Alimentation et masse du composant.

SDA / SCL: Data & Clock I²C

Adressage : L'adresse du MAX1039AEEE en tant qu'esclave sur le bus I²C n'est pas paramétrable, en effet son adresse est fixée d'usine. Néanmoins si il s'avérait nécessaire d'utiliser plusieurs MAX1039AEEE sur le bus I²C du satellite, MAXIM™ offre d'autres composants de la série MAX1039 qui proposent exactement les mêmes caractéristiques que le MAX1039AEEE si ce n'est l'adresse esclave I²C : les MAX1039KAEEE, MAX1039LAEEE et MAX1039MAEEE, ce qui autorise un maximum de 4 composants de la série MAX1039 sur le même bus. Tous les périphériques I²C possèdent une adresse sur 7 bits, le MAX1039AEEE ne déroge pas à cette règle et son adresse est : 0b1100101.

La première étape en vue de récupérer une valeur échantillonnée sur l'ADC est donc d'adresser le périphérique cible en écriture, pour se faire, le maitre du bus (le µC de l'OBC) initie le dialogue en envoyant un « START-BIT » suivi de l'adresse du MAX1039. La communication sur le bus I²C se faisant octet par octet et l'adresse d'un périphérique étant sur 7 bits, le dernier bit (le LSB) va servir à désigner le sens de la communication à effectuer sur l'esclave, un « 0 » logique désigne une opération d'écriture. Si cette étape s'est déroulée correctement, le MAX1039 produira un ACK comme illustré dans la figure 18.

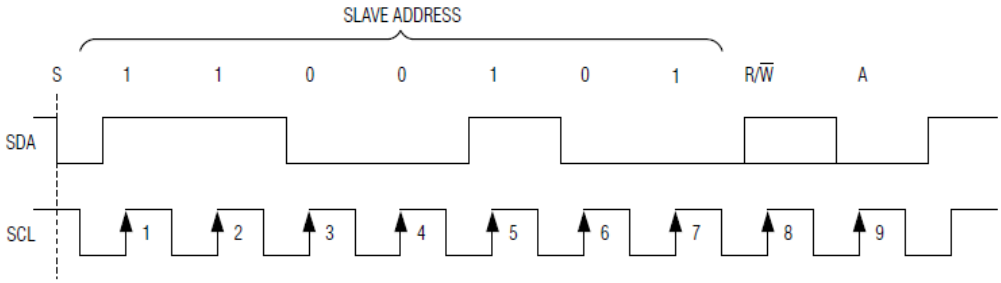

FIGURE 18 - ADRESSAGE DU MAX1039 EN ÉCRITURE

Une fois l'ACK reçu, la seconde étape consiste à configurer le MAX1039, celui-ci dispose de deux registres permettant de le paramétrer grâce au « Setup byte » et au « Configuration byte ».

Setup byte

BIT7 (MSB)	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0 (LSB)
REG	SEL2	SEL1	SEL0	CLK	BIP/$\overline{\text{UNI}}$	$\overline{\text{RST}}$	X
1	**0**	**1**	**0**	**0**	**0**	**0**	**0**

FIGURE 19 - SETUP BYTE DU MAX1039 POUR SON UTILISATION SUR LA CARTE EPS

REG : Permet d'identifier l'octet de configuration (1 : Setup byte / 0 : Configuration byte).

SEL0 – SEL2 : Permet de choisir entre la référence de tension interne ou externe et donc de définir l'utilisation de la broche AIN11, dans notre cas, il a été décidé d'utiliser pour tous les ADCs une référence à 2.5V pour simplifier l'implémentation. Cette tension n'étant pas disponible en référence interne, une référence externe est nécessaire.

CLK : Avec un zéro logique, le convertisseur A/D est cadencé via son horloge interne.

BIP/$\overline{UNI}$: 0 = Entrées analogiques en mode unipolaire, n'autorisant pas les tensions négatives en entrée.

$\overline{RST}$: Permet de demander un reset du « Configuration register », étant donné que pour plus de sécurité les deux registres sont paramétrés à chaque utilisation, ce bit importe peu.

X : non-utilisé.

Configuration byte

BIT7 (MSB)	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0 (LSB)
REG	SCAN1	SCAN0	CS3	CS2	CS1	CS0	SGL/$\overline{DIF}$
0	**1**	**1**	**A**	**B**	**C**	**D**	**1**

FIGURE 20 - CONFIGURATION BYTE DU MAX1039 POUR SON UTILISATION SUR LA CARTE EPS

REG : Permet d'identifier l'octet de configuration (1 : Setup byte / 0 : Configuration byte).

SCAN0 – SCAN1 : Effectuer la conversion uniquement sur le canal sélectionné par CS0 – CS2.

CS0 – CS3 : Sélectionner le canal à convertir.

SGL/$\overline{DIF}$: Sert à choisir entre le mode asymétrique ou le mode différentiel.

La figure 21 illustre une trame typique de configuration du MAX1039 sur le bus I²C.

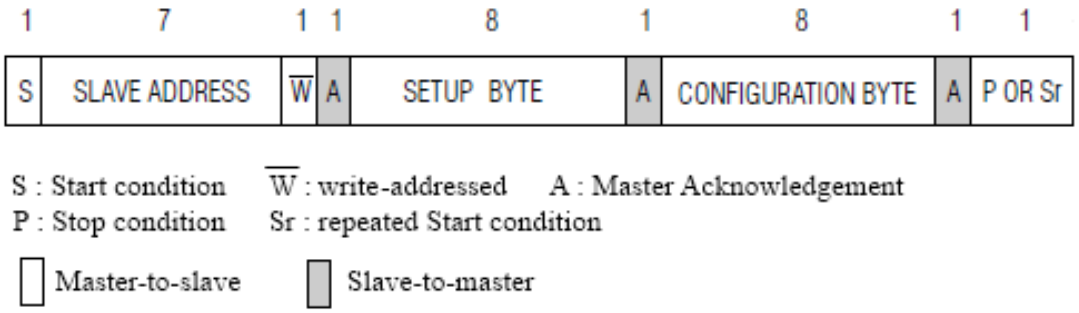

FIGURE 21 - TRAME COMPLETE DU CONFIGURATION DU MAX1039

Une fois configuré, le MAX1039 va initier la conversion et stocker le résultat dans un registre interne. Pour récupérer la donnée, il suffit alors d'adresser le périphérique en lecture et une fois l'octet reçu d'émettre un NACK pour demander au MAX1039 d'arrêter d'envoyer le résultat de la dernière conversion (cf. figure 22).

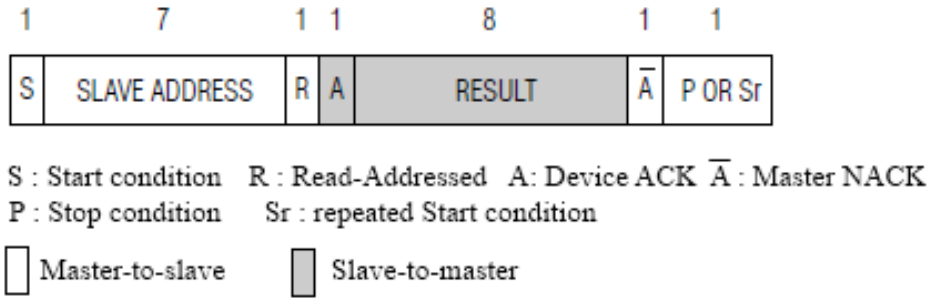

FIGURE 22 - TRAME D'ACQUISITION D'UN RÉSULTAT SUR LE MAX 1039

L'ADS7830, présent sur l'EPS et l'xEPS

Description du brochage du composant (cf. figure 23) selon sa configuration sur l'EPS et l'xEPS :

CH0 – CH7: 8 entrées analogiques en mode unipolaire.

$+V_{DD}$ / GND: Alimentation et masse du composant.

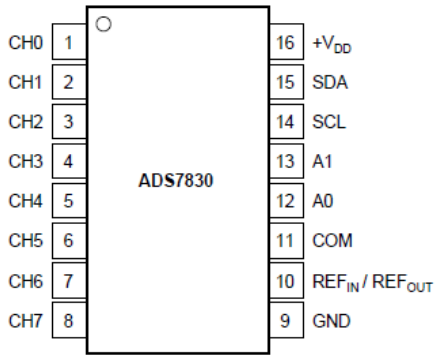

FIGURE 23 - BROCHAGE DE L'ADS7830

SDA / SCL: Data & Clock I²C

A0 – A1: Bits 0 et 1 de l'adresse esclave I²C (voir adressage).

COM: Mis à la masse dans le cas d'une utilisation des entrées en mode unipolaire.

REF_{IN} / REF_{OUT}: Non-utilisé, l'ADS7830 propose une référence interne de 2.5V.

Adressage : Contrairement au MAX1039AEEE, l'adresse I²C de l'ADS7830 est paramétrable grâce aux broches A0 et A1, l'adresse de base est : « 10010 », suivie des valeurs de A1 et A0, ce qui permet d'utiliser au maximum quatre ADS7830 sur le même bus I²C.

Quand une des broches A0 ou A1 est connectée à la masse elle donne un zéro, connectée à $+V_{DD}$ elle donne un 1 logique (cf. figure 24) :

SS	**7**	**6**	**5**	**4**	**3**	**A1** **2**	**A0** **1**	**R/$\overline{W}$** **0**
EPS	1	0	0	1	0	0	0	0
xEPS	1	0	0	1	0	1	1	0

FIGURE 24 - OCTET D'ADRESSE DES ADS7830 DE L'EPS & XEPS

On remarquera qu'à l'instar de l'octet d'adressage du MAX1039, le LSB contient le bit désignant le sens le l'opération (ici, une écriture). Après avoir adressé le convertisseur et obtenu un ACK, il faut le configurer en vue de la lecture, l'ADS7830 ne possède qu'un seul octet de configuration : le Command byte.

Command byte

BIT7 **(MSB)**	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0** **(LSB)**
SD	C2	C1	C0	PD1	PD0	X	X
1	**A**	**B**	**C**	**1**	**1**	**0**	**0**

FIGURE 25 - COMMAND BYTE DE L'ADS7830 POUR L'UTILISATION SUR L'EPS & L'XEPS

SD : Choix unipolaire / différentiel, dans notre cas : unipolaire.

C0 – C2 : Sélectionner le canal à convertir.

PD0 – PD1: Permet de choisir l'état de la référence interne 2.5V, dans notre cas : ON.

X : Non-utilisé (valeur indifférente).

Une fois l'ADS7830 adressé en écriture et le command byte envoyé, on peut récupérer la donnée échantillonnée en adressant l'ADS en lecture. Une fois l'octet reçu il faut, tout comme pour le MAX1039, émettre un NACK.

La figure 26 résume les étapes nécessaires à l'acquisition d'un échantillon sur l'ADS7830.

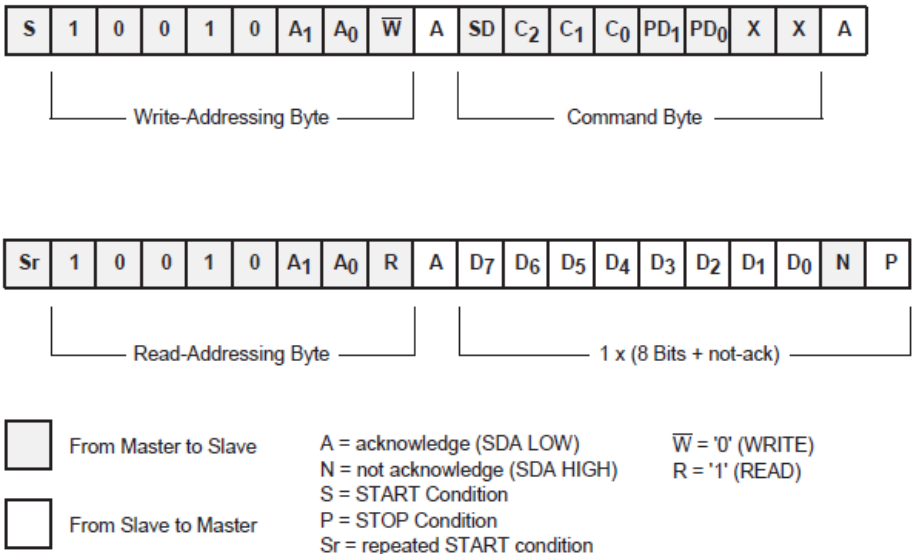

FIGURE 26 - TRAME DE CONFIGURATION ET D'ACQUISITION D'UNE MESURE SUR L'ADS7830

Communication COM-OBC

L'ADC prévu pour échantillonner les mesures sur la carte COM est donc l'ADC12, interne au µC de la carte. Pour rappel, la carte COM n'est pas desservie par le bus I²C, il va donc falloir utiliser un des deux ports série MSP430 de l'OBC afin de créer un canal de communication avec le MSP430 de la COM.

Le MSP430 dispose de deux modules USART (universal synchronous/asynchronous receive/transmit) : USART0 et USART1 ; USART0 étant utilisé par le bus I²C, USART1 sera dédié à la communication série OBC-COM.

Le module USART1 est configuré en mode UART, dans ce mode deux pins sont utilisées pour créer une communication asynchrone bidirectionnelle full-duplex : UTXD1 est le port de transmission, les données déposées sur ce port sont transmises vers la carte COM. URXD1 est le port de réception, lorsque la carte COM envoie des données, elles sont reçues via ce port et génèrent une interruption matérielle (cf. figure 27).

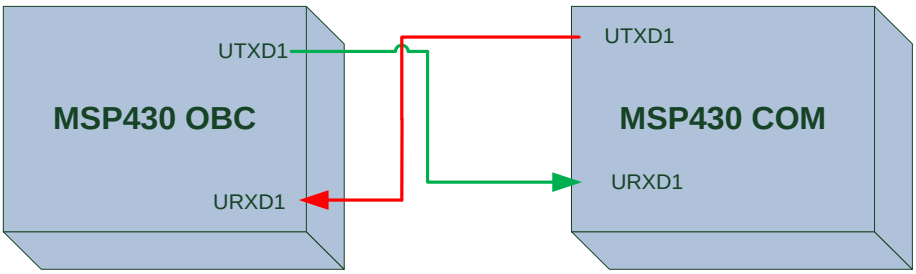

FIGURE 27 - INTERCONECTION DES µC MSP430 DE L'OBC ET DE LA COM VIA UN UART

Ce canal de communication avec la carte COM a d'autres emplois mis à part le rapatriement des mesures COM :

- L'envoi des corrections doppler pour le D-STAR [*1*] ;
- L'activation / désactivation du répéteur D-STAR (voir chapitre 4.5 section « c ») ;

Dans notre communication UART, les données sont transmises par paquets de 8 bits les uns à la suite des autres sur la liaison UTXD1 à une vitesse de 9600 bauds. Une trame UART est constituée comme sur la figure suivante.

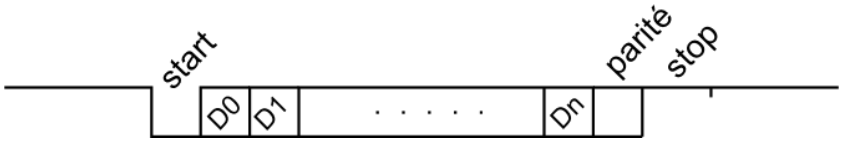

FIGURE 28 - CONSTITUTION D'UNE TRAME UART

Start bit : Il permet de synchroniser la communication, lorsque le récepteur détecte le niveau bas, cella signifie que le paquet de données suit.

D0 - Dn : C'est le paquet de données, il est constitué de 8 bits dans le cas de notre implémentation.

