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

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Filière Systèmes industriels

Orientation Power & Control

Travail de bachelor
Diplôme 2016

Maxime Dubosson

Module de conversion statique

Professeur
Philippe Barrade

Expert
Toufann Chaudhuri

Date de la remise du rapport
15.07.2016

HES-SO Valais

SYND	ETE	TEVI
X	X	X

Données du travail de diplôme
Daten der Diplomarbeit

FO 1.2.02.07.CB
haj/11/01/2016

Filière / Studiengang **SYND**	Année académique / *Studienjahr* **2015/16**	No TD / *Nr. DA* **pc/2016/61**
Mandant / *Auftraggeber* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **Maxime Dubosson** Professeur / *Dozent* **Philippe Barrade**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja [1] ☒ non / nein	Expert / *Experte (données complètes)* **Dr. Toufann Chaudhuri** ABB Sécheron SA | Rue des Sablières 4-6 | 1242 Satigny	

Titre / *Titel*

Module de Conversion Statique

Description / *Beschreibung*

Ce projet s'inscrit dans les projets de développement du groupe d'électronique industrielle. Ceci concerne les outils et notions nécessaires à la conception modulaire du dispositif de conversion statique. Il s'agit de réaliser un premier module de conversion DC/AC, ses protections et ses contrôles associés. L'objectif est de fournir à la HEI un module qui puisse être dupliqué et utilisé tant pour des laboratoires de la Filière que pour les projets de Ra&D nécessitant un dispositif de conversion statique générique, pouvant être rapidement mis en œuvre pour des tests préliminaires.

L'étude d'identification des fonctionnalités et choix de composants a été déjà réalisée.

Objectifs / *Ziele*

L'objectif de ce travail consiste en la validation expérimentale, qui comprend :

- Assemblage et tests sur un module.
- Analyse du comportement, possibilités d'exploitation du microcontrôleur intégré.

Ce travail sera conclu par un rapport, ainsi qu'une soutenance orale.

Signature ou visa / *Unterschrift oder Visum*	Délais / *Termine*
Responsable de l'orientation / filière *Leiter der Vertiefungsrichtung / Studiengang:* ..... [1] Etudiant / Student :	Attribution du thème / *Ausgabe des Auftrags:* **17.05.2016** Remise du rapport / *Abgabe des Schlussberichts:* **15.07.2016, 12:00** Expositions / *Ausstellungen der Diplomarbeiten:* **31.08 – 01.09 – 02.09.2016** Défense orale / *Mündliche Verfechtung:* **Semaine / *Woche* 36**

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

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

Module de conversion statique

Diplômant Maxime Dubosson

Objectif du projet

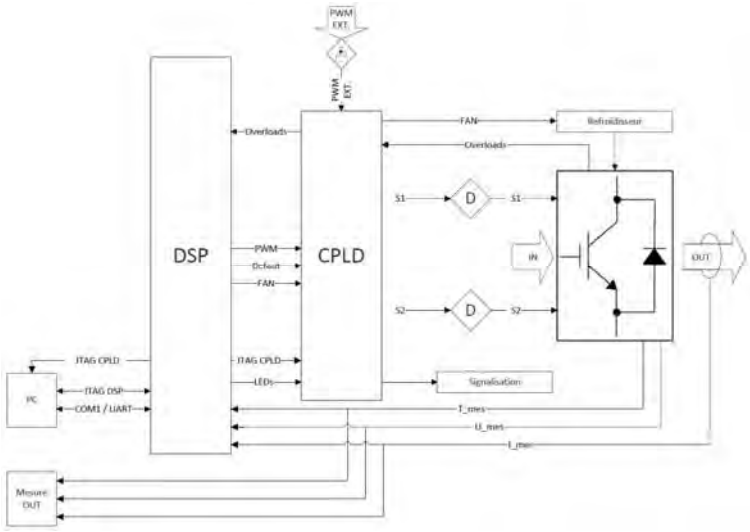
Développer un module de conversion générique, permettant de réaliser les conversions élémentaires, tout en prenant en compte les aspects liés à la sécurité avec la possibilité d'implémenter un réglage interne.

Méthodes | Expériences | Résultats

Une étude complète a été réalisée afin de déterminer les contraintes sur la structure du convertisseur. Une fois ces contraintes identifiées, le choix et le dimensionnement des composants ont permis la conception de la carte électronique.

Le processeur intégré dans le circuit offre une grande possibilité de développement en parallèle de la gestion de la sécurité qui lui a été dédiée.
Les premiers essais réalisés ont permis d'obtenir des résultats positifs. Ceux-ci ont été comparés avec des simulations.
De plus, une communication sérielle a pu être établie entre le convertisseur et une interface de contrôle.

Par la suite, les objectifs sont d'optimiser le module en réduisant la taille de la carte et en permettant la programmation via l'interface de contrôle, et de tester la notion de modularité.
Une étude approfondie sur le refroidisseur devra être réalisée afin de pouvoir réduire ses dimensions.



Représentation fonctionnelle du module de conversion

Travail de diplôme
| édition 2016 |

Filière
Systèmes industriels

Domaine d'application
Power & Control

Professeur responsable
Philippe Barrade
philippe.barrade@hevs.ch

Remerciements

Je souhaite remercier toutes les personnes qui ont œuvrées d'une manière ou d'une autre dans ce projet ainsi que celles qui ont répondu à mes nombreuses questions. J'aimerais porter une attention toute particulière à Messieurs:

- Barrade Philippe qui, en tant que professeur responsable, a été d'une grande disponibilité tout au long du projet et qui a mis toute sa connaissance à disposition pour éclaircir mes incompréhensions et mes doutes.

- Germanier Alain qui, malgré la montagne de travail qu'il avait, a consacré beaucoup de temps pour répondre aux questions et pour discuter des différents choix.

- Horta Rodolfo qui m'a beaucoup aidé pour la mise en place de la communication sérielle.

- Arcudi Carmine qui, dans un premier temps, a rapidement modélisé tous les composants dont j'avais besoin pour la schématique et, dans un second temps, a réalisé le routage de la carte.

- Gallay Steve et ses apprentis qui ont fabriqués la carte de commande du ventilateur et des adaptateurs pour les JTAG.

- Walpen Olivier qui m'a aiguillé dans la recherche de matériel.

Ainsi qu'à Laurane Dubosson pour la relecture du rapport et la correction des fautes d'orthographe.

Résumé

Ce travail de bachelor consiste au développement d'un module de conversion statique, c'est-à-dire un module permettant d'effectuer tous les types de conversion de base qui sont :

- Conversion continue (DC/DC)
- Conversion continue-alternative (DC/AC)
- Conversion alternative-continue (AC/DC)

Ou des conversions plus « complexes » qui nécessitent l'association de plusieurs modules, tels que :

- Conversion alternative-alternative (AC/AC)
- Conversion triphasée

Le critère principal a été de créer un système offrant une large gamme de puissance allant jusqu'à 12kW. Ceci a pu être réalisé en utilisant des facteurs de formes « form factor » bien spécifiques.

De plus, ce module a été conçu pour permettre un fonctionnement autonome (maître « master ») ou un fonctionnement commandé (esclave « slave »). Dans le premier cas, les signaux de commande sont générés par le microprocesseur intégré au module. Celui-ci gère également les aspects liés à la sécurité et à la communication. Dans le deuxième cas, les signaux de commande sont générés par un organe externe via une connexion optique.

Table des matières

Terminologies

$V_{ce,sat}$	Tension de saturation
$I_{c,max}$	Courant maximal
$P_{T,cond}$	Pertes par conduction
$P_{T,com}$	Pertes par commutation
$P_{Tf,max}$	Somme des pertes du composant
$V_{ce,on}$	Tension résultante lorsque le composant est passant
$I_{T,moy}$	Courant moyen
R_T	Résistance interne
$I_{T,eff}$	Courant efficace
E_{Ts}	Pertes par commutation (sur une période de commutation)
$R_{\theta,max}$	Résistance thermique maximale
$\theta_{ambiant}$	Température de l'air ambiant
$\theta_{jonction}$	Température à la jonction du composant
$R_{\theta r,max}$	Résistance thermique du refroidisseur
$R_{\theta,jb}$	Résistance thermique entre la jonction et le boîtier
$R_{\theta,br}$	Résistance thermique entre le boîtier et le refroidisseur
$I_{s,moy}$	Courant moyen circulant dans la charge
D	Rapport cyclique
$I_{e,moy}$	Courant moyen circulant à l'entrée du convertisseur
$I_{s,min}$	Courant minimal
$I_{s,max}$	Courant maximal
ΔI_s	Ondulation de courant
T	Période de commutation
L_s	Inductance de sortie du convertisseur
U_e	Tension continue fournie à l'entrée du convertisseur
$I_{c,eff}$	Courant efficace circulant dans le condensateur

Liste des figures

Liste des tableaux

I. INTRODUCTION

Pour chaque nouveau convertisseur, une étude est réalisée. Celle-ci comprend une série d'opérations répétitives qui sont, dans les grandes lignes :

- Définition des contraintes liées à la structure
- Création de la schématique
- Dimensionnement des composants
- Fabrication de la carte

Une fois ces étapes terminées, il est temps de mesurer et de tester le convertisseur.

L'intérêt de ce projet est donc de simplifier le processus de développement afin de gagner du temps. Une fois le module fabriqué, il sera question de sélectionner les composants voulus et de tester le dispositif.

Le dispositif est doté d'un microprocesseur. Celui-ci gère les sécurités en interne, ce qui a l'avantage de faciliter la mise en place du convertisseur. Ce processeur offre aussi la possibilité de fonctionner en autonomie complète, c'est-à-dire sans aucun lien avec le monde extérieur.

Au final, les limites du système sont définies par la plage de puissance admise et la capacité du microprocesseur.

II. ASPECTS GÉNÉRAUX

Ce chapitre traite des aspects liés à la planification, au déroulement et au but du projet. Il permet de suivre l'évolution du projet au fil du temps.

1 Planning

Tableau 1: Planning prévisionnel du projet

	LUNDI	MARDI	MERCREDI	JEUDI	VENDREDI
SEMAINE 1	*16 mai 16*	*17.05.2016 [1]* Dessiner la schématique	*18.05.2016 [2]* Dessiner la schématique	*19.05.2016 [3]* Dessiner la schématique	*20.05.2016 [4]* Dessiner la schématique
SEMAINE 2	*23.05.2016 [5]* Contrôler et corriger la schématique Prise en main du kit de développement	*24.05.2016 [6]* Programmation du DSP sur le kit de développement Choix des composants	*25.05.2016 [7]* Programmation du DSP sur le kit de développement Choix des composants	*26 mai 16*	*27.05.2016 [8]* Programmation du DSP sur le kit de développement Choix des composants
SEMAINE 3	*30.05.2016 [9]* Programmation du DSP sur le kit de développement Choix des composants	*31.05.2016 [10]* Programmation du DSP sur le kit de développement Choix des composants	*01.06.2016 [11]* Programmation du DSP sur le kit de développement	*02.06.2016 [12]* Programmation du DSP sur le kit de développement	*03.06.2016 [13]* Programmation du DSP sur le kit de développement
SEMAINE 4	*06.06.2016 [14]* Programmation du DSP sur le kit de développement Dimensionnement des résistances, capacités et inductances	*07.06.2016 [15]* Programmation du DSP sur le kit de développement Dimensionnement des résistances, capacités et inductances	*08.06.2016 [16]* Programmation du DSP sur le kit de développement Dimensionnement des résistances, capacités et inductances	*09.06.2016 [17]* Programmation du DSP sur le kit de développement Dimensionnement des résistances, capacités et inductances	*10.06.2016 [18]* Programmation du DSP sur le kit de développement Dimensionnement des résistances, capacités et inductances
SEMAINE 5	*13.06.2016 [19]* Programmation du DSP sur le kit de développement	*14.06.2016 [20]* Programmation du DSP sur le kit de développement	*15.06.2016 [21]* Programmation du DSP sur le kit de développement	*16.06.2016 [22]* Programmation du DSP sur le kit de développement	*17.06.2016 [23]* Programmation du DSP sur le kit de développement
SEMAINE 6	*20.06.2016 [24]* Contrôle du routage de la carte	*21.06.2016 [25]* Pose des composants Contrôle de la soudure	*22.06.2016 [26]* Contrôle des alimentations	*23.06.2016 [27]* Réserve	*24.06.2016 [28]* Réserve
SEMAINE 7	*27.06.2016 [29]* Test de la carte Correction et modification	*28.06.2016 [30]* Test de la carte Correction et modification	*29.06.2016 [31]* Test de la carte Correction et modification	*30.06.2016 [32]* Test de la carte Correction et modification	*01.07.2016 [33]* Test de la carte Correction et modification
SEMAINE 8	*04.07.2016 [34]* Test de la carte Correction et modification	*05.07.2016 [35]* Test de la carte Correction et modification	*06.07.2016 [36]* Test de la carte Correction et modification	*07.07.2016 [37]* Test de la carte Correction et modification	*08.07.2016 [38]* Test de la carte Correction et modification
SEMAINE 9	*11.07.2016 [39]* Réserve	*12.07.2016 [40]* Réserve	*13.07.2016 [41]* Réserve	*14.07.2016 [42]* Réserve	*15.07.2016 [43]* 12h00 remise du rapport

Tableau 2: Planning effectif du projet

	LUNDI	MARDI	MERCREDI	JEUDI	VENDREDI
SEMAINE 1	*16 mai 16*	*17.05.2016 [1]* Dessiner la schématique	*18.05.2016 [2]* Dessiner la schématique	*19.05.2016 [3]* Dessiner la schématique	*20.05.2016 [4]* Dessiner la schématique
SEMAINE 2	*23.05.2016 [5]* Dessiner la schématique	*24.05.2016 [6]* Contrôle et Correction de la schématique Établissement de la liste du matériel	*25.05.2016 [7]* Finalisation de la schématique Choix de l''IGBT Regrouper les fonctions sur le PCB	*26 mai 16*	*27.05.2016 [8]* Regrouper les fonctions sur le PCB
SEMAINE 3	*30.05.2016 [9]* Choisir une capacité de découplage pour les IGBTs Établir un protocole de mesure et de test Liste du matériel manquant	*31.05.2016 [10]* Écriture du rapport Programmation de la CPLD et test sur un board d'évaluation	*01.06.2016 [11]* Programmation de la CPLD et test sur un board d'évaluation Écriture du rapport Calcul du circuit d'overload	*02.06.2016 [12]* Écriture du rapport Prise en main du programme CCS 6 avec le kit de développement Calcul du coût du matériel	*03.06.2016 [13]* Prise en main du programme CCS6 Programmation du DSP Script Matlab pour le calcul des condensateurs de lissage
SEMAINE 4	*06.06.2016 [14]* Programmation du DSP Création d'un module axillaire pour le kit de test + routage de celui-ci	*07.06.2016 [15]* Programmation du DSP Écriture du rapport	*08.06.2016 [16]* Programmation du DSP Calcul des condensateurs Mise à jour des entrées/sorties du DSP	*09.06.2016 [17]* Calculs des condensateurs Programmation du DSP Écriture du rapport	*10.06.2016 [18]* Programmation du DSP
SEMAINE 5	*13.06.2016 [19]* Programmation du DSP	*14.06.2016 [20]* Programmation du DSP Écriture du rapport Calcul des composants (Amplificateur Différentiel) Préparation des composants Routage d'un adaptateur pour le JTAG	*15.06.2016 [21]* Programmation du DSP Préparation des composants Création du module ventilateur Test du ventilateur avec PWM et TACH Résolution d'un problème de connexion avec le DSP	*16.06.2016 [22]* Programmation du DSP Création du module ventilateur Préparation du ventilateur Calcul des amplificateurs différentiels	*17.06.2016 [23]* Écriture du rapport Mise en forme du programme DSP Modification du programme CPLD
SEMAINE 6	*20.06.2016 [24]* Fabrication des adaptateurs pour le JTAG Soudage et test du module ventilateur Écriture du rapport	*21.06.2016 [25]* Écriture du rapport Test du module ventilateur Préparation du refroidisseur Pose des composants sur la carte principale	*22.06.2016 [26]* Pose des composants sur la carte principale Test de la carte	*23.06.2016 [27]* Test de la carte	*24.06.2016 [28]* Test de la carte Ajout du patch pour la tension 1.5V ref Mesure de la puissance dissipée max du refroidisseur
SEMAINE 7	*27.06.2016 [29]* Mesure de la puissance de dissipation max du refroidisseur Écriture du rapport Essai de commutation	*28.06.2016 [30]* Écriture du rapport Mesure des différentes grandeurs et corrections de l'erreur Gestion des temps mort dans la CPLD	*29.06.2016 [31]* Mise en place de la communication Charge du programme sur la mémoire Flash + boot du processeur Écriture du rapport	*30.06.2016 [32]* Essais en puissance BUCK Écriture du rapport	*01.07.2016 [33]* Essais en puissance BUCK Essais en puissance BOOST Préparation pour essais en puissance DC/AC
SEMAINE 8	*04.07.2016 [34]* Simulation Matlab du convertisseur (BUCK et BOOST	*05.07.2016 [35]* Simulation Matlab Mise à jour de la documentation du projet	*06.07.2016 [36]* Essais en puissance (onduleur) Simulation Matlab Programmation de la communication	*07.07.2016 [37]* Programmation de la communication Programmation de l'interface utilisateur Gestion des temps morts dans la CPLD	*08.07.2016 [38]* Programmation de la communication Gestion des temps morts dans la CPLD
SEMAINE 9	*11.07.2016 [39]* Gestion des temps morts dans la CPLD Rangement de l'espace de travail Rédaction du résumé	*12.07.2016 [40]* Mise à jour de la documentation Écriture du rapport	*13.07.2016 [41]* Mise à jour de la documentation Écriture du rapport	*14.07.2016 [42]* Mise à jour de la documentation Écriture du rapport	*15.07.2016 [43]* 12h00 remise du rapport

Différences

Dans les grandes lignes, le planning a été respecté. Le point critique était la réception de la carte. Celle-ci était attendue au 20 juin 2016, elle est arrivée le 22 juin 2016. Cependant les deux jours de réserve planifiés les 23 et 24 juin on permit de combler ce retard.

La dernière semaine du projet a laissé suffisamment de temps pour mettre à jour la documentation et pour la rédaction du rapport.

2 Journal de travail

Dans le cadre du travail de Bachelor et pour avoir un suivi continu de l'avancement du projet, un journal de travail détaillant quotidiennement ce qui a été réalisé a été tenu. Celui-ci peut être consulté en annexe 1.

Des journées de 8h30 ont été prises en compte lors de la planification du projet.

Globalement, peu de temps supplémentaire a été nécessaire pour le bon déroulement du projet. Il en aura fallu deux.

3 Cahier des charges

Le cahier des charges très global a permis une grande souplesse au niveau de la conception et des différents choix effectués. L'objectif était de réaliser un module de conversion statique disposant de sécurités suffisantes et permettant de transmettre des signaux vers l'extérieur.

L'ajout d'un processeur offre une grande possibilité de développement. Il n'a pas été imposé, cependant, il a très vite été évident du besoin d'un tel dispositif pour répondre à la demande de surveillance interne.

4 Coûts

Un des objectifs principal a été la question du coût d'un tel module. Toute la réflexion a été faite afin de minimiser les coûts sans perdre en sécurité et en qualité. Avec le temps à disposition pour réaliser un projet de cette ampleur, le module ne peut pas être totalement optimisé. Un développement approfondi permettrait probablement de réduire d'avantages les coûts.

A noter que certains sacrifices ont été faits, notamment pour la mesure de tension du bus continue. Tous ces choix sont décrits dans le projet de semestre[1].

[1] (Dubosson, 2016)

4.1 Matériel

Les tableaux ci-dessous présentent respectivement un aperçu du coût matériel dépensé pour le projet ainsi que pour la fabrication d'un module. Le tableau complet se trouve en annexe 2. Cependant, il est important de noter que le prix des composants a été défini à une date précise. De plus, l'achat de matériel en grande quantité permet de réduire leur coût unitaire. Le prix des composants SMD (résistances, capacités,…) a été négligé.

Le prix final se situe avec une marge de ±20%.

Le taux de change utilisé pour le calcul du coût de fabrication du PCB est celui du 17.06.2016 (date de facturation). Il est de 0.92407 € = 1 CHF.

Tableau 3: Coût matériel du projet

Description	Quantité [pce]	Prix Unitaire [CHF]	Prix total [CHF]	Annexe
Liste des composants	1	263.4	263.4	2
Fabrication PCB	3	191.9	575.8	3
		Total :	**839.2**	

Tableau 4: Coût matériel pour un module

Description	Quantité [pce]	Prix Unitaire [CHF]	Prix total [CHF]	Annexe
Liste des composants	1	263.4	263.4	2
Fabrication PCB	1	191.9	191.9	3
		Total :	**455.3**	

4.2 Main d'œuvre

A titre informatif, le tableau 5 ci-dessous présente la main d'œuvre qui a été utilisé dans le cadre du projet. Il s'agit de collaborateur de la HES-SO // Valais – Wallis.

Ce coût n'apparaît qu'une seule fois dans le processus de développement.

Tableau 5: Coût main d'œuvre pour le projet

Description	Quantité [h]	Prix Unitaire [CHF]	Prix total [CHF]	Remarque
Routage du PCB	42	70	2940	collaborateur
Modules annexes	8	10	80	apprenti
		Total :	**3020**	

III. DESCRIPTION DU PROJET DE SEMESTRE

Lors du projet de semestre[2], il a été question de définir:

- Les contraintes liées à la structure
- Les fonctions à implémenter
- Le choix ou le « form factor » des composants

Avec pour objectif de terminer le développement de la schématique.

En raison de la migration du programme de schématique, celle-ci n'a pas pu être entièrement terminée. Deux semaines ont été nécessaires pour la finaliser.

Les fonctions sont présentées dans la figure 1 ci-après. Toutes les informations complémentaires se trouvent dans le projet de semestre.

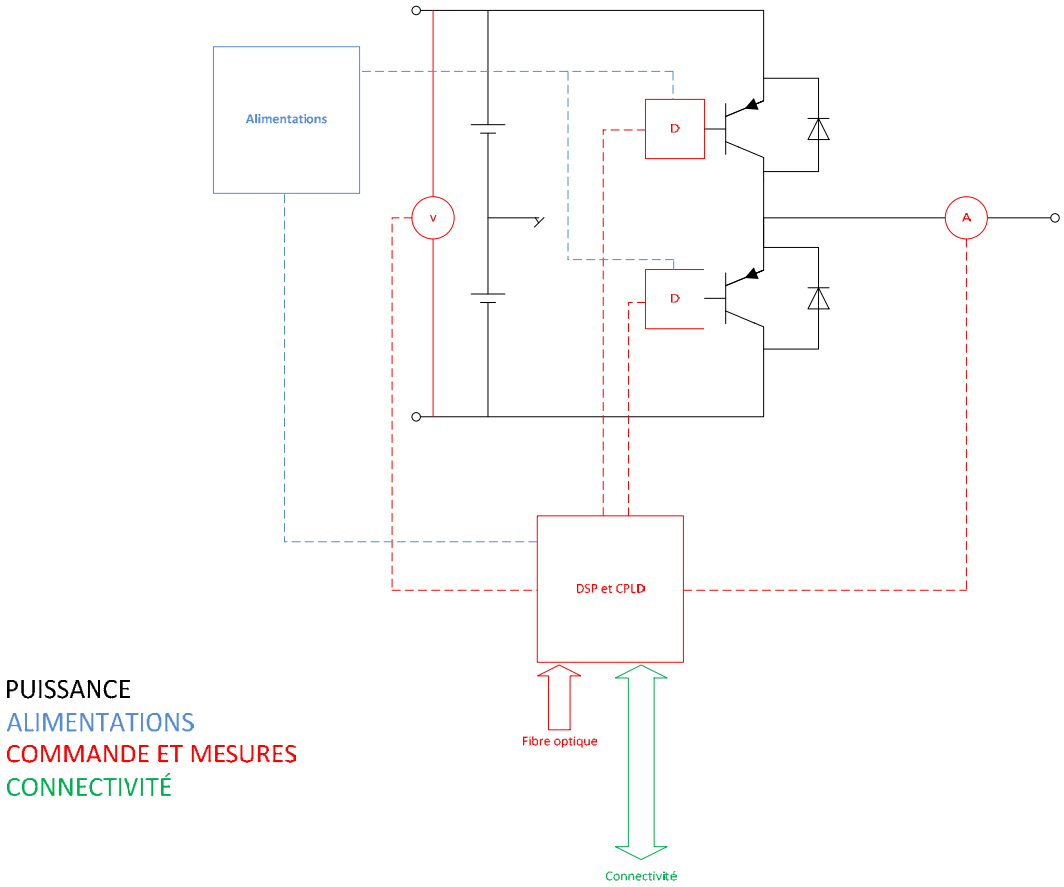


Figure 1: Schéma simplifié définissant les fonctions présentes sur le module

Celles-ci seront définies et expliquées dans le chapitre « IV. HARDWARE ».

[2] (Dubosson, 2016)

IV. HARDWARE

1 Puissance

Le demi-pont de puissance a été conçu de manière que lorsqu'on combine les modules, tous les types de conversions pourront être réalisés.

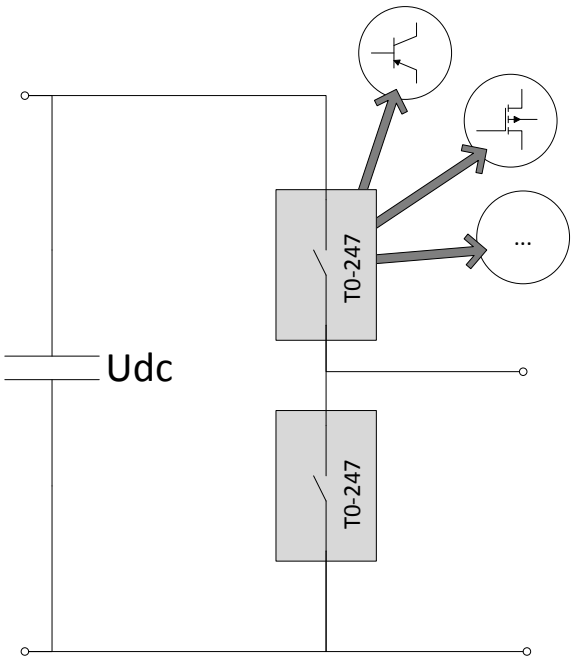


Figure 3: Structure du demi-pont de puissance

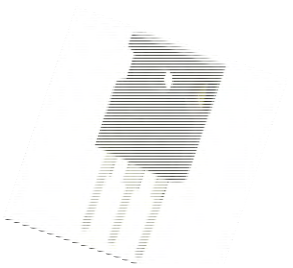
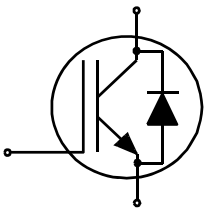


Figure 2: Form Factor T0-247

1.1 Interrupteurs

Les composants de puissance ont été modélisés par un interrupteur dans la figure 3. Le facteur de forme est un boîtier de type « T0-247 ».

Le composant choisi pour le prototype est un IGBT avec la diode en antiparallèle intégrée dans le boîtier, ceci afin de gagner de la place. Le composant a été choisi selon les caractéristiques demandées et parmi les stocks disponibles au laboratoire.



Infineon IKW25N120T2

Figure 4: Représentation du composant utilisé pour le prototype

1.2 Condensateurs

Le critère principal de choix des condensateurs est leur courant efficace limite. Un script Matlab a été implémenté pour calculer ces différents courants selon le type de conversion effectué ainsi que le courant traversant le convertisseur.

Pour le cas d'une conversion DC-DC les paramètres suivants ont été utilisés :

$$F_{com} = 20\ kHz$$

$$F = 50\ Hz$$

$$I_s = 0 \rightarrow 40\ A$$

$$D = 0 \rightarrow 1$$

Ce qui permet d'obtenir le résultat présenté à la figure 5. Pour un courant moyen limite fixé à 20A, Ce courant efficace circulant dans la branche de condensateur vaudrait 10A (point critique à D = 0.5).

Le courant circulant dans une branche de condensateur vaut donc :

$$I_{capa,rms} = \frac{I_{rms}}{\#condo} = \frac{10}{10} = 1\ A_{rms}$$

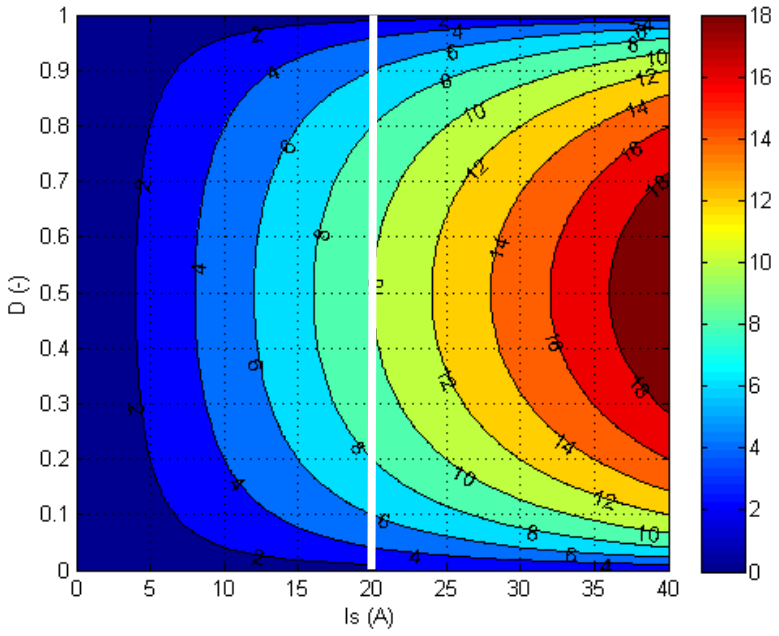


Figure 5: Courant efficace circulant dans la branche de condensateurs (DCDC)

Ce même raisonnement a été fait pour une conversion DC-AC :

$$F_{com} = 20\ kHz$$

$$F = 50\ Hz$$

$$I_s = 0 \rightarrow 40\ A$$

$$D = 0 \rightarrow 0.5$$

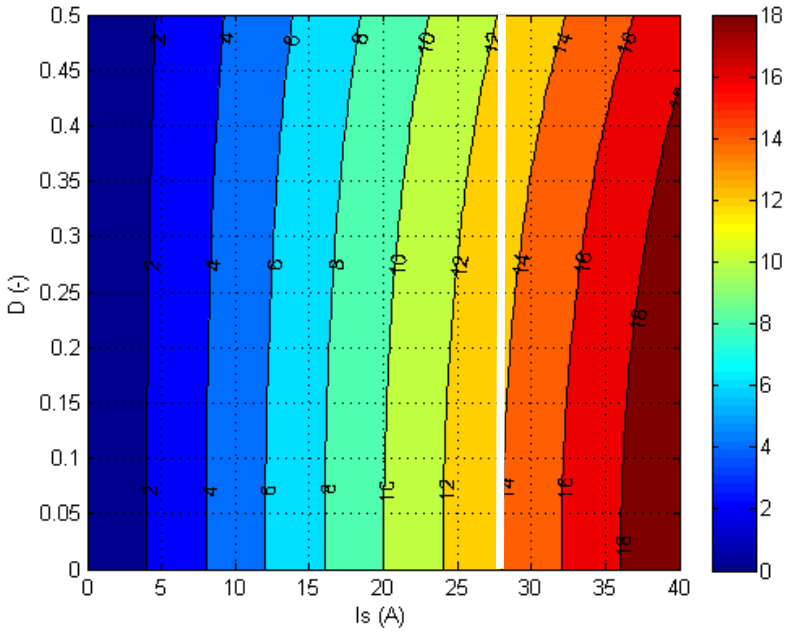


Figure 6: Courant efficace circulant dans la branche de condensateurs (DCAC)

Pour un courant efficace limite fixé à $\sqrt{2} * 20A$, ce courant efficace vaudrait 14A (point critique à D = 0).

A noter que pour la conversion alternative, le rapport cyclique varie sinusoïdalement selon l'expression suivante :

$$Duty = 0.5 + D * \sin(wt + \varphi)$$

Le point D sur le graphique exprime donc la variation du rapport cyclique par rapport au point milieu fixé à 0.5.

Le courant circulant dans une branche de condensateur sera :

$$I_{capa,rms} = \frac{I_{rms}}{\#condo} = \frac{14}{10} = 1.4\ A_{rms}$$

La série de condensateur QXW de la firme Rubycon indique un coefficient d'adaptation selon la fréquence de commutation. Ce coefficient vaut 1.5 pour une fréquence supérieure ou égale à 10kHz.

$$I_{capa,comp,rms} = \frac{I_{capa,rms}}{coeff} = \frac{1.4}{1.5} = 0.93\ A_{rms}$$

Les condensateurs choisis sont «450QXW180MEFC18x45 ». Chacun peut supporter un courant efficace de 1.09A.

1.3 Résistances d'équilibrage

Étant donné qu'un point milieu est créé sur le bus continu et afin de le fixer précisément, il est nécessaire d'insérer une résistance identique entre le commun et les tensions positive et négative.

Les résistances SMD utilisées permettent une chute de tension maximale de 150V ainsi qu'une puissance de dissipation de 0.125W. Ce qui permet de définir la résistance minimale.

$$P = \frac{U^2}{R} \rightarrow R = \frac{U^2}{P} = \frac{200^2}{0.125} = 320\ K\Omega$$

La série de résistance E 24 nous offre la possibilité de choisir une résistance de 330 $K\Omega$.

Trois résistances en série seront placées afin d'avoir une marge de sécurité. Ce qui permet de définir la tension maximale du bus à +600V / -600V.

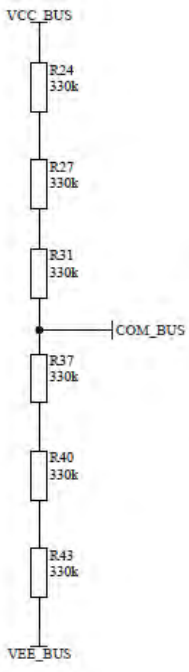


Figure 7: Résistances d'équilibrage du bus DC

1.4 Inductance

La valeur minimale de l'inductance devant être placée sur le module est fonction de la tension à ses bornes et de la variation du courant par rapport au temps.

$$L = U_l * \frac{dt}{di}$$

Pour étudier le cas le plus critique, et selon les composants utilisés, voici les grandeurs des différents paramètres.

$U_l = 900\ V$	Tension maximale possible aux bornes de l'inductance
$dt = 3us$	Delta T maximal entre 2 mesures
$di = 50\ A$	Limite imposée par le composant (§1.1) avec une sécurité

$$L = 900 * \frac{3us}{50} = 54\ uH$$

L'inductance minimale devra donc valoir 54uH.

1.5 Refroidisseur

Le principal phénomène de dissipation utilisé par le refroidisseur est le transfert de chaleur par convection. Cette puissance dissipée est définie selon l'expression suivante :

$$\dot{Q} = h_{conv} * S * (T_{air} - T_s)^3$$

$\dot{Q}$	:	Puissance $[W]$
h_{conv}	:	Coefficient de transfert de chaleur $[\frac{W}{m^2K}]$
T_{air}	:	Température de l'air $[K]$
T_s	:	Température du refroidisseur $[K]$
S	:	Surface du refroidisseur$[m^2]$

Cette équation a été prise du cours de thermodynamique de Monsieur Page Jessen lors du 6ème semestre de formation de la Hes-So.

Pour définir ce coefficient, la décision a été prise de réaliser des essais pratique. Ces essais consistent à chauffer le refroidisseur avec une puissance constante. La température a été mesurée régulièrement jusqu'à atteindre la température d'équilibre, comme le démontre la figure 8 suivante. Ces essais ont été réalisés pour deux puissances qui sont 100 et 200 Watt.

Le matériel utilisé pour ces essais est :

- Alimentation continue : Agilent Technologies A8761A
- Résistance thermique : Dig-Chip BRQ-100R-10L
- Capteur thermique : FLIR i7

[3] (Page)

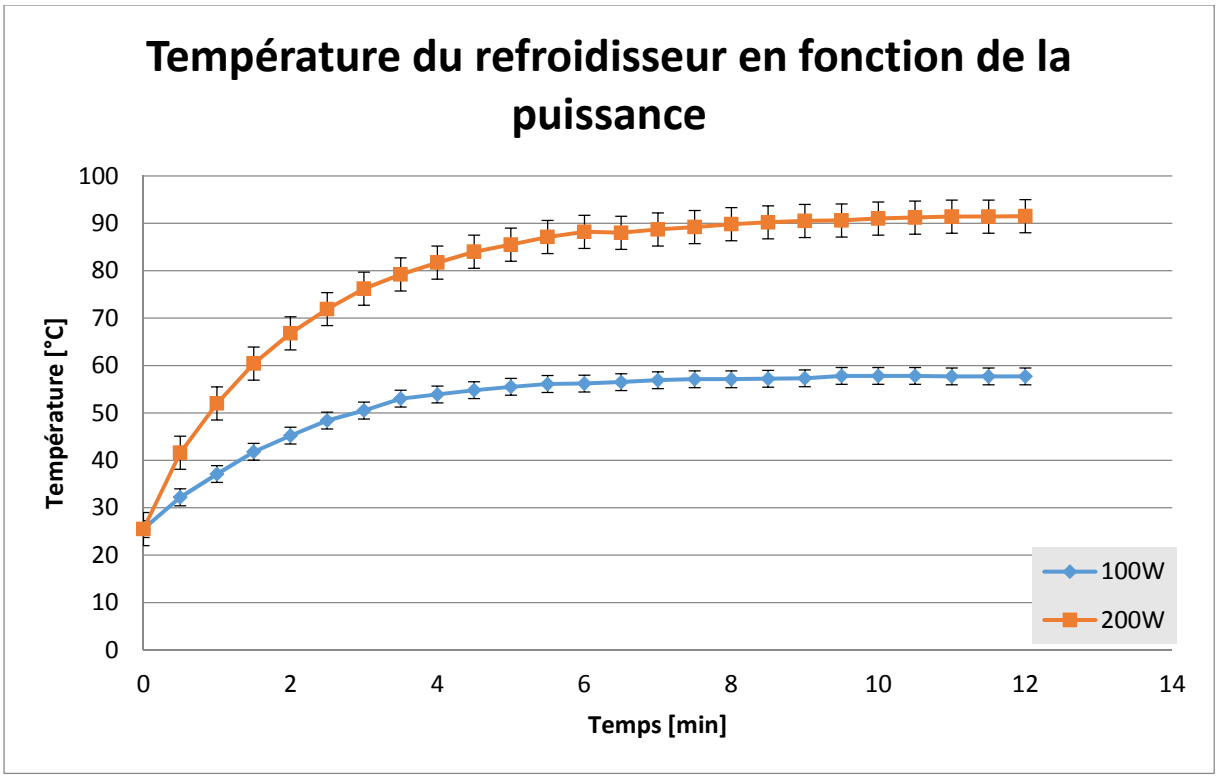


Figure 8: Température du refroidisseur en fonction de la puissance fournie

La surface du refroidisseur a été mesurée et vaut 0.036 m^2. Seules les parois en contact avec l'air ventilé ont été prises en compte. Lors de la mesure, la température du laboratoire était de 25 °C. Ce qui permet donc de définir ce coefficient :

$$h_{conv} = \frac{\dot{Q}}{S * (T_{air} - T_s)}$$

Tableau 6: Coefficient de transfert de chaleur par convection pour le refroidisseur

Puissance $\dot{Q}$	Surface S	Température de l'air T_{air}	Température de surface T_s	Coefficient h_{conv}	Erreur %
100	0.036	25	57.7	-84.95	12
200	0.036	25	91.4	-83.67	6

Ainsi la température de surface a pu être définie en fonction de la puissance dissipée (Figure 9) pour le ventilateur fonctionnant à plein régime. Dans cette figure, l'erreur maximale a été prise en compte soit les 12 % du coefficient de transfert de chaleur.

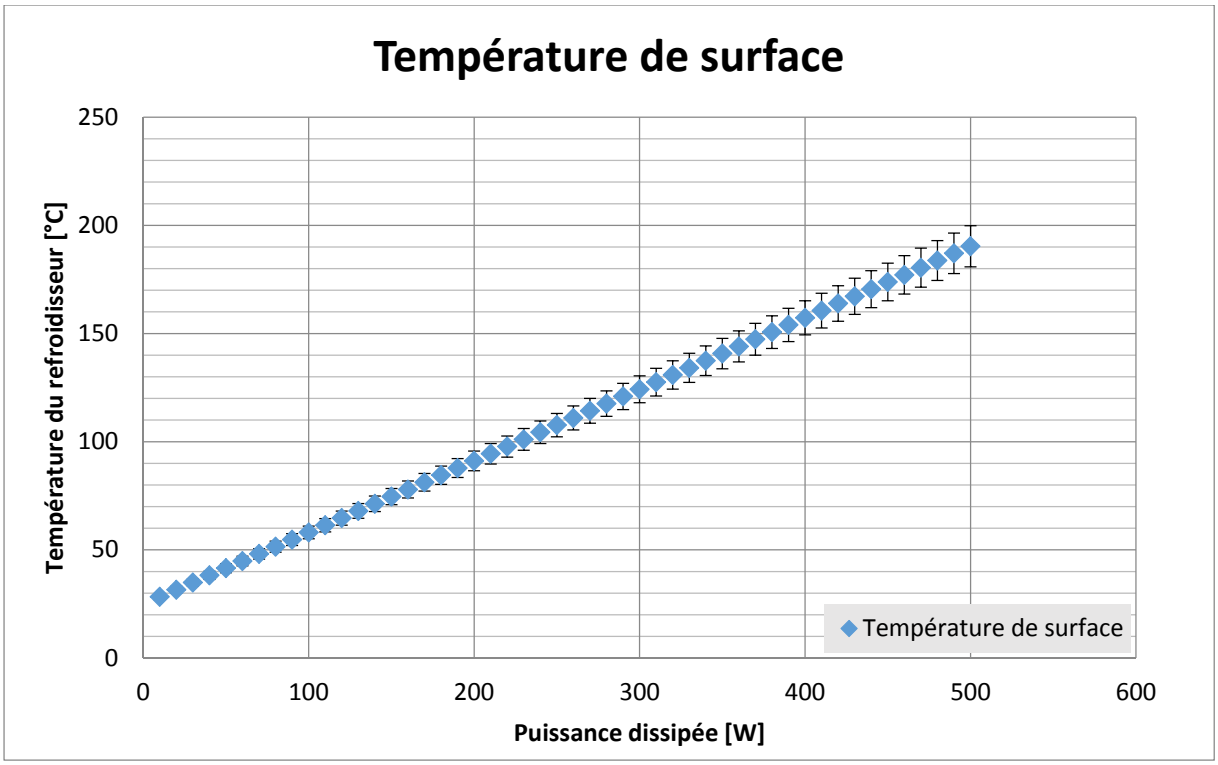


Figure 9: Température de surface en fonction de la puissance à dissiper

2 Alimentations

Toutes les tensions d'alimentations nécessaires seront créées à partir du 24Vdc.

Celles-ci sont :

- +24Vdc Ventilateur
- +15Vdc / -5Vdc Alimentation des drivers
- +5Vdc Capteur de courant LEM LTSR25-NP
- +3.3Vdc Signaux d'entrées / sorties
- +1.8Vdc Alimentation de la CPLD
- +1.2Vdc Alimentation du DSP

La figure 10 décrit le processus de conversion.

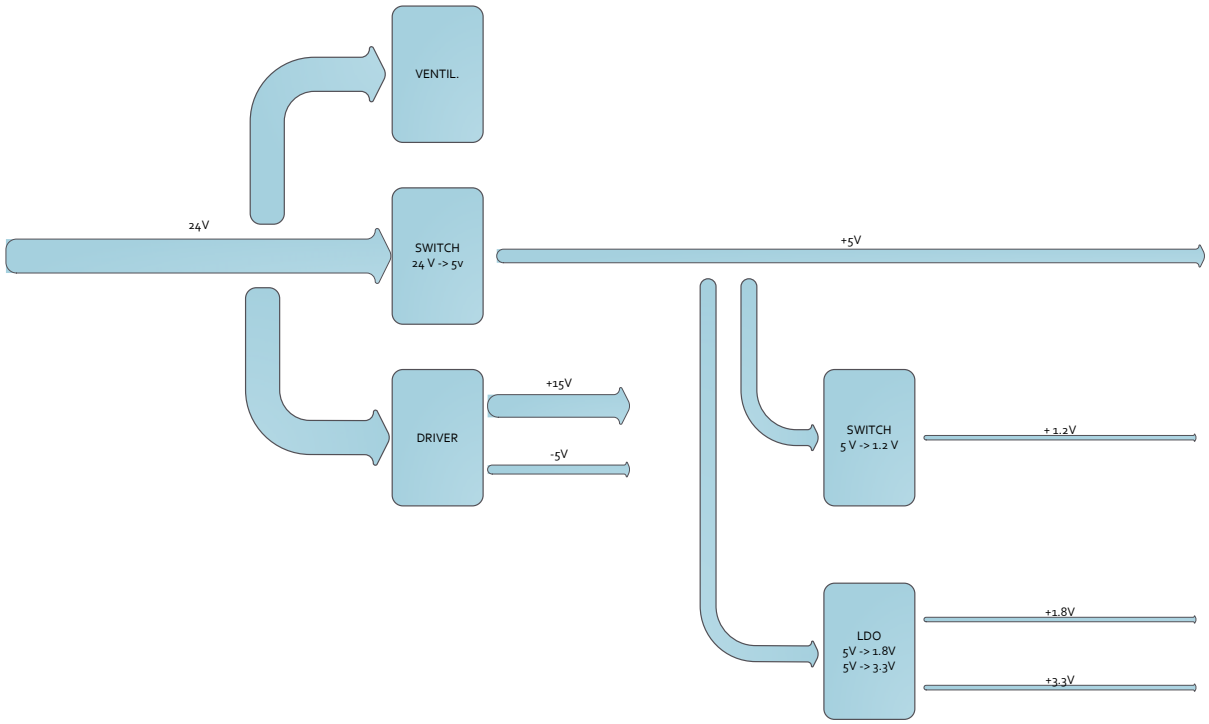


Figure 10: Processus de conversion des tensions d'alimentations

Suivant l'étude effectuée lors du travail de semestre[4], le tableau 7 présente les différents convertisseurs choisis.

Tableau 7: Choix des différents convertisseurs

Conversion	**Fabriquant**	**Type**	**Utilisation**
24Vdc → +5Vdc	Cincon	EC4A11-H	Général
24Vdc → +5Vdc	Murata	MEF1S2405SP3C	Communication Uart
24Vdc → +15Vdc / -5Vdc	Murata	MGJ2D241505SC	Drivers
5Vdc → +1.2Vdc	Texas Instruments	TPS62262DDCT	Microprocesseur
5Vdc → +3.3Vdc / +1.8Vdc	Texas Instruments	TPS767D318-Q1	DSP et CPLD

A noter que lors de l'étude, le choix pour le convertisseur « 5Vdc → 1.2Vdc » s'était porté sur le produit MCP1603L-120 du fabriquant MICRO-CHIP. Après vérification, le produit n'était pas adapté pour ce cas précis d'une part et d'autre part, il était très difficile à obtenir. Il a donc été décidé de changer d'alimentation.

Les essais qui seront réalisés durant ce travail vont permettre de valider ou alors d'écarter ces différents composants.

[4] (Dubosson, 2016)

3 Mesures

Les grandeurs mesurées sur la carte sont utilisées pour la sécurité d'une part et pour le contrôle du système d'autre part. Celles-ci sont :

- La tension positive du bus DC
- La tension négative du bus DC
- La tension d'alimentation 24Vdc
- Le courant de sortie
- La température des composants de puissance
- La tension aux bornes du composant de puissance lorsque celui-ci est passant

Les signaux doivent ensuite être adaptés selon la plage d'entrée du microcontrôleur. Pour ce faire, un amplificateur a été implémenté selon la figure 11.

Les valeurs seront spécifiées selon chaque mesure.

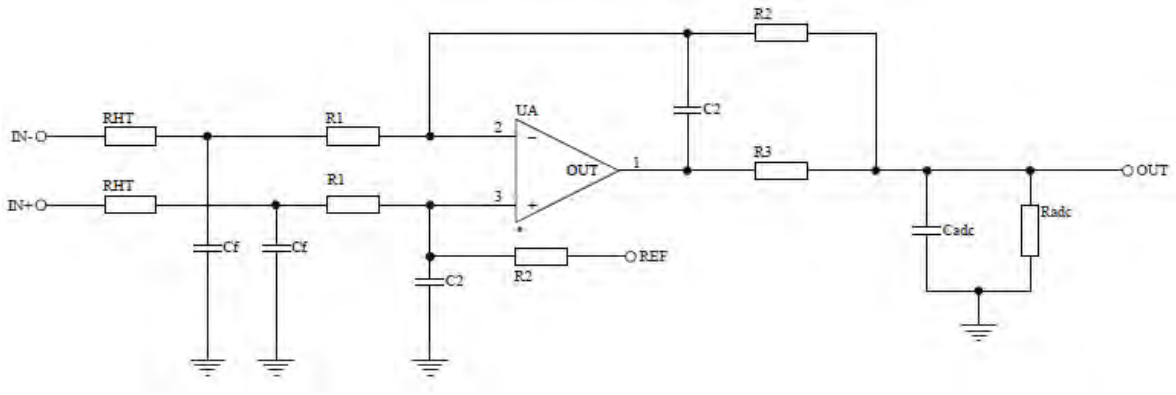


Figure 11: Schéma de l'amplificateur pour les signaux de mesure

Ces valeurs ont été calculées à l'aide d'un fichier Excel développé au laboratoire d'électronique industriel de la HES-SO // Valais – Wallis de Sion. (Annexe 12). Le tableau 8 résume les résultats obtenus avec ce fichier.

Tableau 8: Grandeurs spécifiques de chaque amplificateur différentiel

	24Vdc	**Tension bus DC**	**Courant de sortie**
$V_{nominal}$	24 V	450 V	± 20 A
V_{max}	30 V	500 V	± 80 A → 4 V
V_{out}	3 V	3 V	3 V
Gain souhaité	0.1	0.006	0.75
fg	10 kHz	10 kHz	200 kHz

C_{adc}	10 nF	10 nF	1 nF
R_3	430 Ω	430 Ω	220 Ω
C_2	2.2 nF	2.2 nF	220 pF
R_2	10 kΩ	9.1 kΩ	3.6 kΩ
R_{HT}	82 kΩ	1.5 MΩ	3.6 kΩ
R_1	18 kΩ	16 kΩ	1.2 kΩ
C_f	470 pF	470 pF	470 pF
Gain obtenu	0.1	0.006003	0.75

3.1 Tension du bus DC

Pour mesurer la tension du bus DC qui pourrait être grande, il est nécessaire d'abaisser la tension avec des résistances de grandes valeurs. Ces résistances ayant une chute de tension admissible à leurs bornes, il a été nécessaire d'en disposer deux en parallèle.

Ces résistances (figure 12) représentent les résistances « RHT » de la figure 11.

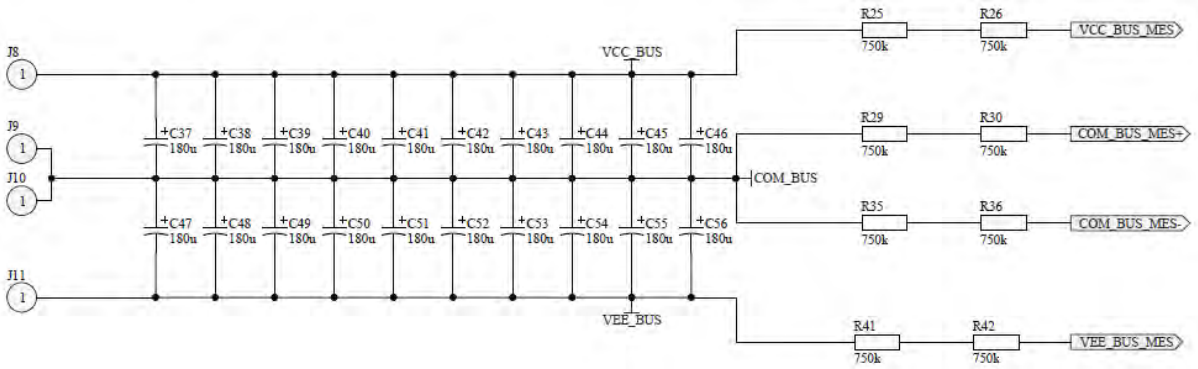


Figure 12: Bus DC avec résistances pour la mesure de la tension

Le circuit de mesure est identique à la figure 11 et le tableau 8 défini la valeur des composants.

3.2 Tension d'alimentation 24Vdc

La mesure de la tension 24Vdc est réalisée avec le même principe que pour les autres grandeurs. La figure 11 indique le circuit de mesure. La différence notable est la valeur des résistances, celles-ci se trouvent dans le tableau 8.

3.3 Courant de sortie

Le courant de sortie est mesuré avec un capteur LEM. Celui-ci fournit une tension comprise entre 0 et 5V. L'expression suivante permet de définir cette tension :

$$U_{out} = 2.5 \pm 0{,}625 * \frac{I_{mes}}{I_{pn}}$$

La figure 13 représente cette équation.

Ce courant nominal vaut 25A pour le capteur choisi « LEM LTSR-25NP ». Les informations du composant sont disponibles sur la liste du matériel en annexe 2.

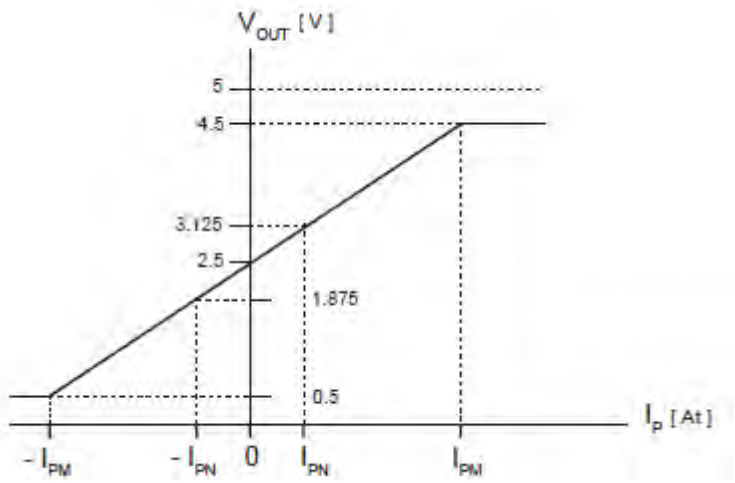


Figure 13: Tension de sortie vs Courant mesuré pour le capteur LEM LTSR-25NP

3.4 Température

La mesure de température est nécessaire pour protéger les composants de puissance. Le capteur utilisé est une thermistance « KTY81-120 » fabriqué par Phillips. Le tableau 9 exprime la résistance en fonction de la température. Ces grandeurs ont été récupérées dans la fiche technique (Valeur TYP.) du composant.

Tableau 9: Résistance du composant en fonction de la température

Température [°C]	Résistance [Ω]	Température [°C]	Résistance [Ω]
0	815	**80**	1490
10	886	**90**	1591
20	961	**100**	1696
25	1000	**110**	1805

30	1040	**120**	1915
40	1122	**125**	1970
50	1209	**130**	2023
60	1299	**140**	2124
70	1392	**150**	2211

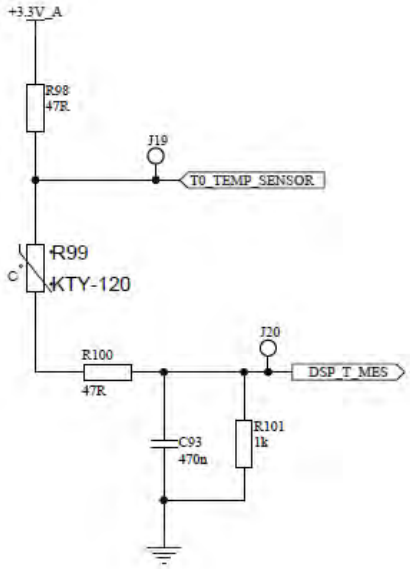


Figure 15: Mesure de la température

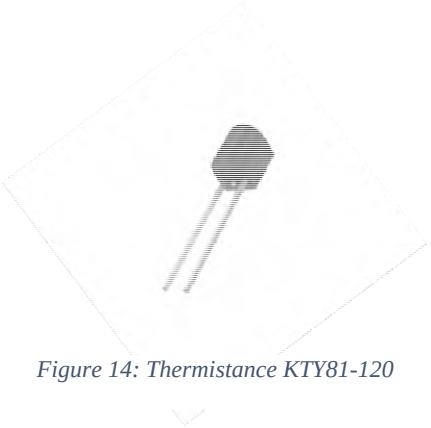


Figure 14: Thermistance KTY81-120

La caractéristique réelle de cette mesure de température est représentée sur la figure 16. L'équation de linéarisation a été définie entre 25 et 80°C. Celle-ci vaut :

$$y = U_{25} - \frac{U_{25} - U_{80}}{T_{80} - T_{25}} * x + 25$$

$$y = 1.566 - \frac{1.566 - 1.35}{80 - 25} * x + 25 = 1.566 - 0.003927 * x + 25$$

L'erreur maximale entre la valeur réelle et la valeur linéarisée vaut 10% à 45°C.

La température mesurée pour une température de 120°C est de 117°C, on constate donc que la courbe reste linéaire sur toute la plage de mesure.

Avec la précision de la mesure, il est nécessaire, pour protéger les composants, de fixer la température de coupure en dessous de la température limite du composant. Cette température devra être inférieure de 15% de la température limite.

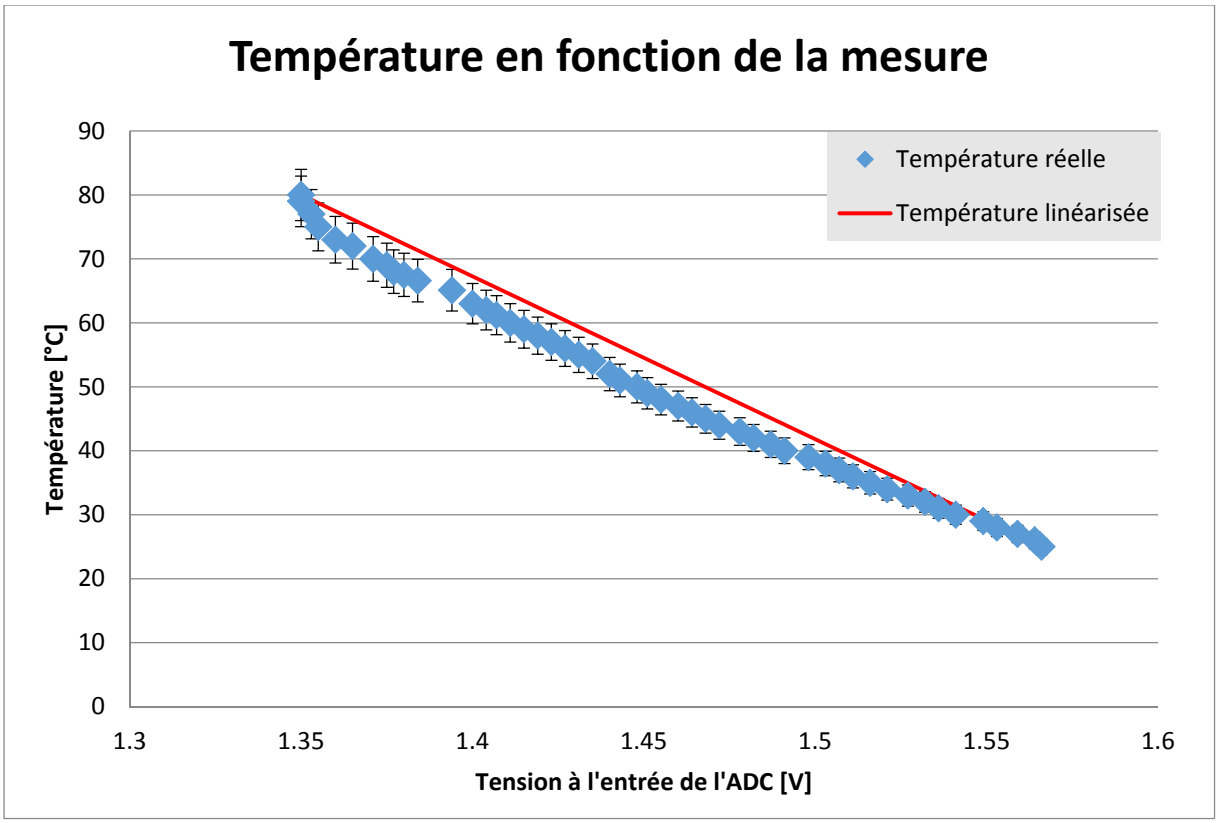


Figure 16: Température en fonction de la mesure de tension

3.5 Overload

La mesure d'overload est une sécurité supplémentaire pour la mesure du courant traversant le composant. La chute de tension entre le collecteur et l'émetteur est fonction du courant, comme le présente les figures 17-18 suivantes. Cette sécurité est présente pour éviter la destruction du composant.

Dans notre situation, la tension entre le gate et l'émetteur vaut 15V. En limitant le courant à 50A, il faudrait imposer le seuil à 2.2V à 25°C et 3.5V à 175°C.

Le composant choisi pouvant supporter des courants allant jusqu'à 100A, la décision a été prise de fixer cette limite à 3.5V, ce qui équivaut à des courants de coupure de 50A dans le meilleur des cas et 85A dans le pire des cas.

Cette mesure d'overload a été récupérée d'un projet existant.

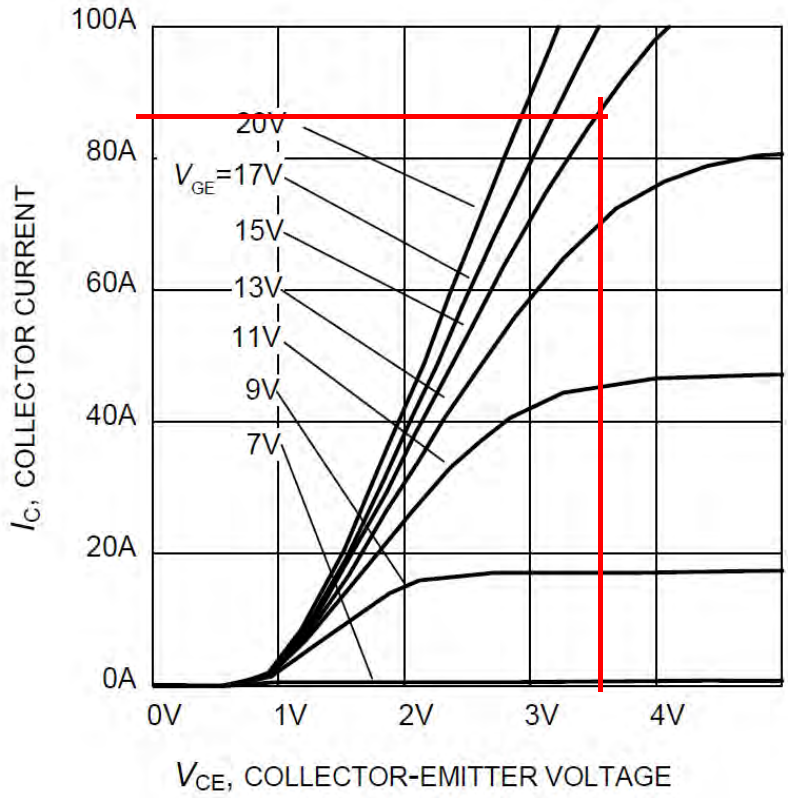


Figure 17: Caractéristique du courant collecteur en fonction de la tension Vce à 25°C

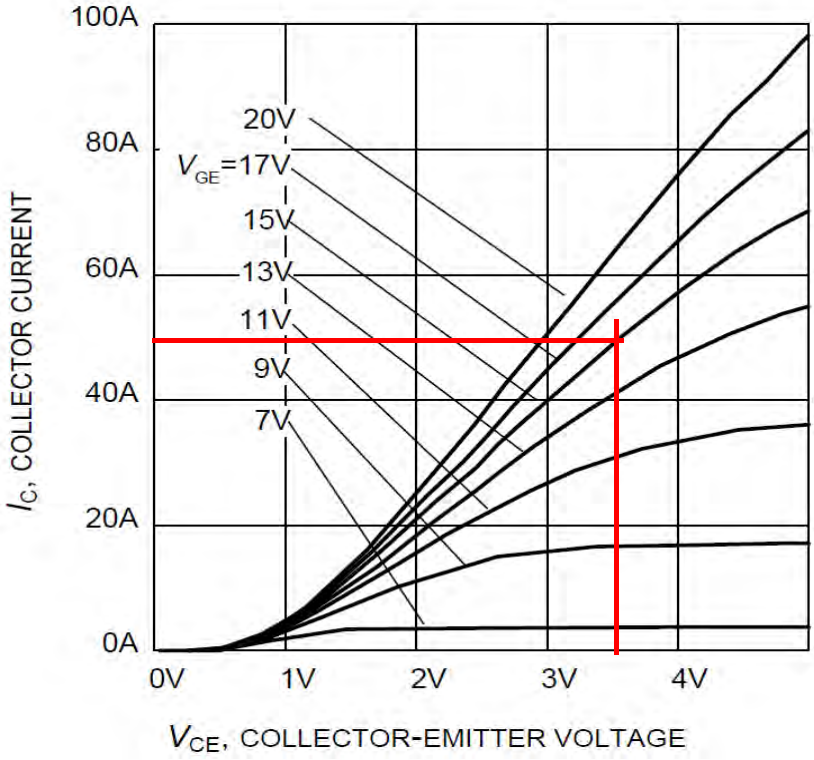


Figure 18: Caractéristique du courant collecteur en fonction de la tension Vce à 175°C

Ce dispositif offre la possibilité de créer n'importe quelle tension de référence (U_{ref}<>1.24V). Dans notre cas, la tension doit être de 3.5V, soit plus grand que les 1.24V du LMV431ACM5. La résistance R_{52} devra donc être court-circuitée et la résistance R_{56} ne devra pas être montée. L'équation permettant de définir la valeur de résistance est :

$$R_x = R_{ref} * \frac{U_{seuil} - 1.24}{1.24}$$

En fixant R_{ref} à 1K2, on obtient :

$$R_x = 1200 * \frac{3.5 - 1.24}{1.24} = 2187\ \Omega$$

Les grandeurs normalisées les plus proches sont $R_x = 2200\ \Omega$ et $R_{ref} = 1200\ \Omega$ ce qui permet d'obtenir une tension de seuil de $U_{seuil} = 3.51\ V$

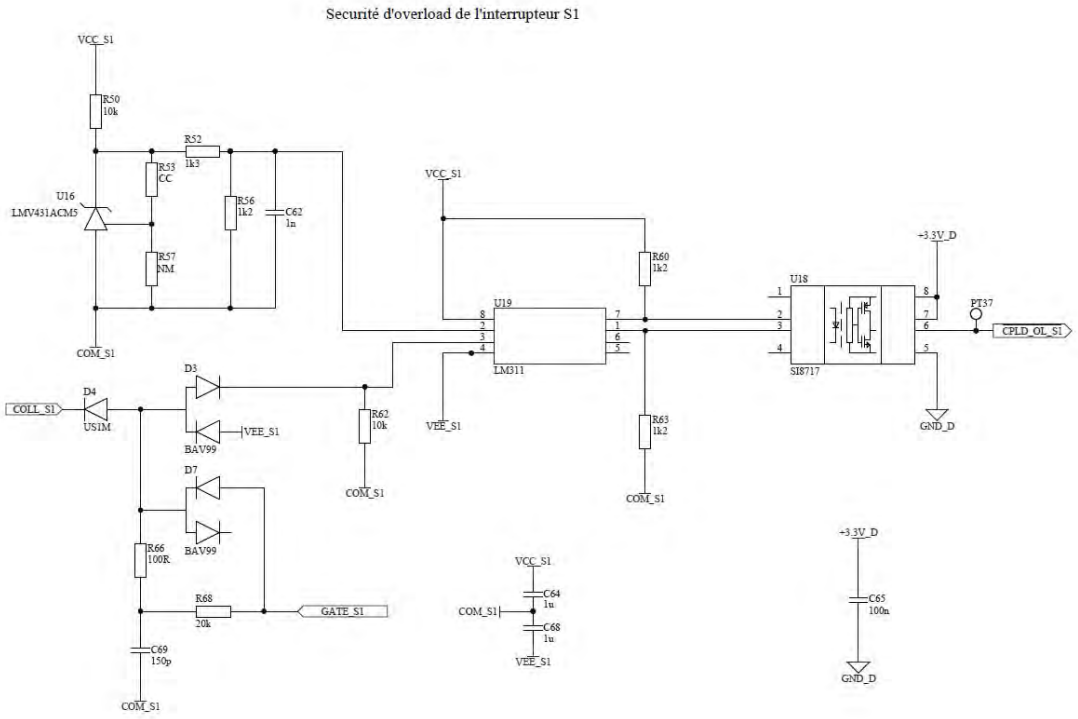


Figure 19: Overload pour le premier interrupteur:

4 Signalisation

Afin de faciliter la détection des états du module, trois LEDs ont été ajoutées.

- LED OK : Le module est quittancé et peut démarrer
- LED ALIM OK : Toutes les alimentations sont présentes
- LED ERR : Une erreur est présente

La figure 20 suivante représente le schéma des LEDs.