Parité : Il est égal à « 1 » si le nombre de 1 dans le paquet de données est pair, il vaut 0 si ce nombre est impair. Le bit de parité constitue un moyen simple de détection d'erreurs, en effet le module récepteur peut vérifier l'intégrité du paquet de données entre autre grâce à ce bit.

Stop bit : Toujours à « 1 », il signale la fin de la trame.

Le premier octet à envoyer, appelé « TOC » (Type Of Communication) est un identifiant unique pour chaque type d'échange sur le canal OBC-COM, il sert à informer le récepteur sur le type de message qui va arriver. Chaque TOC étant unique, le récepteur sait donc le nombre d'octets qu'il va recevoir.

Voici une liste de messages (cf. figure 29) transmis sur l'UART OBC-COM, seuls ceux utiles a notre implémentation ont été retenus.

Nom de la commande	TOC	DATA	Sens	Description
Enable D-STAR	E	/	OBC > COM	Active le relais D-STAR. Sans correction Doppler.
Disable D-STAR	D	/	OBC > COM	Désactive le relais D-STAR.
Ask Mesure	P	Type de Mesure	OBC > COM	Demande une mesure au MSP COM
Mesure	M	Mesures	COM > OBC	Mesure du MSP COM
Mini-Log D-STAR	L	Mini-Log D-STAR	COM > OBC	Informations relatives au relais D-STAR

FIGURE 29 - LISTE DES MESSAGES TRANSMIS ENTRE L'OBC ET LA COM

L'implémentation du protocole applicatif à été réalisée par Nicolas Marchal dans le cadre de son mémoire sur la carte COM [3], les principes majeurs de cette implémentation sont exposés ici.

Etant donné la nature asynchrone de la liaison UART, on ne peut pas déterminer quand la donnée demandée va nous parvenir. Alors que dans la boucle d'échantillonnage des mesures, on voudrait pouvoir enregistrer la mesure juste après l'avoir demandé et pas un ou deux tours de boucle plus tard, dans le but de préserver une fréquence d'échantillonnage correcte. Pour remédier à ce problème, un mécanisme élémentaire de synchronisation sera utilisé : un sémaphore.

Lors de l'émission de la demande de mesure, la première chose à faire est donc de verrouiller le sémaphore afin d'empêcher la tâche d'acquisition de sauvegarder une donnée qui n'est pas

encore arrivée. Il faut ensuite émettre le TOC, l'identifiant de la mesure et le checksum CRC, si la carte COM réponds par un ACK applicatif, tout va bien, on peut attendre la donnée. Si par contre on ne reçoit pas de confirmation après dix essais, il faut déverrouiller le sémaphore afin d'empêcher la tâche d'acquisition d'attendre indéfiniment (pour plus de sécurité, le sémaphore est pourvu d'un compteur d'inactivité).

La réception de la mesure se fait par interruption matérielle, chaque trame reçue va donc déclencher la routine d'interruption. Pour éviter de passer trop de temps dans cette routine et ainsi monopoliser le temps processeur, la routine d'interruption fonctionne selon le principe de la machine à états. A la fin de chaque passage dans la routine, le contexte d'exécution est sauvegardé afin de reprendre là où on en était. Apres avoir reçu la mesure attendue et contrôlé son intégrité grâce au CRC, on peut émettre un ACK vers la carte COM et déverrouiller le sémaphore pour permettre à la tâche « measurement » de sauvegarder la mesure et reprendre son exécution.

Voici le format de la trame UART pour récupérer une mesure :

FIGURE 30 - DONNÉES ÉMISES ET RECUE LORS DE L'ACQUISITION D'UNE MESURE DE LA COM

Les ordinogrammes des fonctions d'émission et de réception spécifiques aux mesures sont détaillés sur la figure 31.

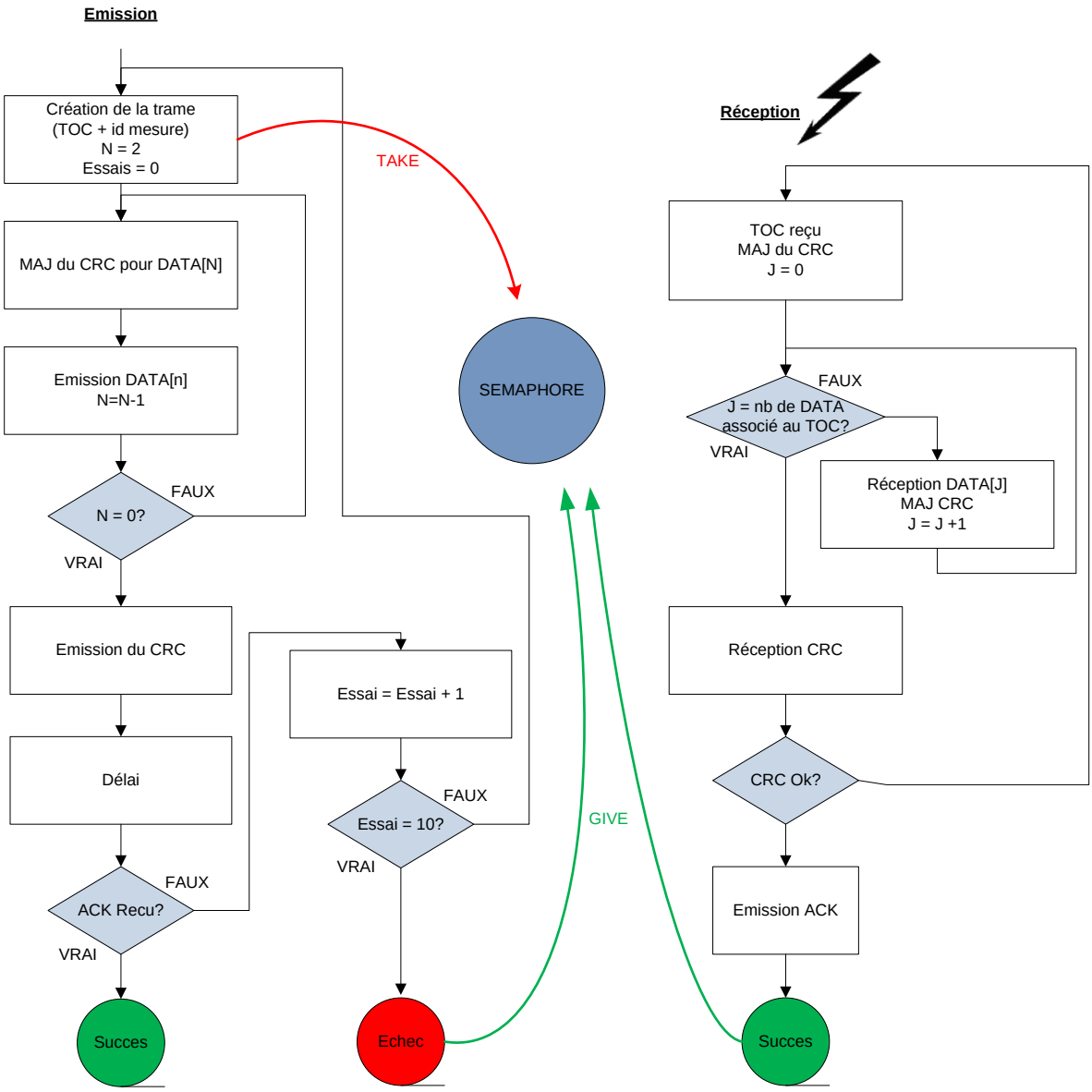

FIGURE 31 - ORDINOGRAMMES DES FONCTIONS D'EMISSION ET DE RÉCEPTION VIA L'UART OBC - COM

Mesures propres à l'OBC

Température du µP de l'OBC : Le MSP430 de l'OBC dispose d'un capteur de température interne connecté sur le port 10 de l'ADC12. Pour récupérer la valeur de la température il faut configurer l'ADC12 pour fournir un résultat sur 8 bits, choisir la référence à 1.5V et sélectionner le canal d'entrée 10(cf. extrait de code de la figure 32).

```
int msp430_getTemperature(void)
{
  ADC12CTL0 = SHT0_8 + REFON + ADC12ON;// Resultat sur 8bits, Ref. ON, ADC ON
  ADC12CTL1 = SHP;                     // sample timer activé
  ADC12MCTL0 = SREF_1 + INCH_10;       // Ref. 1.5V , Capteur de T°

  ADC12CTL0 |= ENC | ADC12SC;          // Debut de conversion

  while (ADC12CTL1 & ADC12BUSY);       // Attente de la fin de conversion

  return ADC12MEM0;                    //renvoyer la température
}
```

FIGURE 32 - FONCTION D'ACQUISITION DE LA TEMPÉRATURE INTERNE DU MSP430

Pour pouvoir interpréter le résultat obtenu et obtenir la valeur en degrés Celsius, il faut utiliser la fonction de transfert de la figure 33 fournie avec la datasheet du composant.

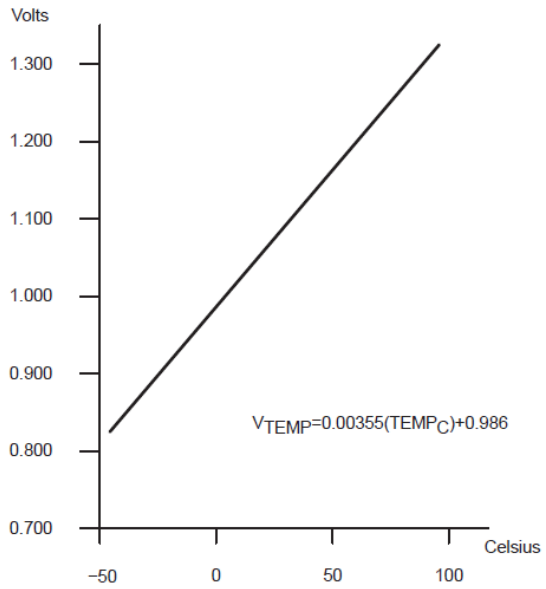

FIGURE 33 - FONCTION DE TRANSFERT DU CAPTEUR DE TEMPÉRATURE DU MSP430

Test du capteur de T° :

L'ADC nous fournis une valeur de 182, sachant que l'ADC est configuré pour une résolution sur 8 bits (256 valeurs) et que la référence est à 1.5V, on peut obtenir la tension de sortie du capteur.

$$\begin{aligned} 1.5 &= \frac{255.V}{182} \\ 273 &= 255.V \\ V &\cong 1.07 \end{aligned}$$

En appliquant le résultat obtenu à la fonction de transfert du capteur, on obtient la température en degrés Celsius.

$$1.07V = 0.00355(TEMPc) + 0.986$$

$$0.084 = 0.00355(TEMPc)$$

$$\frac{0.084}{0.00355} = TEMPc$$

$$TEMPc \cong 23.66°C$$

Pourcentage d'utilisation du CPU : Le principe envisagé pour récupérer le pourcentage d'utilisation moyenne et instantanée du CPU est d'utiliser la tâche « idle » (routine exécutée en boucle par le CPU quand il n'a rien d'autre à faire). En incrémentant un compteur dans cette tâche et en réinitialisant ce compteur toutes les secondes, on peut déterminer le nombre de « ticks » ou le processeur est inactif :

$$\left(1 - \left(\frac{nbIdleTicks/sec}{CPU\ frequency\ (Hz)}\right)\right)100 = utilisation(\%)$$

Malheureusement, cette solution n'est pas une bonne idée dans le contexte d'OUFTI-1, L'énergie est une ressource précieuse et limitée dans les systèmes embarqués alimentés par batteries et le rôle de la tache idle deviens alors de passer le CPU en mode basse consommation. Cette fonctionnalité a tout de même été implémentée et est accessible pour le debugging au sol.

« HighWaterMark » des tâches : La fonction « uxTaskGetHighWaterMark() » de freeRTOS permet de connaitre l'espace restant sur le stack d'une tâche, cette fonction à été fort utile pour dimensionner correctement le stack de chaque tâche (voir chapitre 4.8), elle peut aussi s'avérer utile a bord en tant qu'outil de diagnostique. Le paramètre à passer à cette fonction est le handler de la tâche (ou NULL pour la tâche qui appelle la fonction), la valeur de retour exprime la taille de l'espace restant sur le stack en mots (deux octets pour l'architecture 16 bits du MSP430).

b) ENREGISTREMENT DES MESURES

Après leur échantillonnage, les mesures sont enregistrées dans l'EEPROM connectée au bus I²C. L'EEPROM choisie est la 24xx1025 de Microchips™, ses caractéristiques sont décrites dans le chapitre 3.1 section « c ».

Description du brochage du composant (cf. figure 34) selon sa configuration sur l'OBC:

A0 – A1: Bits 0 et 1 de l'adresse esclave I²C (voir adressage).

A2: Non configurable, doit être relié à $V_{CC.}$

V_{SS} / V_{CC}: Alimentation et masse du composant.

WP: Permet de mettre l'EEPROM en lecture-seule.

SCL / SDA: Data & Clock I²C.

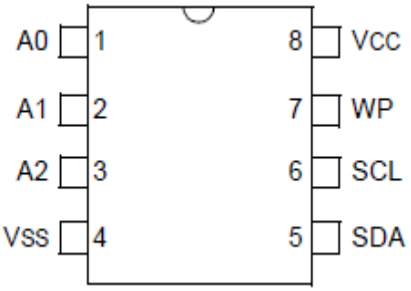

FIGURE 34 - BROCHAGE DE L'EEPROM 24XX1025

Adressage :

Comme les autres périphériques I²C, l'EEPROM 24xx1025 possède une adresse sur 7bits, et le huitième bit de l'octet d'adressage sert à designer l'opération (lecture ou écriture) à effectuer sur le périphérique. L'adresse de base de l'EEPROM est « 1010 ».L'EEPROM à une taille de 1024Kbits, elle est divisée en deux segment logiques de 512Kbits chacun. Le bit suivant l'adresse de base sert a designer dans quel bloc on souhaite aller lire ou écrire. Les bits 5 & 6 sont fixés en fonction du raccordement des broches A0 et A1. L'octet d'adressage pour l'EEPROM 24xx1025 de l'OBC est illustré sur la figure ci-dessous.

BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**A1 BIT2**	**A0 BIT1**	**R/$\overline{W}$ BIT0**
Adresse de base				Bloc	Chip select		Opération
1	0	1	0	0/1	0	0	0/1

FIGURE 35 - OCTET D'ADRESSAGE DE L'EEPROM 24XX1025

Les lectures et les écritures dans l'EEPROM peuvent se faire octet par octet mais il existe également un mode permettant d écrire directement une page de maximum 127 octets sans devoir ré-adresser l'EEPROM a chaque fois. Les mesures étant sur 8 octets (cf. figure 36), nous allons utiliser cette fonctionnalité pour accélérer l'enregistrement.

8 Octets				
mID	Valeur	Fréquence	Timestamp	Vide

FIGURE 36 - STRUCTURE D'UNE MESURE EN EEPROM

L'adressage de la mémoire de stockage de l'EEPROM se fait sur 16 bits car un bloc de l'EEPROM fait 512 Kbits ou 2^{16} bits. Pour aller écrire une mesure, on commence par adresser l'EEPROM en écriture via l'octet d'adressage, une fois l'ACK reçu il faut alors envoyer la partie haute de l'adresse à laquelle on veut aller écrire, recevoir son ACK, ensuite la partie basse et recevoir son ACK. A partir de ce moment la, on peut envoyer les données séquentiellement, octet par octet et signaler la fin de la trame par une condition STOP (un front descendant sur SDA quand SCL est haut). La figure ci-dessous montre la trame nécessaire à l'enregistrement d'une mesure dans l'EEPROM via le bus I²C.