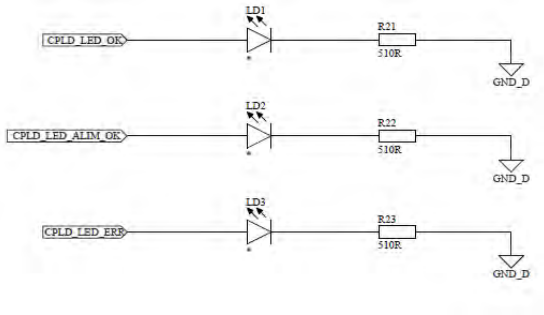


Figure 20: LEDs de signalisation

5 Connectivité

5.1 Connecteur sériel

Il permet la communication entre le microprocesseur et un organe externe (Ordinateur, etc…). Le protocole de communication utilisé est le protocole « UART ».

Le connecteur utilisé est un « D-SUB » neuf pôles avec pins de connexions coudées. La dénomination complète se trouve dans la liste de matériel en annexe 2.

5.2 Connecteurs optiques

Ces deux connecteurs reçoivent le signal PWM venant de l'organe de commande. Ils offrent, d'une part, l'avantage d'avoir une séparation galvanique et, d'autre part, la vitesse de transmission est plus élevée qu'avec une connexion électrique. Ils sont fabriqués par la firme Avago. Leur dénomination est « HFBR2528 ».

5.3 JTAG

Ou « Boundary Scan » est un connecteur permettant de communiquer avec des composants tels que le DSP et la CPLD. Il est généralement utilisé pour la programmation de ceux-ci. Cependant il n'offre aucune séparation galvanique. Il est donc nécessaire d'agir avec précaution. C'est pourquoi, l'idée est de pouvoir programmer le processeur via le connecteur sériel décrit ci-dessus.

5.4 Connecteur SPI

L'interface périphérique sérielle sera utilisée pour le debugge analogique. Il a été implémenté, mais comme la programmation du microprocesseur a été réalisée sur une carte

d'évaluation, celui-ci ne sera pas utilisé dans ce projet. Le temps mis à disposition pour ce projet ne permet pas la programmation de cette partie dans le processeur.

6 Module ventilateur

Cette carte gère la séparation galvanique entre les signaux venant du processeur et le ventilateur. Cette séparation est nécessaire en raison des tensions d'alimentations différentes.

La raison pour laquelle ce dispositif n'a pas été implémenté directement sur la carte principale est que lors de la conception de celle-ci, le choix de la commande du ventilateur n'était pas encore défini.

La schématique se trouve en annexe 7.

Sur cette carte, la tension d'alimentation du ventilateur ainsi que la tension 3.3V sont présentes. Deux optocoupleurs isolent les signaux PWM et TACH.

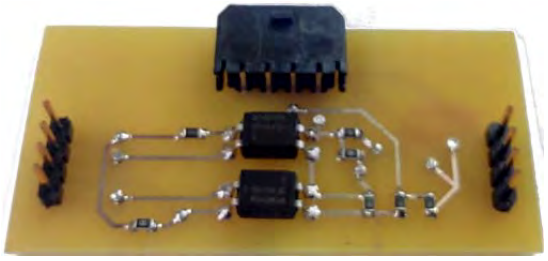


Figure 21: Module annexe ventilateur

6.1 PWM

Les résistances R1 et R2 permettent de définir la tension du signal PWM à envoyer au ventilateur. Celui-ci attend une tension maximale de 6V.

En définissant $R_1 = 4700\ \Omega\ et\ R_2 = 1000\ \Omega$, on obtient :

$$U_{pwm} = \frac{U_{in}}{\frac{R_1 + R_2}{R_1}} = \frac{24}{\frac{4700 + 1000}{1000}} = 4.2V$$

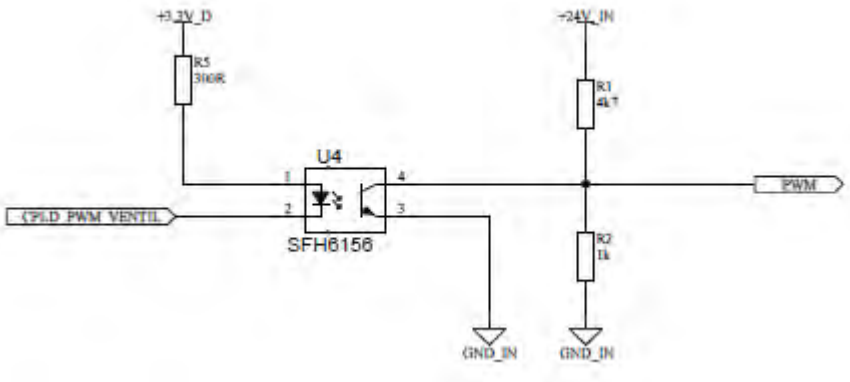


Figure 22: Isolation du signal PWM pour le ventilateur

6.2 TACH

Les résistances ont été calculées afin de fournir un courant de 5mA à l'entrée de l'optocoupleur. Lorsque la led conduit, elle impose une tension de 1 volt à ses bornes. Le courant circulant dans les résistances R4 et R6 vaut :

$$I_4 = I_6 = \frac{U_{in} - U_{diode}}{R_4 + R_6} = \frac{24 - 1}{2200 + 2200} = 5.23mA$$

Le courant circulant dans la résistance R7 vaut :

$$I_7 = \frac{U_{diode}}{R_7} = \frac{1}{5100} = 0.2mA$$

Ce qui permet d'imposer un courant de 5mA aux bornes de l'optocoupleur :

$$I_{opt} = I_6 - I_7 = 5.23 - 0.2 \cong 5mA$$

Le rapport de transfert de courant a été estimé à 0.6 selon la fiche technique du composant. Quand le transistor est activé, un courant de 3mA peut circuler. Avec une résistance$R_1 = 1000\ \Omega$, on peut amener la tension « CPLD_TACH_VENTIL » a :

$$U_{tach} = U_d - R_3 * I = 3.3 - 1000 * 3mA = 300mV$$

Le signal du tachymètre va donc varier entre 3.3V et 300mV.

La puissance de dissipation maximale des résistances est de 125mW. Dans le cas présent, on dissipe :

$$P_{dis} = R_4 * {I_4}^2 = 2200 * 5.23mA^2 = 60mW$$

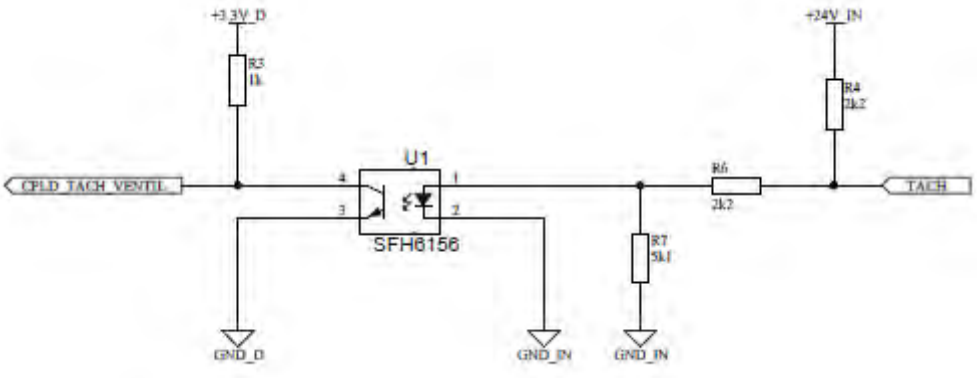


Figure 23: Isolation du signal TACH pour le ventilateur

Ce qui permet d'approuver toutes les décisions prisent antérieurement.

7 Patch 1.5V

Sur la carte prototype réalisée pendant le projet de diplôme, une erreur sur le schéma a imposé la pose d'un patch sur la carte. Il s'agit de la tension de référence de 1.5V. La figure 24 suivante permet de se représenter la pose du patch sur la carte.

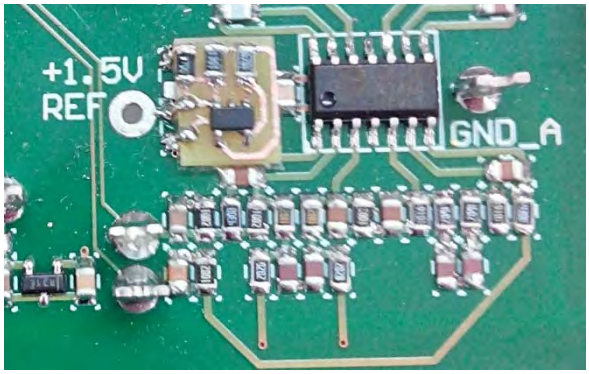


Figure 24: Patch pour la tension de référence 1.5V

8 Adaptateurs JTAG

Lors de la réalisation de la carte, le connecteur JTAG utilisé n'est pas le même que celui de l'interface. Pour cette raison, des adaptateurs ont dû être réalisés pour le JTAG du DSP et celui de la CPLD. La figure 25 montre l'adaptateur réalisé pour le JTAG du processeur.

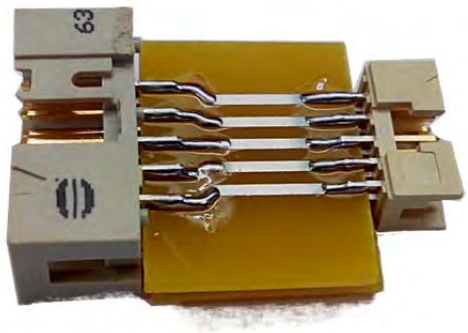


Figure 25: Adaptateur utilisé pour le JTAG du microprocesseur

Le principe reste le même pour le JTAG de la CPLD.

V. SOFTWARE

1 Microprocesseur [DSP]

1.1 Logiciel

« Code Composer Studio, version 6.1.3 » est le logiciel utilisé pour la programmation du microcontrôleur.

1.2 Matériel

Le kit de développement utilisé est le « LaunchXL-F28377S : DEV BOARD ». Le guide utilisateur est référencé dans la bibliographie.

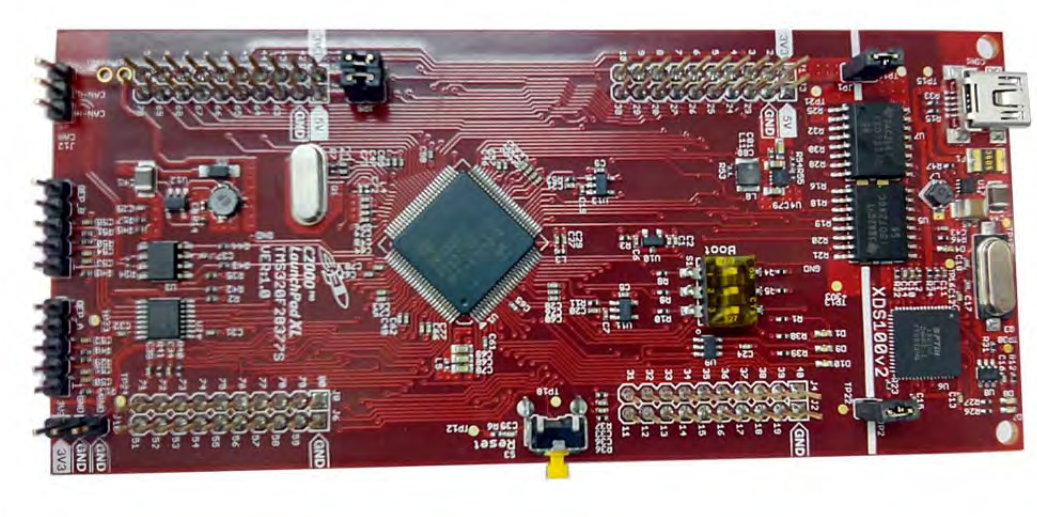


Figure 26: Kit de développement TI LaunchXL-F28377S

Afin de faciliter les mesures lors du développement, une carte de connexion a été créée. Le plan de routage se trouve en annexe 8.

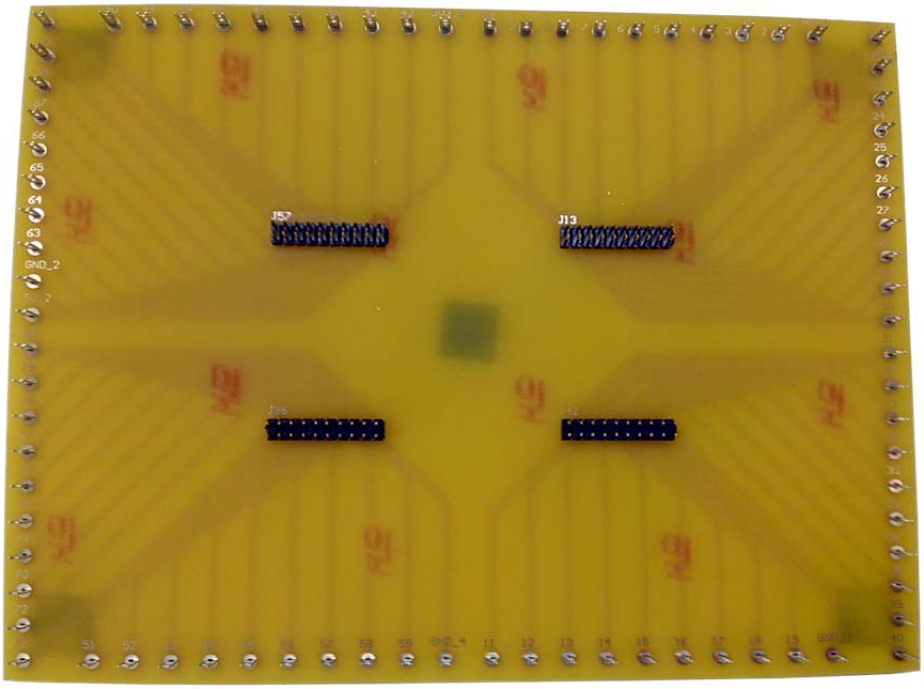


Figure 27: module annexe de connexion pour le kit LaunchXL-F28377S

1.3 Programme

La structure et le contenu du programme ne sont pas optimisés, le changement de processeur en est la cause. Le temps à disposition pour ce projet ne permet pas d'avoir un travail fini et optimal. Le choix a donc été pris d'implémenter les fonctions permettant de garantir la sécurité pour le module et les personnes. Ces fonctions sont :

- Mesure des signaux entrant (ADC)
- Gestion du ventilateur (PWM)
- Gestion des erreurs (sécurité)
- Création et régulation des signaux de commande PWM internes au module
- Interface permettant la modification des paramètres via une connexion sérielle
- Transmission des mesures sur une interface externe via une connexion sérielle

Des idées de développement ont déjà été réfléchies, celles-ci pourront être réalisées lors d'un travail futur.

- Ajout d'une carte mémoire communiquant avec le protocole I^2C
- Communication entre plusieurs modules (master – slave)
- Quittance des erreurs via une interface

Le développement d'un programme abouti pour un nouveau processeur prend plusieurs années, c'est pourquoi ce programme ne peut être considéré comme optimal. Malgré tout, une base solide a été créée.

ADC

1.3.1.1 Temps d'acquisition

Le temps d'acquisition est la période durant laquelle le microprocesseur va lire les données sur l'entrée analogique. Ce temps est configurable dans le registre « ADCSOCxCTL ».

L'exemple suivant défini le temps d'acquisition pour le registre SOC0.

```
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 0x39; // set acquisition prescale for SOC 0
```

Il est calculé comme suit :

$$Acquisition\ window = \frac{sample\ window}{T_{cpu}} = \frac{200ns}{5ns} = 40$$

$$ACQPS = Acquisition\ window - 1 = 40 - 1 = 39$$

Ce chiffre correspond au nombre de cycles systèmes « SYSCLK ».

1.3.1.2 Fréquence des mesures

Deux fréquences de mesures ont été implémentées, 500kHz et 5kHz :

- Mesure à 500kHz : consacrée à la mesure du courant. Elle a besoin d'être rapide pour protéger les composants de puissance de la destruction en cas de défaut. Il s'agit d'une redondance ave la mesure des overloads.
- Mesure à 5kHz : consacrée à la mesure des tensions du bus continu, de la tension d'alimentation 24Vdc et de la mesure de température. Ces mesures sont effectuées alternativement (figure 29) à chaque signal de l'horloge à 3kHz. Ce qui amène la fréquence de mesure à 1kHz pour chaque signal.

La figure 28 exprime les différentes interruptions programmées qui sont :

- Canal 1 : Fréquence à 500kHz (jaune)
  - Mesure du courant (ADCA3)
- Canal 2 : Fréquence à 5kHz (rouge)
  - Mesure des tensions du bus DC (ADCA2 et ADCA5)
  - Mesure de la tension d'alimentation (ADCA4)
  - Mesure de la température (ADCB2)

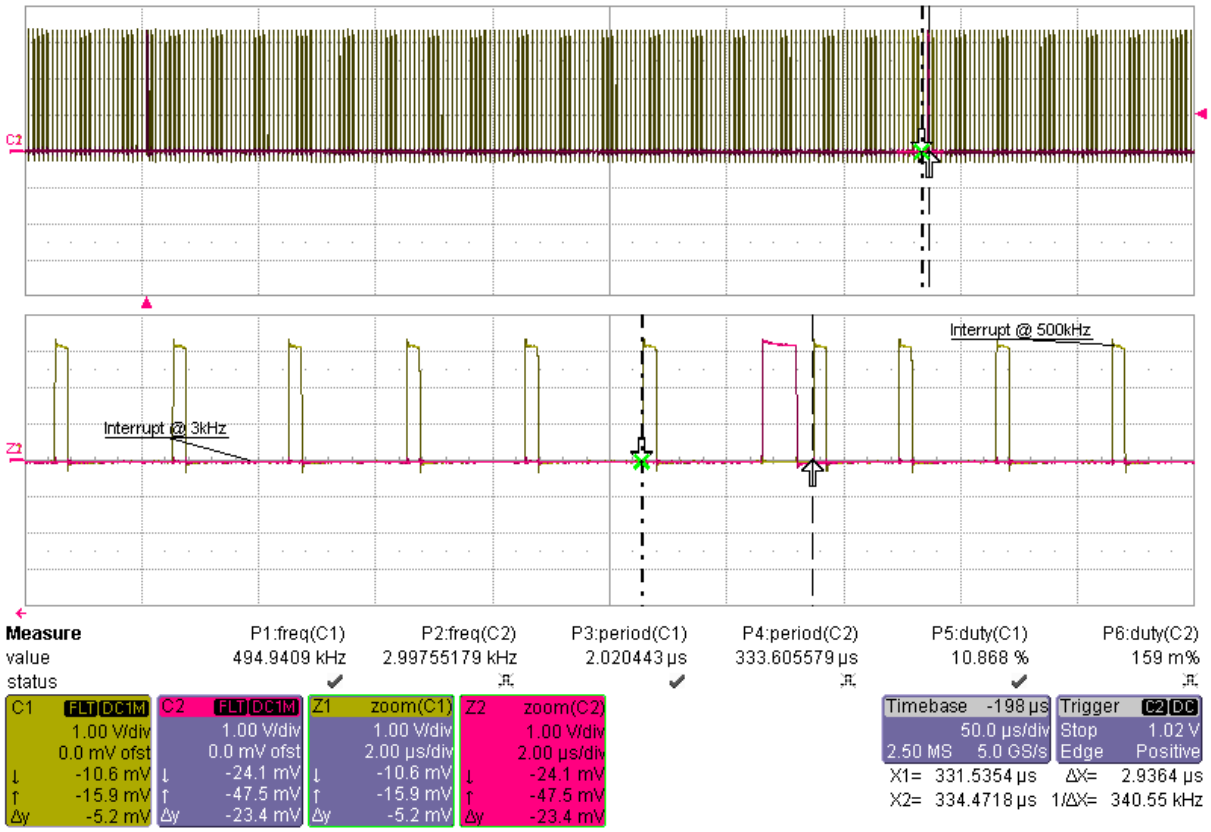


Figure 28: Fréquences des différentes interruptions

Lors du signal d'horloge à 3kHz, un certain retard est pris sur l'horloge à 500kHz, comme le montrent les curseurs sur la figure 28. Ce qui fait que le temps maximal entre 2 mesures n'est plus de 2us (fréquence de 500kHz) mais de 3us (mesure avec les curseurs). C'est ce temps de 3us qui doit être pris en compte pour les calculs de sécurité et de dimensionnement.

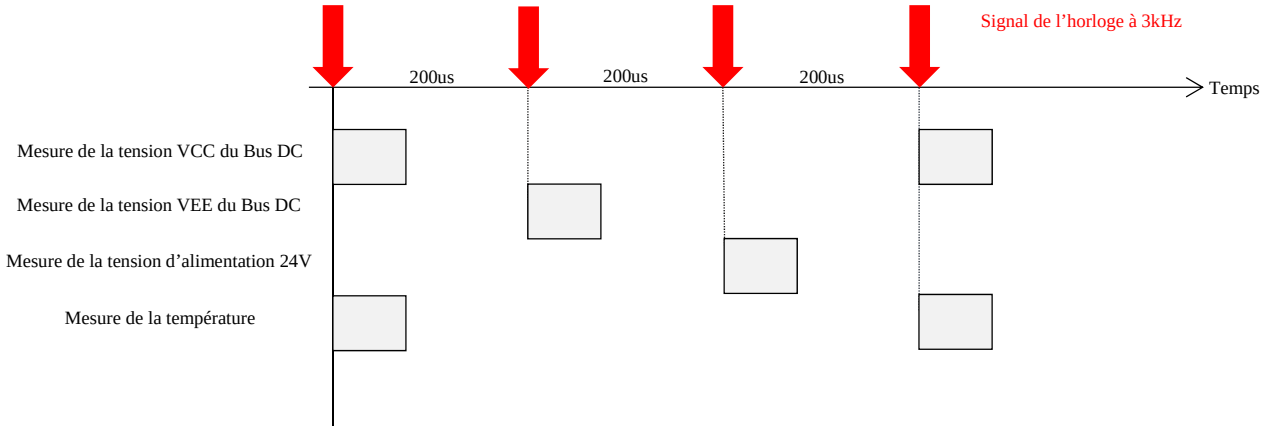


Figure 29: Cycle de mesures pour l'horloge à 3kHz

PWM

1.3.1.3 Refroidisseur

Une commande PWM est divisée en plusieurs modules qui doivent être configurés. Ces modules sont présentés à la figure 30 suivante.

- Time-Base TB
- Counter Compare CC
- Action Qualifier AQ
- Dead Band DB
- PWM-chopper PC
- Trip Zone TZ
- Digital Compare DC
- Event Trigger ET

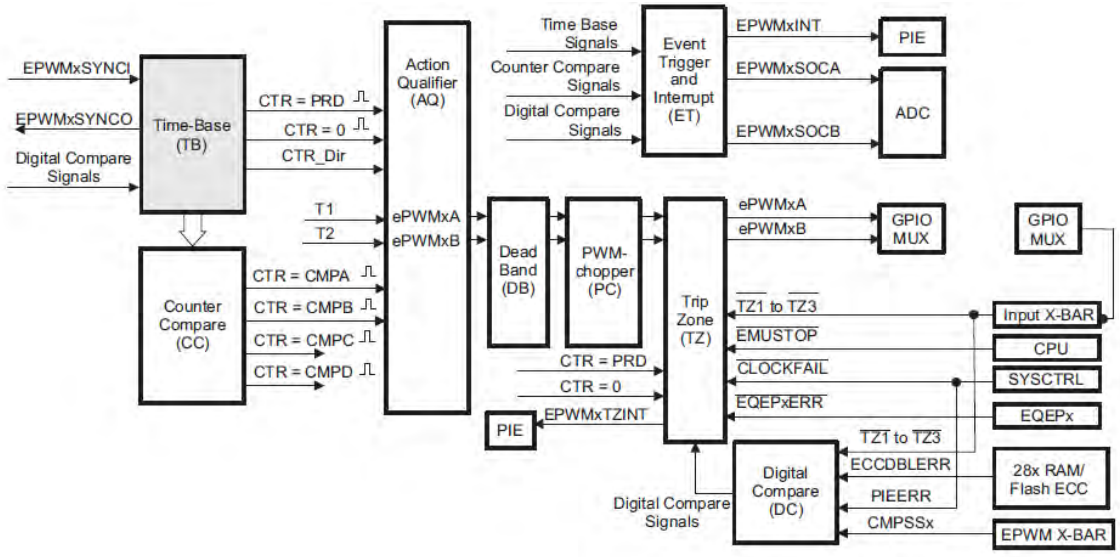


Figure 30: Représentation de la création d'un signal PWM

La fréquence du signal est définie selon l'expression suivante :

$$T_{pwm} = 2 * TBPRD * TBCLK$$

En paramétrant les « CLKDIV » ainsi,

```
EPwm6Regs.TBCTL.bit.HSPCLKDIV = TB_DIV2;
EPwm6Regs.TBCTL.bit.CLKDIV    = TB_DIV2;
```

La période d'horloge « TBCLK » vaut 80ns.

Ce qui permet de déterminer « TBPRD » en fonction de la fréquence désirée.

$$TBPRD = \frac{T_{pwm}}{2 * TBCLK}$$

Pour définir le rapport cyclique, il suffit de varier la grandeur « CMPA ».

```
EPwm6Regs.CMPA.bit.CMPA = value;
```

Cette valeur doit être comprise entre 0 et TBPRD, ce qui définit un rapport cyclique compris entre 0 et 100%.

Les registres de l'AQ permettent de définir l'action à effectuer en fonction du résultat de la comparaison avec le « CMPA » défini précédemment.

```
EPwm6Regs.AQCTLA.bit.CAU  =  AQ_SET;
EPwm6Regs.AQCTLA.bit.CAD = AQ_CLEAR;
```

La figure 31 suivante décrit le fonctionnement du module PWM avec les comparaisons. Lorsque la valeur du registre « TBCTR » dépasse la valeur de comparaison « CMPA », le signal PWM passe à l'état 1 et lorsque le registre « TBCTR » descend en dessous de la valeur « CMPA », le signal passe à l'état 0.

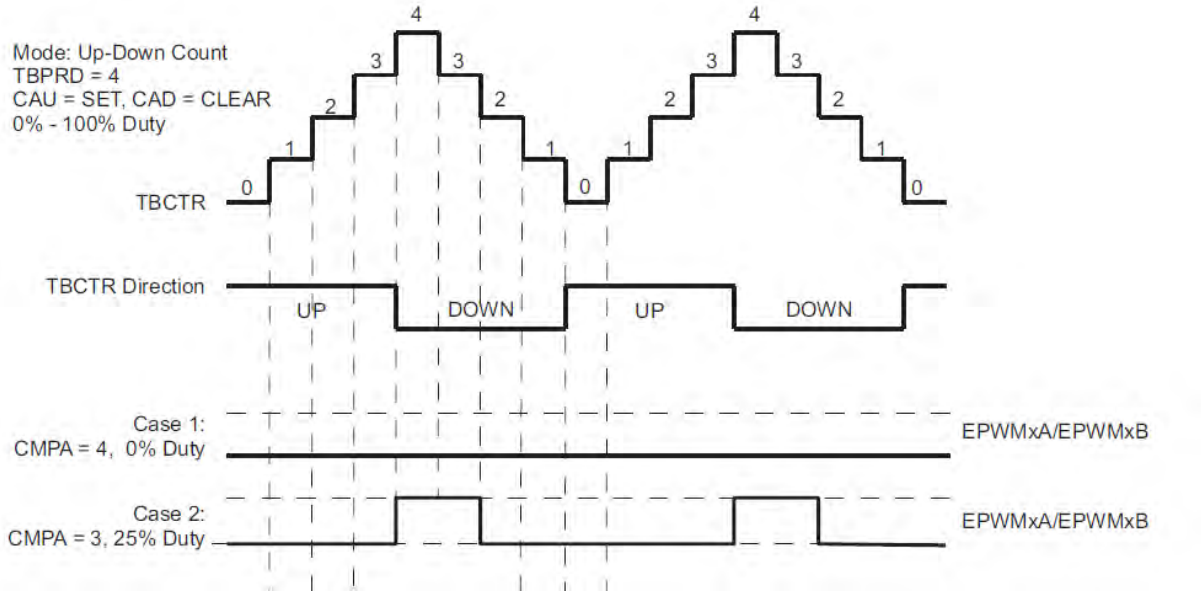


Figure 31: Fonctionnement du module PWM

Sécurité

Lorsqu'une alarme apparaît, celle-ci va rapidement couper la commande afin de mettre le module en sécurité. Ces différents signaux sont :

Signaux Digitaux :

- Overload sur le premier interrupteur
- Overload sur le deuxième interrupteur

Signaux Analogiques

- Courant de sortie trop important
- Tensions sur le bus DC trop grande
- Tension d'alimentation 24Vdc trop basse
- Température des interrupteurs trop élevée

Les valeurs de comparaison sont programmables, celles-ci peuvent être modifiées dans le programme.

```
adc.i_lim    = 20 ;
adc.u_24_lim = 22 ;
adc.vcc_lim  = 400;
adc.vee_lim  = 400;
adc.t_lim    = 90;
```

Le bloc de gestion des erreurs comprend les variables :

- *Bool* Err_Appear : Une erreur est apparue
- *Bool* Err_Disappear : Toutes les erreurs ont disparu
- *Bool* Err_set : Erreurs non quittancées
- *Integer* erreur : Chaque bit correspond à une erreur précise

  - Bit 0 : non utilisé
  - Bit 1 : Erreur sur le courant (ERR_I)
  - Bit 2 : Erreur sur le bus DC (ERR_VCC)
  - Bit 3 : Erreur sur le bus DC (ERR_VEE)
  - Bit 4 : Erreur sur la tension 24V (ERR_U_24)
  - Bit 5 : Erreur overload 1 (ERR_OL_1)
  - Bit 6 : Erreur overload 2 (ERR_OL_ 2)
  - Bit 7 : Erreur sur la température (ERR_T)
  - …

Lorsqu'une erreur apparait, la variable « erreur » est mise à jour et le bit « Err_Appear » est mis à 1, une fois que le bit « Err_set » est mis à 1, le bit « Err_Appear » est mis à 0. Lorsque toutes les erreurs ont disparu « erreur = 0 », le bit « Err_Disappear » est mis à 1. À ce moment-là, la quittance peut être faite, ce qui va mettre à 0 la variable « Err_set ».

Mise à part le reset général « PWRON_RST », aucun bouton de quittance des erreurs a été prévu. Et il est important de ne pas réenclencher le module sans quittance manuelle. Celle-ci sera effectuée par l'entrée « CONFIG4 ». Dans un prochain modèle, il serait judicieux d'ajouter un bouton pour quittancer les erreurs ou alors via l'interface de contrôle.

Contrôle

La régulation du ventilateur est une régulation à 5 niveaux.

- 0% : Température ≤ 30°C
- 25% : 30°C ≤ Température ≤ 40°C
- 50% : 40°C ≤ Température ≤ 55°C
- 75% : 55°C ≤ Température ≤ 70°C
- 100% : Température ≥ 70°C

Le pourcentage correspond au « duty cycle » du PWM.

Le signal du tachymètre du ventilateur remonte jusqu'au processeur. Ceci a été prévu dans l'optique d'une amélioration futur et n'as pas été implémenté dans ce projet.

Communication

Un début de communication a été implémenté, le transfert de données est possible avec l'ordinateur. Une interface de contrôle simplifiée a été développée afin de pouvoir visualiser les différentes grandeurs mesurées. (Voir chapitre V.3)

2 Circuit logique programmable [CPLD]

2.1 Logiciel

« ISE Project Navigator, version finale 14.5» fourni par Xilinx, le fabriquant du composant est le logiciel utilisé pour la programmation et la compilation du programme.

« ISE iMPACT, version finale 14.5 » est le logiciel utilisé pour le transfert du programme dans la CPLD.

2.2 Matériel

La carte d'évaluation « EVAL BOARD CPLD XC2C32A » a été développée à l'HES-SO // Valais – Wallis de Sion. La figure 32 montre cette carte de test.

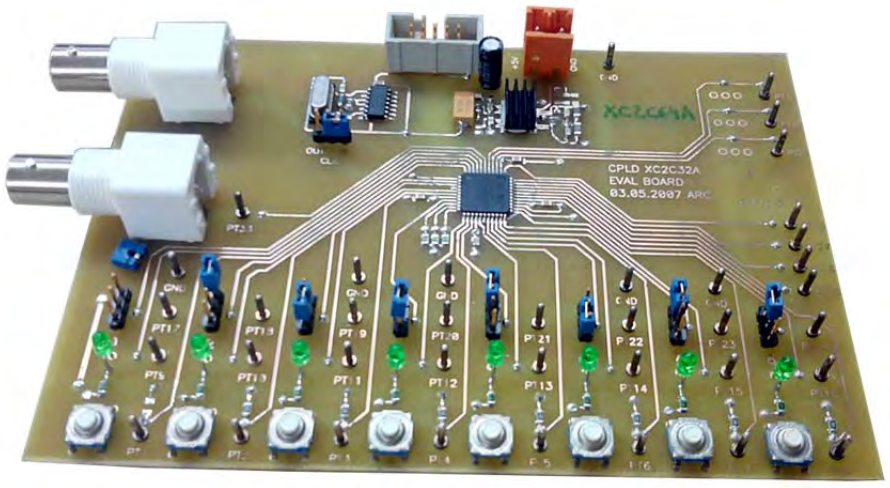


Figure 32: Carte d'évaluation pour la CPLD

La figure 33 définie le matériel de communication entre la carte d'évaluation et l'ordinateur. Il s'agit du produit « Platform Cable USB II » développé par Xilinx.

Figure 33: Module de communication « Platform Cable USB II » pour la CPLD

2.3 Programme

Le but principal de ce programme est de gérer le transfert des signaux en prenant en compte les aspects liés à la sécurité.

- Multiplexage des signaux PWM entrants
- Coupure du transfert des signaux PWM en cas d'overload, de sur-courant, de sur-tension ou de température élevée.
- Transfert des signaux de commande des LEDs.
- Transfert du signal de vitesse du ventilateur vers le microprocesseur
- Gestion des temps morts

Le code se trouve en annexe 4.

Temps mort

Une gestion des temps est effectuée à l'intérieur de la CPLD afin de garantir qu'aucun court-circuit entre les deux interrupteurs n'apparaisse. Ce temps mort peut être paramétré entre 0 et 5us. La capacité de la CPLD ne permet pas d'implémenter un compteur plus grand, ce qui limite cette plage de temps mort.

Le temps mort est défini de la manière suivante :

$$Temps\ mort = \frac{1}{F_{CLK}} * paramètre = \frac{1}{50MHz} * x$$

Le compteur étant un compteur 8 bits, le paramètre x peut valoir entre 0 et 255. Il doit être inséré au format hexadécimal [0 ; FF].

Par défaut, ce temps mort est défini à 1µs.