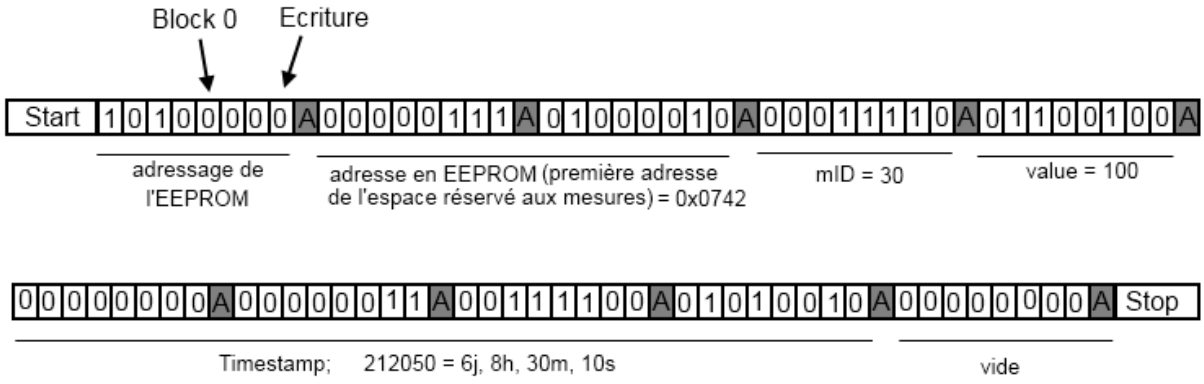

FIGURE 37 - TRAME PERMETANT L'ENREGISTREMENT D'UNE MESURE DANS L'EEPROM 24XX1025

Même si les mesures occupent un espace important au sein de l'EEPROM, tout l'espace disponible ne leur est pas réservé, la figure 38 représente l'allocation des différents espaces de l'EEPROM.

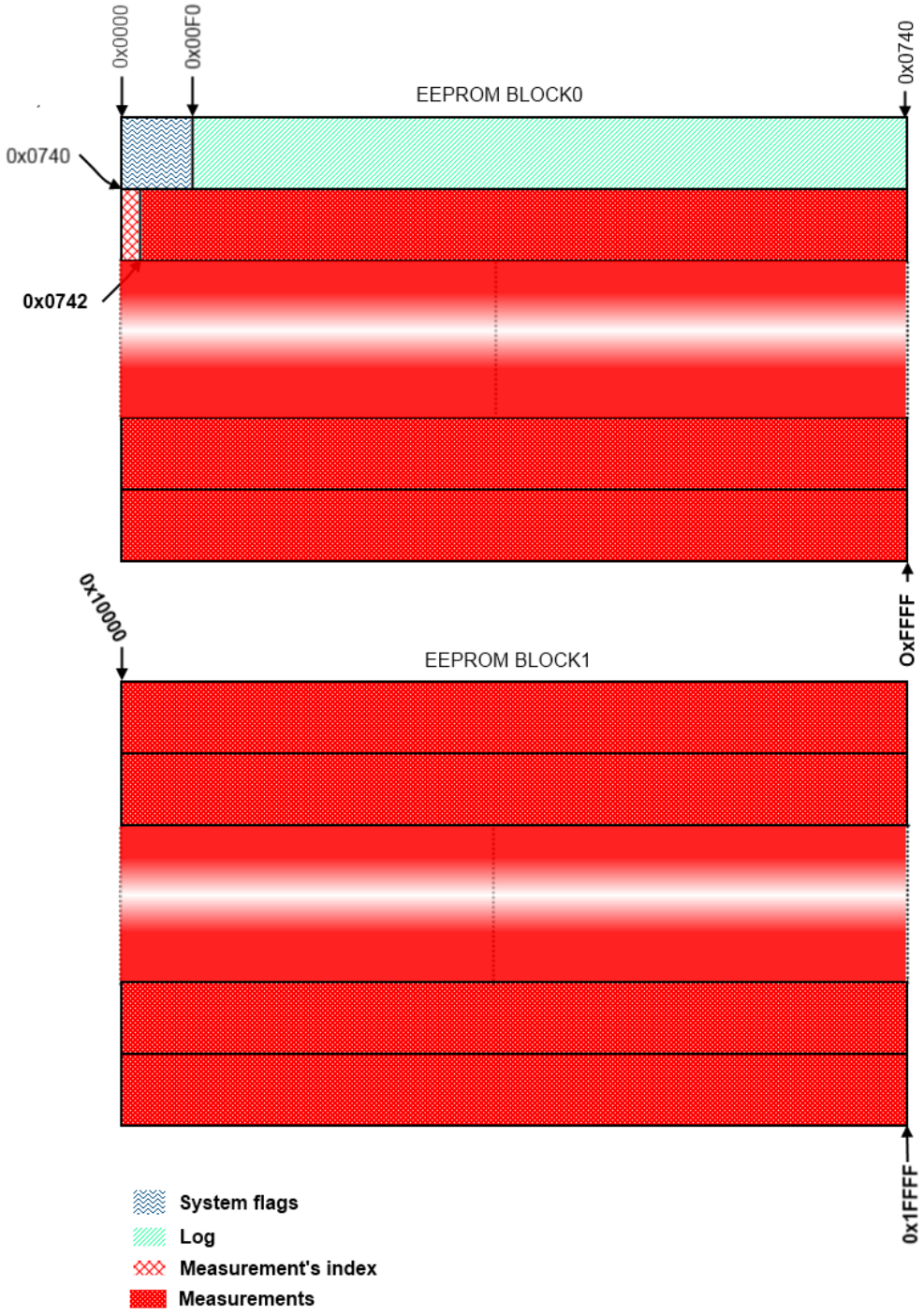

FIGURE 38 - ALLOCATION DE L'EEPROM

L'espace réservé aux mesures va donc de l'adresse 0x0742(bloc 0) à l'adresse 0x1FFFF (bloc 1). Cet espace permet donc d'enregistrer 129213 octets ou 16151 mesures.

L'enregistrement dans l'EEPROM est cyclique, lorsque l'on arrive à la dernière adresse possible (0x1FFF7), le prochain enregistrement se fera dans le premier espace destiné aux mesures.

Le temps nécessaire pour remplir tout l'espace alloué aux mesures, en échantillonnant toutes les mesures (40) à la fréquence maximum (1 Hz) est donc de :

$$40\ mesures/_{sec.} \longrightarrow 16151/_{40} \cong 404\ sec.$$

OU 6 minutes et 44 secondes.

En vue de cette donnée, un problème récurant aux mémoires Flash devoir être étudier : le « memory wearing ». Ce phénomène apparait lorsqu'une mémoire flash arrive en fin de vie prématurément à cause d'une utilisation trop intensive. En effet les mémoires de type flash peuvent endurer un nombre de cycle d'écriture limité, une fois ce nombre atteins, la mémoire se dégrade et certains blocs peuvent devenir illisibles. Microchips™, le fabricant de l'EEPROM donne dans la datasheet un maximum de 1M cycles d'écriture, mais fournis également un logiciel permettant de calculer plus précisément la durée de vie de ses EEPROM pour des applications spécifiques.

Les paramètres choisis pour la simulation constituent le pire scénario :

- Toutes les mesures sont échantillonnées et enregistrées (40 mesures).
- Echantillonnage a fréquence maximum (1 Hz).
- OUFTI-1 en orbite constamment exposé aux rayonnements du soleil (hot case) [*Jaques 08-09*].

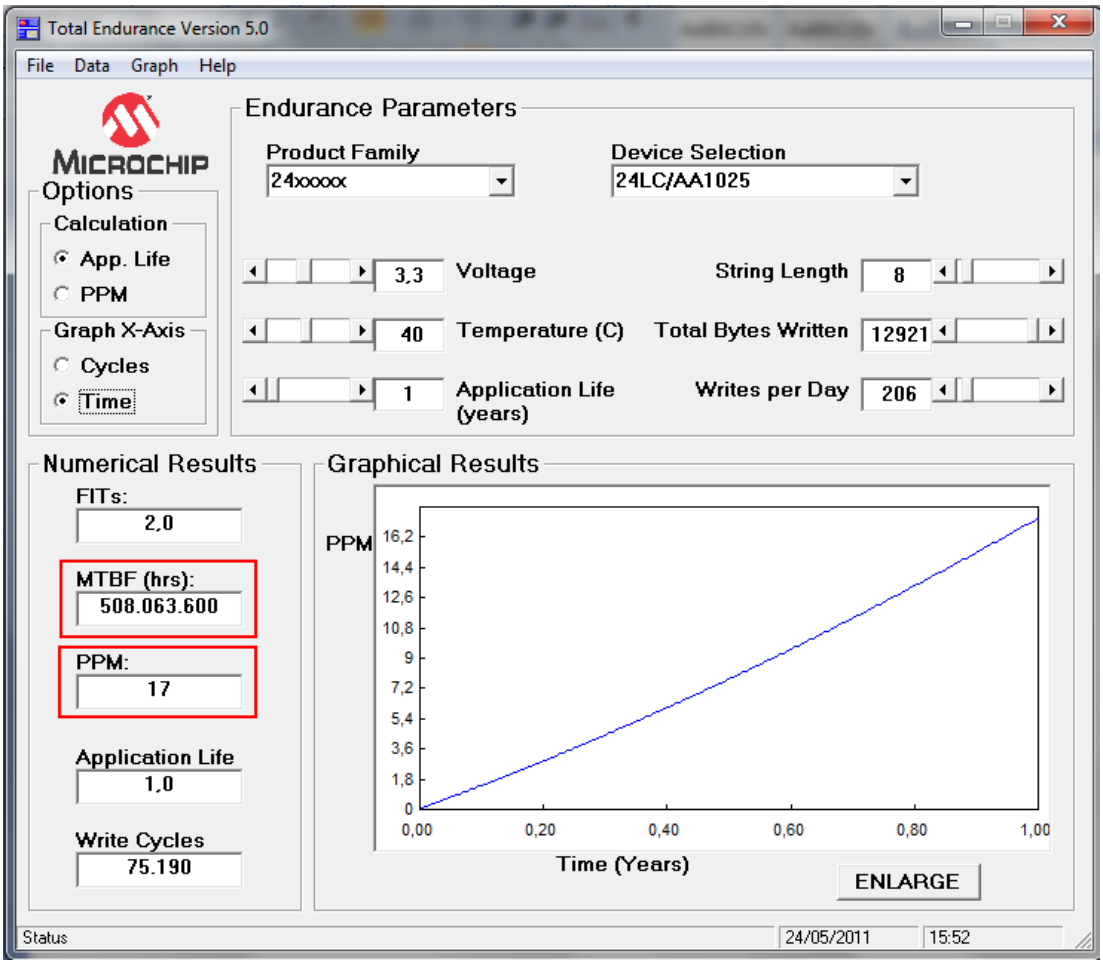

FIGURE 39 - ESTIMATION DE LA DURÉE DE VIE DE L'EEPROM 24XX1025 GRÂCE AU LOGICIEL "TOTAL ENDURENCE V5.0" DE MAXIM

La valeur à observer concernant le résultat de cette simulation :

PPM (failure rate in Part Per Millon): « 17 » après un an, ce qui donne 0,0017% de chances de défaillance au terme de la mission, ce qui est relativement négligeable. Le résultat s'exprime aussi en MTBF (Mean Time Between Failures), c'est le temps moyen entre pannes que l'on peut espérer pour un équipement donné.

c) RAPATRIEMENT DES MESURES

Pour pouvoir relire les mesures précédemment enregistrées dans l'EEPROM on commence par adresser l'EEPROM en écriture afin de lui donner l'adresse du début de la mesure désirée. Une fois l'ACK émis par l'EEPROM reçu, on peut alors lui envoyer les parties hautes et basses de l'adresse de la mesure et envoyer l'octet d'adressage en lecture. L'EEPROM va alors envoyer le premier octet de la mesure, il faut alors émettre un ACK pour recevoir l'octet suivant et répéter l'opération jusqu'à la réception de l'avant-dernier octet. Le dernier octet doit être acquitté avec un NACK pour dire à l'EEPROM d'arrêter l'émission (cf. figure 40).

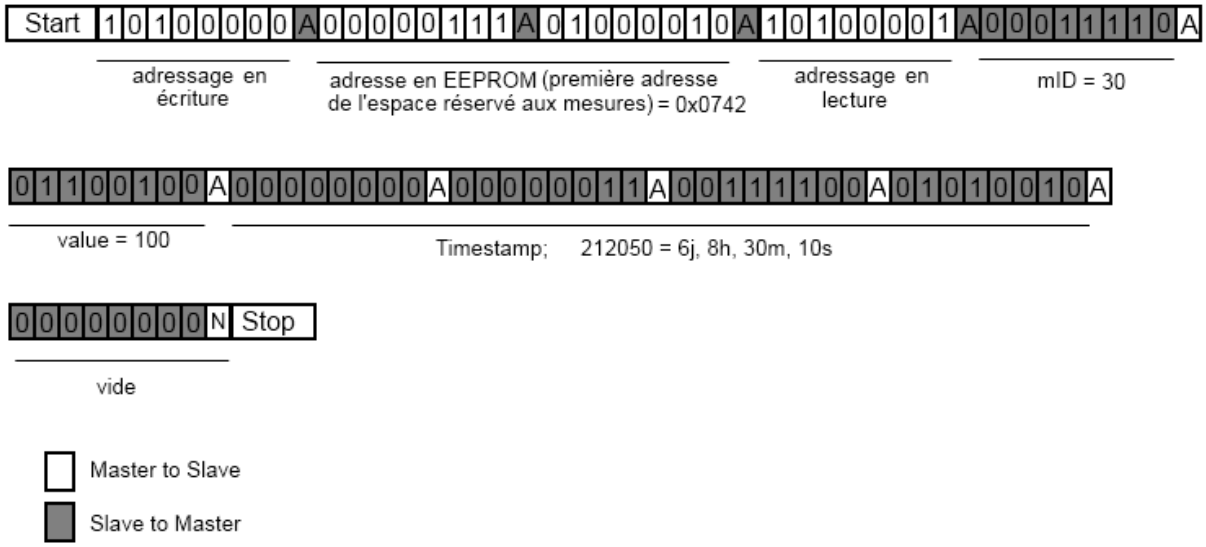

FIGURE 40 - TRAME PERMETANT LA LECTURE D'UNE MESURE DANS L'EEPROM 24XX1025

Deux paramètres entrent en compte concernant le rapatriement des mesures : le type de mesure désiré (mID) et la période à rapatrier. Les mesures étant enregistrées par ordre temporel, on commence par chercher la mesure qui correspond au début de la période à rapatrier. On lit ensuite séquentiellement dans l'EEPROM jusqu'à la dernière mesure de la période en ne prenant que les mesures qui correspondent au type désiré. Pour économiser de la mémoire vive, on écrit directement dans le buffer d'émission du module COM Tx qui aura été préalablement réservé.

5 MODULE MONITOR

La tâche monitor est la tâche la plus prioritaire de l'OBSW et remplis des rôles variés :

- Initialisation du hardware.
- Déploiement des antennes.
- Surveillance des HKP (Housekeeping parameters).
- Reset des sous-systèmes en surconsommation de courant.
- Gestion de la redondance.
- Gestion des modes du satellite (activation / désactivation des S-S et PP).
- Reset du watchdog logiciel.