Développement

Le système conçu offre une grande possibilité d'amélioration et d'optimisation. Notamment trois signaux de configuration ont été réservés « CONFIG_1, CONFIG_2, CONFIG_3 ».

Cette partie du travail ne fait pas partie du projet de Bachelor. Elle pourra faire l'objet d'un prochain travail pour l'amélioration du module de conversion.

3 Interface de contrôle

3.1 Logiciel

Le logiciel de visualisation utilisé est le programme « DSP Control Center v0.7» qui a été développé par l'HES-SO // Valais – Wallis.

3.2 Programme

Une communication de base a été implémentée, celle-ci permet de visualiser les grandeurs mesurées qui sont :

- La tension du bus DC
- Le courant de sortie
- La température du refroidisseur

Et de régler la grandeur du signal PWM d'une valeur comprise entre 10% et 90%. La figure 34 représente l'interface de contrôle programmée.

Le code se trouve sur le CD-ROM.

Figure 34: Interface de contrôle créée pour la commande et la visualisation du module

Remarque :

A noter que cette interface a été réalisée dans le laps de temps mis à disposition. Elle reste donc très sommaire. L'objectif futur étant de créer une interface complète permettant la programmation du processeur et la modification des différents paramètres sans nécessiter une connexion JTAG.

Cette visualisation a été créée afin de pouvoir contrôler le dispositif et montrer le travail effectué lors de la présentation du projet.

VI. MESURES & TESTS

1 Hardware

Figure 35: Module de conversion, premier prototype♥

1.1 Protocole de mesure

Ce protocole a été établi pendant le routage du PCB ceci afin de gagner du temps lors des mesures. Toutes les étapes ont été implémentées de façon à tester de façon optimale le module. Ce protocole est disponible en annexe 9.

1.2 Matériels

Toutes les mesures ont été effectuées avec le matériel décrit dans le tableau ci-dessous. Leur « ID » sera utilisé par la suite pour les annoncer.

Tableau 10: Liste du matériel utilisé pour les mesures

ID	Appareil	Fabricant	Type	Sérial	Inv.
A1	Alimentation	Topward	TPS4000	894814	AE02/0590.07
A2	Alimentation	Agilent Tech.	N8761A	US10E5864A	-
G1	Générateur Fonc.	Hewlett	3314A	A612081	-
M1	Multimètre	Fluke	175	97020093	AE02/7622.03
M2	Multimètre	Fluke	175	97020094	AE02/7622.04
O1	Oscilloscope	Teleyne Lecroy	HDO6054	LCRY3561N17749	-
T1	Caméra Therm.	FlIR	i7	-	-

1.3 Mesures

Overload

Cette mesure a été réalisée en insérant une self à la sortie du convertisseur. En modulant la tension d'alimentation « A1 » et le duty cycle du signal de commande pwm « G1 », on peut varier le courant traversant les composants.

La figure 36 suivante représente la détection du premier overload.

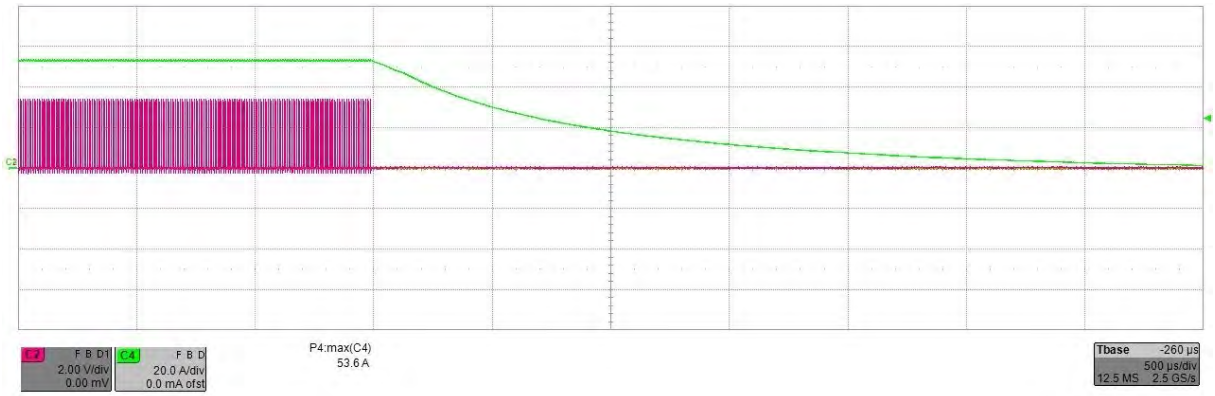


Figure 36: Détection du premier overload

La grandeur limite est de 53.6 A. Lors de la mesure, le composant avait déjà été utilisé, donc chaud. Pour cette raison, la valeur s'approche de la valeur minimale.

La figure 37 suivante représente la détection du deuxième overload.

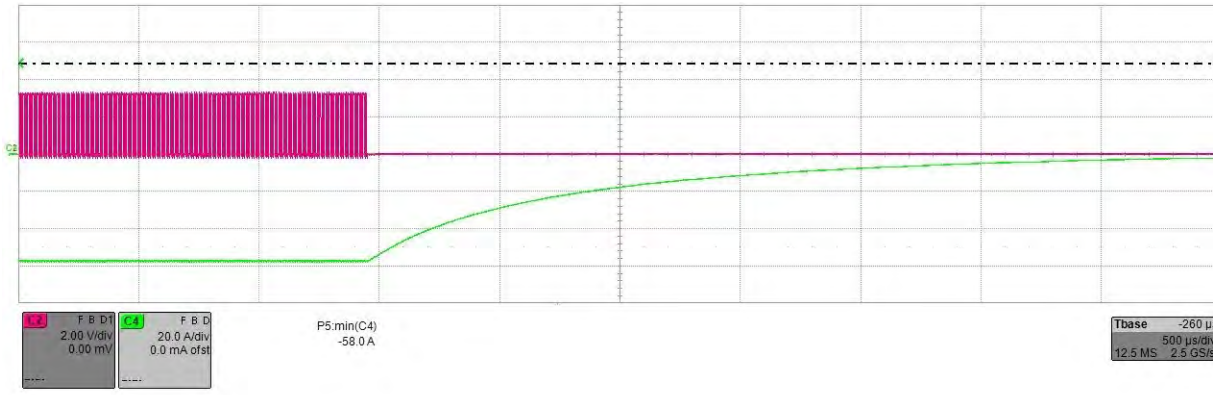


Figure 37: Détection du deuxième overload

La grandeur limite est, dans ce cas, 58A. La mesure a été faite dans la continuité de la première, le résultat coïncide parfaitement avec la grandeur attendue.

Courant de sortie

Lors du test, il s'est avéré que le courant réel était différent du courant mesuré « O1 ». Il a donc été nécessaire de faire un ajustement. La figure 38 suivante représente le courant mesuré par rapport au courant réel.

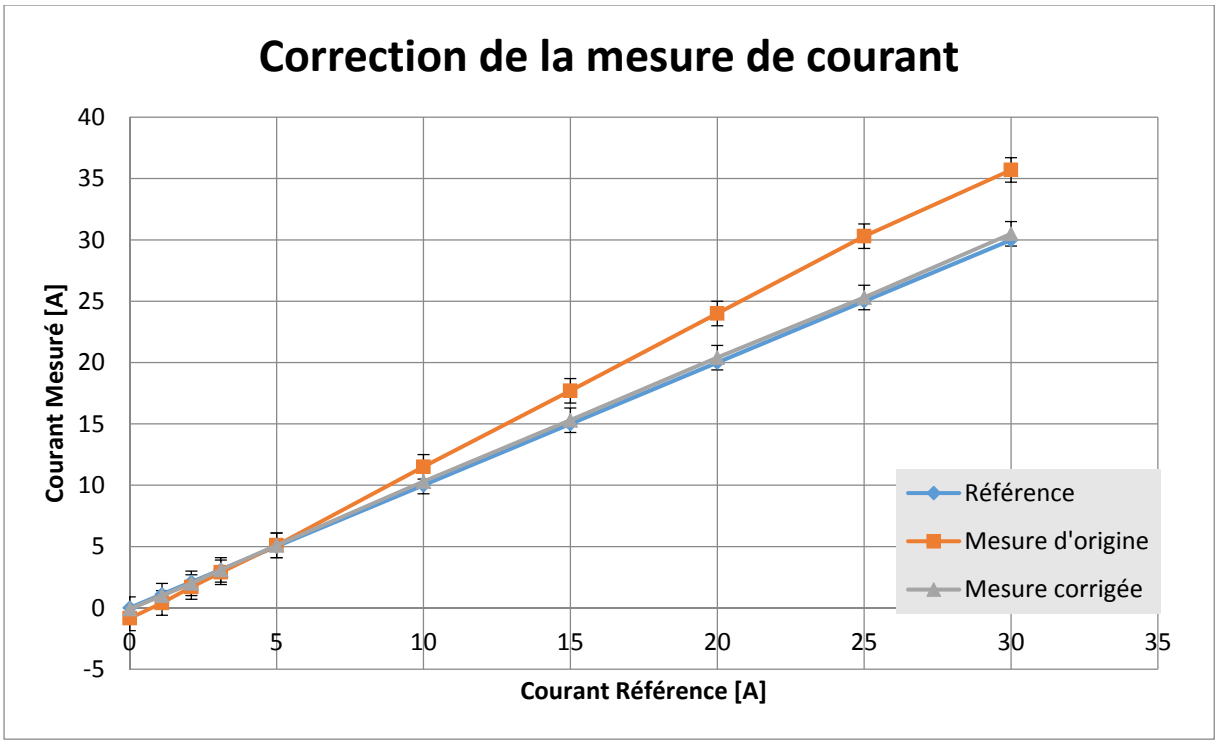


Figure 38: Correction de la mesure de courant

Premièrement, un décalage de 0.85A a été effectué afin d'obtenir un courant nul lorsque le système est en arrêt.

Ensuite un gain de 0.83 a été inséré dans l'échelle de correction afin de corriger l'erreur grandissante. La courbe corrigée correspond à la grandeur réelle.

Tension d'alimentation 24V

La figure 39 ci-dessous indique que la mesure de tension n'a pas besoin d'être corrigée, la grandeur affichée correspond à la réalité à moins 1% prêt.

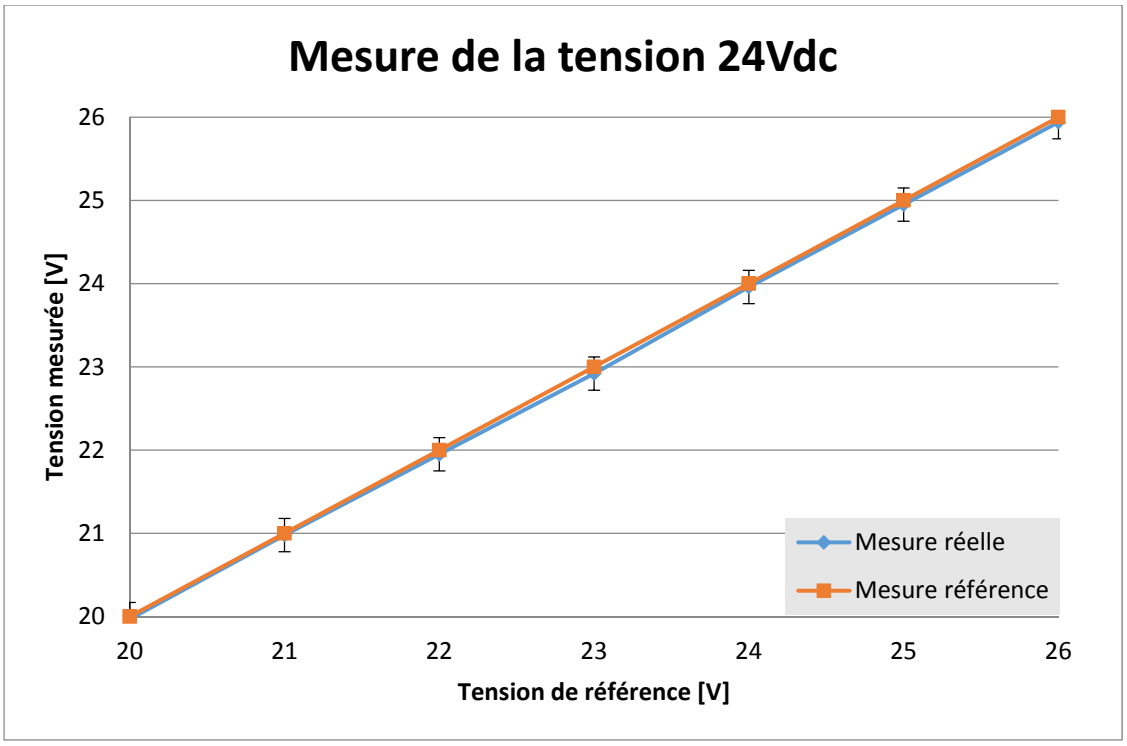


Figure 39: Mesure de la tension d'alimentation 24Vdc

Cette mesure a été réalisée avec le multimètre « M1 ».

Tension du bus DC

Les multimètres utilisés pour mesurer les tensions VCC et VEE sont respectivement l'appareil « M1 » et l'appareil « M2 ».

La référence a volontairement été prise à l'inverse pour la tension VEE afin de rendre le graphique plus lisible.

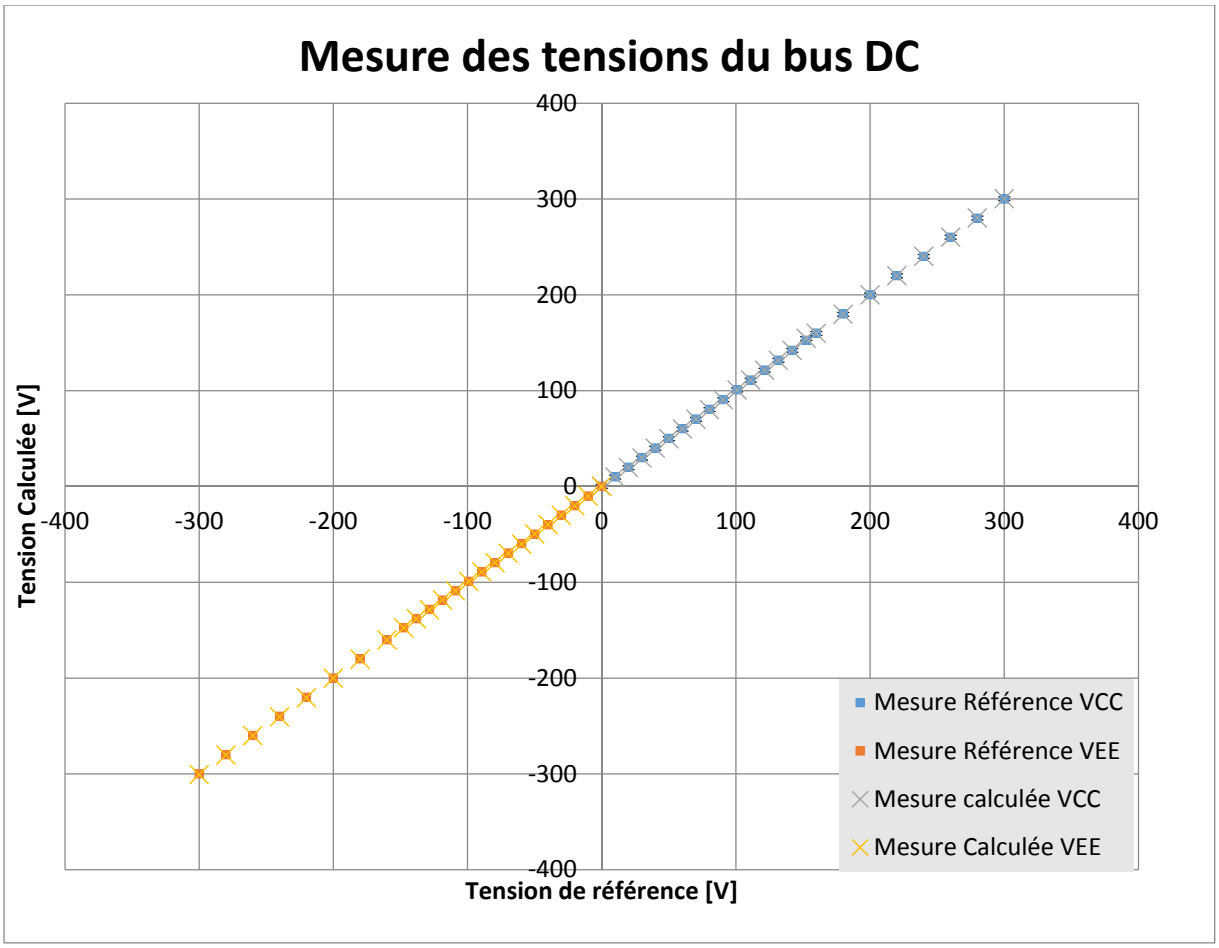


Figure 40: Mesure des tensions du bus DC

Température

2 Software

2.1 Protocole de test

Le programme a été testé selon le protocole établi (annexe 10).

VII. Essais en puissance

Ces essais ont pour but de tester les différents types de conversion et de vérifier le bon fonctionnement de celles-ci. Ces tests sont réalisés à de faibles puissances de l'ordre de 400W à 1000W.

1 Matériels

Le tableau 11 décrit le matériel utilisé pour les essais en puissance.

Tableau 11: Liste du matériel utilisé pour les essais en puissance

ID	Appareil	Fabricant	Type	Sérial	Inv.
A1	Alimentation	TDK Lambda	GEN3300W	-	B103/5259-2
O1	Oscilloscope	Teleyne Lecroy	HDO6054	LCRY3561N17749	-
	Sonde diff. « Ue »	Lecroy	HDV3106	-	AE02/7705.01
	Sonde diff. « Us »	Lecroy	HDV3106	-	AE02/705.03
	Sonde courant « Ie »	Lecroy	CP030	0314	-
	Sonde courant « Is »	Lecroy	CP030	0322	-
R1	RCL Meter	Philipps	PM6303	845206303001	AE02/3039.01
R2	Résistances	-	-	-	AE02/609.03

2 Abaisseur de tension

2.1 Montage

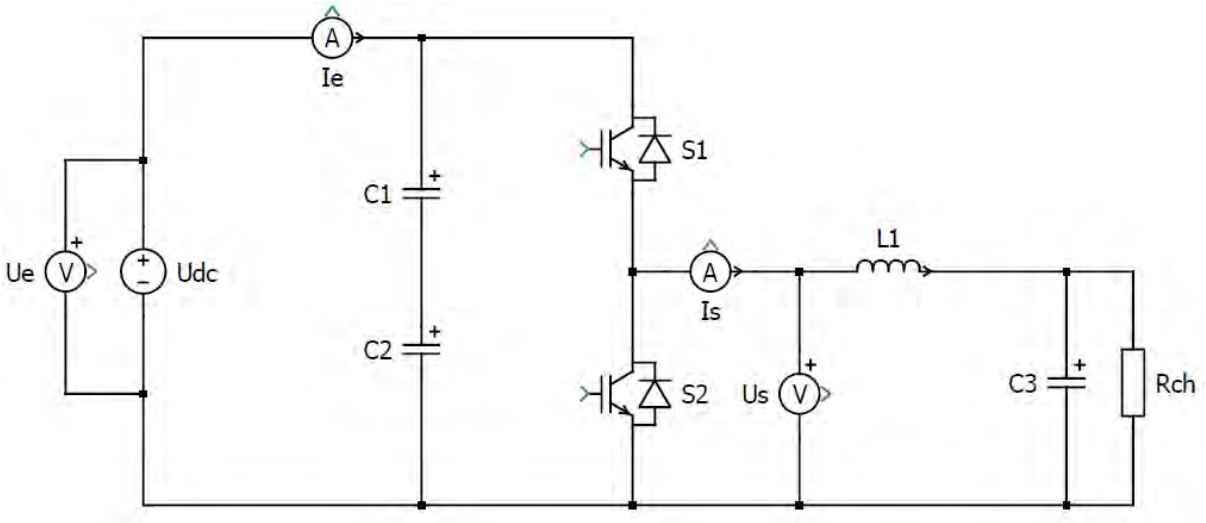


Figure 41: Montage du convertisseur abaisseur de tension pour essais avec PWM fixe

La résistance de charge a été fixée à 62Ω. L'inductance peut être calculée par l'expression suivante :

$$L_1 = \frac{U_l * d_t}{d_I}$$

La grandeur de l'inductance a été mesurée avec l'appareil « R1 » défini au tableau 11. Elle a été mesurée à 1.5mH.

2.2 Simulations

Une simulation du convertisseur en mode abaisseur de tension a été réalisée, les paramètres utilisés pour cette visualisation se trouvent en annexe 42. Ces paramètres ont été sélectionnés afin de représenter au mieux le circuit existant.

Les figures suivantes représentent cette simulation avec un PWM à 50% et pourront donc être comparées avec la première mesure effectuée au point suivant.

- Les courbes jaunes représentent respectivement la tension et le courant d'entrée.
- Les courbes roses représentent respectivement la tension et le courant de sortie avant le filtre LC.
- Les courbes bleues représentent respectivement la tension et le courant de sortie après le filtre LC.

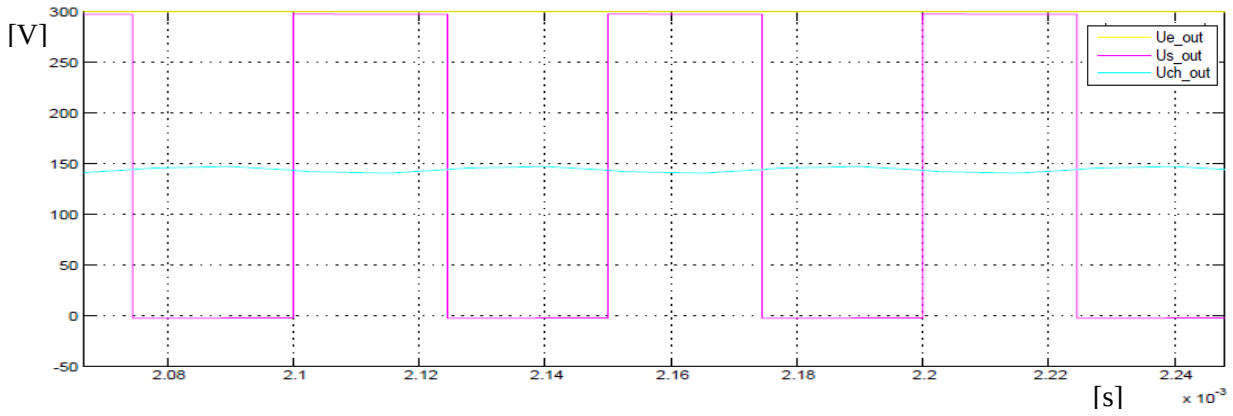


Figure 42: Simulation de la tension pour le convertisseur abaisseur de tension

La tension moyenne obtenue est de 144.9V.

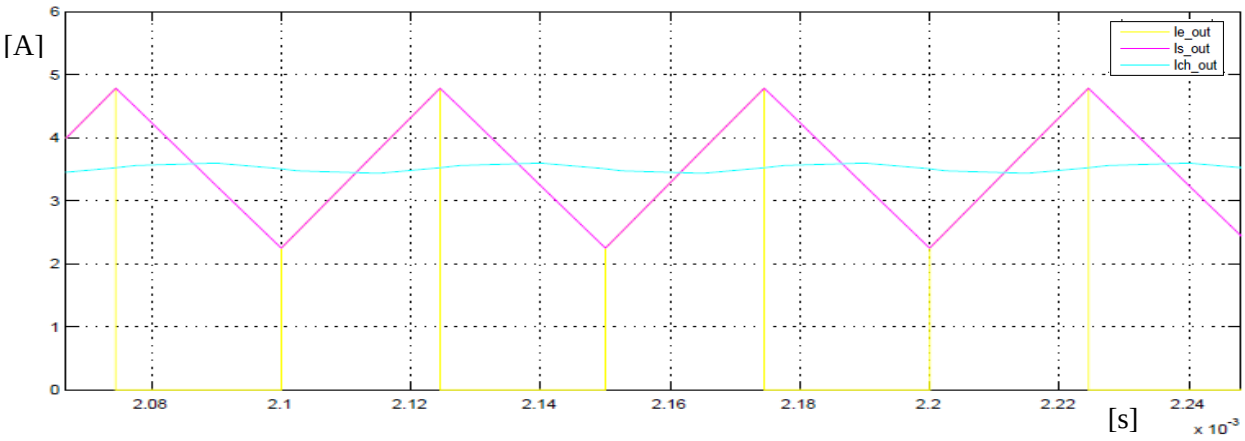


Figure 43: Simulation du courant pour le convertisseur abaisseur en tension

Le courant moyen de sortie vaut 3.506A et les ondulations sont de 2.54A.

2.3 Mesures

1ère Mesure

La première mesure a été réalisée avec une tension d'alimentation de 300V et un rapport cyclique de 50%.

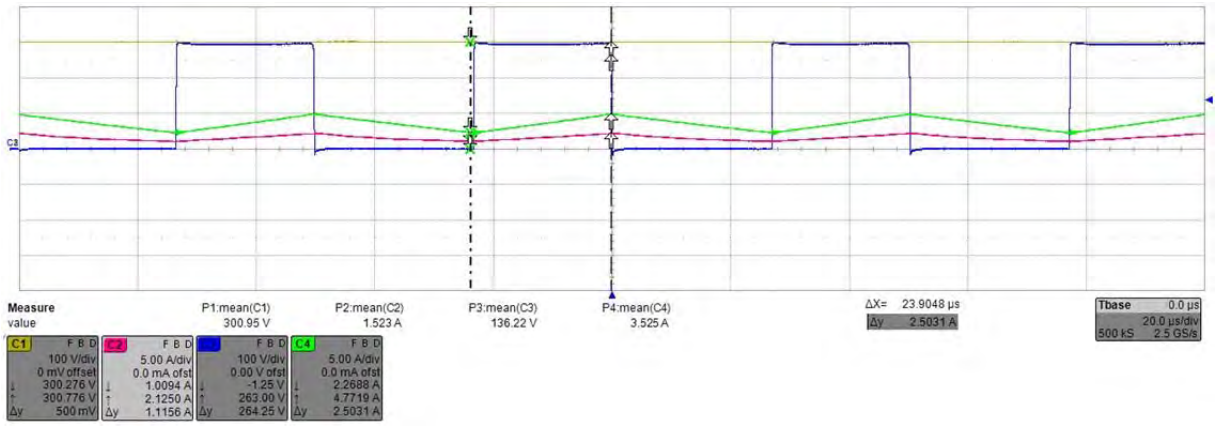


Figure 44: Convertisseur abaisseur de tension avec rapport cyclique à 50%

La figure 44 ci-dessus est le résultat obtenu avec un rapport cyclique fixe, ce qui équivaut à un convertisseur buck (abaisseur de tension).

La tension moyenne en sortie du convertisseur vaut 140V. La différence avec le résultat attendu (150V) est due principalement, aux temps morts du signal de commande PWM. Ceci afin d'éviter de court-circuiter la tension d'entrée.

L'ondulation de courant vaut 2.54A, cette valeur est définie par l'inductance à la sortie du convertisseur.

$$U_l = L * \frac{d_I}{d_t} \rightarrow d_I = \frac{U_l * d_t}{L} = \frac{140 * 24e-6}{1.5e-3} = 2.24A$$

En admettant une erreur de 15%, le calcul théorique correspond à la mesure. L'erreur est réparti selon les proportions suivantes.

- 5% mesure de la tension
- 5% mesure de temps
- 5% grandeur de l'inductance

2ème Mesure

Pour cette deuxième mesure, le rapport cyclique a été augmenté à 75%.

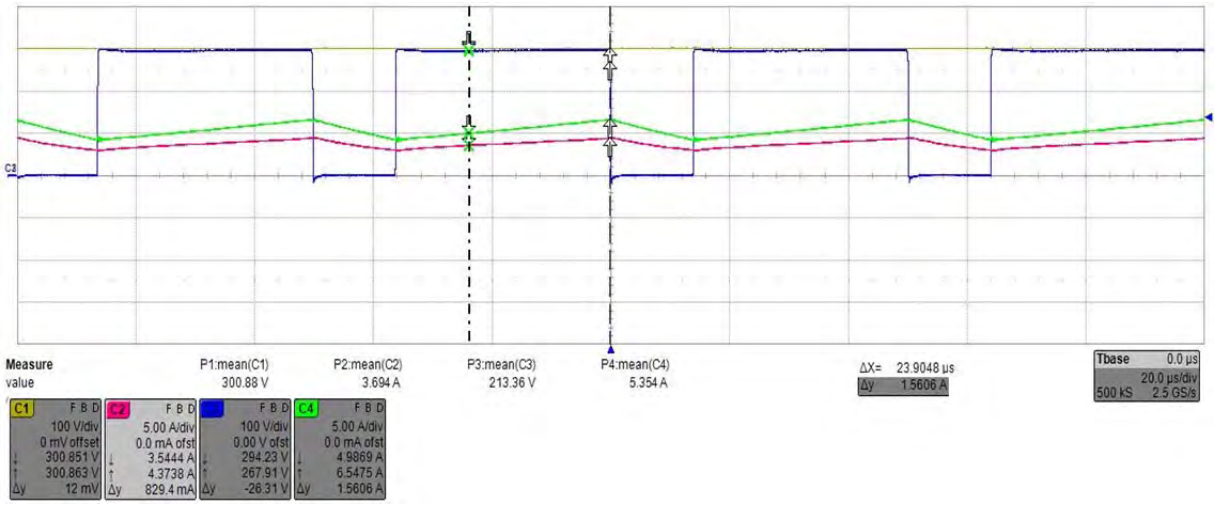


Figure 45: Convertisseur abaisseur de tension avec rapport cyclique à 75%

On constate bien que la tension moyenne à la sortie du convertisseur a augmenté. Elle vaut désormais 214V. Dans un cas idéal, celle-ci vaudrait 225V.

2.4 Commentaires

Les résultats obtenus entre la simulation et l'essai pratique pour une pwm de 50% est intéressant. Le tableau 12 ci-dessous compare les différentes grandeurs.

Tableau 12: Comparaison entre la simulation et la mesure pour l'essai buck

	Uin [V]	Uout [V]	Iin [A]	Iout [A]	ΔIout [A]
Simulation	300	144.9	1.72	3.51	2.54
mesure	301	136.22	1.52	3.525	2.50
Différence [%]	0.33	6	11.63	0.42	1.57

On peut donc en conclure que le résultat mesuré s'approche de la simulation effectuée. Soit :

- Les paramètres de simulation ont été correctement définis
- Le convertisseur fonctionne correctement

3 Élévateur de tension

3.1 Montage

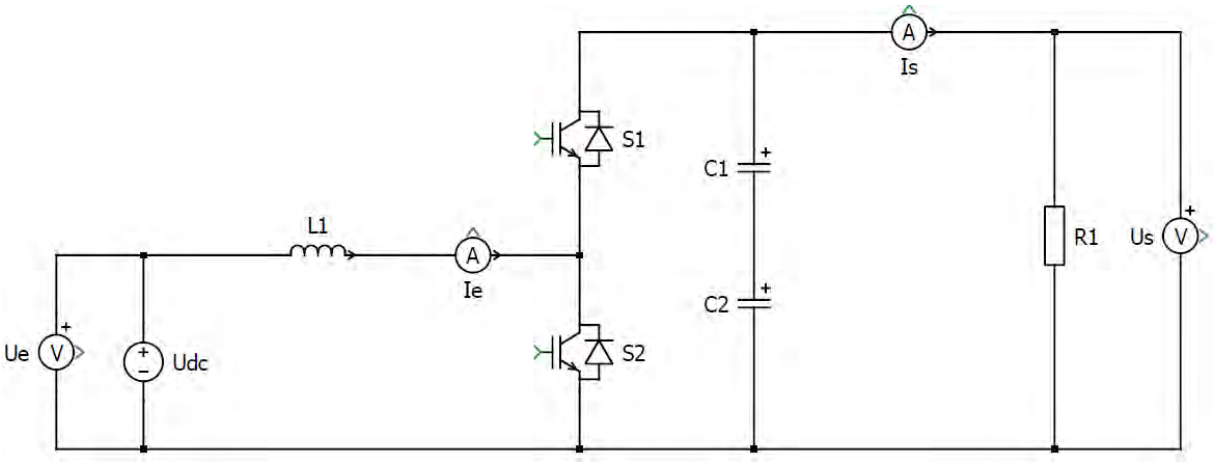


Figure 46: Montage du convertisseur élévateur de tension pour essais avec PWM fixe

La résistance de charge et l'inductance sont identiques à celles utilisées pour le montage abaisseur de tension.

3.2 Simulations

Une simulation du convertisseur en mode élévateur de tension a été réalisée, les paramètres utilisés pour cette visualisation se trouvent en annexe 11. Ces paramètres ont été sélectionnés afin de représenter au mieux le circuit existant.

Les figures suivantes représentent cette simulation avec un PWM à 15% et pourront donc être comparées avec la dernière mesure effectuée au point suivant.

- Les courbes jaunes représentent respectivement la tension et le courant d'entrée.
- Les courbes roses représentent respectivement la tension et le courant de sortie avant le filtre LC.
- Les courbes bleues représentent respectivement la tension et le courant de sortie après le filtre LC.

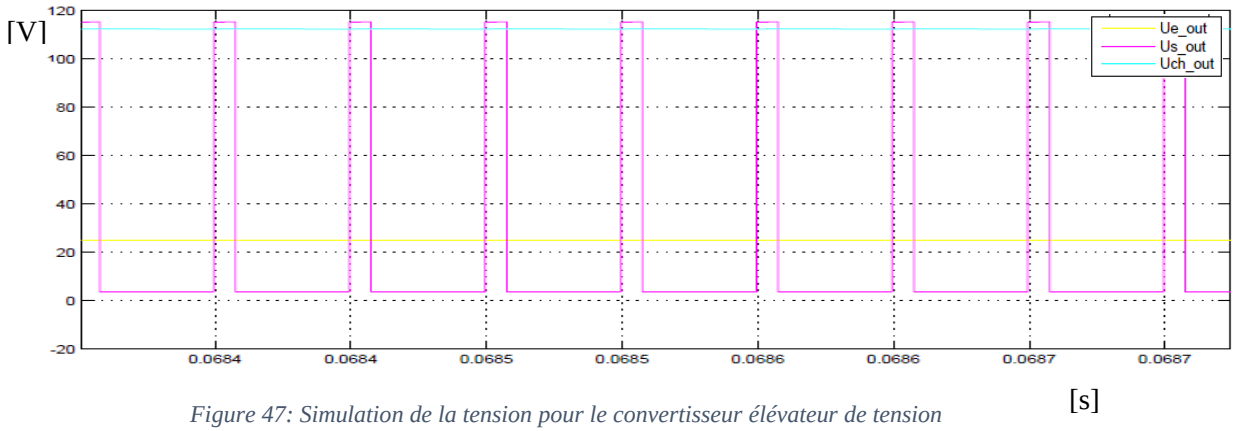


Figure 47: Simulation de la tension pour le convertisseur élévateur de tension

La tension d'entrée a été défini à 25V et la tension de sortie a vaut 110.4V.