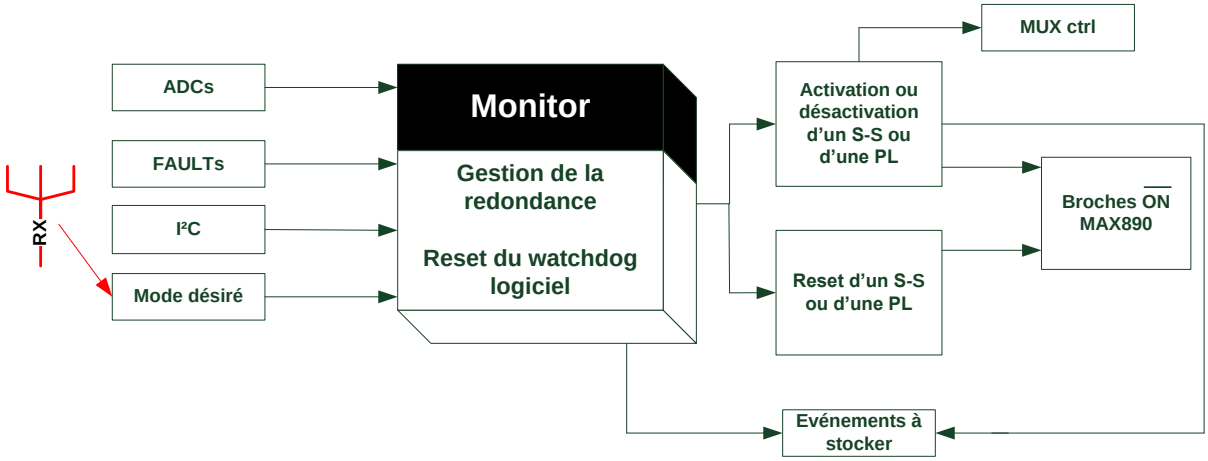

FIGURE 41 - BLOCK DIAGRAM DE LA TÂCHE MONITOR

L'ordinogramme global du module monitor figure ci-dessous, les détails concernant ces routines sont expliqués dans les chapitres suivants.

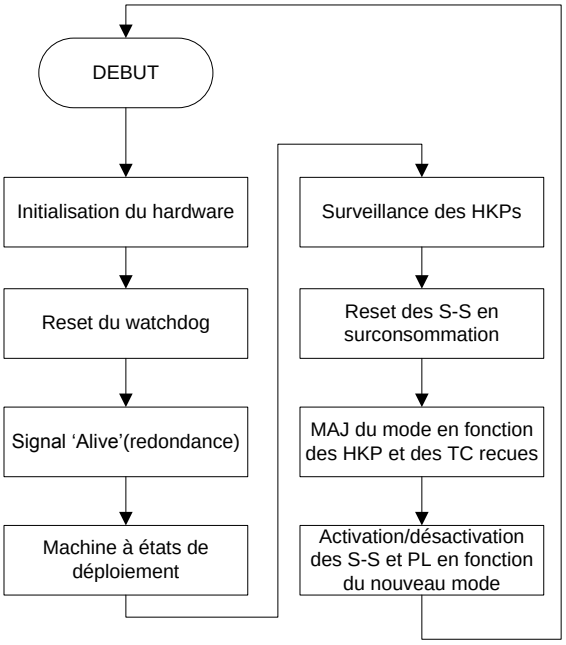

FIGURE 42 - ORDINOGRAME DE LA TÂCHE MONITOR

a) DÉPLOIEMENT DES ANTENNES

Le standard CubeSat exige que les antennes ne puissent êtres déployées que 30 minutes après que le satellite soit éjecté du module P-POD afin d'éviter d'endommager les autres CubeSat et la payload principale du lanceur, un compteur digital est donc nécessaire. Une seconde condition doit être remplie avant de pouvoir commencer à déployer les antennes : la tension délivrée par le bus des batteries doit être suffisante pour activer les couteaux thermiques.

Le signal de déploiement des antennes est donné par l'OBC. L'entièreté du mécanisme de déploiement est redondant, deux signaux vont de l'OBC à l'EPS qui alimente deux couteaux thermiques permettant de couper le fil maintenant les antennes en place.

Description du problème :

La tâche monitor, comme toutes les autres tâches est cyclique, sa périodicité est de 1 Hz. Un des rôles de la tâche est de réinitialiser le watchdog logiciel et d'envoyer le signal « Alive » nécessaire à la redondance (cf. chapitre 4.5 sect. « d »). Ces deux rôles doivent être remplis de façon régulière ou alors l'OBC risque de redémarrer ou le backup OBC risque de prendre la main. On ne peut donc pas interrompre le comportement cyclique de la tâche monitor pour la phase de déploiements des antennes.

Solution

La solution envisagée pour résoudre ce problème est d'utiliser un système de machine à états pour le déploiement des antennes, ce qui permet d'effectuer à chaque tour de boucle du monitor uniquement quelques opérations élémentaires qui ne perturbent pas le reset du watchdog ou l'émission du signal « Alive ».

Avant de rentrer dans sa boucle pour la première fois, le monitor verrouille les autres tâches car elles ne servent à rien pendant les déploiements des antennes. La tâche entre ensuite dans sa boucle, elle réinitialise le watchdog et en envoie le signal « Alive ». La tâche rentre ensuite pour la première fois dans la machine à états de déploiement dont l'ordinogramme est illustré ci-dessous (cf. figure 43). Le processus décrit dans cet ordinogramme va être illustré grâce aux différents scénarios potentiels qu'OUFTI-1 peut subir.

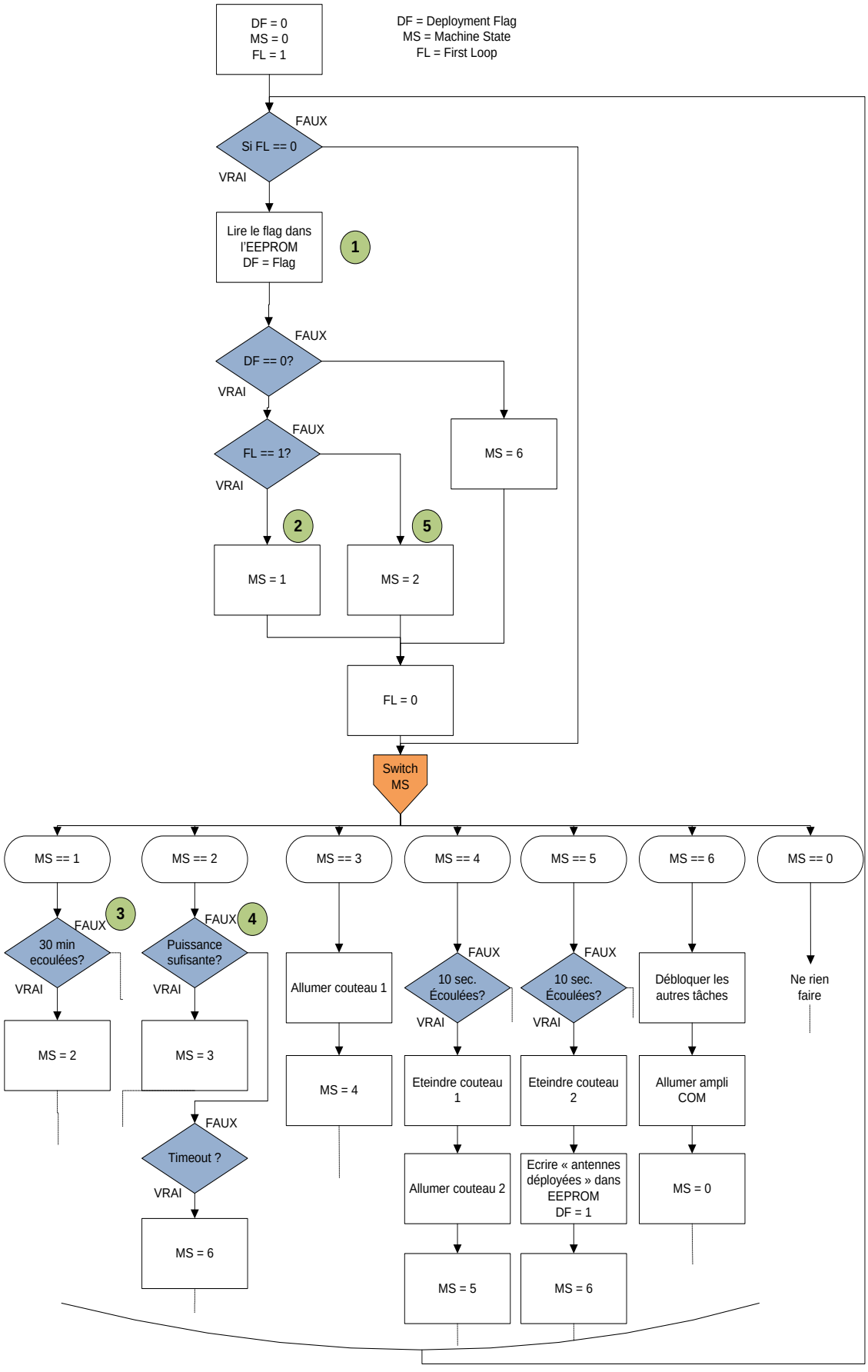

FIGURE 43 - MACHINE À ÉTATS DU DÉPLOIMENT DES ANTENNES

Scénario 1 : premier démarrage du satellite et batteries chargées

Le processus entre dans la machine à états et lit le flag de déploiement dans l'EEPROM(1), celui-ci est à sa valeur initiale (zéro) et c'est aussi la première fois que l'on passe dans la boucle (FL = 1). La procédure de déploiement normale est donc amorcée et la variable de statuts de la machine à états (MS) prend la valeur « 1 ». La tâche monitor va alors boucler normalement en attendant que la référence temporelle du satellite atteigne le 1800ème tick (30 minutes) pour pouvoir commencer à déployer les antennes(3). Il faut ensuite vérifier que la puissance délivrée par les batteries soit suffisante pour allumer les couteaux thermiques(4). Pour se faire, il faut effectuer une lecture sur le canal 8 du MAX1039 de l'EPS (voir annexe B). Pour terminer, on allumer chaque couteau pour dix secondes chacun, on déverrouille les autres tâches du logiciel de bord et on allume l'ampli d'émission de la carte COM.

Scénario 2 : premier démarrage et les batteries sont vides

Tout se déroule exactement comme dans le scénario 1 jusqu'au moment de vérifier si la puissance est suffisante pour dépoiler les antennes(4). À ce moment là, un décompte (timeout) se met en route et dure le temps de plusieurs orbites (TBD) afin de laisser le temps aux batteries de se recharger grâce aux panneaux solaires. Deux scénarios peuvent se présenter :

I. La charge des batteries devient suffisante avant la fin du décompte, le déploiement est alors amorcé.
II. Le décompte arrive à échéance et les batteries ne sont pas suffisamment chargées pour se permettre de déployer les antennes, le déploiement est alors annulé, les autres tâches sont débloquées et l'amplificateur d'émission est allumé. Le flag de déploiement n'a cependant pas été mis à « 1 », le monitor tentera donc de recommencer une phase de déploiement dés le prochain tour de boucle sans attendre 30 minutes(5) avec comme seule différence que le reste du satellite est entièrement fonctionnel puisque toutes les tâches de l'OBSW on été lancées.

Scénario 3 : l'OBC redémarre, les antennes ont été déployées

Le flag en EEPROM étant à « 1 » vu que les antennes ont été déployées ultérieurement, le monitor déverrouille les autres tâches et allume l'amplificateur d'émission de la carte COM. Il peut ensuite reprendre son fonctionnement normal.

Scénario 4 : l'OBC redémarre, les antennes n'étaient pas déployées

Il se produit alors exactement la même chose que lors du scénario 1, comme si il s'agissait du premier démarrage.

Remarque : Le flag en EEPROM qui permet d'enregistrer le fait que les antennes soient déployées ou non est, potentiellement sujet à des erreurs dues aux SEU (voir chapitre 4.2). Apres chaque lecture de ce flag, s'il ne contient pas uniquement des « 1 » ou uniquement des « 0 », un vote est réalisé afin de déterminer sa vraie valeur. Le flag étant sur 16 bits si une égalité devait survenir, le cas « non-déployé » l'emporte.

```
void seu_vote(unsigned int *ant_flag)
{
  //vote to determine the state of the antennas in case of Single Event Upsets (SEU)
  unsigned int trueBitsCount = 0, falseBitsCount = 0, c=1, i=0;
  for(i=0;i<16;i++)
  {
    if(*ant_flag & c)
      trueBitsCount++;
    else
      falseBitsCount++;
    c=c*2;
  }
  if(trueBitsCount > falseBitsCount)
    *ant_flag = 0xFFFF;
  else
    *ant_flag = 0x0000;
}
```

FIGURE 44 - CODE PERMETANT LA RÉSOLUTION DES SEU PAR VOTE

b) SURCONSOMMATION D'UN SOUS-SYSTÈME

Surconsommation d'un sous-système

Chaque sous-système d'OUFTI-1 possède une protection qui sert à limiter le courant entre la sortie du bus et le sous-système. Le composant qui rempli ce rôle est le MAX890L de chez MAXIM™.

Description du brochage du composant (cf. figure 45) :

IN (1/2) : entrée du bus d'alimentation.

$\overline{ON}$: un niveau bas indique que le composant doit laisser passer le courant.

GND : masse

$\overline{FAULT}$: cette broche passe à l'état bas quand la limitation de courant est active.

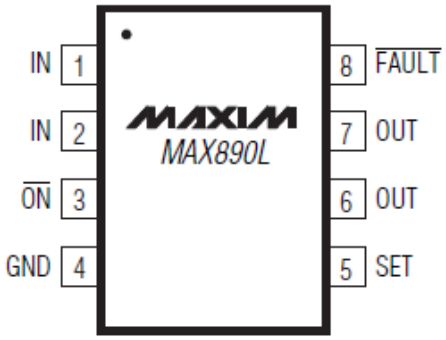

FIGURE 45 - BROCHAGE DU COMPOSANT MAX890

SET : sert à dimensionner le niveau de protection.

Quand le MAX890L détecte une surconsommation sur sa sortie, il l'indique via sa broche $\overline{FAULT}$ et coupe puis rallume sa sortie. Il a néanmoins été décidé que pour plus de sécurité il faudrait aussi couper le MAX890L via broche $\overline{ON}$ pendant un certain temps avant de le rallumer.

Les broches $\overline{FAULT}$ des protections des sous-systèmes ont été connectées au PORT1 du MSP430 de l'OBC via le bus PC/104. Le PORT1 du MSP430 est interruptible et permet donc de réagir rapidement en cas de surconsommation de courant. La routine d'interruption pour les broches $\overline{FAULT}$ des MAX890L se contente de couper le sous-système fautif et de setter un flag afin de déléguer le reste du travail à la tâche monitor.

L'ordinogramme de la figure 46 illustre le fonctionnement de la routine d'interruption et de la portion de la tâche monitor qui reset les sous-systèmes fautifs.