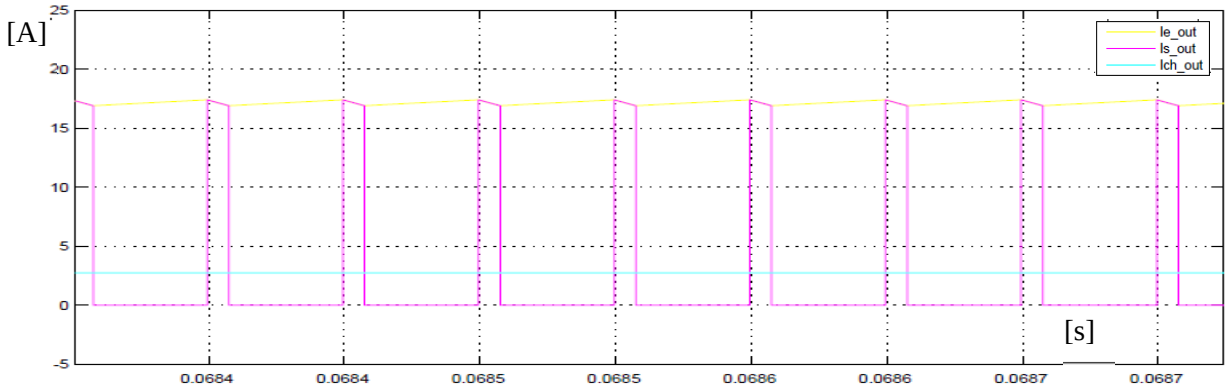


Figure 48: Simulation du courant pour le simulateur élévateur de tension

Le courant d'entrée a été simulé à 17.37A et le courant de sortie à 2.78A. L'ondulation de courant quant à elle vaut 0.5A.

3.3 Mesures

1ère Mesure

La première mesure a été réalisée avec un rapport cyclique de 50% avec une tension d'entrée de 25V.

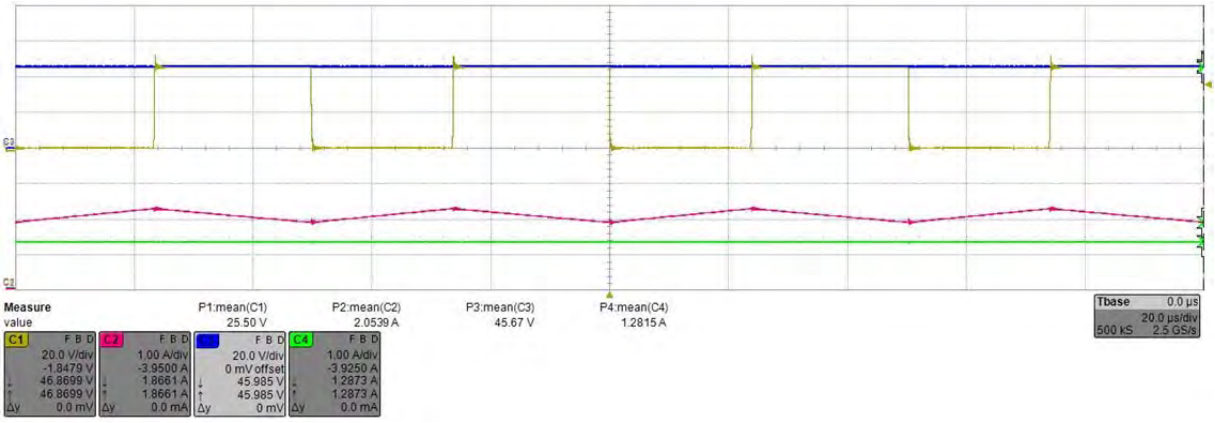


Figure 49: Convertisseur élévateur de tension avec le rapport cyclique à 50%

La tension moyenne à la sortie du convertisseur est bien supérieure à celle à l'entrée.

2ème Mesure

Le rapport cyclique a été ajusté à 25%.

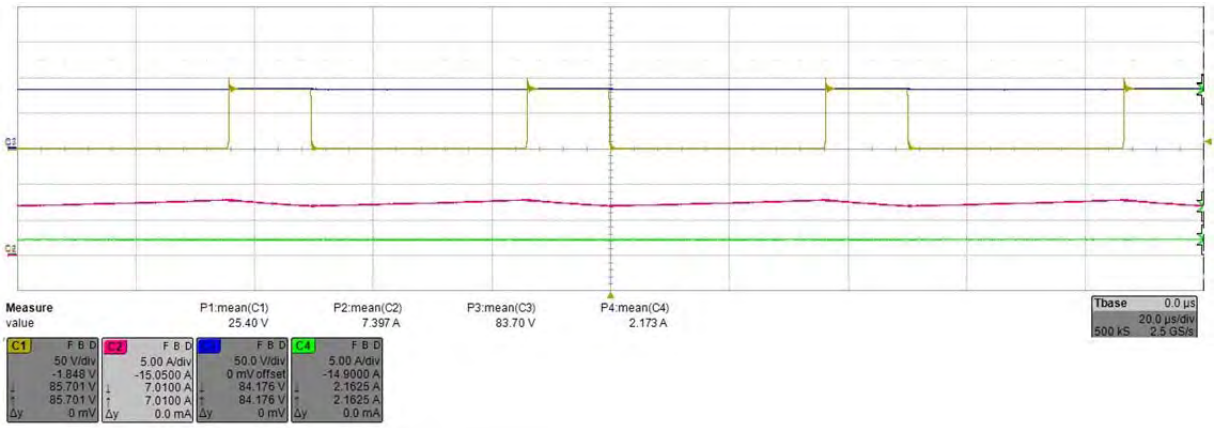


Figure 50: Convertisseur élévateur de tension avec le rapport cyclique à 25%

3ème Mesure

Le rapport cyclique a été abaissé à 15%

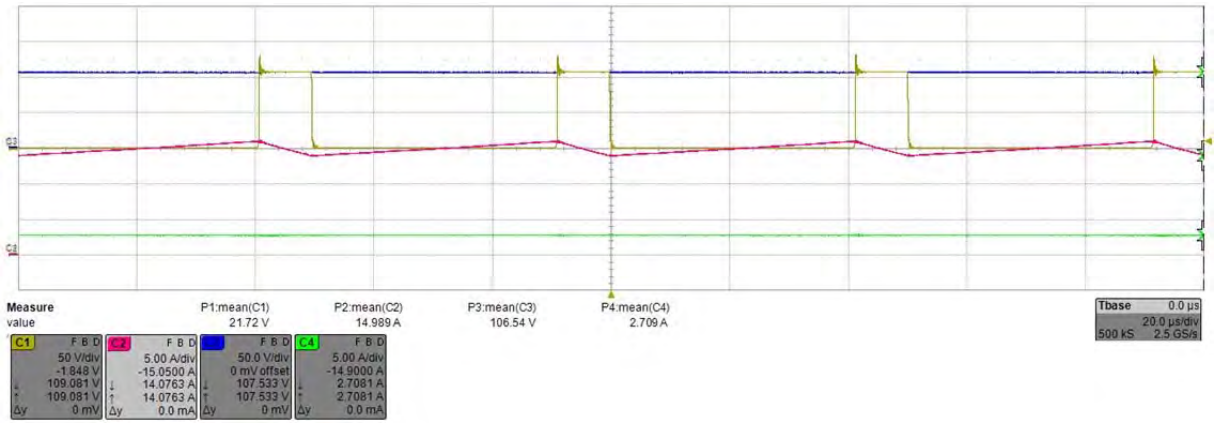


Figure 51: Convertisseur élévateur de tension avec le rapport cyclique à 15%

3.4 Commentaires

Tableau 13: Comparaison entre la simulation et la mesure pour le convertisseur boost

	Uin [V]	Uout [V]	Iin [A]	Iout [A]	ΔIout [A]
Simulation	25	110.4	17.37	2.78	0.5
mesure	21.72	106.5	14.99	2.7	-
Différence [%]	13.12	3.53	14.2	2.9	-

Malgré le fait que les résultats sont moins précis que ceux obtenus pour le fonctionnement en mode abaisseur de tension, la différence mesurée est inférieure à 15%. Ces différences peuvent avoir plusieurs causes :

- Les paramètres de simulation ne sont pas les plus correctes.
- Des effets non prévus n'ont pas été pris en compte lors de la simulation

4 Convertisseur DC/AC

4.1 Montage

Pour cet essai, le montage est identique à l'essai abaisseur de tension, comme le démontre la figure 52.

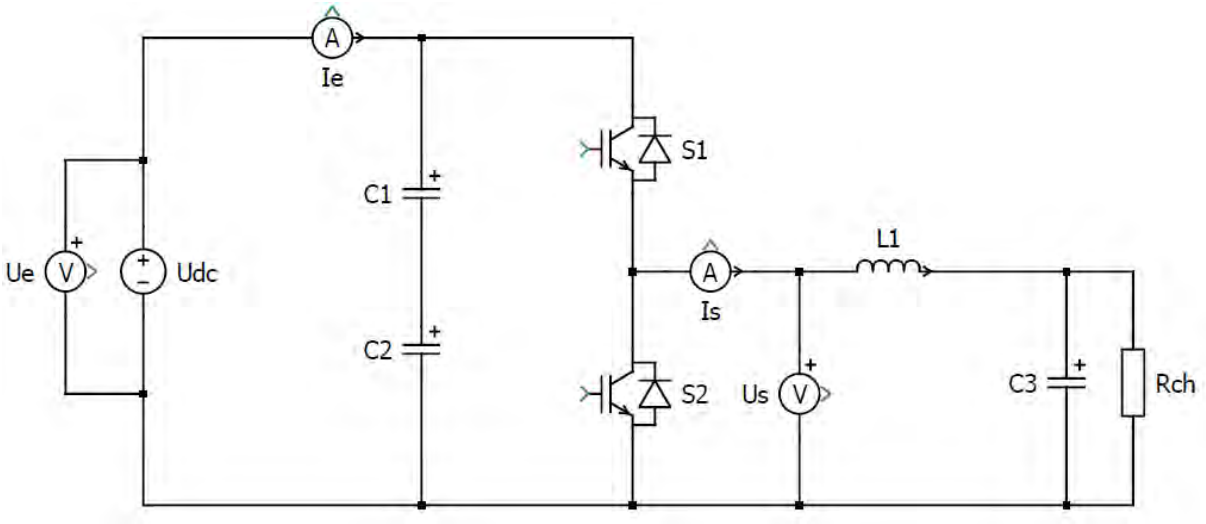


Figure 52: Montage du convertisseur onduleur avec PWM variable

4.2 Simulation

Une simulation du convertisseur en mode onduleur a été réalisée, les paramètres utilisés pour cette visualisation se trouvent en annexe 11. Ces paramètres ont été sélectionnés afin de représenter au mieux le circuit existant.

Les figures suivantes représentent cette simulation avec un PWM variant sinusoïdalement avec une fréquence de 50Hz. et pourront donc être comparées avec la mesure effectuée au point suivant.

- Les courbes jaunes représentent respectivement la tension et le courant d'entrée.
- Les courbes roses représentent respectivement la tension et le courant de sortie avant le filtre LC.
- Les courbes bleues représentent respectivement la tension et le courant de sortie après le filtre LC.

Sur les deux prochaines figures (53 et 54), on peut observer la variation de la tension et du courant dans le temps.

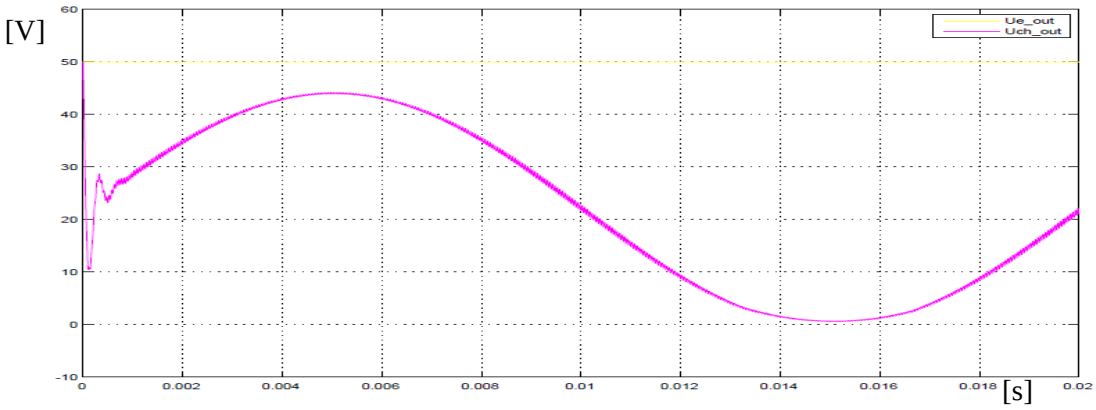


Figure 53: Tension du convertisseur avec PWM sinusoïdale

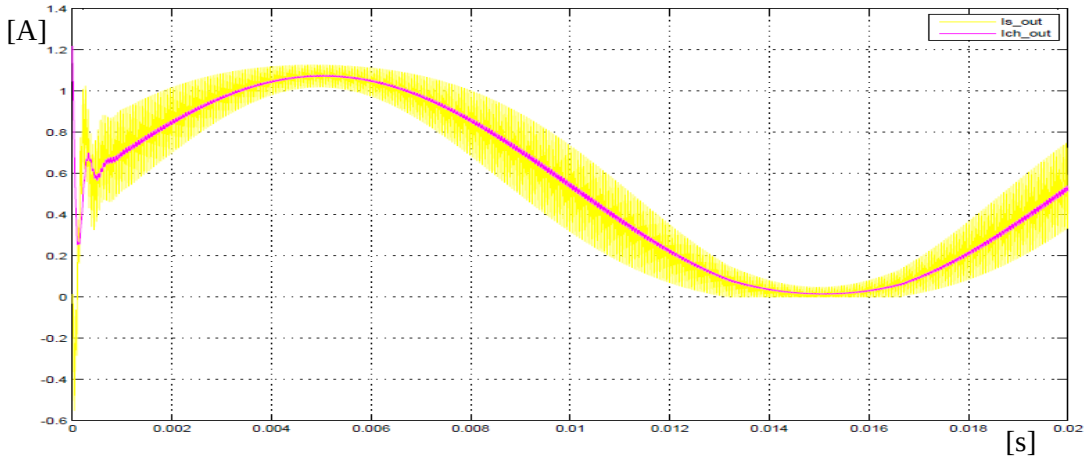


Figure 54: Courant du convertisseur avec PWM sinusoïdale

Les prochaines figures vont représenter ces essais pour des points précis de fonctionnement qui sont :

- Point haut du sinus (PWM max)
- Point bas du sinus (PWM min)
- Point milieu du sinus (PWM 50%)

PWM MAX

Sur les deux images ci-dessous, le temps est suffisamment court pour estimer que la PWM est fixe, c'est-à-dire qu'il s'agit d'un convertisseur « BUCK ». Il y aura donc le même effet que pour l'essai en abaisseur de tension.

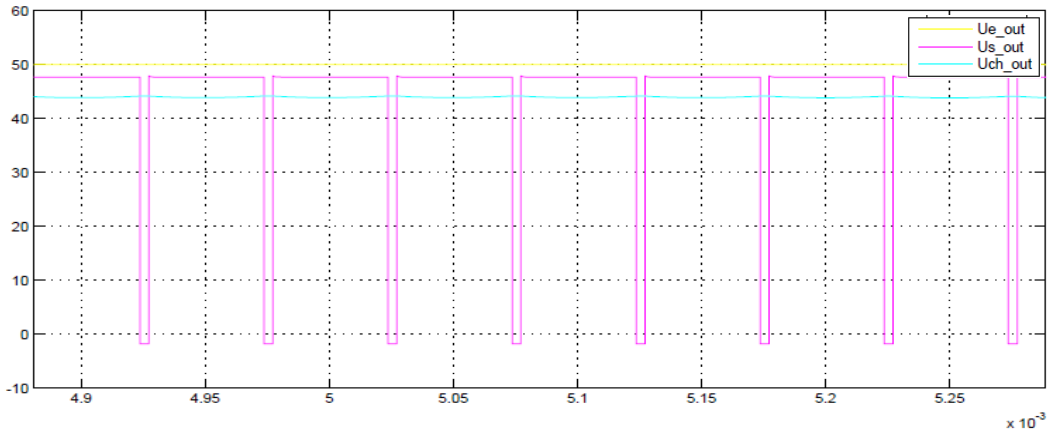


Figure 55: Tension du convertisseur quand la PWM est maximale

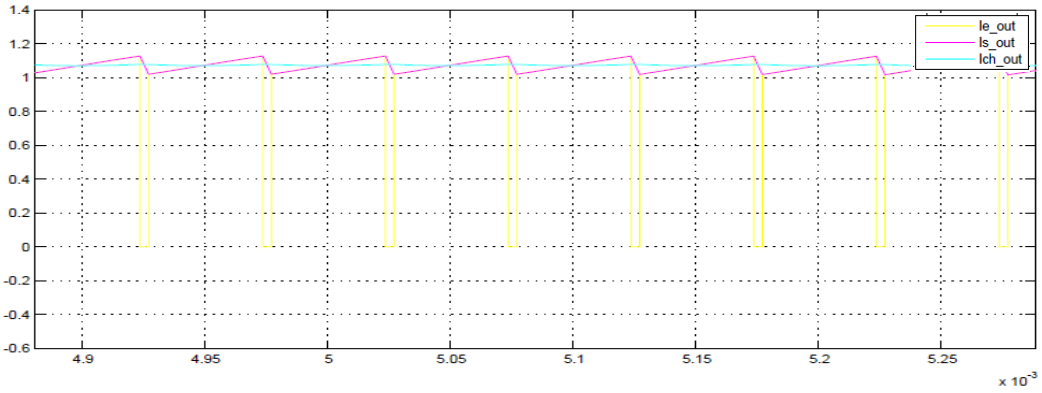


Figure 56: Courant du convertisseur quand la PWM est maximale

PWM 50%

Avec une PWM à 50%, la tension moyenne à cet instant vaut la moitié de la tension imposée aux bornes du convertisseur.

À noter que si le convertisseur était alimenté en ±VCC, cette tension moyenne vaudrait 0.

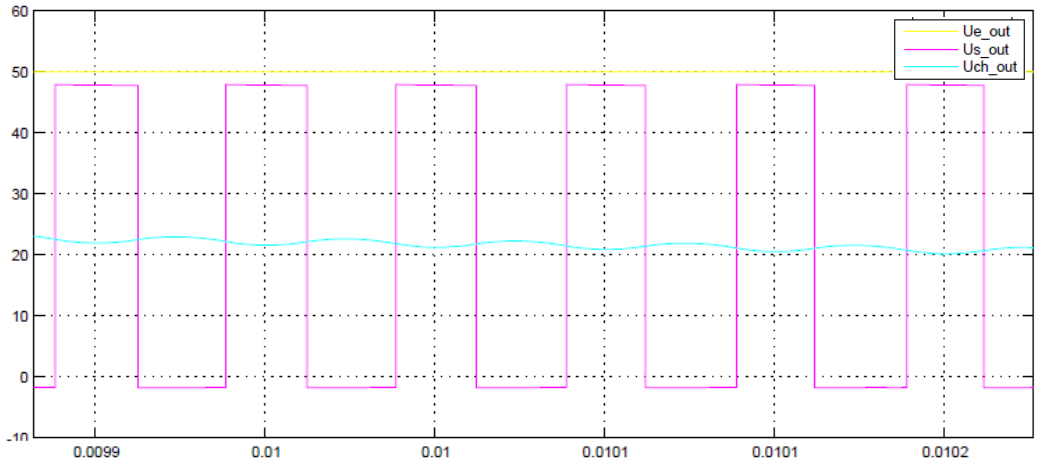


Figure 57: Tension du convertisseur quand la PWM est moyenne

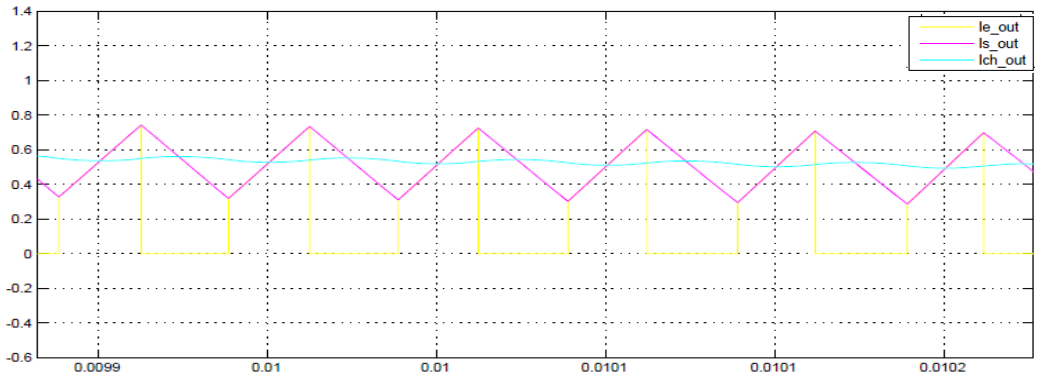


Figure 58: Courant du convertisseur quand la PWM est moyenne

PWM MIN

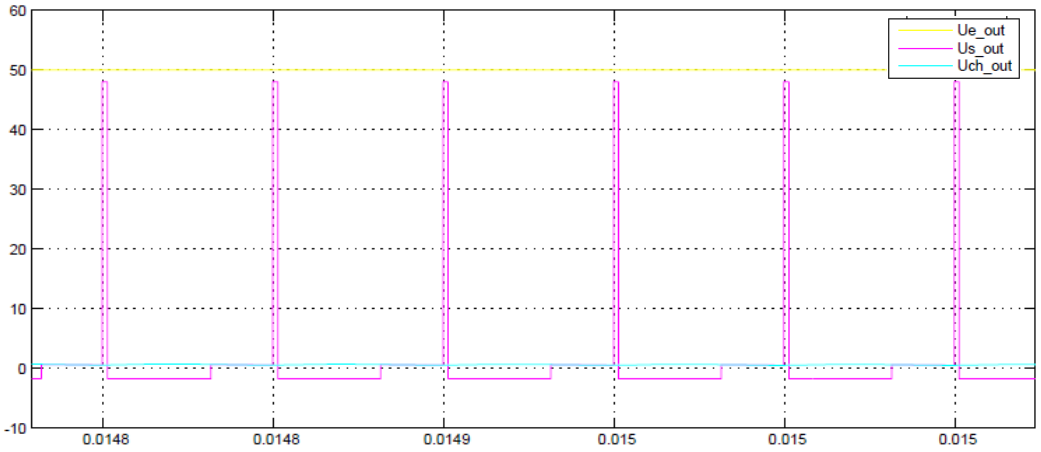


Figure 59: Tension du convertisseur quand la PWM est minimale

On constate que lorsque le courant devient nul, la diode ne conduit plus. La tension « Us_out » vaut donc 0. Cette tension vaut 1.7V lorsque le courant diminue.

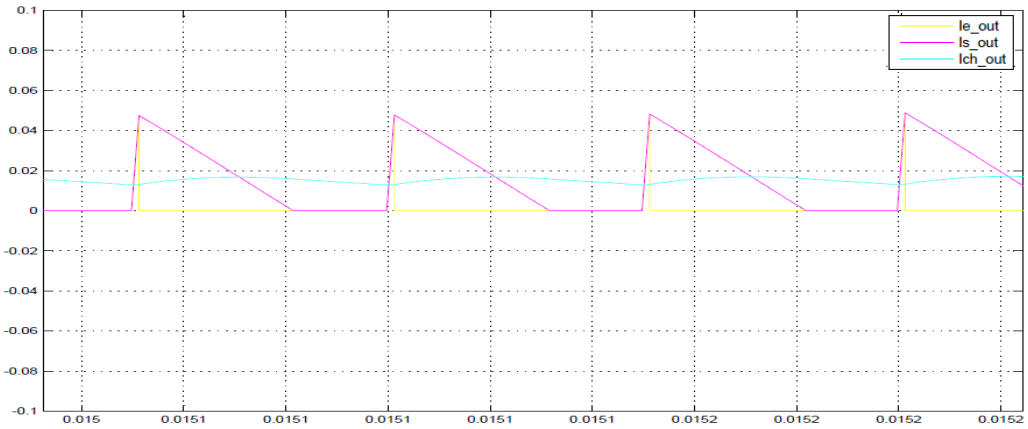


Figure 60: Courant du convertisseur quand la PWM est minimale

En lien avec la figure 60 ci-dessus, lorsque le courant devient nul, la diode ne conduit plus et donc le courant ne circule plus.

4.3 Mesure

Cet essai en onduleur permet d'observer que le courant est en phase avec la tension (charge résistive) et que les ondulations de courant sont maximales pour un rapport cyclique de 50%.

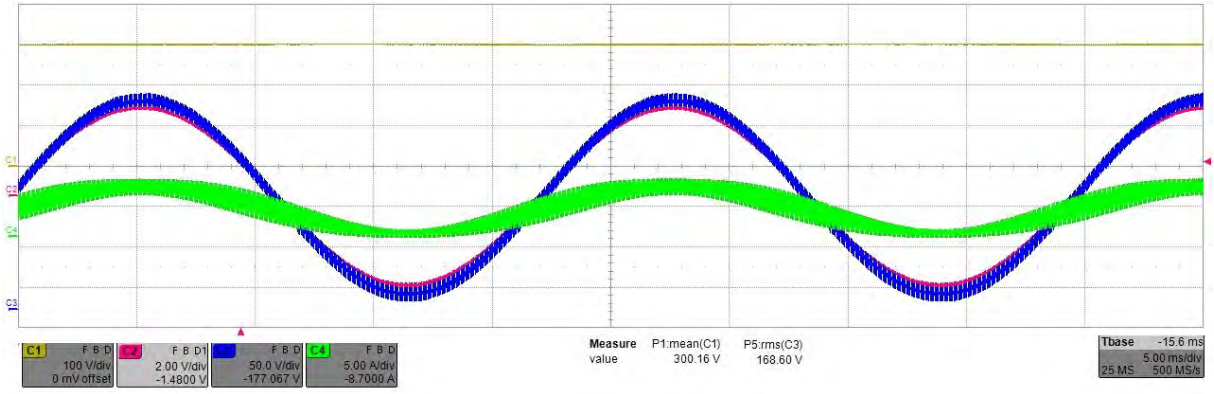


Figure 61: Essai du convertisseur en onduleur de tension

4.4 Commentaires

En comparant les figures 60 et 61 (simulation et mesure), on constate que les formes de courbes sont semblables. Lorsque le sinus est à son point minimal, le courant passe à zéro et la diode se bloque ce qui diminue l'ondulation de courant, comme on peut le voir sur la figure 61 ci-dessus.

Les éléments de comparaison pour cet essai sont limités, la décision a été prise de programmer la communication sérielle du processeur afin de pouvoir visualiser les grandeurs sur une interface de contrôle plutôt que de continuer des tests qui avaient déjà permis de valider le fonctionnement du module.

VIII. PROCHAINES ÉTAPES

Essais à « haute » puissance (<12kW)

Il peut être intéressant de tester le système aux environs de la puissance maximale, afin de vérifier sa réaction.

Essais continu sur une longue période

Tester le système dans un emploi industriel sur une période de plusieurs jours, voire plusieurs semaines non interrompu. Ceci permettrait de tester la robustesse du convertisseur.

Améliorer l'interface de contrôle

Créer une interface de contrôle complète permettant la modification des paramètres, la programmation du processeur ainsi que des sauvegardes continues des différentes grandeurs.

Utiliser une entrée comparaison pour la mesure du courant

Il existe une entrée comparaison sur le processeur, cette entrée permet la comparaison de plusieurs signaux analogique. Il peut être intéressant d'approfondir la recherche sur le sujet.

Essais de plusieurs modules ensemble (AC/AC, triphasé, etc…)

Utilisation des modules pour effectuer une conversion alternative-alternative ou même une conversion triphasée

Réduire les dimensions du module

Réduire la taille en regroupant les composants serait un point à étudier.

IX. CONCLUSION

Dans un premier temps, il est important de préciser que les objectifs du projet ont été atteints. C'est-à-dire avoir un module fonctionnel avec une sécurité interne autonome et la possibilité de commander les interrupteurs via le microprocesseur.

De plus, les premiers essais effectués sont prometteurs, les résultats obtenus sont conformes aux résultats attendus. Ceux-ci ont été réalisés à faible puissance (1kW) et sur une courte durée. Il serait intéressant d'augmenter la durée et la puissance des essais lors des prochains tests.

Tous les composants choisis fonctionnent correctement selon leur utilisation. Le changement de l'alimentation 1.2V de Micro-chip pour TI s'est révélé judicieux.

Le type de connecteur pour le JTAG pourrait être modifié dans un prochain prototype afin d'éviter l'utilisation d'un adaptateur.

La correction de la schématique a déjà été effectuée, cependant les cartes déjà fabriquées comprennent une erreur pour la tension de référence de 1.5V, il sera nécessaire de leur ajouter le patch de correction.

Les prochaines étapes du projet ont pu être définies. L'amélioration de l'interface de contrôle est un des points important avec les essais en puissance.

X. RÉFÉRENCES

Agilent Technologies. 2002. *HFBR-0508 Series.* 2002. Datasheet. 5988-5674EN.

Analog Device. 2009. *AD8648 rev.C.* 2009. Datasheet.

Barrade, Philippe. 2006. *Électronique de puissance Méthodologie et convertisseurs élémentaires.* Lausanne : Presses polytechniques et universitaires romandes, 2006. 2-88074-566-7.

Bonvin, Nicolas. 2015. *Convertisseur DC/DC 3KW pour photovoltaïque.* SION : HES-SO VALAIS/WALLIS, 2015. Travail de Semestre.

—. 2015. *Convertisseur DC/DC 3KW pour photovoltaïque.* Sion : HES-SO VALAIS/WALLIS, 2015. Travail de Bachelor.

Cincon. 2014. *EC4A Series Application note.* 2014. Datasheet.

Coulinge, Emilien. 2015. *Power electronics building block - design.* Lausanne : EPFL, 2015. Master's thesis.

Dubosson, Maxime. 2016. *Module de conversion statique.* Sion : HES-SO VALAIS/WALLIS, 2016. Travail de Semestre.

Fairchild. 2008. *FOD3180 Rev. 1.0.6.* 2008. Datasheet.

Infineon. 2013. *IKW25N120T2 Rev. 2.2.* 2013. Datasheet. IFAG IPC TD VLS.

—. 2000. *SFH6156.* 2000. Datasheet.

Lem. 2014. *Current Transducer LTSR 25-NP.* 2014. Datasheet. 110214/10.

Murata. 2014. *MEF1 Series.* 2014. Datasheet. KDC_MEF1.CO2.

—. 2014. *MGJ2 Series.* 2014. Datasheet. KDC_MJG2_CO4.

Page, Jessen. *Transfert de chaleur par convection.* Document de cours.

Rhopoint Components LTD. 2001. *High Load and Braking Resistors.* 2001. Datasheet.

Rubycon. *QXW Series.* Datasheet.

Sunon. 2009. *DC Brushless Fan.* 2009. Datasheet. E10800420.

Texas Instruments Incorporated. 2015. *Launchxl-F28377S User's Guide.* 2015. Datasheet. SPRUI25A.

—. 2015. *LM311 Comparator.* 2015. Datasheet. SLCS007I.

—. 2014. *LMV431 Shunt regulator.* 2014. Datasheet. SNVS041G.

—. 2014. *MAX232 driver/receiver.* 2014. Datasheet. SLLS047M.

—. 2015. *REF3030 voltage reference.* 2015. Datasheet. SBVS032G.

—. 2014. *TMS320F28377S Delfino* . 2014. Datasheet. SPRS881B.

—. 2015. *TMS320F28377S Technical Reference Manual.* 2015. Datasheet. SPRUHX5C.

—. 2015. *TPS62262 step-down converter.* 2015. Datasheet. SLVS763E.

—. 2008. *TPS767D318-Q1 voltage regulator.* 2008. Datasheet. SGLS231A.

Xilinx. 2008. *XC2C32A CoolRunner II.* 2008. Datasheet. DS310 (v2.1).

Zbären, Nikolas. 2015. *Power electronics building block - protection.* Lausanne : EPFL, 2015. Master's thesis.

XI. ANNEXES

1 Papier

[1] Journal de travail
[2] Liste du matériel
[3] Facture de fabrication du PCB
[4] Programme de la CPLD
[5] Programme de l'interface de contrôle
[6] Schématique et plan du convertisseur
[7] Schématique et plan du module ventilateur
[8] Plan de routage du module « testpoint » pour le kit Launchxl-F28377S
[9] Protocole de mesure (hardware)
[10] Protocole de test (software)
[11] Script Matlab de simulation du convertisseur
[12] Fichier Excel de dimensionnement des amplificateurs

2 CD-ROM

- Page de garde
- Donnée du travail de diplôme
- Résumé
- Rapport
- Annexes

- ALTIUM
  - Schématique du projet
- MATLAB
  - Scripts
- CCS
  - Programme du DSP
  - DSP control center
- XILINX
  - Programme de la CPLD
- FICHIERS VISIO
- EXCEL
  - Dimensionnement des amplificateurs différentiels
  - Fichier de simulation
- DATASHEET

ANNEXE 1

Semaine 1 / Jour 1 (17 mai 2016)

Activité(s)	Durée(s)
- Avancement de la schématique - Choix des alimentations	7h 1.5h

Prochain(s) Objectif(s) :

- Avancement la schématique

Semaine 1 / Jour 2 (18 mai 2016)

Activité(s)	Durée(s)
- Avancement de la schématique	8.5h

Prochain(s) Objectif(s) :

- Avancement de la schématique

Semaine 1 / Jour 3 (19 mai 2016)

Activité(s)	Durée(s)
- Avancement de la schématique - Recherche d'un ventilateur	6h 2.5h

Prochain(s) Objectif(s) :

- Terminer la schématique
- Choisir un ventilateur

Semaine 1 / Jour 4 (20 mai 2016)

Activité(s)	Durée(s)
- Avancement de la schématique - Recherche d'un ventilateur	6h 2.5h

Prochain(s) Objectif(s) :

- Contrôle et modification de la schématique
- Terminer la schématique

Remarque(s) :

- Certains composants n'étaient pas modélisés correctement et devront être modélisés

Semaine 2 / Jour 5 (23 mai 2016)

Activité(s)	Durée(s)
- Avancement et correction de la schématique	5.5h
- Demande d'informations pour un ventilateur	0.5h
- Établissement d'un planning	0.5h
- Recherche d'information pour le kit de développement F28377S	2h

Prochain(s) Objectif(s) :

- Correction de la schématique
- Liste du matériel

Semaine 2 / Jour 6 (24 mai 2016)

Activité(s)	Durée(s)
- Correction de la schématique	6h
- Liste du matériel	2.5h

Prochain(s) Objectif(s) :

- Mise en route du kit de développement
- Fournir la schématique pour le routage

Semaine 2 / Jour 7 (25 mai 2016)

Activité(s)	Durée(s)
- Finalisation de la schématique	3.5h
- Recherche des composants	1h
- Placement des composants par fonction sur le PCB	4h

Prochain(s) Objectif(s) :

- Établir une liste de matériel
- Placement des composants sur le PCB
- Écriture du rapport

Semaine 2 / Jour 8 (27 mai 2016)

Activité(s)	Durée(s)
- Placement des composants par fonction sur le PCB	4h
- Discussion avec le professeur du travail de Semestre	1h

Prochain(s) Objectif(s) :

- Ajout d'une capacité de découplage au plus près des IGBTs

Semaine 3 / Jour 9 (30 mai 2016)

Activité(s)	Durée(s)
- Établir un protocole de mesure et de test	1h
- Choix d'une capacité de découplage pour les IGBTs	1h
- Ajouts de résistances d'équilibrage sur le banc de capacités	1h
- Avancement et mise à jour du PCB	3h
- Liste du matériel manquant	3h

Prochain(s) Objectif(s) :

- Prise en main du kit de développement

Semaine 3 / Jour 10 (31 mai 2016)

Activité(s)	Durée(s)
- Écriture du rapport	5h
- Programmation de la CPLD	4h

Prochain(s) Objectif(s) :

- Programmation de la CPLD
- Commencer la prise en main du processeur
- Écriture du rapport

Semaine 3 / Jour 11 (1 juin 2016)

Activité(s)	Durée(s)
- Programmation de la CPLD	3h
- Écriture du rapport	4h
- Calcul du circuit pour les overloads	2h

Prochain(s) Objectif(s) :

- Commencer le développement du DSP (Si le programme a pu être installé)
- Écriture du rapport
- Carte annexe pour le ventilateur

Remarque(s) :

- Le DSP n'est pas disponible sur le programme CSS 5, le 6 devra être installé, ce qui va encore induire du retard sur le projet.

Semaine 3 / Jour 12 (2 juin 2016)

Activité(s)	Durée(s)
- Calcul du coût du matériel	2h
- Écriture du rapport	2h
- Prise en main du programme CCS 6	5h

Prochain(s) Objectif(s) :

- Prise en main du programme CCS 6

Semaine 3 / Jour 13 (3 juin 2016)

Activité(s)	Durée(s)
- Prise en main du programme CSS6 et programmation du DSP - Script Matlab pour calcul des condensateurs de lissage	8h 1h

Prochain(s) Objectif(s) :

- Programmation du DSP
- Calcul des condensateurs

Semaine 4 / Jour 14 (6 juin 2016)

Activité(s)	Durée(s)
- Programmation du DSP	7h
- Création d'un module auxiliaire pour le Launchxl-f28377s	2h

Prochain(s) Objectif(s) :

- Programmation du DSP
- Correction des Pins dû au routage

Semaine 4 / Jour 15 (7 juin 2016)

Activité(s)	Durée(s)
- Programmation du DSP	8h
- Écriture du rapport	1h

Prochain(s) Objectif(s) :

- Programmation du DSP
- Calcul des Condensateurs

Semaine 4 / Jour 16 (8 juin 2016)

Activité(s)	Durée(s)
- Programmation du DSP	5h
- Calcul des condensateurs	2h
- Mise à jour des entrées sorties du DSP	1h
- Mise en place d'un système de version (sauvegarde) Mercurial	1h

Prochain(s) Objectif(s) :

- Programmation du DSP
- Calcul des condensateurs

Semaine 4 / Jour 17 (9 juin 2016)

Activité(s)	Durée(s)
- Calcul des condensateurs	2h
- Programmation du DSP	4h
- Écriture du rapport	2h

Prochain(s) Objectif(s) :

- Programmation du DSP

Semaine 4 / Jour 18 (10 juin 2016)

Activité(s)	Durée(s)
- Programmation du DSP	9h

Prochain(s) Objectif(s) :

- Programmation du DSP
- Calcul des composants (Amplificateur différentiel)

Semaine 5 / Jour 19 (13 juin 2016)

Activité(s)	Durée(s)
- Programmation du DSP	9h

Prochain(s) Objectif(s) :
- Programmation du DSP
- Calcul des composants (Amplificateur différentiel)

Semaine 5 / Jour 20 (14 juin 2016)

Activité(s)	Durée(s)
- Programmation du DSP	3h
- Écriture du rapport	2h
- Calcul des composants (Amplificateur différentiel)	1h
- Routage d'un adaptateur pour les JTAG	2h
- Préparation des composants	1h

Prochain(s) Objectif(s) :
- Création du module pour le ventilateur
- Écriture du rapport
- Programmation du DSP

Semaine 5 / Jour 21 (15 juin 2016)

Activité(s)	Durée(s)
- Problème de connexion à l'EvalBoard, résolution du problème	3h
- Préparation des composants	1h
- Programmation du DSP	2h
- Création du module ventilateur	1h
- Test du ventilateur avec PWM et TACH	1h

Prochain(s) Objectif(s) :
- Création du module ventilateur
- Programmation du DSP

Remarque(s) :
- Le problème de connexion était dû à un problème de driver pour communiquer entre l'ordi et le kit. Après désinstallation et réinstallation du programme CCS, tout fonctionnait comme il faut

Semaine 5 / Jour 22 (16 juin 2016)

Activité(s)	Durée(s)
- Programmation du DSP	4.5h
- Création du module ventilateur	2h
- Préparation du ventilateur	0.5h
- Calcul des amplificateurs différentiels	1h

Prochain(s) Objectif(s) :

- Écriture du rapport
- Programmation du DSP
- Finaliser le protocole de mesure et de test

Semaine 5 / Jour 23 (17 juin 2016)

Activité(s)	Durée(s)
- Écriture du rapport - Mise en forme du programme DSP - Modification et assignation du programme CPLD	2h 2h 1h

Prochain(s) Objectif(s) :

- Réception de la carte, début du protocole de mesure et de test
- Écriture du rapport

Remarque(s) :

- Entretien pour un travail l'après –midi

Semaine 6 / Jour 24 (20 juin 2016)

Activité(s)	Durée(s)
- Fabrication des adaptateurs pour le JTAG - Soudage et test du module pour le ventilateur - Écriture du rapport	3h 4h 2.5h

Prochain(s) Objectif(s) :
- Réception de la carte, début du protocole de mesure et de test

Remarque(s) :
- La carte n'est pas arrivée, changement de planning

Semaine 6 / Jour 25 (21 juin 2016)

Activité(s)	Durée(s)
- Écriture du rapport - Test du module pour le ventilateur - Préparation du refroidisseur - Pose des composants sur la carte principale	2h 2h 1h 4h

Prochain(s) Objectif(s) :
- Pose des composants sur la carte principale

Semaine 6 / Jour 26 (22 juin 2016)

Activité(s)	Durée(s)
- Pose des composants sur la carte principale - Test de la carte	6h 3.5h

Prochain(s) Objectif(s) :
- Test de la carte

Semaine 6 / Jour 27 (23 juin 2016)

Activité(s)	Durée(s)
- Test de la carte	9.5h

Prochain(s) Objectif(s) :
- Test de la carte

Semaine 6 / Jour 28 (24 juin 2016)

Activité(s)	Durée(s)
- Test de la carte - Ajout du patch pour le 1.5V_ref - Mesure de la puissance dissipée max du refroidisseur	6h 1h 2h

Prochain(s) Objectif(s) :

- Mesure et essai du refroidisseur
- Écriture du rapport

Semaine 7 / Jour 29 (27 juin 2016)

Activité(s)	Durée(s)
- Mesure de la puissance de dissipation du refroidisseur - Écriture du rapport - Essai de commutation	4.5h 2h 2h

Prochain(s) Objectif(s) :

- Écriture du rapport
- Programmation de la communication

Semaine 7 / Jour 30 (28 juin 2016)

Activité(s)	Durée(s)
- Écriture du rapport - Mesure des différentes grandeurs et corrections si nécessaire - Gestion des temps mort dans la CPLD	1h 3.5h 4h

Prochain(s) Objectif(s) :

- Programmation de la communication
- Écriture du rapport

Remarque(s) :

Semaine 7 / Jour 31 (29 juin 2016)

Activité(s)	Durée(s)
- Mise en place de la communication - Charge du programme sur la mémoire flash + boot du microprocesseur - Écriture du rapport	4h 3h 1.5h

Prochain(s) Objectif(s) :

- Mise en place de la communication Uart
-

Remarque(s) :

Semaine 7 / Jour 32 (30 juin 2016)

Activité(s)	Durée(s)
- Essais en puissance buck - Écriture du rapport	6h 3h

Prochain(s) Objectif(s) :

- Essais en puissance

Remarque(s) :

Semaine 7 / Jour 33 (1 juillet 2016)

Activité(s)	Durée(s)
- Essais en puissance buck - Essais en puissance boost - Préparation pour essais en puissance DC/AC	4h 3h 2h

Prochain(s) Objectif(s) :

- Effectuer des simulations avec Matlab
- Tester le convertisseur avec PWM variable DC/AC

Remarque(s) :

Semaine 8 / Jour 34 (4 juillet 2016)

Activité(s)	Durée(s)
- Simulations Matlab Buck et Boost	8.5h

Prochain(s) Objectif(s) :

- Simulations Matlab
- Tester le convertisseur avec PWM variable DC/AC

Semaine 8 / Jour 35 (5 juillet 2016)

Activité(s)	Durée(s)
- Simulation Matlab - Mise à jour de la documentation du projet	3h 6h

Prochain(s) Objectif(s) :

- Simulation Matlab

Semaine 8 / Jour 36 (6 juillet 2016)

Activité(s)	Durée(s)
- Essais en puissance onduleur - Simulation Matlab - Programmation de la communication	3h 1h 5h

Prochain(s) Objectif(s) :

- Programmation de la communication
- Programmation de l'interface utilisateur

Semaine 8 / Jour 37 (7 juillet 2016)

Activité(s)	Durée(s)
- Programmation de la communication - Programmation de l'interface utilisateur - Gestion des temps morts dans la CPLD	2h 2h 4h

Prochain(s) Objectif(s) :

- Programmation de la communication
- Gestion des temps morts dans la CPLD

Semaine 8 / Jour 38 (8 juillet 2016)

Activité(s)	Durée(s)
- programmation de la communication - Gestion des temps morts dans la CPLD	7h 3h

Prochain(s) Objectif(s) :

- Gestion du temps mort dans la CPLD
- Programmation de l'interface de contrôle

Semaine 9 / Jour 39 (11 juillet 2016)

Activité(s)	Durée(s)
- Gestion des temps morts dans la CPLD - Rangement de l'espace de travail - Rédaction du résumé, du rapport et du journal de travail	3h 2h 3.5h

Prochain(s) Objectif(s) :

- Mise à jour de la documentation
- Écriture du rapport

Semaine 9 / Jour 40 (12 juillet 2016)

Activité(s)	Durée(s)
- Mise à jour de la documentation - Écriture du rapport	5h 3.5h

Prochain(s) Objectif(s) :

- Mise à jour de la documentation
- Écriture du rapport

Semaine 9 / Jour 41 (13 juillet 2016)

Activité(s)	Durée(s)
- Mise à jour de la documentation - Écriture du rapport	3h 5.5h

Prochain(s) Objectif(s) :

- Mise à jour de la documentation
- Écriture du rapport

Semaine 9 / Jour 42 (14 juillet 2016)

Activité(s)	Durée(s)
- Mise à jour de la documentation - Écriture du rapport	2h 6.5h

Prochain(s) Objectif(s) :

- Mise à jour de la documentation
- Écriture du rapport

Semaine 9 / Jour 43 (15 juillet 2016)

Activité(s)	Durée(s)
- Remise des documents demandés	

ANNEXE 2

Description	Valeur	Désignation	Fabricant	Type	Footprint	Quantité	Prix unité [CHF]	Prix [CHF]
Condensateur	100n	C1, C2, C3, C4, C6, C8, C9, C11, C12, C14, C19, C27, C28, C31, C32, C59, C65, C67, C82, C94, C97, C98, C99, C100, C105, C106, C107, C108, C109, C110, C111, C112, C113, C114, C115, C116, C117, C118, C119, C120, C121, C122, C123, C124, C125, C126, C127, C128, C129, C131, C132	-	-	SMD 2012 (0805)	51	0	0
Condensateur	22u	C5, C10	-	-	SMD 2012 (0805)	2	0	0
Condensateur polarisé	10u	C7, C21	AVX	TAJC106K035RNJ	TAJC	2	1.14	2.28
Condensateur	nm	C13, C17, C18, C26, C101, C102	-	-	SMD 2012 (0805)	6	0	0
Condensateur	4u7	C15, C16	-	-	SMD 2012 (0805)	2	0	0
Condensateur	1u	C20, C22, C23, C24, C29, C57, C58, C60, C61, C64, C66, C68, C70, C130	-	-	SMD 2012 (0805)	14	0	0
Condensateur	2.2u	C25, C30	-	-	SMD 2012 (0805)	2	0	0
Condensateur	1u/50V	C33, C34, C36	-	-	SMD 2012 (0805)	3	0	0
Condensateur	1n	C35, C62, C63, C75	-	-	SMD 2012 (0805)	4	0	0
Capacitor polarized	180u	C37, C38, C39, C40, C41, C42, C43, C44, C45, C46, C47, C48, C49, C50, C51, C52, C53, C54, C55, C56	Rubycon	450QXW80MEFC18X45	18X45 P7.5	20	4.15	83
Condensateur	150p	C69, C71	-	-	SMD 2012 (0805)	2	0	0
Condensateur	2.2n	C72, C80, C83, C84, C91, C92	-	-	SMD 2012 (0805)	6	0	0
Condensateur	220p	C73, C81	-	-	SMD 2012 (0805)	2	0	0
Condensateur	10n	C74, C85, C86	-	-	SMD 2012 (0805)	3	0	0
Condensateur	470p	C76, C77, C78, C79, C87, C88, C89, C90	-	-	SMD 2012 (0805)	8	0	0
Condensateur	470n	C93	-	-	SMD 2012 (0805)	1	0	0
Condensateur	2n2	C95, C136	-	-	SMD 2012 (0805)	2	0	0
Condensateur	15p	C103, C104	-	-	SMD 2012 (0805)	2	0	0
Condensateur	2u2	C133, C134	ICEL	PHB1604220*HSD	-	2	0	0
Diode	-	D1, D2	-	L4148	DIODE MINI MELF DO213AA	2	0	0
Diode Schottky, Double	-	D3, D5, D7, D8	-	BAV99	SOT23 P0.95 C1.6x3.0 H1.3	4	0	0
Diode	-	D4, D6	-	US1M	DIODE DO-241AC (SMA)	2	0	0
Connecteur D-SUB 9P	-	J1	-	D-SUB-9-FEM-ANGLE		1	13	13
Récepteur Optique	-	J2, J7	AVAGO	HFBR-2528		2	19.02	38.04

PCB terminal block with screw	-	J3, J4	-	ScrewHeader3 P5.0	ScrewHeader3 P2.54	2	0	0
Connecteur rapide	-	J5, J6, J8, J9, J10, J11, J12	-	SOCKET 4MM Vertical	SOCKET 4MM VERTICAL	7	0	0
Plug connector	-	J13, J14	-	PinHeader1x4	PinHeader1x4 P2.54	2	0	0
Connecteur plat	-	J21	-	FlatCable2x5	FlatCable2x5 2MM Straight TH FCI 98414-G06-10ULF	1	0	0
Connecteur plat	-	J22	-	FlatCable2x7	FlatCable2x7 2MM Straight TH FCI 98414-G06-14ULF	1	0	0
Connecteur plat	-	J23	-	FlatCable2x3	FlatCable2x3 2MM Straight TH FCI 98414-G06-06ULF	1	0	0
Connecteur plat	-	J24	-	FlatCable2x4	FlatCable2x4 2MM Straight TH FCI 98414-G06-08ULF	1	0	0
Jumper 3 pos	-	JP2, JP20	-	-	SMD 2012 (0805)	2	0	0
Jumper 2 pos	-	JP3, JP4, JP5, JP6, JP7, JP8, JP9, JP10, JP12, JP13, JP14, JP15, JP15, JP16, JP17, JP18, JP19	-	-	SMD 2012 (0805)	17	0	0
Inductance	2u2	L1	-	-	DR73	1	1	1
Inductance	cc	L2	-	-	DR73	1	0	0
Led		LD1, LD2, LD3	-	-	SMD 2012 (0805)	3		0
Test Point	-	PT1, PT2, PT3, PT4, PT5, PT6, PT7, PT8, PT9, PT10, PT11, PT12, PT13, PT14, PT15, PT16, PT17, PT18, PT19, PT20, PT21, PT22, PT23, PT24, PT25, PT26, PT27, PT28, PT29, PT30, PT31, PT32, PT33, PT34, PT35, PT36, PT37, PT38, PT39, PT40, PT41, PT42, PT43, PT44, PT45, PT46, PT47, PT48	-	-	Test Point Keystone Compact	48	0	0
Résistance	1k5	R1	-	-	SMD 2012 (0805)	1	0	0
Résistance	300R	R2	-	-	SMD 2012 (0805)	1	0	0
Résistance	1k3	R3, R19, R52, R54	-	-	SMD 2012 (0805)	4	0	0
Résistance	5k1	R4, R111, R134	-	-	SMD 2012 (0805)	3	0	0
Résistance	1k	R5, R10, R11, R12, R15, R16, R101	-	-	SMD 2012 (0805)	7	0	0
Résistance	470R	R6, R18	-	-	SMD 2012 (0805)	2	0	0
Résistance	220R	R7, R76	-	-	SMD 2012 (0805)	2	0	0
Résistance	0R	R8, R33, R46	-	-	SMD 2012 (0805)	3	0	0
Résistance	2R7	R9, R14	-	-	SMD 2012 (0805)	2	0	0
Résistance	560m	R13, R17	-	-	SMD 2012 (0805)	2	0	0

Résistance	6k2	R20	-	-	SMD 2012 (0805)	1	0	0
Résistance	510R	R21, R22, R23	-	-	SMD 2012 (0805)	3	0	0
Résistance	330k	R24, R27, R31, R37, R40, R43	-	-	SMD 2012 (0805)	6	0	0
Résistance	750k	R25, R26, R29, R30, R35, R36, R41, R42	-	-	SMD 2012 (0805)	8	0	0
Résistance	75R	R28, R38, R44, R48	-	-	SMD 2012 (0805)	4	0	0
Résistance	NM	R32, R45, R57, R59, R115, R116	-	-	SMD 2012 (0805)	6	0	0
Résistance	56R	R34, R47	-	-	SMD 2012 (0805)	2	0	0
Résistance	10k	R39, R49, R50, R51, R62, R64, R80, R81, R85, R94, R95, R97, R104, R105, R106, R110, R113, R114, R117, R118, R119, R120, R121, R122, R124, R125, R126, R127, R129, R130, R131, R132, R133, R135, R136, R137, R138, R139	-	-	SMD 2012 (0805)	38	0	0
Résistance	CC	R53, R55	-	-	SMD 2012 (0805)	2	0	0
Résistance	1k2	R56, R58, R60, R61, R63, R65, R74, R79	-	-	SMD 2012 (0805)	8	0	0
Résistance	100R	R66, R67, R128	-	-	SMD 2012 (0805)	3	0	0
Résistance	20k	R68, R69	-	-	SMD 2012 (0805)	2	0	0
Résistance	9.1k	R70, R82, R84, R96	-	-	SMD 2012 (0805)	4	0	0
Résistance	3k6	R71, R73, R78, R83	-	-	SMD 2012 (0805)	4	0	0
Résistance	16k	R72, R77, R86, R91	-	-	SMD 2012 (0805)	4	0	0
Résistance	430R	R75, R89, R90	-	-	SMD 2012 (0805)	3	0	0
Résistance	82k	R87, R92	-	-	SMD 2012 (0805)	2	0	0
Résistance	18k	R88, R93	-	-	SMD 2012 (0805)	2	0	0
Résistance	47R	R98, R100	-	-	SMD 2012 (0805)	2	0	0
Résistance thermique	KTY-120	R99	PHILIPS	-	TO92	1	0	0
Résistance	820R	R102, R103	-	-	SMD 2012 (0805)	2	0	0
Résistance	2k2	R107, R108, R109	-	-	SMD 2012 (0805)	3	0	0
Résistance	1M	R123	-	-	SMD 2012 (0805)	1	0	0
Bouton Poussoir		S1	-	-	Push Button C7.4x7.4 C&K KSA0M210	1	0	0
IGBT & DIODE	-	T1, T2	INFINEON	IKW25N120T2	T0247	2	6	12
Optocoupleur	-	U1, U4	VISHAY	SFH6156	SO4L P2.54 C4.8x6.2 H3.8	2	0.5	1
Convertisseur de signal RS232	-	U2	TEXAS INSTRUMENTS	MAX232EIDR	SO16 P1.27 C4.0x10.3 H2.65 - duplicate	1	1	1

Alimentation régulée 1.8V et 3.3V	-	U3	TEXAS INSTRUMENTS	TPS767D318QPWPRQ1	TSSOP28 P0.65 T6.5x2.4 B9.7x4.4 H1.2	1	5.66	5.66
Alimentation régulée 1.2V	-	U5	TEXAS INSTRUMENTS	TPS62262DDCT	SOT23-5 P0.95 C1.6x3.0 H1.3	1	1.79	1.79
Amplificateur opérationnel	-	U6			SO8 P1.27 W5.84 H1.75	1	1.75	1.75
Alimentation référence 3V	-	U7	TEXAS INSTRUMENTS	REF3030	SOT23 P0.95 C1.6x3.0 H1.3	1	2.1	2.1
Alimentation 24V to 5V	-	U8	CINCON	EC4A11-H	CINCON CASE A	1	12.93	12.93
Alimentation 24V to 5V	-	U9	MURATA	MEF1S2405SP3C	MURATA SP3C	1	5.56	5.56
Alimentation driver	-	U10, U12	MURATA	MGJ2D241505SC	MURATA MGJ2	2	8.43	16.86
Shunt régulateur de précision	-	U11, U16, U17	TEXAS INSTRUMENTS	LMV431ACM5	SOT23-5 P0.95 C1.6x3.0 H1.3	3	0.5	1.5
Optocoupleur	-	U13, U15	FAIRCHILD	FOD3180S	SO8L P2.54 C6.6X9.7 H5.0	2	1.5	3
Capteur de courant	-	U14	LEM	LTSR-25NP	LTSR xx NP	1	14.74	14.74
Optocoupleur	-	U18, U20	Silicon Labs	SI8717	SO8 P1.27 C4.0X5.0 H1.75	2	1.23	2.46
Comparateur	-	U19, U21	TEXAS INSTRUMENTS	LM311	SO8 P1.27 W5.84 H1.75	2	0.5	1
Amplificateur opérationnel	-	U22	-	-	SO14 P1.27 W5.84 H1.75	1	1.75	1.75
CPLD	-	U23	XILINX	XC2C32A-6VQG44C	VQ44 P0.8 C10X10 H1.2	1	1.5	1.5
processeur	-	U24	TEXAS INSTRUMENTS	TMS320F28377S	TSQFP50P1600X1600X120_ HS-101L	1	23.72	23.72
oscillateur	10MHz	X1	IQD	HC49S	Quartz HC49/4H SMX	1	0	0
Ventilateur	-	-	SUNON			1	15.8	15.8

Total	263.44

ANNEXE 3

ORDER

We thank you for your order and are pleased to confirm it to you as follows

10 juin 2016

Administrative details

Your references

Order nr.	E822866	Purchase reference	TRC_PC16123
Service	STANDARD pool	Project reference	TRC_PC16123
Board name	Module Conversion Statique	Article number	TRC_PC16123

Invoicing & delivery details

Invoice to:
Haute Ecole Valaisanne
Costa Christian
. rte du rawyl 47
1950 SION
Switzerland
0041276068721
0041 27 606 87 11
cos@hevs.ch

Delivered to:
Haute Ecole Valaisanne
Costa Christian
rte du rawyl 47
1950 SION
Switzerland
0041276068721
0041 276068711
cos@hevs.ch

PCB Visualizer

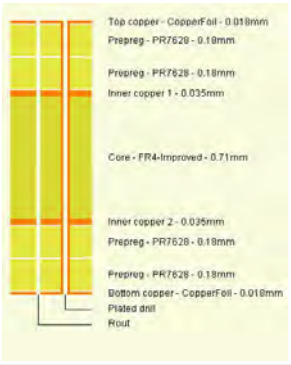
PCB images

Top view :

Bottom view :

Buildup & Mechanical plan

Board buildup :



Technology & options

Board definition

Number of layers	4	Delivery format	No
PCB width (X)	400.00 mm	PCB height (Y)	100.00 mm
eC-registration compatible	No		

Board definition

Top soldermask	Vert	Bottom soldermask	Vert
Top legend	Blanc	Bottom legend	Blanc
Surface finish	Any lead free finish		
Bare board testing	Yes		

Board technology

Pattern class	6	Drill class	C
Outer layer trackwidth (OL-TW)	0.150 mm	Hole density	<1000/dm2
Outer layer isolation distance (OL-TT-TP-PP)	0.150 mm	Holes <= may be reduced	0.45 mm
Outer layer annular ring (OAR)	0.125 mm		
Inner layer trackwidth (IL-TW)	0.150 mm		
Inner layer isolation distance (IL-TT-TP-PP)	0.150 mm		
Inner layer annular ring (IAR)	0.125 mm		

Material definition

Board thickness	1.55 mm	Board buildup	Standard buildup
Base material	FR4IMP	Material Tg	145-150°C
Outer layer copper foil	18µm(End-Cu +/-35µm)	Inner layer copper foil	35µm
Extra PTH runs	0	Extra press cycles	0
Reversed buildup	No	Inner layer core thickness	Standard

Advanced options

Copper up to board edge	No	Plated holes on the board edge	No
Specific tolerances	No	Specific marking	No
Press-fit holes	No	Depth routing	No
Round-edge plating	No	Chamfered mechanical holes	No

Pricing

Printed circuits

Order nr.	Shipment date	Quantity	Unit price	Transport price	Transport mode	Total price	VAT	Gross
E822866	17 juin 2016	3	160.35 EUR	11.58 EUR	Express	492.63 EUR	8.00 %	532.04 EUR

Payment terms & conditions

The payment term is 30 days from invoice date.

This quotation is valid for 30 days. All our deliveries are according to our general terms and conditions of delivery. These are available on the website , and agreed upon between us during the registration procedure. All above mentioned prices are an indication on the basis of the information at our disposal on the moment of quotation. These prices may be reviewed at the moment of order on the basis of the final documentation and conditions. The final quantity to be delivered can vary up to 5% of the ordered quantity. Delivery terms start counting upon receipt of the complete documentation and firm order.

Eurocircuits Sàrl
Place du Midi 30 - CP 97
4 eme etage
droite
1951 Sion
Switzerland