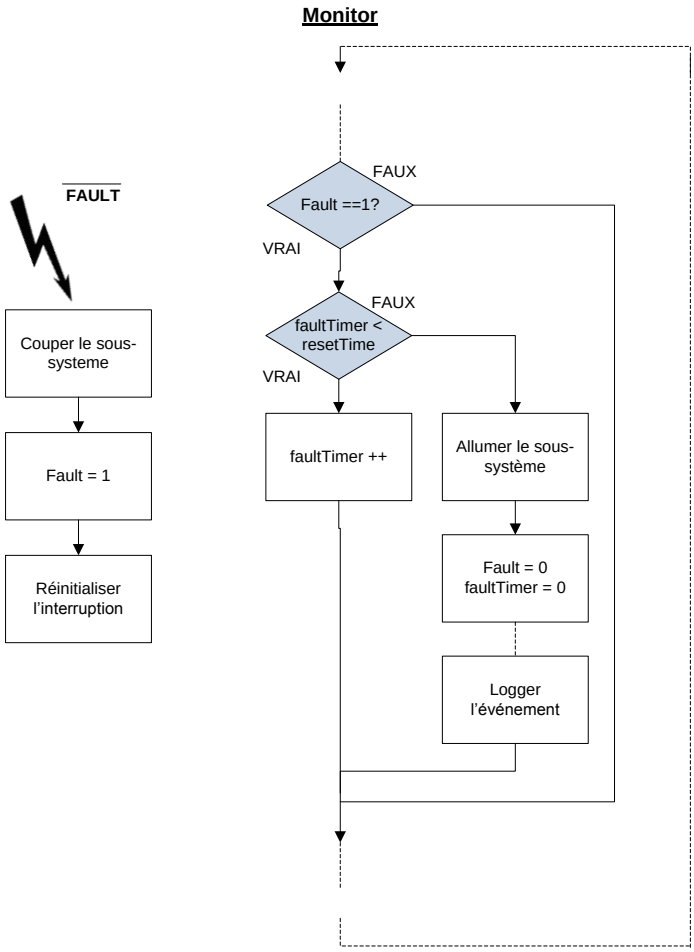

FIGURE 46 - FONCTIONNEMENT DE LA ROUTINE D'INTERRUPTION ET DE LA PORTION DE LA TÂCHE MONITOR QUI RESET LES SOUS-SYSTÈMES FAUTIFS

c) LES MODES D'OUFTI-1

Un mode représente une configuration des sous-systèmes et payloads du satellite qui peut survenir pendant sa durée de vie. Les modes d'OUFTI-1 ont été choisis lors de la phase d'analyse de mission.

Le tableau qui suit représente la configuration pour chaque mode d'OUFTI-1(cf. figure 47).

	EPS	BCN	OBC	RX AX-25	TX AX-25	D-STAR	X-EPS
OFF							
SAFE							
DEFAULT							
SILENCE							
D-STAR							
XEPS							
FUNMODE							

FIGURE 47 - LES MODES D'OUFTI-1 ET LEUR CONFIGURATION MATERIELLE

Description des modes :

OFF

C'est le mode initial du satellite quand il est toujours dans le module P-POD, c'est aussi le mode dans lequel il entre quand les batteries sont complètement vides.

- **Transitions:**
  De OFF à SAFE, lorsque OUFTI-1 est éjecté du module P-POD ou lorsque les convertisseurs de l'EPS reçoivent suffisamment de courant pour alimenter l'OBC.
- **États:**
  OFF: EPS, BCN, OBC, RX AX.25, TX AX.25, D-STAR, xEPS.

SAFE

Ce mode est le mode de sécurité du satellite, il permet de diagnostiquer les potentiels problèmes au sein d'OUFTI-1. On reste en mode SAFE tant que le problème n'a pas été résolu. En mode SAFE, seules les fonctions critiques sont activées et les payloads (D-STAR & xEPS) sont coupées. C'est aussi le mode dans lequel le satellite démarre afin de s'assurer que tout fonctionne correctement avant de passer en mode DEFAULT.

- **Transitions :**
  De SAFE à OFF ;
  De SAFE à DEFAULT, au premier démarrage et lorsqu'un problème a été corrigé.

- **États :**
  ON: EPS, BCN, OBC, RX AX.25.
  OFF: TX AX.25, D-STAR, xEPS.

DEFAULT

C'est le mode de fonctionnement normal d'OUFTI-1. Dans ce mode les payloads xEPS et D-STAR sont désactivées, le satellite ne fait qu'exécuter les télécommandes, prendre des mesures, générer des télémétries, etc. Au niveau de la communication, l'AX.25 est activé en émission et réception ainsi que la BEACON.

- **Transitions :**
  De DEFAULT à OFF ;
  De DEFAULT à SAFE ;
  De DEFAULT à D-STAR ;
  De DEFAULT à XEPS ;
  De DEFAULT à FUN .
- **États :**
  ON : EPS, BCN, OBC, RX AX.25, TX AX.25.
  OFF: D-STAR, xEPS.

SILENCE

Le mode SILENCE est un mode dans lequel toute émission radio est impossible. Tout satellite doit être capable de couper totalement ses émissions radio dans l'éventualité ou il perturberait des communications prioritaires. L'émission AX.25, le relais D-STAR et même la balise morse de secours (BCN) sont coupées.

- **Transitions :**
  De SILENCE à OFF ;
  De SILENCE à SAFE ;
  De SILENCE à DEFAULT.

- **États :**
  ON : EPS, OBC, RX AX.25.
  OFF: xEPS, TX AX.25, BCN, D-STAR.

D-STAR

Dans ce mode, OUFTI-1 fait office de relais D-STAR, tout fonctionne normalement à l'exception de la transmission AX.25 qui ne peut pas être active en même temps que le D-STAR.

- **Transitions :**
  De D-STAR à OFF ;
  De D-STAR à SAFE ;
  De D-STAR à DEFAULT.
- **États :**
  ON : EPS, BCN, OBC, RX AX.25, D-STAR.
  OFF: xEPS, TX AX.25.

xEPS

Ce mode sert à tester l'alimentation numérique expérimentale, toutes les fonctions classiques sont actives en plus de l'xEPS. Dans ce mode, il est possible de récupérer les données concernant le fonctionnement de l'xEPS.

- **Transitions :**
  De xEPS à OFF ;
  De xEPS à SAFE ;
  De xEPS à DEFAULT.
- **États:**
  ON: EPS, BCN, OBC, RX AX.25, TX AX.25.
  OFF: D-STAR.

FUNMODE

Le mode fun est le mode où le plus de fonctions sont actives, toutes les payloads sont en marche. La seule fonction désactivée est la transmission AX.25 étant donné que le D-STAR est ON.

- **Transitions :**
  De FUNMODE à OFF ;
  De FUNMODE à SAFE ;
  De FUNMODE à DEFAULT.
- **États:**
  ON: EPS, BCN, OBC, RX AX.25, D-STAR, xEPS.
  OFF: TX AX.25.

La figure 48 montre les transitions possibles entre les différents modes d'OUFTI-1.

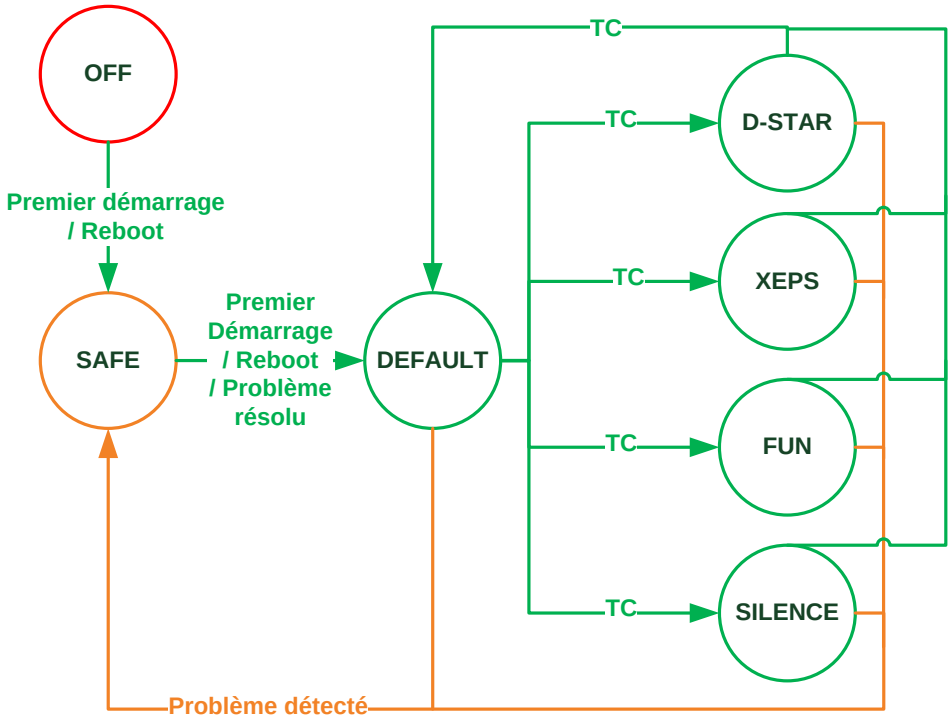

FIGURE 48 - SCHÉMA RÉCAPITULATIF DES TRANSITIONS DE MODES POSSIBLES

Activation et Désactivation des différents sous-systèmes

La transition entre deux modes requiert d'activer ou de désactiver certains sous-systèmes, les méthodes pour activer ou désactiver chaque sous-système sont listées ci-dessous.

- **EPS**

  L'EPS se met en route automatiquement lorsque la tension délivrée par les batteries est suffisante pour alimenter les convertisseurs (3.3V, 5.0V et 7.2V). Il est désactivé automatiquement quand la tension des batteries est trop faible.

- **OBC**

  Il n'est jamais désactivé, aucune transition d'un mode à l'autre ne désactive l'OBC à l'exception du passage en mode OFF qui n'est pas intentionnel. Aucune implémentation logicielle n'est donc nécessaire pour l'OBC.

- **BCN**

  Activation : La balise morse émet en continu, il n'y a aucune action particulière à entreprendre pour l'activer si ce n'est de s'assurer que l'amplificateur de puissance d'émission de la carte COM est activé.

  Désactivation **:** Il n'y a que dans le mode SILENCE que la balise morse est désactivée, dans ce mode aucune autre émission radio n'est autorisée. Afin de désactiver la balise morse, on peut donc couper l'amplificateur de puissance de la carte COM via la broche $\overline{\text{ON}}$ de son MAX890. Cela empêche par la même occasion l'émission de trames AX.25 et D-STAR.

- **RX AX.25**

  La réception AX.25 est toujours active, sans elle le satellite ne peut plus recevoir aucun ordre.

- **D-STAR**

  Activation : Un multiplexeur (MUX) est utilisé sur la carte COM pour effectuer le routage de signaux soit entre le MSP COM et le système d'emission (un ADF7021[3]) pour le D-STAR, soit entre le MSP OBC et le système d'emission pour l'AX.25 (illustré sur la figure 49).

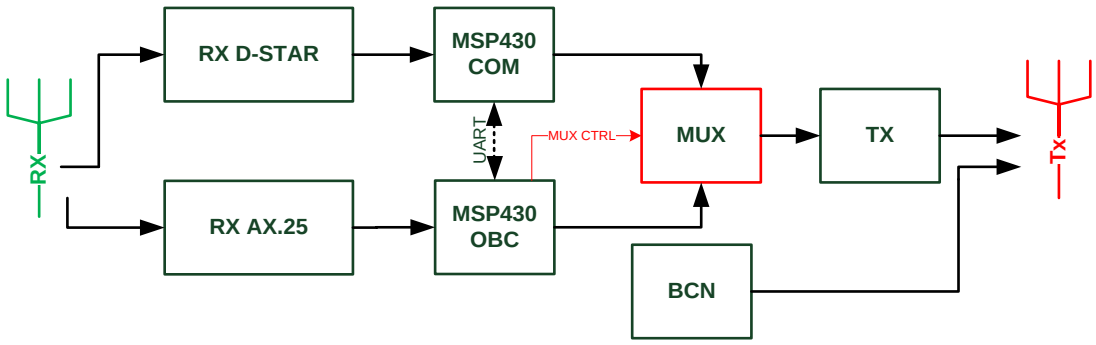

FIGURE 49 - BLOCK DIAGRAM DES FLUX DE DONNÉES ENTRE L'OBC ET LA COM

Pour que le multiplexeur effectue le routage entre le MSP COM et la transmission, il faut passer son entrée « CTRL » à « 1 ».

La deuxième étape pour activer le D-STAR est de prévenir la carte COM qu'elle doit activer le relais D-STAR, pour cela il faut envoyer un message sur l'UART OBC-COM en utilisant le même protocole que celui définit dans le chapitre 4.4, sect. « a ». Le TOC (Type Of Communication) à envoyer sur l'UART est « E ».

La dernière étape consiste à suspendre la tâche COM Rx dans le scheduler. Cette tâche a comme rôle d'encoder les trames AX.25 [1].Étant donné qu'il est impossible d'envoyer des télémétries AX.25 en même temps que du D-STAR, cette tâche est inutile.

Désactivation : voir « Activation TX AX.25 ».

- **TX AX.25**

  Activation : Si l'on vient du mode D-STAR, il faut rendre la main au MSP OBC sur système d'émission en passant l'entrée « CTRL » du multiplexeur à « 0 » (voir figure 49). Ensuite, il s'agit de demander à la carte COM de couper le relais D-STAR en envoyant la commande « DISABLE D-STAR » sur l'UART OBC-COM (TOC :

« D »). Il faut finalement modifier le statut de la tâche COM TX en « READY » dans le scheduler, afin que celle-ci puisse recommencer à encoder des trames AX.25.
Désactivation : voir « Activation D-STAR ».

- **xEPS**

  Activation : La condition préalable à l'activation de l'xEPS est que la tension des batteries soit supérieure à 4.0V, cette mesure faisant partie des HKP échantillonnées par le monitor, il suffit donc de lire cette valeur en RAM. La seconde étape consiste à sélectionner la charge résistive, cette charge va permettre de contrôler le bon fonctionnement de l'xEPS avant de lui faire alimenter le bus 3.3V. On peut finalement activer l'xEPS via la broche $\overline{\text{ON}}$ de son MAX890.
  La mise en marche de l'xEPS a pour conséquence le monitoring d'une nouvelle HKP, il faut surveiller la tension de sortie de l'xEPS afin de savoir si l'on peut passer de la charge résistive au bus 3.3V du satellite. Les modes d'opérations de l'xEPS sont détaillés dans le mémoire de Ledent P. [*6*].

  Désactivation : Pour désactiver l'xEPS, il faut couper l'alimentation de la carte via la broche $\overline{\text{ON}}$ de sa protection en courant MAX890. L'xEPS peut être coupé via télécommande mais il serra automatiquement coupé si la tension des batteries chute en dessous de 3.6V. Il faut également placer sa sortie sur la charge résistive afin d'éviter d'alimenter le bus 3.3V lors de la remise en marche de l'xEPS.

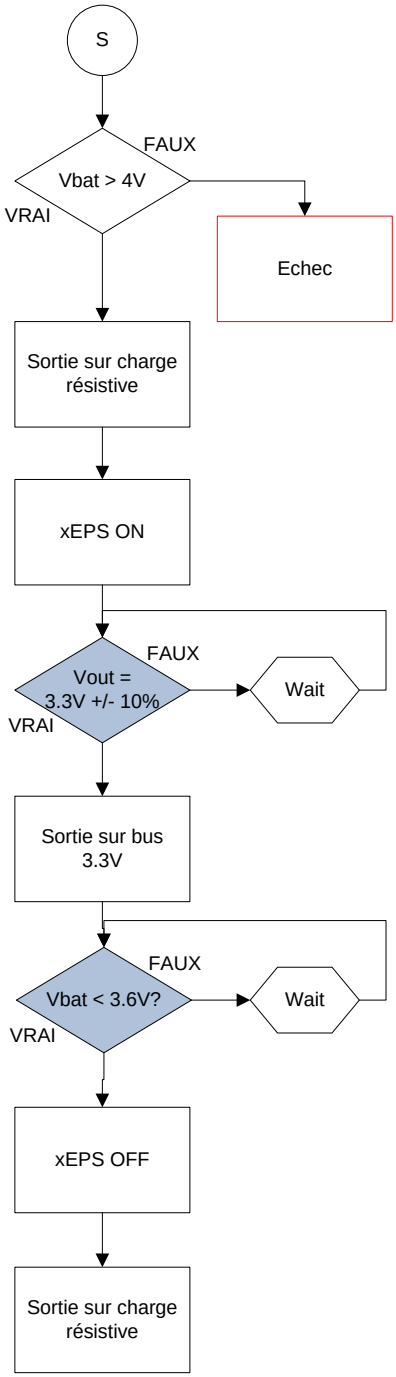

FIGURE 50 - ORDINOGRAMME REPRÉSENTANT LE CYCLE DE FONCTIONNEMENT DE L'EPS. SOURCE : [LEDENT 09]

d) REDONDANCE

Le principe de fonctionnement de la redondance des OBC au sein d'OUFT-1 à été réalisé par K.Can dans le cadre de son mémoire. Ce chapitre résume ce fonctionnement et en propose une correction. Pour plus de détails et des scénarios de pannes alternatifs, se référer a son mémoire [4].