www.eurocircuits.com

Phone:+41225480563
Fax:+41 27 329 05 6
E-mail:euro@eurocircuits.com

ANNEXE 4

```
 1 /*****************************************************************/
 2 /*   Module de Conversion Statique                  Global.h     */
 3 /*                                                               */
 4 /*   Travail de Bachelor                                         */
 5 /*                                                               */
 6 /*   HES-SO VALAIS / WALLIS                        13.07.2016    */
 7 /*                                                               */
 8 /*   AUTHOR : DUBOSSON MAXIME                                    */
 9 /*   PROFESSOR : BARRADE PHILIPPE                                */
10 /*                                                               */
11 /*****************************************************************/
12
13 #include "F28x_Project.h"
14 #include "Interrupt.h"
15 #include "Adc.h"
16 #include "Pwm.h"
17 #include "Gpio.h"
18 #include "Alarme.h"
19 #include "Control.h"
20 #include "vardebug.h"
21 #include "remote_ctrl.h"
22
23 /*****************************************************************/
24 /* Macro pour la compréhension du programme                      */
25 /*****************************************************************/
26 #define ENABLE_INTERRUPTS       EINT
27 #define DISABLE_INTERRUPTS      DINT
28
29 #define ENABLE_REALTIME_MODE    ERTM
30 #define DISABLE_REALTIME_MODE   DRTM
31
32 #define ENABLE_PROTECTED_REGISTER   EALLOW
33 #define DISABLE_PROTECTED_REGISTER  EDIS
34
35 #define SOFTWARE_BREAKPOINT ESTOP0
36
37 #define INTERRUPT_ENABLE_REGISTER IER
38 #define INTERRUPT_FLAG_REGISTER IFR
39
40 #define TRUE 1
41 #define FALSE 0
42
43 #define LSPCLK_FREQ 25000000
44 #define T_S_REGUL 0.000002
45
46 /*****************************************************************/
47 /* Paramètres minimum et maximum des grandeurs                   */
48 /*****************************************************************/
49 #define I_MAX 50
50 #define I_MIN -50
51 #define U_MAX 30
52 #define U_MIN 0
53 #define VCC_MAX 500
54 #define VEE_MIN 500
55 #define T_MAX 200
56 #define T_MIN 0
57
58
```

```c
/*****************************************************************/
/*   Module de Conversion Statique                      main.c    */
/*                                                                */
/*   Travail de Bachelor                                          */
/*                                                                */
/*   HES-SO VALAIS / WALLIS                        13.07.2016     */
/*                                                                */
/*   AUTHOR : DUBOSSON MAXIME                                     */
/*   PROFESSOR : BARRADE PHILIPPE                                 */
/*                                                                */
/*****************************************************************/

#include "Global.h"

/*****************************************************************/
/* Déclaration des structures                                    */
/*****************************************************************/
struct count_t count;
struct adc_t adc;
struct epwm_t epwm6;
struct epwm_t epwm7;
struct err_t err;
struct pwm_t ventil;
struct channel_t adc_channel;
union param_t param;

/*****************************************************************/
/* Fonction main                                                 */
/*****************************************************************/
void main(void) {

    /*****************************************************************/
    /* Initialisation                                                */
    /*****************************************************************/
    InitValue();
    InitSysCtrl();
    InitADC();
    InitGPIO();
    init_vardebug();
    InitPWM_FAN();
    InitPWM_CONV();
    InitInterrupt();

    /*****************************************************************/
    /* Démarrage des mesures ADC                                     */
    /*****************************************************************/
    start_soc0();
    start_soc7();

    /*****************************************************************/
    /* Boucle infinie                                                */
    /*****************************************************************/
    do {
        updateErreur(); //actualise les erreurs
        activeOutEnable();

        /*****************************************************************/
        /* Boucle de contrôle --> 10ms                                   */
        /*****************************************************************/
        if (count.start_update) {
            adc.i_lim_up_not_scale = (ADC_MAX / 2)
```

```
62                      + adc.i_lim * ADC_MAX / 2 / I_MAX;
63              adc.i_lim_down_not_scale = (ADC_MAX / 2)
64                      - adc.i_lim * ADC_MAX / 2 / I_MAX;
65              updateValue(); //actualise les valeurs des mesures
66              regulFan();
67              count.start_update = 0;
68              count.timer_update = 0;
69
70          }
71
72          /*****************************************************************/
73          /* Boucle de mesure --> 200us                                    */
74          /*****************************************************************/
75          if (count.start_adc) {
76              mesADC();
77              count.start_adc = 0;
78              count.timer_adc = 0;
79              process_vardebug();
80
81          }
82
83      } while (1);
84  }
85
86
```

```
 1 /*****************************************************************/
 2 /*   Module de Conversion Statique                    Gpio.h     */
 3 /*                                                               */
 4 /*   Travail de Bachelor                                         */
 5 /*                                                               */
 6 /*   HES-SO VALAIS / WALLIS                        13.07.2016    */
 7 /*                                                               */
 8 /*   AUTHOR : DUBOSSON MAXIME                                    */
 9 /*   PROFESSOR : BARRADE PHILIPPE                                */
10 /*                                                               */
11 /*****************************************************************/
12
13 typedef unsigned int boolean;
14
15 #define OE_DISABLE 1
16 #define OE_ENABLE 0
17
18 #define GPIO_INPUT 0
19 #define GPIO_OUTPUT 1
20
21 #define GPIO_MUX_IO 0
22 #define GPIO_MUX_SELECT_1 1
23 #define GPIO_MUX_SELECT_2 2
24 #define GPIO_MUX_SELECT_3 3
25
26 #define GPIO_PULL_UP_ENABLE 0
27 #define GPIO_PULL_UP_DISABLE 1
28
29 /*****************************************************************/
30 /* Macro pour la définition des entrées                          */
31 /*****************************************************************/
32 #define XINT GpioDataRegs.GPADAT.bit.GPIO4
33 #define DIGITAL_INPUT_1 GpioDataRegs.GPBDAT.bit.GPIO41
34 #define CONFIG_1 GpioDataRegs.GPBDAT.bit.GPIO42
35 #define CONFIG_2 GpioDataRegs.GPBDAT.bit.GPIO43
36 #define DIGITAL_INPUT_2 GpioDataRegs.GPBDAT.bit.GPIO58
37 #define OL_S2_N GpioDataRegs.GPCDAT.bit.GPIO64
38 #define OL_S1_N GpioDataRegs.GPCDAT.bit.GPIO65
39 #define CONFIG_3 GpioDataRegs.GPCDAT.bit.GPIO69
40 #define CONFIG_4 GpioDataRegs.GPCDAT.bit.GPIO70
41 #define TACH_VENTIL GpioDataRegs.GPCDAT.bit.GPIO71
42 #define DEFAUT GpioDataRegs.GPADAT.bit.GPIO20
43
44 /*****************************************************************/
45 /* Macro pour la définition des sorties                          */
46 /*****************************************************************/
47 inline void set_led_err(const boolean val) {
48     if (val)
49         GpioDataRegs.GPBSET.bit.GPIO59 = 1;
50     else
51         GpioDataRegs.GPBCLEAR.bit.GPIO59 = 1;
52 }
53
54 inline void set_led_alim_ok(const boolean val) {
55     if (val)
56         GpioDataRegs.GPBSET.bit.GPIO60 = 1;
57     else
58         GpioDataRegs.GPBCLEAR.bit.GPIO60 = 1;
59 }
60 inline void toggle_led_err(const boolean val) {
61     if (val)
```

```
62          GpioDataRegs.GPATOGGLE.bit.GPIO10 = 1;
63      else
64          GpioDataRegs.GPATOGGLE.bit.GPIO10 = 0;
65  }
66
67  inline void set_led_ok(const boolean val) {
68      if (val)
69          GpioDataRegs.GPBSET.bit.GPIO61 = 1;
70      else
71          GpioDataRegs.GPBCLEAR.bit.GPIO61 = 1;
72  }
73  inline void debug_io_1(const boolean val) {
74      if (val)
75          GpioDataRegs.GPCSET.bit.GPIO86 = 1;
76      else
77          GpioDataRegs.GPCCLEAR.bit.GPIO86 = 1;
78  }
79  inline void debug_io_2(const boolean val) {
80      if (val)
81          GpioDataRegs.GPCSET.bit.GPIO87 = 1;
82      else
83          GpioDataRegs.GPCCLEAR.bit.GPIO87 = 1;
84  }
85  inline void debug_io_3(const boolean val) {
86      if (val)
87          GpioDataRegs.GPASET.bit.GPIO2 = 1;
88      else
89          GpioDataRegs.GPACLEAR.bit.GPIO2 = 1;
90  }
91  inline void debug_io_4(const boolean val) {
92      if (val)
93          GpioDataRegs.GPASET.bit.GPIO3 = 1;
94      else
95          GpioDataRegs.GPACLEAR.bit.GPIO3 = 1;
96  }
97  inline void reset_ol(const boolean val) {
98      if (val)
99          GpioDataRegs.GPASET.bit.GPIO10 = 1;
100     else
101         GpioDataRegs.GPACLEAR.bit.GPIO10 = 1;
102 }
103 inline void defaut(const boolean val) {
104     if (val)
105         GpioDataRegs.GPASET.bit.GPIO20 = 1;
106     else
107         GpioDataRegs.GPACLEAR.bit.GPIO20 = 1;
108 }
109 inline void output_enable(const boolean val) {
110     if (val)
111         GpioDataRegs.GPCSET.bit.GPIO66 = 1;
112     else
113         GpioDataRegs.GPCCLEAR.bit.GPIO66 = 1;
114 }
115
116 /*************************************************************/
117 /* Déclaration des prototypes de fonctions                   */
118 /*************************************************************/
119 void InitGPIO(void);
120
```

```c
/*****************************************************************/
/*   Module de Conversion Statique                    gpio.c     */
/*                                                               */
/*   Travail de Bachelor                                         */
/*                                                               */
/*   HES-SO VALAIS / WALLIS                        13.07.2016    */
/*                                                               */
/*   AUTHOR : DUBOSSON MAXIME                                    */
/*   PROFESSOR : BARRADE PHILIPPE                                */
/*                                                               */
/*****************************************************************/

#include "Global.h"

/*****************************************************************/
/* Initialisation des GPIOs                                      */
/*****************************************************************/
void InitGPIO(void) {

    ENABLE_PROTECTED_REGISTER;
    /*
     * utilisé pour les leds sur le kit de test
     */
    //GpioCtrlRegs.GPADIR.bit.GPIO12 = 1;
    //GpioCtrlRegs.GPADIR.bit.GPIO13 = 1;


    /*****************************************************************/
    /* Multiplexage des entrées / sorties                            */
    /*****************************************************************/
    GpioCtrlRegs.GPAMUX1.bit.GPIO2 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPAMUX1.bit.GPIO3 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPAMUX1.bit.GPIO4 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPAMUX1.bit.GPIO10 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPAMUX1.bit.GPIO11 = GPIO_MUX_SELECT_1; //EPWM6B - Ventilateur
    GpioCtrlRegs.GPAMUX1.bit.GPIO12 = GPIO_MUX_SELECT_1; //EPWM7A
    GpioCtrlRegs.GPAMUX1.bit.GPIO13 = GPIO_MUX_SELECT_1; //EPWM7B
    //GpioCtrlRegs.GPAMUX1.bit.GPIO14 = GPIO_MUX_SELECT_1; //EPWM8A
    //GpioCtrlRegs.GPAMUX1.bit.GPIO15 = GPIO_MUX_SELECT_1; //EPWM8B
    GpioCtrlRegs.GPAMUX1.bit.GPIO15 = GPIO_MUX_IO; //GPIO - Temps mort
    GpioCtrlRegs.GPAMUX2.bit.GPIO16 = GPIO_MUX_SELECT_1; //SPISIMOA
    GpioCtrlRegs.GPAMUX2.bit.GPIO17 = GPIO_MUX_SELECT_1; //SPISOMIA
    GpioCtrlRegs.GPAMUX2.bit.GPIO18 = GPIO_MUX_SELECT_1; //SPICLKA
    GpioCtrlRegs.GPAMUX2.bit.GPIO19 = GPIO_MUX_SELECT_1; //NON-SPISTEA
    GpioCtrlRegs.GPAMUX2.bit.GPIO20 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPBMUX1.bit.GPIO41 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPBMUX1.bit.GPIO42 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPBMUX1.bit.GPIO43 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPBMUX2.bit.GPIO58 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPBMUX2.bit.GPIO59 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPBMUX2.bit.GPIO60 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPBMUX2.bit.GPIO61 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPBGMUX2.bit.GPIO62 = GPIO_MUX_SELECT_1; //CANRXA
    GpioCtrlRegs.GPBMUX2.bit.GPIO62 = GPIO_MUX_SELECT_2; //CANRXA
    GpioCtrlRegs.GPBGMUX2.bit.GPIO63 = GPIO_MUX_SELECT_1; //CANRXA
    GpioCtrlRegs.GPBMUX2.bit.GPIO63 = GPIO_MUX_SELECT_2; //CANTXA
    GpioCtrlRegs.GPCMUX1.bit.GPIO64 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPCMUX1.bit.GPIO65 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPCMUX1.bit.GPIO66 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPCMUX1.bit.GPIO69 = GPIO_MUX_IO; //GPIO
    GpioCtrlRegs.GPCMUX1.bit.GPIO70 = GPIO_MUX_IO; //GPIO
```

```
62      GpioCtrlRegs.GPCMUX1.bit.GPIO71 = GPIO_MUX_IO; //GPIO
63      GpioCtrlRegs.GPCMUX1.bit.GPIO73 = GPIO_MUX_SELECT_3; //XCLKOUT
64      GpioCtrlRegs.GPCMUX2.bit.GPIO86 = GPIO_MUX_IO; //GPIO
65      GpioCtrlRegs.GPCMUX2.bit.GPIO87 = GPIO_MUX_IO; //GPIO
66      GpioCtrlRegs.GPCGMUX2.bit.GPIO89 = GPIO_MUX_SELECT_1; //SCITXDC
67      GpioCtrlRegs.GPCMUX2.bit.GPIO89 = GPIO_MUX_SELECT_2; //SCITXDC
68      GpioCtrlRegs.GPCGMUX2.bit.GPIO90 = GPIO_MUX_SELECT_1; //SCIRXDC
69      GpioCtrlRegs.GPCMUX2.bit.GPIO90 = GPIO_MUX_SELECT_2; //SCIRXDC
70      GpioCtrlRegs.GPCGMUX2.bit.GPIO91 = GPIO_MUX_SELECT_1; //SDAA
71      GpioCtrlRegs.GPCMUX2.bit.GPIO91 = GPIO_MUX_SELECT_2; //SDAA
72      GpioCtrlRegs.GPCGMUX2.bit.GPIO92 = GPIO_MUX_SELECT_1; //SCLA
73      GpioCtrlRegs.GPCMUX2.bit.GPIO92 = GPIO_MUX_SELECT_2; //SCLA
74
75      /*************************************************************/
76      /* Gestion des entrées / sorties                             */
77      /*************************************************************/
78      GpioCtrlRegs.GPADIR.bit.GPIO2 = GPIO_OUTPUT; //DEBUG IO 3
79      GpioCtrlRegs.GPADIR.bit.GPIO3 = GPIO_OUTPUT; //DEBUG IO 4
80      GpioCtrlRegs.GPADIR.bit.GPIO4 = GPIO_INPUT; //RESERVE / XINT (ev.)
81      GpioCtrlRegs.GPADIR.bit.GPIO15 = GPIO_INPUT; //Temps mort
82      GpioCtrlRegs.GPADIR.bit.GPIO10 = GPIO_OUTPUT; //RESET OL
83      GpioCtrlRegs.GPADIR.bit.GPIO20 = GPIO_OUTPUT; //DSP_DEFAUT
84      GpioCtrlRegs.GPBDIR.bit.GPIO41 = GPIO_INPUT; //DIGITAL INPUT 1
85      GpioCtrlRegs.GPBDIR.bit.GPIO42 = GPIO_INPUT; //CONFIG 1
86      GpioCtrlRegs.GPBDIR.bit.GPIO43 = GPIO_INPUT; //CONFIG 2
87      GpioCtrlRegs.GPBDIR.bit.GPIO58 = GPIO_INPUT; //DIGITAL INPUT 2
88      GpioCtrlRegs.GPBDIR.bit.GPIO59 = GPIO_OUTPUT; //DSP LED ERR
89      GpioCtrlRegs.GPBDIR.bit.GPIO60 = GPIO_OUTPUT; //DSP LED ALIM OK
90      GpioCtrlRegs.GPBDIR.bit.GPIO61 = GPIO_OUTPUT; //DSP LED OK
91      GpioCtrlRegs.GPCDIR.bit.GPIO64 = GPIO_INPUT; //N DSP OL S2
92      GpioCtrlRegs.GPCDIR.bit.GPIO65 = GPIO_INPUT; //N DSP OL S1
93      GpioCtrlRegs.GPCDIR.bit.GPIO66 = GPIO_OUTPUT; //N OE DSP
94      GpioCtrlRegs.GPCDIR.bit.GPIO69 = GPIO_INPUT; //CONFIG 3
95      GpioCtrlRegs.GPCDIR.bit.GPIO70 = GPIO_INPUT; //CONFIG 4
96      GpioCtrlRegs.GPCDIR.bit.GPIO71 = GPIO_INPUT; //DSP TACH VENTIL
97      GpioCtrlRegs.GPCDIR.bit.GPIO86 = GPIO_OUTPUT; //DEBUG IO 1
98      GpioCtrlRegs.GPCDIR.bit.GPIO87 = GPIO_OUTPUT; //DEBUG IO 2
99
100     /*************************************************************/
101     /* Gestion des Pull-up                                       */
102     /*************************************************************/
103     GpioCtrlRegs.GPCPUD.bit.GPIO64 = GPIO_PULL_UP_ENABLE;
104     GpioCtrlRegs.GPCPUD.bit.GPIO65 = GPIO_PULL_UP_ENABLE;
105
106     // Gestion de Sélection de la synchronisation 0: SYSCLK, 1: 3 SAMPLES, 2: 6 SAMPLES, 3:
    ASYNC
107     GpioCtrlRegs.GPCQSEL1.bit.GPIO73 = 0;
108
109     /*************************************************************/
110     /* Gestion du Clock out XCLKOUT                              */
111     /*************************************************************/
112     ClkCfgRegs.CLKSRCCTL3.bit.XCLKOUTSEL = 0x010; //CPU1.SYSCLK
113     ClkCfgRegs.XCLKOUTDIVSEL.bit.XCLKOUTDIV = 0x11; // divisé par 8
114
115     DISABLE_PROTECTED_REGISTER;
116
117     GPIO_SetupXINT3Gpio(73); // Configure la GPIO73 sur l'interrupt externe 3
118
119     defaut(0);
120     reset_ol(0);
121 }
```

122

```
/*****************************************************************/
/*   Module de Conversion Statique                Interrupt.h    */
/*                                                               */
/*   Travail de Bachelor                                         */
/*                                                               */
/*   HES-SO VALAIS / WALLIS                        13.07.2016    */
/*                                                               */
/*   AUTHOR : DUBOSSON MAXIME                                    */
/*   PROFESSOR : BARRADE PHILIPPE                                */
/*                                                               */
/*****************************************************************/

/*****************************************************************/
/* Définition de la structure count                              */
/*****************************************************************/
struct count_t {
    uint16_t timer_adc;
    uint16_t start_adc;
    uint16_t timer_update;
    uint16_t start_update;
};

/*****************************************************************/
/* Définition de la structure channel                            */
/*****************************************************************/
struct channel_t {
    uint16_t channel;
};

extern struct count_t count;
extern struct channel_t adc_channel;

/*****************************************************************/
/* Déclaration des prototypes de fonctions                       */
/*****************************************************************/
__interrupt void cpu_timer0_isr(void);
__interrupt void cpu_timer1_isr(void);
__interrupt void cpu_timer2_isr(void);
__interrupt void adca2int_isr(void);
__interrupt void adca3int_isr(void);
__interrupt void xint3_isr(void);
__interrupt void epwm6_isr(void);
__interrupt void epwm7_isr(void);

void InitInterrupt(void);

```

```c
/*************************************************************/
/*   Module de Conversion Statique                interrupt.c    */
/*                                                             */
/*   Travail de Bachelor                                       */
/*                                                             */
/*   HES-SO VALAIS / WALLIS                        13.07.2016    */
/*                                                             */
/*   AUTHOR : DUBOSSON MAXIME                                  */
/*   PROFESSOR : BARRADE PHILIPPE                              */
/*                                                             */
/*************************************************************/

#include "Global.h"

/*************************************************************/
/* Initialisation des interruptions                          */
/*************************************************************/
void InitInterrupt(void) {

    DISABLE_INTERRUPTS;

    InitPieCtrl();

    INTERRUPT_ENABLE_REGISTER = 0x0000;
    INTERRUPT_FLAG_REGISTER = 0x0000;

    InitPieVectTable();

    /*************************************************************/
    /* Assignation des ISR d'interruption                        */
    /*************************************************************/
    ENABLE_PROTECTED_REGISTER;
    PieVectTable.TIMER0_INT = &cpu_timer0_isr;
    PieVectTable.TIMER1_INT = &cpu_timer1_isr;
    //PieVectTable.TIMER2_INT = &cpu_timer2_isr;
    PieVectTable.ADCA2_INT = &adca2int_isr;
    PieVectTable.ADCA3_INT = &adca3int_isr;
    PieVectTable.XINT3_INT = &xint3_isr;
    PieVectTable.EPWM6_INT = &epwm6_isr;
    PieVectTable.EPWM7_INT = &epwm7_isr;
    DISABLE_PROTECTED_REGISTER;

    InitCpuTimers();

    /*************************************************************/
    /* Configuration des Timer d'interruption                    */
    /*************************************************************/
    ConfigCpuTimer(&CpuTimer0, 200, 10); //Timer 0, 1 us Periode
    ConfigCpuTimer(&CpuTimer1, 200, 1000); //Timer 1, 1ms Periode
    //ConfigCpuTimer(&CpuTimer2, 200, 100000); //Timer 2, 100ms Periode

    CpuTimer0Regs.TCR.all = 0x4000; // Use write-only instruction to set TSS bit = 0
    CpuTimer1Regs.TCR.all = 0x4000; // Use write-only instruction to set TSS bit = 0
    //CpuTimer2Regs.TCR.all = 0x4000; // Use write-only instruction to set TSS bit = 0

    /*************************************************************/
    /* Activation des groupes d'interruptions                    */
    /*************************************************************/
    PieCtrlRegs.PIECTRL.bit.ENPIE = 1; // Active PIE block
    PieCtrlRegs.PIEIER1.bit.INTx7 = 1; // Active PIE Groupe 1 TIMER0
    PieCtrlRegs.PIEIER3.bit.INTx6 = 1; // Active PIE Groupe 3 EPWM6
```

```
62      PieCtrlRegs.PIEIER10.bit.INTx2 = 1; // Active PIE Groupe 10 ADCA2
63      PieCtrlRegs.PIEIER10.bit.INTx3 = 1; // Active PIE Groupe 10 ADCA3
64      PieCtrlRegs.PIEIER12.bit.INTx1 = 1; // Active PIE Groupe 12 XINT3
65
66      /*************************************************************/
67      /* Activation des registre d'interruption                    */
68      /*************************************************************/
69      INTERRUPT_ENABLE_REGISTER |= M_INT1; // Interrupt Timer 0
70      INTERRUPT_ENABLE_REGISTER |= M_INT3; // Interrupt EPWM6
71      INTERRUPT_ENABLE_REGISTER |= M_INT10; // Interrupt ADCs
72      //INTERRUPT_ENABLE_REGISTER |= M_INT12; // Interrupt XINT
73      INTERRUPT_ENABLE_REGISTER |= M_INT13; // Interrupt Timer 1
74      //INTERRUPT_ENABLE_REGISTER |= M_INT14; // Interrupt Timer 2
75
76      /*************************************************************/
77      /* Activation de l'interruptions externe                     */
78      /*************************************************************/
79  //  XintRegs.XINT3CR.bit.POLARITY = 1; //1: flanc montant, 0: flanc descendant
80  //  XintRegs.XINT3CR.bit.ENABLE = 1; // active l'interruption externe 3
81      ENABLE_INTERRUPTS;
82      ENABLE_REALTIME_MODE;
83  }
84
85  /*************************************************************/
86  /* Interruption timer 0 --> 1us                              */
87  /*************************************************************/
88  __interrupt void cpu_timer0_isr(void) {
89      //GpioDataRegs.GPCSET.bit.GPIO86 = 1;
90      CpuTimer2.InterruptCount++;
91
92      PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //Quittance du PIE pour le groupe 1
93
94      AdcaRegs.ADCSOCFRC1.bit.SOC0 = 1;
95      if (count.timer_adc == 10) {
96          count.start_adc = 1;
97      }
98      //GpioDataRegs.GPCCLEAR.bit.GPIO86 = 1;
99  }
100
101 /*************************************************************/
102 /* Interruption timer 1 --> 1ms                              */
103 /*************************************************************/
104 __interrupt void cpu_timer1_isr(void) {
105     CpuTimer1.InterruptCount++;
106
107     count.timer_update++;
108     if (count.timer_update == 10) {
109         count.start_update = 1;
110     }
111
112     // La CPU quittance l'interruption
113 }
114
115 /*************************************************************/
116 /* Interruption timer 2 --> 100ms   (pas utilisé)            */
117 /*************************************************************/
118 __interrupt void cpu_timer2_isr(void) {
119     CpuTimer2.InterruptCount++;
120
121     // La CPU quittance l'interruption
122 }
```

```
123
124 /*****************************************************************/
125 /* Interruption adc courant                                      */
126 /*****************************************************************/
127 __interrupt void adca2int_isr(void) {
128     GpioDataRegs.GPCSET.bit.GPIO87 = 1;
129
130     adc.i_mes = AdcaResultRegs.ADCRESULT0;
131
132     if (adc.i_mes > adc.i_lim_up_not_scale
133             || adc.i_mes < adc.i_lim_down_not_scale) {
134         err.Err_appear = 1;
135         defaut(1);
136     }
137
138     PieCtrlRegs.PIEACK.all = PIEACK_GROUP10; //Quittance du PIE pour le groupe 10
139
140     count.timer_adc++;
141
142     GpioDataRegs.GPCCLEAR.bit.GPIO87 = 1;
143 }
144
145 /*****************************************************************/
146 /* Interruption adc tensions/température                         */
147 /*****************************************************************/
148 __interrupt void adca3int_isr(void) {
149
150     switch (adc_channel.channel) {
151     case 1:
152         adc.vcc_bus_mes = AdcaResultRegs.ADCRESULT6;
153         adc_channel.channel = 2;
154         break;
155     case 2:
156         adc.vee_bus_mes = AdcaResultRegs.ADCRESULT6;
157         adc_channel.channel = 3;
158         break;
159     case 3:
160         adc.u_24_mes = AdcaResultRegs.ADCRESULT6;
161         adc_channel.channel = 1;
162         break;
163     default:
164         break;
165     }
166
167 //Sauvegarde la valeur de la température mesurée sur l'entrée ADCB2
168     if (AdcbRegs.ADCINTFLG.bit.ADCINT3 == 1) {
169         adc.t_mes = AdcbResultRegs.ADCRESULT7;
170         AdcbRegs.ADCINTFLGCLR.bit.ADCINT3 = 1;
171     }
172
173     PieCtrlRegs.PIEACK.all = PIEACK_GROUP10; //Quittance du PIE pour le groupe 10
174
175 }
176 /*****************************************************************/
177 /* Interruption externe                                          */
178 /*****************************************************************/
179 __interrupt void xint3_isr(void) {
180
181 }
182
183 /*****************************************************************/
```

```
184 /* Interruption epwm6                                                */
185 /*******************************************************************/
186 __interrupt void epwm6_isr(void) {
187
188     /*
189      if (EPwm6_DB_Direction == DB_UP) {
190      if (EPwm6Regs.DBFED.bit.DBFED < EPWM6_MAX_DB) {
191      EPwm6Regs.DBFED.bit.DBFED++;
192      EPwm6Regs.DBRED.bit.DBRED++;
193      } else {
194      EPwm6_DB_Direction = DB_DOWN;
195      EPwm6Regs.DBFED.bit.DBFED--;
196      EPwm6Regs.DBRED.bit.DBRED--;
197      }
198      } else {
199      if (EPwm6Regs.DBFED.bit.DBFED == EPWM6_MIN_DB) {
200      EPwm6_DB_Direction = DB_UP;
201      EPwm6Regs.DBFED.bit.DBFED++;
202      EPwm6Regs.DBRED.bit.DBRED++;
203      } else {
204      EPwm6Regs.DBFED.bit.DBFED--;
205      EPwm6Regs.DBRED.bit.DBRED--;
206      }
207      }
208      */
209     epwm6.EPwmTimerIntCount++;
210
211     // Clear INT flag for this timer
212     EPwm6Regs.ETCLR.bit.INT = 1;
213
214     // Acknowledge this interrupt to receive more interrupts from group 3
215     PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
216 }
217
218 /*******************************************************************/
219 /* Interruption epwm7                                              */
220 /*******************************************************************/
221 __interrupt void epwm7_isr(void) {
222
223     /*
224      if (EPwm6_DB_Direction == DB_UP) {
225      if (EPwm6Regs.DBFED.bit.DBFED < EPWM6_MAX_DB) {
226      EPwm6Regs.DBFED.bit.DBFED++;
227      EPwm6Regs.DBRED.bit.DBRED++;
228      } else {
229      EPwm6_DB_Direction = DB_DOWN;
230      EPwm6Regs.DBFED.bit.DBFED--;
231      EPwm6Regs.DBRED.bit.DBRED--;
232      }
233      } else {
234      if (EPwm6Regs.DBFED.bit.DBFED == EPWM6_MIN_DB) {
235      EPwm6_DB_Direction = DB_UP;
236      EPwm6Regs.DBFED.bit.DBFED++;
237      EPwm6Regs.DBRED.bit.DBRED++;
238      } else {
239      EPwm6Regs.DBFED.bit.DBFED--;
240      EPwm6Regs.DBRED.bit.DBRED--;
241      }
242      }
243      */
244     //epwm7.EPwmTimerIntCount++;
```

```
245     // Clear INT flag for this timer
246     EPwm7Regs.ETCLR.bit.INT = 1;
247
248     // Acknowledge this interrupt to receive more interrupts from group 3
249     PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
250 }
251
```

```
/*****************************************************************/
/*   Module de Conversion Statique                       Adc.h   */
/*                                                               */
/*   Travail de Bachelor                                         */
/*                                                               */
/*   HES-SO VALAIS / WALLIS                        13.07.2016    */
/*                                                               */
/*   AUTHOR : DUBOSSON MAXIME                                    */
/*   PROFESSOR : BARRADE PHILIPPE                                */
/*                                                               */
/*****************************************************************/

/*****************************************************************/
/* Définition de la structure ADC                                */
/*****************************************************************/
struct adc_t {
    float vee_bus_mes;
    float vcc_bus_mes;
    float u_24_mes;
    float i_mes;
    float t_mes;
    float vee_bus_scale;
    float vcc_bus_scale;
    float u_24_scale;
    float i_scale;
    float t_scale;

    float i_lim;
    float i_lim_up_not_scale;
    float i_lim_down_not_scale;

    float u_24_lim;
    float vcc_lim;
    float vee_lim;
    float t_lim;

};

extern struct adc_t adc;

/*****************************************************************/
/* Définitions de grandeurs typiques                             */
/*****************************************************************/
#define ADC_MAX 4095
#define ADC_MIN 0
#define U_HIGH 3
#define U_LOW 0

#define ADC_MES_VCC 2
#define ADC_MES_VEE 5
#define ADC_MES_U24 4

/*****************************************************************/
/* Macro pour la mesure des ADCs                                 */
/*****************************************************************/
inline void start_soc0(void) {
    AdcaRegs.ADCSOCFRC1.bit.SOC0 = 1;
}

inline void start_soc7(void) {
    AdcaRegs.ADCSOCFRC1.bit.SOC7 = 1;
```

```
62 }
63
64 /*************************************************************/
65 /* Déclaration des prototypes de fonctions                   */
66 /*************************************************************/
67 void InitADC(void);
68 void mesADC(void);
69
```

```c
/*****************************************************************/
/*   Module de Conversion Statique                       adc.c   */
/*                                                               */
/*   Travail de Bachelor                                         */
/*                                                               */
/*   HES-SO VALAIS / WALLIS                        13.07.2016    */
/*                                                               */
/*   AUTHOR : DUBOSSON MAXIME                                    */
/*   PROFESSOR : BARRADE PHILIPPE                                */
/*                                                               */
/*****************************************************************/

#include "Global.h"

/*****************************************************************/
/* Initialisation des ADCs                                       */
/*****************************************************************/
void InitADC(void) {

    ENABLE_PROTECTED_REGISTER;

    /*****************************************************************/
    /* Configuration du mode des ADCs                                */
    /*****************************************************************/
    AdcSetMode(ADC_ADCA, ADC_RESOLUTION_12BIT, ADC_SIGNALMODE_SINGLE);
    AdcSetMode(ADC_ADCB, ADC_RESOLUTION_12BIT, ADC_SIGNALMODE_SINGLE);

    /*****************************************************************/
    /* Paramétrage du diviseur (prescale)                            */
    /*****************************************************************/
    AdcaRegs.ADCCTL2.bit.PRESCALE = 0x0000; //Sélection la division de l'horloge à 1
    AdcbRegs.ADCCTL2.bit.PRESCALE = 0x0000; //Sélection la division de l'horloge à 1

    /*****************************************************************/
    /* Paramétrage des interruptions continues                       */
    /*****************************************************************/
    AdcaRegs.ADCINTSEL1N2.bit.INT2CONT = 1;

    /*****************************************************************/
    /* Paramétrage des interruptions                                 */
    /*****************************************************************/
    AdcaRegs.ADCINTSEL1N2.bit.INT2E = 1; // active/désactive interrupt ADCAINT2
    AdcaRegs.ADCINTSEL3N4.bit.INT3E = 1; // active/désactive interrupt ADCAINT3
    AdcbRegs.ADCINTSEL3N4.bit.INT3E = 1; // active/désactive interrupt ADCBINT3

    AdcaRegs.ADCINTSEL1N2.bit.INT2SEL = 0x0; //Sélection du trigger ADCA pour le SOC0
    AdcaRegs.ADCINTSEL3N4.bit.INT3SEL = 0x6; //Sélection du trigger ADCA pour le SOC6
    AdcbRegs.ADCINTSEL3N4.bit.INT3SEL = 0x7; //Sélection du trigger ADCB pour le SOC7

    AdcaRegs.ADCSOCPRICTL.bit.SOCPRIORITY = 0x04; // Active en priorité SOC0 à SOC3
    AdcaRegs.ADCINTSOCSEL1.bit.SOC0 = 0x10; // ADCINT2 va trigger le SOC0

    /*****************************************************************/
    /* Paramétrage des canaux de mesures                             */
    /*****************************************************************/
    AdcaRegs.ADCSOC0CTL.bit.CHSEL = 0x3; //Sélection du canal ADCA3 pour le SOC0
    AdcaRegs.ADCSOC0CTL.bit.ACQPS = 0x39; //Assignation du temps d'acquisition pour le SOC0

    AdcaRegs.ADCSOC6CTL.bit.CHSEL = 0x2; //Sélection du canal ADCA2 pour le SOC6
    AdcaRegs.ADCSOC6CTL.bit.ACQPS = 0x039; //Assignation du temps d'acquisition pour le
  SOC6
```

```
61
62      AdcbRegs.ADCSOC7CTL.bit.CHSEL = 0x2; //Sélection du canal ADCB2 pour le SOC7
63      AdcbRegs.ADCSOC7CTL.bit.ACQPS = 0x039; //Assignation du temps d'acquisition pour le
    SOC7
64
65      /*****************************************************************/
66      /* Activation des ADCs                                           */
67      /*****************************************************************/
68      AdcaRegs.ADCCTL1.bit.ADCPWDNZ = 1;
69      AdcbRegs.ADCCTL1.bit.ADCPWDNZ = 1;
70      //DELAY_US(1000);
71
72      DISABLE_PROTECTED_REGISTER;
73
74
75  }
76
77  /*****************************************************************/
78  /* Fonction de mesure des ADCs                                   */
79  /*****************************************************************/
80  void mesADC(void) {
81
82      /*****************************************************************/
83      /* Sélection du canal à mesurer                                  */
84      /*****************************************************************/
85      switch (adc_channel.channel) {
86      case 1:
87          ENABLE_PROTECTED_REGISTER;
88          AdcaRegs.ADCSOC6CTL.bit.CHSEL = ADC_MES_VCC;
89          DISABLE_PROTECTED_REGISTER;
90          break;
91      case 2:
92          ENABLE_PROTECTED_REGISTER;
93          AdcaRegs.ADCSOC6CTL.bit.CHSEL = ADC_MES_VEE;
94          DISABLE_PROTECTED_REGISTER;
95          break;
96      case 3:
97          ENABLE_PROTECTED_REGISTER;
98          AdcaRegs.ADCSOC6CTL.bit.CHSEL = ADC_MES_U24;
99          DISABLE_PROTECTED_REGISTER;
100         break;
101     default:
102         break;
103     }
104     AdcaRegs.ADCINTFLGCLR.bit.ADCINT3 = 1; //réinitialise le flag INT3
105
106     PieCtrlRegs.PIEACK.all = PIEACK_GROUP10; //Quittance du PIE pour le groupe 10
107
108     /*****************************************************************/
109     /* Démarre l'acquisition des mesures                             */
110     /*****************************************************************/
111     AdcaRegs.ADCSOCFRC1.bit.SOC6 = 1;
112     AdcbRegs.ADCSOCFRC1.bit.SOC7 = 1;
113 }
114
115
```

```
 1 /*****************************************************************/
 2 /*   Module de Conversion Statique                       Pwm.h   */
 3 /*                                                               */
 4 /*   Travail de Bachelor                                         */
 5 /*                                                               */
 6 /*   HES-SO VALAIS / WALLIS                        13.07.2016    */
 7 /*                                                               */
 8 /*   AUTHOR : DUBOSSON MAXIME                                    */
 9 /*   PROFESSOR : BARRADE PHILIPPE                                */
10 /*                                                               */
11 /*****************************************************************/
12
13 /*****************************************************************/
14 /* Définition de la structure ADC                                */
15 /*****************************************************************/
16 struct epwm_t {
17     uint16_t EPwm_DB_Direction;
18     uint16_t EPwmTimerIntCount;
19     uint32_t pwm;
20 };
21
22 extern struct epwm_t epwm6;
23 extern struct epwm_t epwm7;
24
25 /*****************************************************************/
26 /* Définitions de grandeurs typiques                             */
27 /*****************************************************************/
28 // Maximum Dead Band values
29 #define EPWM6_MAX_DB   1000
30 #define EPWM6_MIN_DB   0
31
32 // To keep track of which way the Dead Band is moving
33 #define DB_UP   1
34 #define DB_DOWN 0
35
36 /*****************************************************************/
37 /* Déclaration des prototypes de fonctions                       */
38 /*****************************************************************/
39 void InitPWM_FAN(void);
40 void InitPWM_CONV(void);
41
42
```

```c
/*****************************************************************/
/*   Module de Conversion Statique                       pwm.c   */
/*                                                               */
/*   Travail de Bachelor                                         */
/*                                                               */
/*   HES-SO VALAIS / WALLIS                          13.07.2016  */
/*                                                               */
/*   AUTHOR : DUBOSSON MAXIME                                    */
/*   PROFESSOR : BARRADE PHILIPPE                                */
/*                                                               */
/*****************************************************************/

#include "Global.h"

/*****************************************************************/
/* Initialisation de la PWM pour le ventilateur                  */
/*****************************************************************/
void InitPWM_FAN(void) {

    ENABLE_PROTECTED_REGISTER;
    CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 0;
    DISABLE_PROTECTED_REGISTER;

    /*****************************************************************/
    /* Paramétrage du Time-Base (TB)                                 */
    /*****************************************************************/
    /*
     * Tpwm = 2*TBPRD *TBCLK --> TBCLK = 80ns
     * F = 23kHz -> Tpwm = 43us
     * TBPRD = Tpwm / TBCLK / 2 => 43/0.08/2 = 269
     */
    EPwm6Regs.TBPRD = 269;                          // Set timer period
    EPwm6Regs.TBPHS.bit.TBPHS = 0x0000;             // Phase is 0
    EPwm6Regs.TBCTR = 0x0000;                       // Clear counter

    // Setup TBCLK
    EPwm6Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Count up
    EPwm6Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Disable phase loading
    EPwm6Regs.TBCTL.bit.HSPCLKDIV = TB_DIV2;        // Clock ratio to SYSCLKOUT
    EPwm6Regs.TBCTL.bit.CLKDIV = TB_DIV2;

    /*****************************************************************/
    /* Parmétrage du Counter Compare (CC)                            */
    /*****************************************************************/
    EPwm6Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;   // Load registers every ZERO
    EPwm6Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm6Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
    EPwm6Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;

    EPwm6Regs.CMPA.bit.CMPA = 20;
    //EPwm6Regs.CMPB.bit.CMPB = 1000;

    /*****************************************************************/
    /* Paramétrage de l'Action Qualifier (AQ)                        */
    /*****************************************************************/
    EPwm6Regs.AQCTLA.bit.CAU = AQ_SET;             // Set PWM6A on Zero
    EPwm6Regs.AQCTLA.bit.CAD = AQ_CLEAR;

    EPwm6Regs.AQCTLB.bit.CAU = AQ_CLEAR;          // Set PWM6A on Zero
    EPwm6Regs.AQCTLB.bit.CAD = AQ_SET;