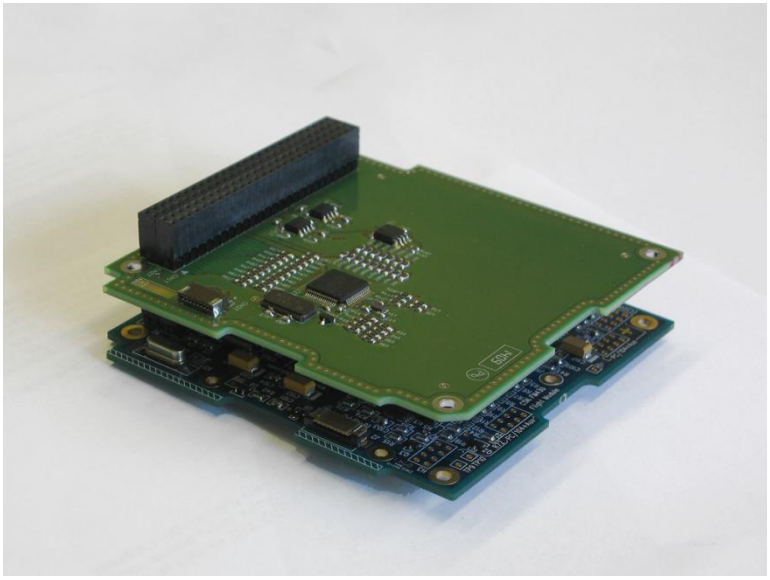

FIGURE 51 - INTERCONNECTION DES DEUX OBC VIA LE BUS PC/104

Principe de fonctionnement:

Chaque broche des deux OBC (Default et Backup) est connectée à la broche similaire sur l'autre OBC. Pour éviter les courts-circuits dans le cas ou les deux OBC auraient la même broche configurée en sortie, des résistances ont été placées en série à chaque broche. Initialement, Le Default fonctionne normalement alors que le Backup est en veille et surveille le bon fonctionnement du Default en réceptionnant un signal appelé « Alive 1». Ce signal est régulièrement envoyé par la tâche monitor du Default sur le bus I^2C, tant que ce signal est réceptionné par le Backup, pas de problème. Si le Backup ne reçoit plus le signal, il sort alors de veille et prend la main.

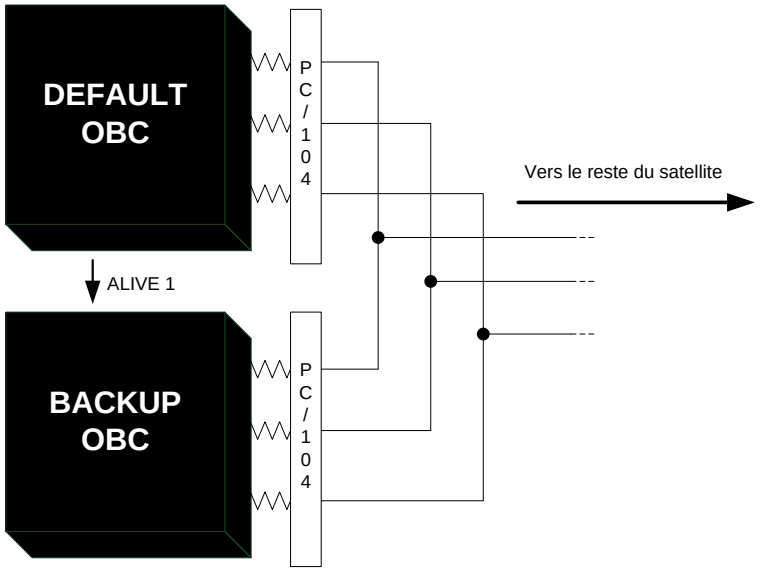

FIGURE 52 - PROTECTION DES OBCS VIA RÉSISTANCES ET SIGNAL ALIVE1

Pour pouvoir piloter le bus I²C, le µC qui a la main doit être maître du bus. Le Default démarrera donc en maître afin de pouvoir utiliser les différents périphériques esclaves du bus (EEPROM, ADCs), le Backup quand à lui démarre en mode esclave afin de ne pas interférer avec le Default et de pouvoir recevoir le signal « ALIVE 1 ».

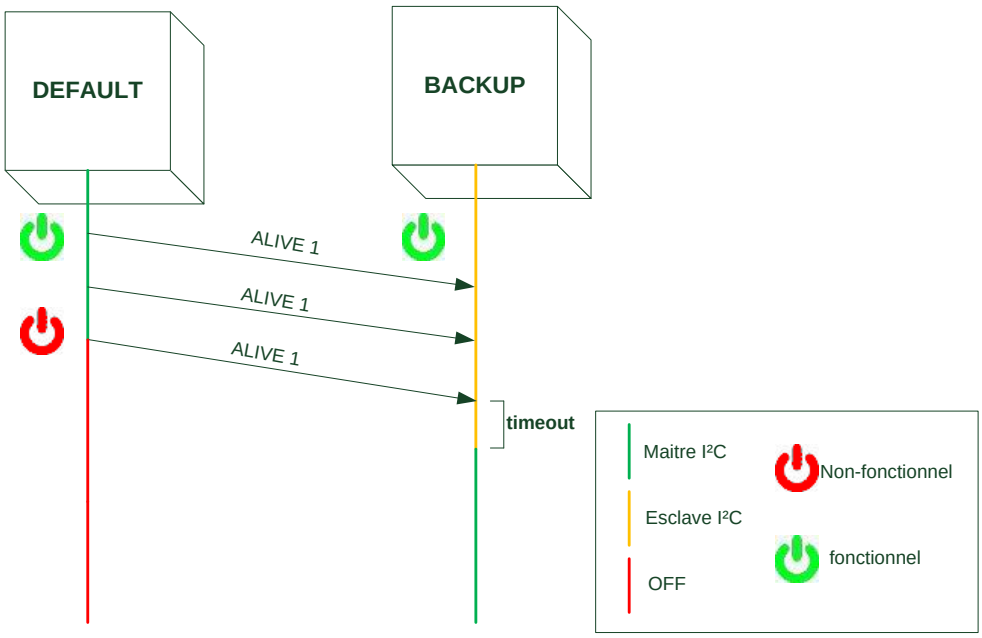

FIGURE 53 - REDONDANCE SIMPLE

Il faudrait que le Default, conçu par les étudiants du projet soit celui qui fonctionne le plus souvent possible. Pour cela il est nécessaire que si après avoir perdu la main, le Default venait à redémarrer correctement, il reprenne le contrôle du satellite. On ne peut malheureusement pas utiliser le signal « ALIVE 1 » car si le Default redémarre, c'est le Backup qui configuré en tant que maître I²C et le Default n'aura pas le droit d'émettre sur le bus.

Pour se faire, il faut demander au Backup d'émettre lui aussi un signal sur le bus I²C mais uniquement lorsqu'il a la main et qu'il est donc maître du bus I²C. Ce signal, que nous appellerons « ALIVE 2 » permet de savoir si le Default a redémarré correctement. Si après réception du signal « ALIVE 2 » le Default répond par un ACK, cela veux dire qu'il aura redémarré correctement, le Backup peut alors se mettre en veille et rendre la main.

Correction du principe original : il faut aussi pour pouvoir mettre ce procédé en place que le Default démarre en esclave du bus ou alors il sera incapable de recevoir le signal « ALIVE 2 » étant donné que même si deux maîtres peuvent cohabiter, ils sont incapables de communiquer entre eux. Les deux OBC démarreront donc en esclave et un décompte devra arriver à terme avant de passer en maître du bus. Ce décompte doit être deux fois plus long pour le Backup afin que lors de la mise sous tension des cartes, le Default prenne la main en premier et empêche le Backup de se configurer en maître du bus grâce au signal « ALIVE 1 ».

Ce scénario est illustré sur la figure 45.

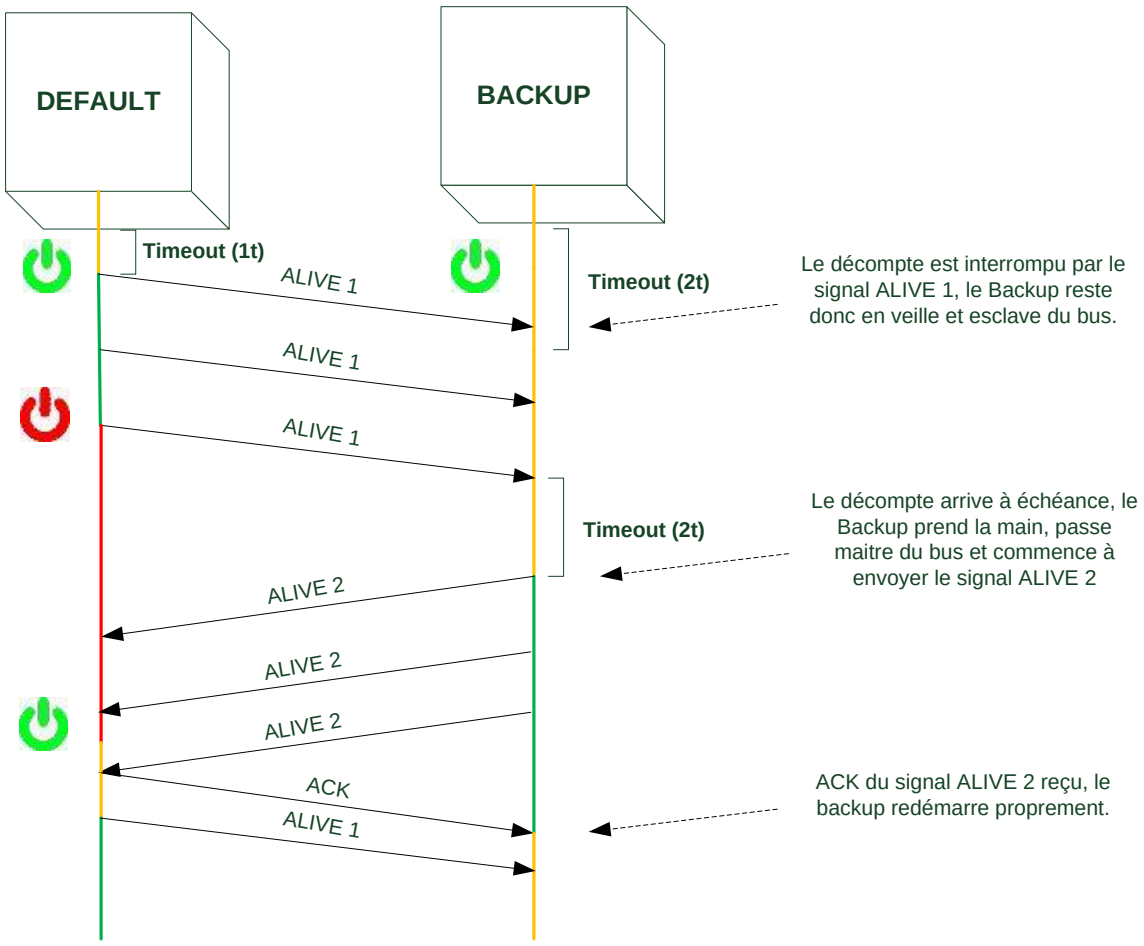

FIGURE 54 - PANNE DU DEFAULT OBC SUIVIE D'UN REDÉMARAGE DE CELUI-CI

Implémentation :

Le canal de communication pour la redondance entre le Default et le backup est donc le bus I²C. Pour passer en maître le MSP430 doit juste configurer sont module USART en mode I²C et configurer quelques paramètres (baud rate, clock pour SCL, ect.). Pour passer en esclave il faut en plus choisir une adresse sur 7 bits et la placer dans le registre I2COA (I²C Own Address).

Le TIMER_B du MSP430 qui est utilisé pour la référence temporelle du satellite est utilisé pour faire le décompte qui une fois arrivé à échéance, fait prendre la main à l'OBC.

L'ordinogramme ci-dessous décrit l'implémentation de la redondance dans les deux OBC.

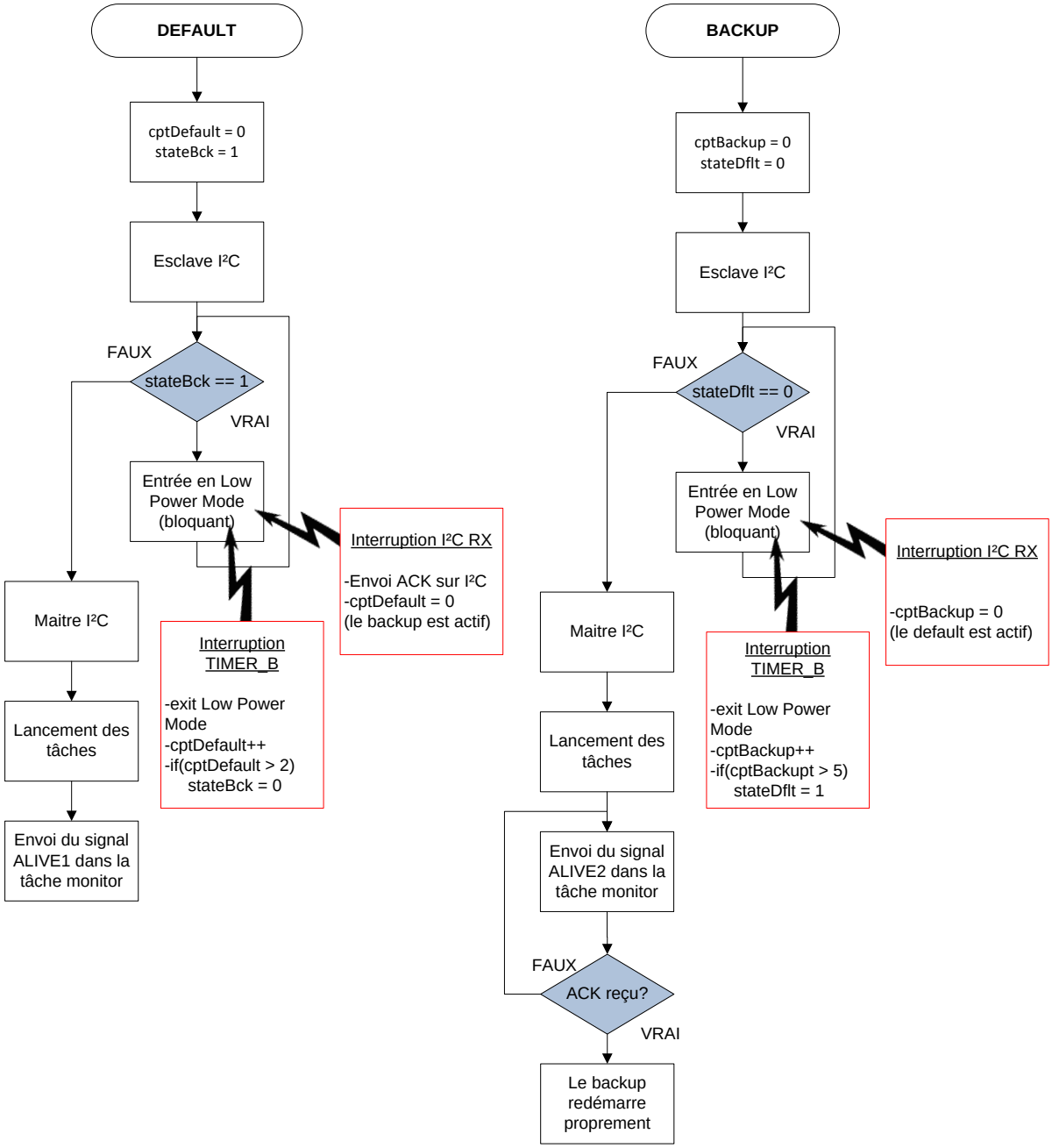

FIGURE 55 - IMPLEMENTATION DE LA REDONDANCE

6 MODULE LOG

Le module log sert à maintenir un journal de bord des événements important qui se produisent au sein du satellite.

Par exemple :

- Démarrage de l'ordinateur de bord.
- Changement de mode de fonctionnement d'un sous-système / d'une payload.
- Surconsommation d'un sous-système.
- Ect.

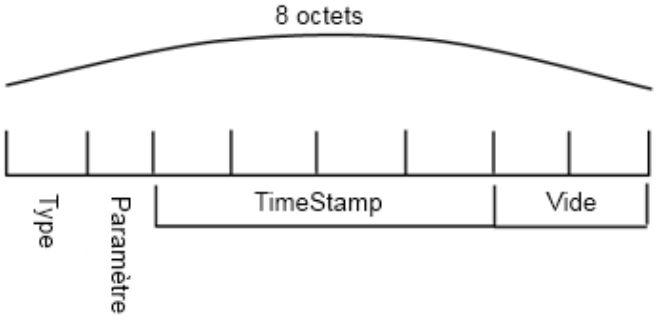

FIGURE 56 - STRUCTURE D'UN ÉVÉNEMENT EN RAM / EEPROM

Les événements sont donc codés sur 8 octets et non sur 6 car il s'est avéré que l'enregistrement de structures de données sur 6 octets en EEPROM produit des inconsistances lors du rapatriement de ces données. Le type d'un événement est codé sur 1 octet, permettant 256 événements différents soit ce qui laisse une bonne marge. Un événement particulier peut posséder un paramètre codé lui aussi sur un octet. Le timestamp est quant à lui codé sur 32 bits et possède une résolution de 1 seconde, ce qui lui permet d'être valide pendant 140 ans, cela qui peut paraitre excessif mais un codage sur 16 bits donne un maximum de 18 heures de validité. La liste complète des événements et la description de leurs paramètres est disponible dans l'annexe A.

a) AJOUT D'UNE ÉVÉNEMENT

Pour ajouter un événement au journal de bord, la tâche qui détecte cet événement doit aller l'écrire dans un buffer FIFO circulaire. C'est la tâche log qui se chargera de l'écriture en EEPROM et de l'entretien du journal. Le fait de passer par une zone mémoire tampon permet de soulager la tâche concernée de l'écriture dans l'EEPROM. L'emplacement dans le buffer ou aller écrire le prochain événement est indiqué par le pointeur « pTail », à chaque ajout ce pointeur avance d'un emplacement et reviens au début du buffer lorsqu'il en atteint la fin. Un compteur « nbElem » est également incrémenté à chaque ajout dans le buffer. L'ajout d'événements étant relativement sporadique, un buffer de cinq éléments est suffisant.

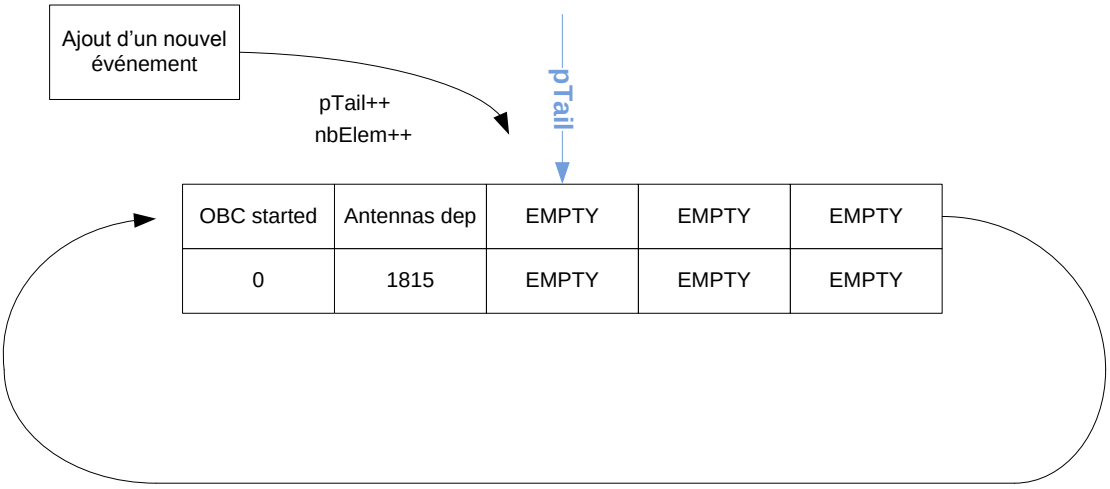

FIGURE 57 - AJOUT D'UN ÉVENEMENT DANS LE TAMPON DU LOG

b) LA TÂCHE LOG

La tâche log va donc « scanner » le buffer grâce à un second pointeur appelé « pHead » qui avancera d'un élément pour chaque événement correctement enregistré en EEPROM. Tout comme le pointeur d'ajout « pTail », le pointeur de retrait « pHead » retourne au début du vecteur lorsqu'il arrive au bout de celui-ci. Chaque seconde (fréquence choisie), la tâche log regarde s'il y a des événements à sauvegarder grâce a « nbElem », enregistre l'événement, incrémente « pHead » et diminue « nbElem » d'une unité. Si au contraire le buffer FIFO est vide, la tâche profite du temps qui lui est alloué pour faire une sauvegarde de l'index

d'enregistrement qui contient l'adresse du prochain événement en EEPROM. Un exemple du fonctionnement global est illustré sur la figure 58 ci-dessous.

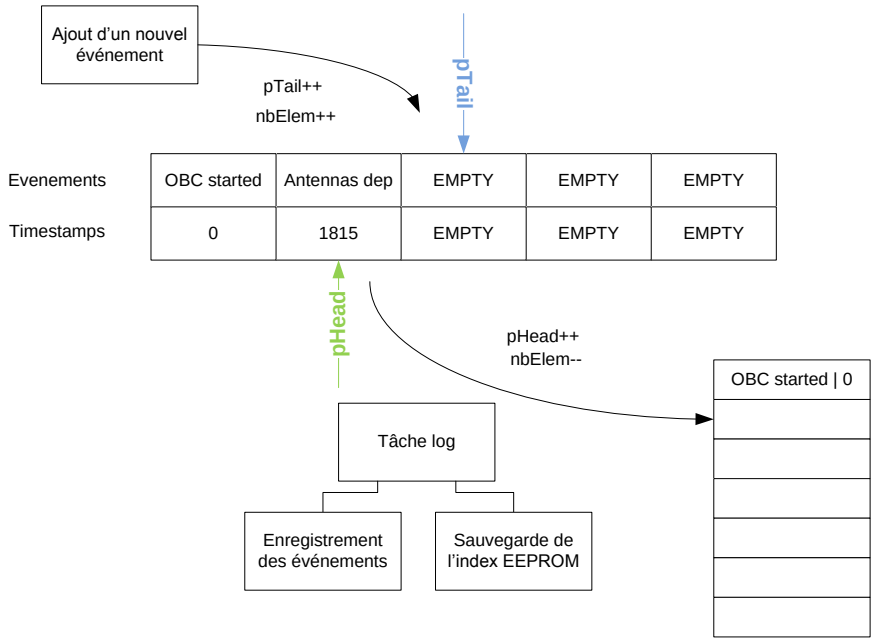

FIGURE 58 - ENREGISTREMENT DES ÉVENEMENT EN EEPROM PAR LA TÂCHE LOG

Allocation EEPROM

Les enregistrements du log sont situés au début du bloc0 de l'EEPROM, juste après les flags systèmes. La zone située à l'adresse 0x00F0 est utilisée pour régulièrement faire un backup de l'index sur lequel le prochain enregistrement serra sauvé, cela s'avère utile au cas l'OBC redémarre afin de reprendre la journalisation là ou elle en était.

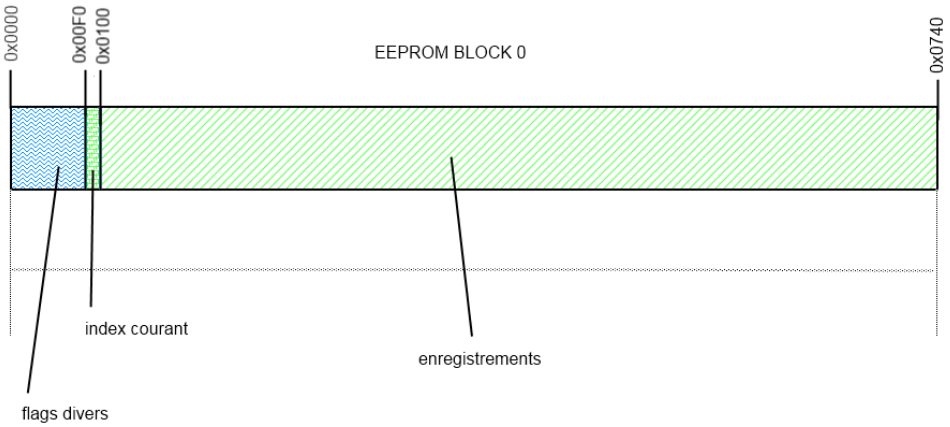

FIGURE 59 - ALLOCATION DU JOURNAL DE BORD EN EEPROM

Un événement étant codé sur 8 octets, les 1600 octets alloués en EEPROM permettent d'enregistrer 200 événements, ce qui est plus que suffisant étant donné la faible fréquence de leurs occurrences. Une fois arrivée a la fin de la zone allouée, l'enregistrement reprendra au début.

Pour plus de détails concernant la façon d'enregistrer des données en EEPROM, consultez le chapitre 4.4 section « b » à propos de l'enregistrement des mesures.

c) RAPATRIEMENT DU LOG

Le rapatriement du journal de bord peut être demandé sous deux formes :

- Soit on demande le rapatriement complet, il suffit alors de balayer tout l'espace réservé au log en EEPROM.
- Soit on demande de rapatrier un période en particulier, il faut alors chercher dans l'EEPROM l'événement dont le timestamp est le premier de cette période et rapatrier tant que le timespamp lu est valide.

Le données à rapatrier sont mises en forme et écrites dans un buffer d'émission qui est fournit par le module COM Tx [1]. Le buffer doit être préalablement réservé afin de ne pas interférer avec l'émission de données provenant d'autres tâches de l'OBSW. Un fois le buffer d'émission remplis de toutes les données à émettre, on le signale à la tâche COM Tx qui peut alors commencer l'encodage de la trame.

7 TESTS

Pour se rapprocher le plus possible de la configuration de vol du satellite, les tests on été réalisés avec un maximum d'interconnexions avec d'autres sous-systèmes du satellite, pour remplacer les éléments indisponibles, des boutons-poussoirs et des diodes lumineuse on été utilisées. La photo ci-dessous montre la configuration matérielle de test.

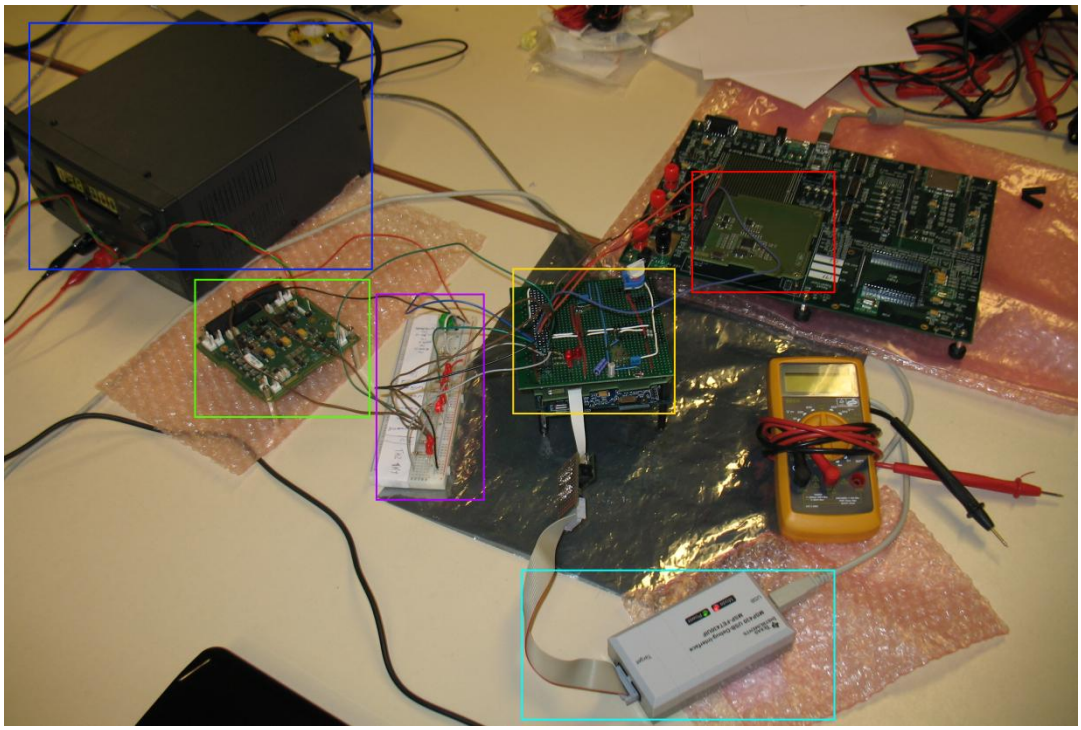

FIGURE 60 - CONFIGURATION MATERIELLE DE TESTS

Légende :

➲ Alimentation de laboratoire, simule les batteries.

➲ Carte EPS.

➲ Breadboard avec boutons-poussoirs permettant de simuler le signal FAULT des MAX890 et diodes lumineuses indiquant l'état de la broche $\overline{ON}$ les MAX890 indiquant l'état de marche des S-S.

➲Trois cartes sont interconnectées, de bas en haut : Backup OBC, Default OBC, carte de test.

➲Carte du type « Default OBC » elle est configurée pour simuler le comportement de la carte COM du satellite. Elle permet de relayer les télécommandes entre le PC de test et l'OBC et de transmettre les télémétries émise par l'OBC au PC de test.

➲Programmateur / débugger JTAG

Les tests on été effectues grâce à une application JAVA développée pour envoyer des télécommandes en AX.25 à l'OBC simulant la carte COM et recevoir et d'afficher sous une forme compréhensible les télémétries envoyées par le satellite.

Cette méthode à permis de tester les modules des satellites dans des conditions les plus proches possibles du modèle final. Toutes les fonctions des modules ont pu être testées : changement de mode, configuration des mesures, rapatriement des mesures, rapatriement du log (figure 61), redondance, etc.

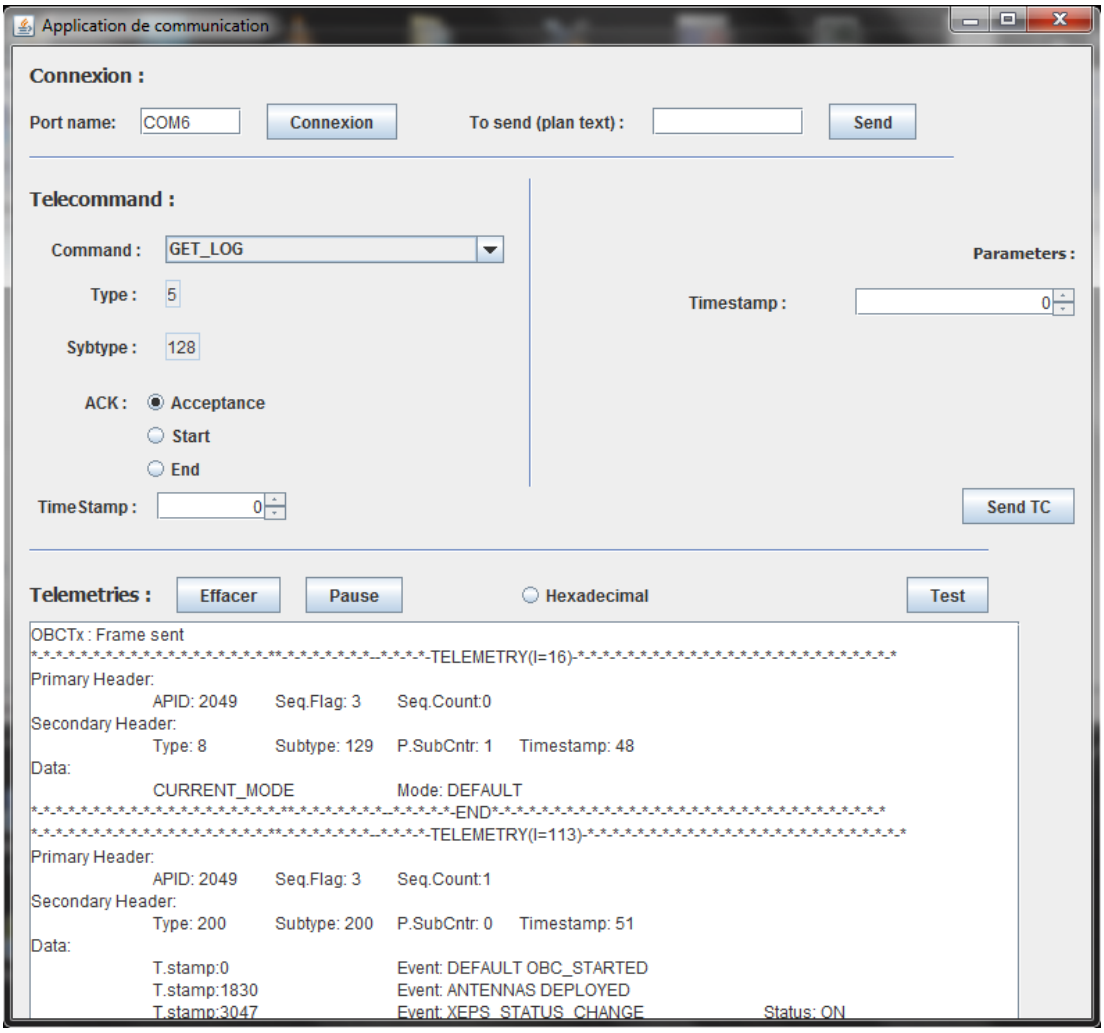

FIGURE 61 - RAPATRIEMENT DU JOURNAL DE BORD

Cette phase de test a permit de confirmer le bon fonctionnement le l'OBSW, seule la redondance est encore affectée par un problème du à une erreur sur le bus I²C : le passage du

Default OBC au Backup OBC en cas de défaillance fonctionne très bien, mais si le Default OBC vient à redémarrer le Backup ne lui rend pas la main. Ce problème n'a malheureusement pas pu être résolu car la défaillance dans le design original n'a été identifiée que très tard et le peu de temps restant n'a pas été suffisant à sa correction.

8 ALLOCATION MÉMOIRE DES TÂCHES

Chaque tâche dispose de son propre espace en mémoire vive (stack) pour enregistrer le contenu de ses variables. Chaque tâche crée doit éviter d'allouer de la mémoire en dehors de son espace d'adressage ou « stack ». La taille du stack alloué à chaque tâche est fixée lors de sa création et ne peut pas être modifiée ultérieurement. Une tâche qui utilise plus de mémoire que celle allouée dans son stack va causer une erreur appelée « stack overflow » qui plantera le programme à coup sur. La fonction « uxTaskGetHighWaterMark() » de freeRTOS permet de connaitre l'espace restant sur le stack d'une tâche, la valeur obtenue est exprimée en mots (2 octets pour l'architecture 16 bits du MSP430. Le peu de RAM disponible dans le MSP430 (5 Ko) nous à obligé à dimensionner au mieux le stack des tâches afin de préserver de l'espace mémoire pour les varibles globales. Pour ce faire nous avons demandé à chaque tâche d'enregistrer en EEPROM le nombre d'octets encore disponible sur son stack et en employant la configuration de test décrite dans le chapitre précédent, nous avons pu récupérer cette valeur. Nous avons en premier lieu alloué beaucoup de RAM à chaque tâche pour ensuite la diminuer pas-à-pas jusqu'à arriver à avoir constamment 100 octets de libre sur le stack. Il a bien entendu fallut utiliser le maximum de fonctionnalité pendant les tests de dimensionnement car à chaque fois qu'une tâche entre dans une fonction de l'espace est alloué pour les varibles locales a la fonction ainsi que pour la valeur de retour de celle-ci.

Chapitre V - CONCLUSION

La programmation des systèmes embarqués complexes peut parfois être un vrai challenge quand on tient compte du peu de ressource disponibles et des contraintes de temps parfois serrées. Cela est encore plus vrai dans le domaine spatial où toutes les précautions possibles doivent êtres prisent afin d'assurer la stabilité du logiciel étant donné l'impossibilité de dépanner sur place.

L'objectif de ce travail était d'implémenter les modules logiciels d'OUFTI-1 ou de compléter le travail des étudiants précédents. Il a d'abord fallu appréhender l'architecture de la solution et se familiariser avec le matériel de développement avant de pouvoir attaquer la programmation proprement dite.

Le premier module à avoir été terminé est le module log qui était déjà partiellement programmé, cependant certaines optimisations on pu être appliquées et la fonction de rapatriement du journal à été entièrement refaite.

Fort des connaissances acquises à propos du bus I²C et de l'EEPROM en travaillant sur le module Log, le module Measurement n'a pas trop posé trop de problèmes au point de vue programmation et tout les objectifs le concernant on pu êtres remplis avec succès.

Le module monitor propose un mécanisme de déploiement des antennes robuste ; l'activation, la désactivation et le reset des sous-systèmes fonctionnent également très bien. Le seul point noir réside dans la gestion de la redondance qui ne fonctionne actuellement que partiellement.

Certaines idées ont été envisagées concernant les développements futurs du sous-système OBC comme par exemple l'ajout d'un mode sol pour effectuer des tests et surveille la charge des batteries avant le lancement, ou encore la possibilité de télécharger une image de la configuration actuelle de la RAM de l'OBC pour diagnostic.

Pour terminer je voudrais ajouter que cette période de stage au sein de l'équipe d'OUFTI-1 fut très enrichissante, tant sur le plan technique que sur le plan humain, un réel esprit de camaraderie est né entre les membres de l'équipe et l'expérience n'en fut que plus agréable.

BIBLIOGRAPHIE

[1] DE DIJCKER (S.), Implémentation de la gestion des télécommandes et des télémétries au sein de l'ordinateur de bord du nanosatellite OUFTI-1, 2011.

[2] CROSSET (N.), Implémentation du relais D-STAR à bord du nanosatellite OUFTI-1, 2010.

[3] MARCHAL (N.), Implémentation, design et test de la carte électronique de télécommunication du CubeSat OUFTI-1, 2010.

[4] CAN (K.), Projet de nanosatellite Oufti-1: Développement du logiciel embarqué, 2010.

[5] TENEY (D.), Design and implementation of on-board processor and software of student nanosatellite OUFTI-1

[6] LEDENT (P.), *Design and Implementation of On-board Digitally Controlled Electrical Power Supply of Student Nanosatellite OUFTI-1 of University of Liege*, Université de Liège, 2009.

[7] http://www.freertos.org

[8] BARRY (R.), freeRTOS reference manual (API Functions and Configuration Options), 2009.

[9] BARRY (R.), Using the freeRTOS Real Time Kernel (A Practical Guide), 2010.

[10] FORTESCUE (P.), STARK (J.) & SWINERD (G.), Spacecraft Systems Engineering 3rd edition, 2003.

[11] http://www.esa.int

[12] CLARKE (T.), Real-Time Operating Systems: pt1 & 2, 2009

[13] http://www.i2c-bus.org

[14]BEUKELAERS (V.), From mission analysis to space flight simulation of the OUFTI-1 nanosatellite, 2009.

[15] Texas Instruments, *MSP320x1xx Family User's Guide*, 2006 .

[16] CubeSat KitTM, *CubeSatKit Motherboard*, Hardware revision D, septembre 2009

ANNEXES

A : LISTE DES ÉVÉNEMENTS DU JOURNAL

Event name	Value	Parameter
EVENT_OBC_STARTED	1	0=BACKUP OBC / 1=DEFAULT OBC
EVENT_ANTENNA_DEPLOYED	2	/
EVENT_xEPS_STATUS_CHANGE	3	0=ON / 1=OFF / 2=RDY
EVENT_MECH_STATUS_CHANGE	4	0=ON / 1=OFF / 2=RDY
EVENT_DSTAR_STATUS_CHANGE	5	0=ON / 1=OFF / 2=RDY
EVENT_RX_STATUS_CHANGE	6	0=ON / 1=OFF / 2=RDY
EVENT_TX_STATUS_CHANGE	7	0=ON / 1=OFF / 2=RDY
EVENT_BCN_STATUS_CHANGE	8	0=ON / 1=OFF / 2=RDY
EVENT_xEPS_FAULT	9	/
EVENT_MEAS_FAULT	10	/
EVENT_COM33_FAULT	11	/
EVENT_COM72_FAULT	12	/
EVENT_EEPROM_FAULT	13	/
EVENT_VBAT_LOW	14	/
EVENT_COM_OBC_FAILED	15	0= unable to get measurement
	//	1= unable to enable D-STAR
	//	2 = unable to enable AX.25 Tx

B: LISTE DES MESURES

Liste des mesures Hardware

Sous-Système	Nom	Identifiant	Capteur	Pt diviseu	ADC	Entrée Beacon	Allocation Bus	HKP
EPS	Courant cellule solaire 1	I_SC1	MAX9938		ADS 1			
EPS	Courant cellule solaire 2	I_SC2	MAX9938		ADS 1			
EPS	Courant cellule solaire 3	I_SC3	MAX9938		ADS 1			
EPS	Courant cellule solaire 4	I_SC4	MAX9938		ADS 1			
EPS	Courant cellule solaire 5	I_SC5	MAX9938		ADS 1			
EPS	Courant cellule Total	I_SCT	MAX9938			X	H1-47	
EPS	Temp Batterie 1	T_BAT1	?		ADS 1	X	H1-48	
EPS	Temp Batterie 2	T_BAT2	?		ADS 1	X	?	
EPS	Temp transistor dissip.	T_EPS	?		ADS 1	X	H1-49	
EPS	Tension batteries	V_BAT		oui	MAX1039	X	H1-50	X
EPS	LIBRE sur MAX1039	/	/		MAX1039			
EPS	3.3 bus voltage	V_3.3		oui	MAX1039	X	H1-51	
EPS	5.0 bus voltage	V_5.0		oui	MAX1039	X	H1-52	
EPS	7.2 bus voltage	V_7.0		oui	MAX1039	X	H2-47	
EPS	Temp face 1	T_F1	?		MAX1039			
EPS	Temp face 2	T_F2	?		MAX1039			
EPS	Temp face 3	T_F3	?		MAX1039			
EPS	Temp face 4	T_F4	?		MAX1039			
EPS	Temp face 5	T_F5	?		MAX1039			
EPS	Temp face 6	T_F6	?		MAX1039			
EPS	Courant 3.3V	I_3.3	MAX9938	?	?	X	H2-48	
EPS	Courant 5.0V	I_5.0	MAX9938	?	?	?	?	
EPS	Courant 7.2V	I_7.2	MAX9938	?	?	?	H2-49	
COM	Courant D-STAR	I_COM_DSTAR	?	?	ADC MSP COM		UART OBC-COM	
COM	Courant AX.25	I_COM_AX25	?	?	ADC MSP COM		UART OBC-COM	
COM	Courant ADF TX	I_COM_TX	?	?	ADC MSP COM		UART OBC-COM	
COM	Temp Ampli COM	T_COM7.2	?	?	ADC MSP COM	X	UART + H2-51(BCN)	?
COM	COM TOS(1)	TOS1_COM	?	?	ADC MSP COM		UART OBC-COM	?
COM	COM TOS(2)	TOS2_COM	?	?	ADC MSP COM		UART OBC-COM	?
xEPS	Tension convertiseur 3.3	V_OUT_EPS2		oui	ADS2			X
xEPS	Courant convertiseur	I_OUT_EPS2	MAX9938		ADS2			
xEPS	Tension circuit digit(max4.2V)	V_DIG_EPS2		oui	ADS2			
xEPS	Courant circuit digital	I_DIG_EPS2	MAX9938		ADS2			
xEPS	T°convertisseur	T_FLY_EPS2	LM94022		ADS2			
xEPS	T°circuit digital	T_DIG_EPS2	LM94022		ADS2			
OBC	Temp F1611	T_OBC	interne					

Liste des mesures Software

Sous-Système	Nom	Identifiant	Capteur	Pt diviseu	ADC	Entrée Beacon	Allocation Bus	HKP
COM	mini-log D-STAR	LOG_DSTAR	/		/			
OBC	Stack Log	HWM_log	/		/			
OBC	Stack COM Tx	HWM_ctx	/		/			
OBC	Stack COM Rx	HWM_crx	/		/			
OBC	Stack Monitor	HWM_mon	/		/			
OBC	Stack Sequencer	HWM_seq	/		/			
OBC	Stack Measurement	HWM_mea	/		/			

C : Code-Source

Le code-source de l'OBC d'OUFTI-1 est disponible à l'adresse suivante :
http://cid-8aa6832f4c7db47a.office.live.com/self.aspx/.Public/OBC%20unifi%c3%a9.rar

D : ACTIVITÉS

JOURNÉE RADIOAMATEURS

Les radioamateurs du Radio club de Gembloux nous ont fait l'honneur de leur visite à l'université de Liège. La journée a commencée par une petite introduction au projet OUFTI, ils nous on ensuite parlés de leur passion pour se domaine extrêmement technique qu'est le radio-amateurisme. La journée s'est poursuivie par différents ateliers, au programme : taillage d'antennes, démonstrations divers et bien sûr établissement de contacts avec entre-autres : les Pays-Bas, l'Italie, et même un opérateur Belge en station-mobile à vélo !

DESIGN REVIEW

Chaque étudiants responsables d'un sous-système ont dû élaborer un résumé technique de leur sous-système respectif. Ce document appelé Data Package à ensuite été soumis à l'équipe système ainsi qu'a un panel d'expert. Le panel ainsi que l'équipe système a alors rédigé toute une série de RID's (Review Item Disposition), c'est-à-dire des questions, des suggestions ou des demandes de clarifications concernant certains points du DataPackage. Tout le monde c'est alors retrouvé à l'EuroSpaceCenter pour un week-end durant lequel des réunions avec les experts du panel étaient organisées pour discuter des solutions trouvées pour répondre aux RID's. Ce week-end fût très chargé en travail mais également très constructif. Nous avons bien sûr également pu profiter des activités proposées par l'EuroSpaceCenter pour nous détendre quand cella s'avérai nécessaire.