```

```
62      /*************************************************************/
63      /* Parmétrage du Dead Band (DB)                              */
64      /*************************************************************/
65      /*
66       EPwm6Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;
67       EPwm6Regs.DBCTL.bit.POLSEL = DB_ACTV_LO;
68       EPwm6Regs.DBCTL.bit.IN_MODE = DBA_ALL;
69       EPwm6Regs.DBRED.bit.DBRED = EPWM6_MIN_DB;
70       EPwm6Regs.DBFED.bit.DBFED = EPWM6_MIN_DB;
71       epwm6.EPwm_DB_Direction = DB_UP;
72       */
73
74      /*************************************************************/
75      /* Paramétrage de l'Event Trigger (ET)                       */
76      /*************************************************************/
77      EPwm6Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;    // Select INT on Zero event
78      EPwm6Regs.ETSEL.bit.INTEN = 1;               // Enable INT
79      EPwm6Regs.ETPS.bit.INTPRD = ET_3RD;          // Generate INT on 3rd event
80
81      // enable PWM6
82      CpuSysRegs.PCLKCR2.bit.EPWM6 = 1;
83
84      ENABLE_PROTECTED_REGISTER;
85      CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;
86      DISABLE_PROTECTED_REGISTER;
87
88  }
89
90  /*************************************************************/
91  /* Initialisation de la PWM pour les interrupteurs           */
92  /*************************************************************/
93  void InitPWM_CONV(void) {
94
95      ENABLE_PROTECTED_REGISTER;
96      CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 0;
97      DISABLE_PROTECTED_REGISTER;
98
99      /*************************************************************/
100     /* Paramétrage du Time-Base (TB)                             */
101     /*************************************************************/
102     /*
103      * Tpwm = 2*TBPRD *TBCLK --> TBCLK = 80ns
104      * F = 20kHz -> Tpwm = 50us
105      * TBPRD = Tpwm / TBCLK / 2 => 50/0.08/2 = 312
106      */
107     EPwm7Regs.TBPRD = 312;                       // Set timer period
108     EPwm7Regs.TBPHS.bit.TBPHS = 0x0000;           // Phase is 0
109     EPwm7Regs.TBCTR = 0x0000;                     // Clear counter
110
111     // Setup TBCLK
112     EPwm7Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Count up
113     EPwm7Regs.TBCTL.bit.PHSEN = TB_DISABLE;        // Disable phase loading
114     EPwm7Regs.TBCTL.bit.HSPCLKDIV = TB_DIV2;       // Clock ratio to SYSCLKOUT
115     EPwm7Regs.TBCTL.bit.CLKDIV = TB_DIV2;
116
117     /*************************************************************/
118     /* Parmétrage du Counter Compare (CC)                        */
119     /*************************************************************/
120     EPwm7Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;   // Load registers every ZERO
121     EPwm7Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
122     EPwm7Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
```

```
123     EPwm7Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;
124
125     EPwm7Regs.CMPA.bit.CMPA = epwm7.pwm * EPwm7Regs.TBPRD / 100;
126
127     /*****************************************************************/
128     /* Paramétrage de l'Action Qualifier (AQ)                        */
129     /*****************************************************************/
130     EPwm7Regs.AQCTLA.bit.CAU = AQ_SET;            // Set PWM7A on Zero
131     EPwm7Regs.AQCTLA.bit.CAD = AQ_CLEAR;
132
133     EPwm7Regs.AQCTLB.bit.CAU = AQ_CLEAR;         // Set PWM7B on Zero
134     EPwm7Regs.AQCTLB.bit.CAD = AQ_SET;
135
136     /*****************************************************************/
137     /* Parmétrage du Dead Band (DB)                                  */
138     /*****************************************************************/
139     /*
140      EPwm7Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;
141      EPwm7Regs.DBCTL.bit.POLSEL = DB_ACTV_LO;
142      EPwm7Regs.DBCTL.bit.IN_MODE = DBA_ALL;
143      EPwm7Regs.DBRED.bit.DBRED = EPWM6_MIN_DB;
144      EPwm7Regs.DBFED.bit.DBFED = EPWM6_MIN_DB;
145      epwm7.EPwm_DB_Direction = DB_UP;
146      */
147
148     /*****************************************************************/
149     /* Paramétrage de l'Event Trigger (ET)                           */
150     /*****************************************************************/
151     EPwm7Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;    // Select INT on Zero event
152     EPwm7Regs.ETSEL.bit.INTEN = 1;               // Enable INT
153     EPwm7Regs.ETPS.bit.INTPRD = ET_3RD;          // Generate INT on 3rd event
154
155     // enable PWM7
156     CpuSysRegs.PCLKCR2.bit.EPWM7 = 1;
157
158     ENABLE_PROTECTED_REGISTER;
159     CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;
160     DISABLE_PROTECTED_REGISTER;
161
162 }
163
164
```

```
/*****************************************************************/
/*   Module de Conversion Statique                  Control.h    */
/*                                                               */
/*   Travail de Bachelor                                         */
/*                                                               */
/*   HES-SO VALAIS / WALLIS                        13.07.2016    */
/*                                                               */
/*   AUTHOR : DUBOSSON MAXIME                                    */
/*   PROFESSOR : BARRADE PHILIPPE                                */
/*                                                               */
/*****************************************************************/

/*****************************************************************/
/* Définition de la structure pwm                                */
/*****************************************************************/
struct pwm_t {
    uint16_t pwm;
    uint16_t speed;
    uint16_t tach;
    uint16_t tach_old;
    uint16_t count;
};

extern struct pwm_t ventil;

/*****************************************************************/
/* Déclaration des prototypes de fonctions                       */
/*****************************************************************/
float convert(float PlageMax, float PlageMin, float currentValue);
float convert_temp(float currentValue);
void updateValue(void);
void updateErreur(void);
void regulFan(void);
void InitValue(void);
void activeOutEnable(void);

```

```c
/*****************************************************************/
/*   Module de Conversion Statique                  control.c    */
/*                                                               */
/*   Travail de Bachelor                                         */
/*                                                               */
/*   HES-SO VALAIS / WALLIS                        13.07.2016    */
/*                                                               */
/*   AUTHOR : DUBOSSON MAXIME                                    */
/*   PROFESSOR : BARRADE PHILIPPE                                */
/*                                                               */
/*****************************************************************/

#include "Global.h"

/*****************************************************************/
/* Régulation du ventilateur avec un comparateur à 5 niveaux     */
/*****************************************************************/
void regulFan(void) {

    if (adc.t_scale >= 70) {
        ventil.pwm = 100;
    } else if (adc.t_scale < 70 && adc.t_scale >= 60) {
        ventil.pwm = 75;
    } else if (adc.t_scale < 55 && adc.t_scale >= 45) {
        ventil.pwm = 50;
    } else if (adc.t_scale < 40 && adc.t_scale >= 35) {
        ventil.pwm = 25;
    } else if (adc.t_scale < 30) {
        ventil.pwm = 0;
    }

    EPwm6Regs.CMPA.bit.CMPA = EPwm6Regs.TBPRD * ventil.pwm / 100;

}

/*****************************************************************/
/* Mise à jour des valeurs mesurées                              */
/*****************************************************************/
void updateValue(void) {

    adc.i_scale = -1.67 * (I_MAX - convert(I_MAX, I_MIN, (adc.i_mes + 15)));
    adc.vcc_bus_scale = convert(VCC_MAX, 0, adc.vcc_bus_mes);
    adc.vee_bus_scale = convert(VEE_MIN, 0, adc.vee_bus_mes);
    adc.u_24_scale = convert(U_MAX, U_MIN, adc.u_24_mes);
    adc.t_scale = convert_temp(adc.t_mes);
    EPwm7Regs.CMPA.bit.CMPA = epwm7.pwm * EPwm7Regs.TBPRD / 100; //Mise à jour PWM

    /*****************************************************************/
    /* Transfert des grandeurs dans le tableau de paramètres / COM   */
    /*****************************************************************/
    param.all[1] = adc.vcc_bus_scale;
    param.all[2] = adc.vee_bus_scale;
    param.all[3] = adc.i_scale;
    param.all[4] = adc.u_24_scale;
    param.all[5] = adc.t_scale;
    epwm7.pwm = param.all[6];
    param.all[7] = param.all[6];

}

/*****************************************************************/
```

```
62 /* Conversion de la grandeur mesurée selon la plage              */
63 /*****************************************************************/
64 float convert(float PlageMax, float PlageMin, float currentValue) {
65
66     return currentValue * (PlageMax - PlageMin) / (ADC_MAX - ADC_MIN);
67
68 }
69
70 /*****************************************************************/
71 /* Mise à l'échelle de la mesure de température                  */
72 /*****************************************************************/
73 float convert_temp(float currentValue) {
74     return ((currentValue + 5) * 3 / 4.095 - 1566) / -3.927 + 25;
75 }
76 /*****************************************************************/
77 /* Initialisation des valeurs par défaut                         */
78 /*****************************************************************/
79 void InitValue(void) {
80
81     //Limites de sécurité
82     adc.i_lim = 20;
83     adc.i_lim_up_not_scale = (ADC_MAX / 2) + adc.i_lim * ADC_MAX / 2 / I_MAX;
84     adc.i_lim_down_not_scale = (ADC_MAX / 2) - adc.i_lim * ADC_MAX / 2 / I_MAX;
85     adc.u_24_lim = 22;
86     adc.vcc_lim = 400;
87     adc.vee_lim = 400;
88     adc.t_lim = 80;
89
90     //Réinisialisation des erreurs
91     err.Err_appear = 0;
92     err.Err_disappear = 0;
93     err.erreur = 0;
94     err.err_set = 0;
95
96     //Réinisialisation de la ventilation
97     ventil.pwm = 0;
98     ventil.speed = 1;
99     ventil.count = 0;
100    ventil.tach = 0;
101
102    //Sélection du premier canal de mesure
103    adc_channel.channel = 1;
104
105    //Réinisialisation des timers
106    count.timer_adc = 0;
107    count.start_adc = 0;
108    count.timer_update = 0;
109    count.start_update = 0;
110
111    //Réinisialisation de la PWM
112    epwm6.EPwmTimerIntCount = 0;
113    epwm6.EPwm_DB_Direction = 0;
114
115    vardebug.state = VARDEBUG_IDLE_STATE;
116
117    output_enable(OE_DISABLE);
118
119 }
120
121 /*****************************************************************/
122 /* Activation des sorties de la CPLD (DSP_OE)   output enable    */
```

```
123 /*************************************************************/
124 void activeOutEnable(void) {
125 
126     if (CONFIG_4) {
127         output_enable(OE_ENABLE);
128     } else {
129         output_enable(OE_DISABLE);
130     }
131 }
132 
```

```
1 /*****************************************************************/
2 /*   Module de Conversion Statique                  Alarme.h     */
3 /*                                                               */
4 /*   Travail de Bachelor                                         */
5 /*                                                               */
6 /*   HES-SO VALAIS / WALLIS                        13.07.2016    */
7 /*                                                               */
8 /*   AUTHOR : DUBOSSON MAXIME                                    */
9 /*   PROFESSOR : BARRADE PHILIPPE                                */
10 /*                                                              */
11 /*****************************************************************/
12
13 /*****************************************************************/
14 /* Définition de la structure erreur                            */
15 /*****************************************************************/
16 struct err_t {
17
18     uint16_t erreur;
19     uint16_t err_set;
20     uint16_t Err_disappear;
21     uint16_t Err_appear;
22     uint16_t erreur_old;
23     uint16_t overload;
24
25 };
26
27 extern struct err_t err;
28
29 /*****************************************************************/
30 /* Définitions de grandeurs typiques                            */
31 /*****************************************************************/
32 #define ERR_I  1
33 #define ERR_VCC  2
34 #define ERR_VEE  3
35 #define ERR_U_24  4
36 #define ERR_OL_1  5
37 #define ERR_OL_2  6
38 #define ERR_T 7
39
40 /*****************************************************************/
41 /* Déclaration des prototypes de fonctions                      */
42 /*****************************************************************/
43 void set_bit_erreur(uint16_t position);
44 void clear_bit_erreur(uint16_t position);
45 void compare(void);
46
```

```
/*****************************************************************/
/*   Module de Conversion Statique                    alarme.c    */
/*                                                                */
/*   Travail de Bachelor                                          */
/*                                                                */
/*   HES-SO VALAIS / WALLIS                          13.07.2016   */
/*                                                                */
/*   AUTHOR : DUBOSSON MAXIME                                     */
/*   PROFESSOR : BARRADE PHILIPPE                                 */
/*                                                                */
/*****************************************************************/

#include "Global.h"

/*****************************************************************/
/* Fonction de mise à jour des erreurs                           */
/*****************************************************************/
void updateErreur(void) {
    /*
     * Gestion des Erreurs
     * Err_appear : erreur apparaissante
     * Err_diseppear : erreur disparue
     * erreur : erreur présente non quittancées
     */

    /*****************************************************************/
    /* Comparaisons des erreurs                                      */
    /*****************************************************************/
    compare();

    /*****************************************************************/
    /* Des erreurs sont apparues mais des erreurs étaient présentes  */
    /*****************************************************************/
    if (err.erreur != 0 & err.erreur != err.erreur_old) {
        err.Err_appear = 1;
    }

    /*****************************************************************/
    /* Toutes les erreurs ont disparu                                */
    /*****************************************************************/
    if (err.erreur == 0 & err.err_set) {
        err.Err_disappear = 1;
    }

    /*****************************************************************/
    /* Des erreurs sont apparues                                     */
    /*****************************************************************/
    if (err.Err_appear) {
        err.Err_appear = 0;
        defaut(1);
        err.err_set = 1;
        err.erreur_old = err.erreur;
    }

    /*****************************************************************/
    /* Quittance des erreurs (si plus aucune erreur)                 */
    /*****************************************************************/
    if (err.Err_disappear & !CONFIG_3) {
        err.Err_disappear = 0;
        defaut(0);
        err.err_set = 0;
```

```
62          err.erreur_old = 0;
63      }
64      if (CONFIG_2) {
65          reset_ol(0);
66      } else {
67          reset_ol(1);
68          err.overload = 0;
69      }
70
71      /*****************************************************************/
72      /* Gestion de la led d'alimentation                              */
73      /*****************************************************************/
74      if (adc.u_24_scale > adc.u_24_lim) {
75          set_led_alim_ok(1);
76      } else {
77          set_led_alim_ok(0);
78      }
79
80      /*****************************************************************/
81      /* Gestion de la led d'erreur                                    */
82      /*****************************************************************/
83      if (err.erreur || err.overload) {
84          set_led_err(1);
85      } else {
86          set_led_err(0);
87      }
88
89      /*****************************************************************/
90      /* Gestion de la led ok ( plus aucune erreur)                    */
91      /*****************************************************************/
92      if (!DEFAUT) {
93          set_led_ok(1);
94      } else {
95          set_led_ok(0);
96      }
97  }
98
99  /*****************************************************************/
100 /* Mise à 1 du but pour l'erreur correspondante                  */
101 /*****************************************************************/
102 void set_bit_erreur(uint16_t position) {
103
104     err.erreur |= 1 << position;
105 }
106
107 /*****************************************************************/
108 /* Mise à 0 du but pour l'erreur correspondante                  */
109 /*****************************************************************/
110 void clear_bit_erreur(uint16_t position) {
111     err.erreur &= ~(1 << position);
112 }
113
114 /*****************************************************************/
115 /* Gestion des erreurs (comparaison des valeurs limites          */
116 /*****************************************************************/
117 void compare(void) {
118     // Erreur d'Overload 1
119     if (OL_S1_N) {
120         clear_bit_erreur(ERR_OL_1);
121     }
122     if (!OL_S1_N) {
```

```
123         set_bit_erreur(ERR_OL_1);
124         err.overload = 1;
125     }
126 
127 // Erreur d'Overload 2
128     if (OL_S2_N) {
129         clear_bit_erreur(ERR_OL_2);
130     }
131     if (!OL_S2_N) {
132         set_bit_erreur(ERR_OL_2);
133         err.overload = 1;
134     }
135 
136 // Limite de courant
137     if (adc.i_scale < adc.i_lim) {
138         clear_bit_erreur(ERR_I);
139     }
140     if (adc.i_scale > adc.i_lim) {
141         set_bit_erreur(ERR_I);
142     }
143 
144 // Limite de tension 24V
145     if (adc.u_24_scale > adc.u_24_lim) {
146         clear_bit_erreur(ERR_U_24);
147     }
148     if (adc.u_24_scale < adc.u_24_lim) {
149         set_bit_erreur(ERR_U_24);
150     }
151 
152 //Limite du bus DC VCC
153     if (adc.vcc_bus_scale < adc.vcc_lim) {
154         clear_bit_erreur(ERR_VCC);
155     }
156     if (adc.vcc_bus_scale > adc.vcc_lim) {
157         set_bit_erreur(ERR_VCC);
158     }
159 
160 //Limite du bus DC VEE
161     if (adc.vee_bus_scale < adc.vee_lim) {
162         clear_bit_erreur(ERR_VEE);
163     }
164     if (adc.vee_bus_scale > adc.vee_lim) {
165         set_bit_erreur(ERR_VEE);
166     }
167 
168 //Limite de température
169     if (adc.t_scale < adc.t_lim - 20) {
170         clear_bit_erreur(ERR_T);
171     }
172     if (adc.t_scale > adc.t_lim) {
173         set_bit_erreur(ERR_T);
174 
175     }
176 }
177 
```

```
 1 /*************************************************************/
 2 /*   Module de Conversion Statique                vardebug.h   */
 3 /*                                                             */
 4 /*   Travail de Bachelor                                       */
 5 /*                                                             */
 6 /*   HES-SO VALAIS / WALLIS                       13.07.2016   */
 7 /*                                                             */
 8 /*   AUTHOR : HES-SO VALAIS / WALLIS                           */
 9 /*   PROFESSOR : BARRADE PHILIPPE                              */
10 /*                                                             */
11 /*************************************************************/
12
13 #define VARDEBUG_MAX_REQUEST_DATA_LENGTH 256
14
15 #define VARDEBUG_MAX_REPLY_DATA_LENGTH 256
16
17 void init_vardebug (void);
18
19
20 /**
21  * states of the OSI 2 state machine
22  */
23 enum vardebug_state_t {
24     VARDEBUG_IDLE_STATE = 1,
25     VARDEBUG_RECEIVE_STATE = 2,
26     VARDEBUG_PROCESS_REPLY_STATE = 3,
27     VARDEBUG_REPLY_STATE = 4,
28
29     VARDEBUG_DROP_STATE = 5
30 };
31
32
33 enum vardebug_cmd_id_t {
34     VARDEBUG_NACK                   = 0x0,
35     VARDEBUG_WRITE_PARAM            = 0x1,
36     VARDEBUG_READ_PARAM             = 0x2,
37     VARDEBUG_WRITE_MEM              = 0x3,
38     VARDEBUG_READ_MEM               = 0x4,
39     VARDEBUG_READ_FIRMWARE_INFO     = 0x5,
40     VARDEBUG_RESET_DSP              = 0x8,
41     VARDEBUG_READ_HARDWARE_INFO     = 0xB
42
43 // command valid only for debug in RAM or if code builded for firmware
44 #if defined(BUILDED_FOR_RAM) || defined (BUILDED_FOR_FIRMWARE)
45     ,
46     VARDEBUG_READ_BOOTLOADER_INFO   = 0xC
47 #endif
48
49 };
50
51
52
53 struct vardebug_request_frame_t {
54     Uint16 packet_id;
55     enum vardebug_cmd_id_t cmd;
56     Uint16 length;
57     Uint16 data[VARDEBUG_MAX_REQUEST_DATA_LENGTH];
58     Uint16 checksum;
59     Uint16* data_ptr;
60 };
61
```

```
62 struct vardebug_reply_frame_t {
63     Uint16 header;
64     Uint16 packet_id;
65     Uint16 cmd;
66     Uint16 length;
67     Uint16 data[VARDEBUG_MAX_REPLY_DATA_LENGTH];
68     Uint16 checksum;
69     Uint16* data_ptr;
70 };
71
72 /**
73  * data for the vardebug code
74  */
75 struct vardebug_t {
76
77     enum vardebug_state_t state;
78
79     unsigned int current_char;
80
81     struct vardebug_request_frame_t request;
82
83     struct vardebug_reply_frame_t reply;
84
85     boolean reply_ready;
86
87     boolean make_reset;
88
89     Uint16 timeout_counter;
90
91 };
92
93 extern struct vardebug_t vardebug;
94
95
96 void process_vardebug (void);
97
98
99
100
101
102
```

```c
/*****************************************************************/
/*   Module de Conversion Statique                  vardebug.c   */
/*                                                               */
/*   Travail de Bachelor                                         */
/*                                                               */
/*   HES-SO VALAIS / WALLIS                        07.06.2016    */
/*                                                               */
/*   AUTHOR : HES-SO VALAIS / WALLIS                              */
/*   PROFESSOR : BARRADE PHILIPPE                                */
/*                                                               */
/*****************************************************************/

#include "Global.h"

/*************************** user configuration ***************************/

/**
 * bit rate in bit/s of the UART.
 *
 */
//#define VARDEBUG_BITRATE  9600.0L
//#define VARDEBUG_BITRATE  19200.0L
#define VARDEBUG_BITRATE    57600.0L
//#define VARDEBUG_BITRATE  115200.0L

/**
 * timeout time between 2 characters.
 * max is 2^16 * T_S, about 3 s for 20 kHz
 * a big value of 0.3 s is used for interactive, non realtime communication
 *
 * 1 start bit, 1 stop bit, 8 datas = 10 bits / char
 * 57600 / 10 = 5760 byte /s
 * 173 us / byte
 */
#define VARDEBUG_RECEIVE_TIMEOUT (300e-3L)

/*************************** local decaration ***************************/

/* global variables declared as extern */
struct vardebug_t vardebug;

#define HEADER_BYTE 0xAA

void vardebug_process_reply(void);

/************************* initialisation functions *************************/

#define VARDEBUG_SCI_REGS  ScicRegs

void init_vardebug(void) {

    VARDEBUG_SCI_REGS.SCICCR.all =
    BIT7 & 0 |  // one stop bit
            BIT6 |  // even parity
            BIT5 & 0 |  // no parity
            BIT4 & 0 |  // no loop back
            BIT3 & 0 |  // Idle-line mode protocol
            BIT2 |  // 111b = 8 data bits
            BIT1 |
            BIT0;

```

```
62      VARDEBUG_SCI_REGS.SCICTL1.all =
63      BIT7 & 0 |  // reserved
64              BIT6 & 0 |  // no error interrupt
65              BIT5 & 0 |  // make reset, do not affect config bits
66              BIT4 & 0 |  // reserved
67              BIT3 & 0 |  // enable transmit feature
68              BIT2 & 0 |  // do not sleep
69              BIT1 |  // transmitter enable
70              BIT0;   // receiver enable
71
72      /* set bautrate, see SPRU051B, sci ref guide, p. 38 */
73  #define BRR ( (LSPCLK_FREQ / (VARDEBUG_BITRATE * 8.0L)) - 1.0L)
74  //#define BRR 82
75      VARDEBUG_SCI_REGS.SCIHBAUD.all = (((Uint16) BRR) >> 8) & 0xFF;
76
77      VARDEBUG_SCI_REGS.SCILBAUD.all = ((Uint16) BRR) & 0xFF;
78
79      VARDEBUG_SCI_REGS.SCICTL2.all =
80      BIT7 & 0 |  // TXRDY, read-only
81              BIT6 & 0 |  // TXEMPY, read-only
82              BIT5 & 0 |  // bit5-2 reserved
83              BIT4 & 0 |
84              BIT3 & 0 |
85              BIT2 & 0 |
86              BIT1 & 0 |  // no rx interrupt
87              BIT0 & 0;   // no tx interrupt
88
89      VARDEBUG_SCI_REGS.SCIFFTX.all =
90      BIT15 & 0 | // reset fifo, do not affect config bits
91              BIT14 | // use fifo
92              BIT13 & 0 | // reset fifo
93              BIT12 & 0 | // bit12-8 read-only
94              BIT11 & 0 |
95              BIT10 & 0 |
96              BIT9 & 0 |
97              BIT8 & 0 |
98              BIT7 & 0 |  // read-only
99              BIT6 |  // clear interrupt flag
100             BIT5 & 0 |  // interrupt match disabled
101             BIT4 & 0 |  // interrupt level
102             BIT3 & 0 |
103             BIT2 & 0 |
104             BIT1 & 0 |
105             BIT0 & 0;
106
107     VARDEBUG_SCI_REGS.SCIFFRX.all =
108     BIT15 & 0 | // read-olny
109             BIT14 | // clear bit overflow bit
110             BIT13 & 0 | // reset fifo
111             BIT12 & 0 | // bit12-8 read-only
112             BIT11 & 0 |
113             BIT10 & 0 |
114             BIT9 & 0 |
115             BIT8 & 0 |
116             BIT7 & 0 |  // read-only
117             BIT6 |  // clear interrupt flag
118             BIT5 & 0 |  // interrupt match disabled
119             BIT4 & 0 |  // interrupt level
120             BIT3 & 0 |
121             BIT2 & 0 |
122             BIT1 & 0 |
```

```
123             BIT0 & 0;
124
125     VARDEBUG_SCI_REGS.SCIFFCT.all =
126     BIT15 & 0 | // read-olny
127             BIT14 | // clear autobaud flag
128             BIT13 & 0 | // disable auto-baud alignement
129             BIT12 & 0 | // bit12-8 reserved
130             BIT11 & 0 |
131             BIT10 & 0 |
132             BIT9 & 0 |
133             BIT8 & 0 |
134             BIT7 & 0 |  // bit7-0: inter char delay in baud
135             BIT6 & 0 |
136             BIT5 & 0 |
137             BIT4 & 0 |
138             BIT3 & 0 |
139             BIT2 & 0 |
140             BIT1 & 0 |
141             BIT0 & 0;
142
143     /* complete sequence if emulator supend */
144     VARDEBUG_SCI_REGS.SCIPRI.bit.FREESOFT = 1;
145
146     /*#ifdef BUILDED_FOR_RAM
147      VARDEBUG_SCI_REGS.SCIPRI.bit.FREE = 0;
148      #else
149      VARDEBUG_SCI_REGS.SCIPRI.bit.FREE = 1;
150      #endif
151      */
152     /* reenable sci after reset */
153     VARDEBUG_SCI_REGS.SCICTL1.bit.SWRESET = 1;
154     VARDEBUG_SCI_REGS.SCIFFTX.bit.SCIFFENA = 1;
155     VARDEBUG_SCI_REGS.SCIFFRX.bit.RXFIFORESET = 1;
156     VARDEBUG_SCI_REGS.SCIFFTX.bit.TXFIFORESET = 1;
157     VARDEBUG_SCI_REGS.SCIFFTX.bit.SCIRST = 1;   // fifo enable
158
159     vardebug.state = VARDEBUG_IDLE_STATE;
160
161     vardebug.reply.header = HEADER_BYTE;
162
163 }
164
165 /*************************** runtime functions ***************************/
166
167 inline void vardebug_write_char(char c) {
168     VARDEBUG_SCI_REGS.SCITXBUF.bit.TXDT = c;
169 }
170
171 inline char vardebug_read_char(void) {
172     return VARDEBUG_SCI_REGS.SCIRXBUF.bit.SAR & 0x00FF;
173 }
174
175 inline boolean vardebug_char_received(void) {
176     return VARDEBUG_SCI_REGS.SCIFFRX.bit.RXFFST;
177
178 }
179
180 inline boolean vardebug_can_send_char(void) {
181     return (VARDEBUG_SCI_REGS.SCIFFTX.bit.TXFFST == 0);
182 }
183
```

```
184 inline void vardebug_change_state(enum vardebug_state_t state) {
185     vardebug.state = state;
186 }
187 
188 char test = '0';
189 /*
190  * perf:
191  * with code in RAM (svn 479)
192  * min 67 cycles -> 0.47 us
193  * max 149 cycles -> 0.99 us
194  */
195 //#pragma CODE_SECTION(process_vardebug, "ramFuncs");
196 void process_vardebug(void) {
197     switch (vardebug.state) {
198 
199     /*-------------------------------------------------------------------------*/
200     case VARDEBUG_IDLE_STATE: {
201 
202         // reste en attente de char received, et commence la reception de trame si recoit
    HEADER_BYTE
203         if (vardebug_char_received()) {
204             vardebug.current_char = vardebug_read_char();
205 
206             if (vardebug.current_char == HEADER_BYTE) {
207                 vardebug.timeout_counter = 0;
208 
209                 /* init new request packet */
210                 vardebug.request.data_ptr = &vardebug.request.packet_id;
211                 vardebug.request.length = 0; /* start to minimal value */
212                 vardebug.request.checksum = HEADER_BYTE;
213 
214                 vardebug_change_state(VARDEBUG_RECEIVE_STATE);
215             }
216 
217         }
218 
219         // si demande de reset et buffer envoi vide
220         else if (vardebug.make_reset && SciaRegs.SCICTL2.bit.TXEMPTY) {
221 
222             /* write a wrong WDCHECK (0,1,0) create a reset */
223             ENABLE_PROTECTED_REGISTER;
224             WdRegs.WDCR.bit.WDCHK = 0;
225         }
226 
227     }
228         break;
229 
230         /*-------------------------------------------------------------------------*/
231     case VARDEBUG_RECEIVE_STATE: {
232         vardebug.timeout_counter++;
233 
234         if (vardebug_char_received()) {
235             vardebug.timeout_counter = 0;
236             vardebug.current_char = vardebug_read_char();
237             //vardebug_write_char(ScicRegs.SCIRXBUF.bit.SAR);
238 
239             /* if checksum received */
240             if (vardebug.request.data_ptr
241                     >= &vardebug.request.data[0] + vardebug.request.length) {
242 
243                 /* check checksum */
```

```
244                 if ((vardebug.request.checksum & 0xFF)
245                         == vardebug.current_char) {
246                     /* checksum ok  -> process reply */
247
248                     /* init data for process function */
249                     vardebug.request.data_ptr = &vardebug.request.data[0];
250                     vardebug.reply.data_ptr = &vardebug.reply.data[0];
251                     vardebug.reply.length = 0;
252                     vardebug.reply_ready = FALSE;
253                     vardebug_change_state(VARDEBUG_PROCESS_REPLY_STATE);
254                 } else {
255                     /* wrong checksum -> drop packet */
256                     vardebug_change_state(VARDEBUG_DROP_STATE);
257                 }
258             }
259             /* else store received data */
260             else {
261                 vardebug.request.checksum += vardebug.current_char;
262                 *vardebug.request.data_ptr++ = vardebug.current_char;
263                 //*vardebug.request.data_ptr++ = ScicRegs.SCIRXBUF.bit.SAR;
264
265                 /* and check for maximum length */
266                 if (vardebug.request.length > VARDEBUG_MAX_REQUEST_DATA_LENGTH) {
267                     vardebug_change_state(VARDEBUG_DROP_STATE);
268                 }
269             }
270         } else if (vardebug.timeout_counter
271                 > (Uint16) (VARDEBUG_RECEIVE_TIMEOUT / T_S_REGUL)) {
272             /* if timeout, return in idle, without sending nack */
273             vardebug_change_state(VARDEBUG_IDLE_STATE);
274         }
275
276     }
277         break;
278
279         /*-------------------------------------------------------------------------*/
280     case VARDEBUG_DROP_STATE: {
281         /* send nack */
282         vardebug.reply.packet_id = vardebug.request.packet_id;
283         vardebug.reply.cmd = VARDEBUG_NACK;
284         vardebug.reply.length = 0;
285         vardebug.reply.data_ptr = &vardebug.reply.header;
286         vardebug.reply.checksum = 0;
287         vardebug_change_state(VARDEBUG_REPLY_STATE);
288
289     }
290         break;
291
292         /*-------------------------------------------------------------------------*/
293     case VARDEBUG_PROCESS_REPLY_STATE: {
294
295         if (!vardebug.reply_ready) {
296             /* process function */
297             vardebug_process_reply();
298         } else {
299             /* init data to send reply */
300             vardebug.reply.packet_id = vardebug.request.packet_id;
301             vardebug.reply.cmd = vardebug.request.cmd;
302             vardebug.reply.data_ptr = &vardebug.reply.header;
303             vardebug.reply.checksum = 0;
304
```

```
305             vardebug_change_state(VARDEBUG_REPLY_STATE);
306         }
307
308     }
309         break;
310
311         /*-------------------------------------------------------------------------*/
312     case VARDEBUG_REPLY_STATE: {
313         if (vardebug_can_send_char()) {
314             if (vardebug.reply.data_ptr
315                     < (&vardebug.reply.data[0] + vardebug.reply.length)) {
316                 /* send char */
317                 vardebug.current_char = *vardebug.reply.data_ptr++;
318                 vardebug.reply.checksum += vardebug.current_char;
319                 vardebug_write_char(vardebug.current_char & 0xFF);
320             } else {
321                 /* end of frame, send checksum */
322                 vardebug_write_char(vardebug.reply.checksum & 0xFF);
323                 vardebug_change_state(VARDEBUG_IDLE_STATE);
324             }
325         }
326
327     }
328         break;
329
330     } /* end of switch */
331 }
332
333 /*
334  * you can expect for the first call that:
335  *      reply.length = 0
336  *      request.data_ptr = &request.data[0]
337  *      reply.data_ptr = &reply.data[0]
338  *
339  * To send the message, you have to set vardebug.reply_ready.
340  *
341  * You don't need to set the packet_id or vardebug.reply.cmd, it is set to the right
342  * value wenn you set vardebug.reply_ready to TRUE.
343  *
344  * To send a nack , call vardebug_change_state(VARDEBUG_DROP_STATE);
345  *
346  */
347 void vardebug_process_reply(void) {
348     switch (vardebug.request.cmd) {
349
350     /*-------------------------------------------------------------------------*/
351     case VARDEBUG_WRITE_PARAM: {
352
353         Uint16 index;       // of parameter, for 0 to 128
354         Uint32 value;       // of parameter
355
356         // get actual index of parameter and postincrement ptr for
357         // extraction of actual value of parameter
358         index = *vardebug.request.data_ptr++ & 0xFF;
359
360         // reconstruction of actual parameter-value (Uint32) by concatenation
361         // of the 4 received bytes
362         value = (Uint32) *vardebug.request.data_ptr++ & 0xFF;
363         value |= (Uint32) (*vardebug.request.data_ptr++ & 0xFF) << 8;
364         value |= (Uint32) (*vardebug.request.data_ptr++ & 0xFF) << 16;
365         value |= (Uint32) (*vardebug.request.data_ptr++ & 0xFF) << 24;
```

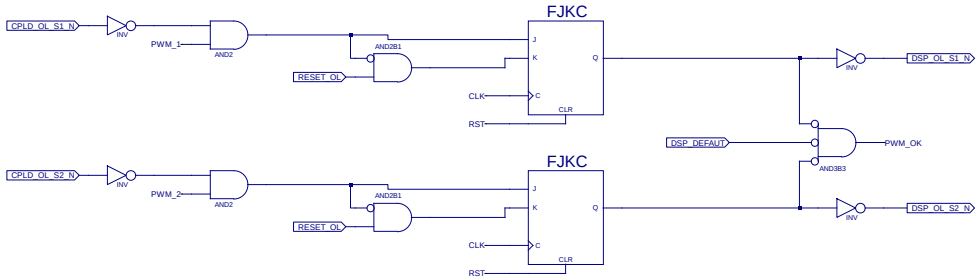
```
366
367         // set actual parameter
368         param.all[index] = value;
369         // decrement request frame by 5 cells:  1 cell for 1 parameter index
370         //                                      4 cells for 1 parameter value
371         vardebug.request.length -= 5;
372
373         // check if all parameters extracted (multiple parameter acquisitions)
374         vardebug.reply_ready = (vardebug.request.length == 0);
375
376     }
377         break;
378
379         /*---------------------------------------------------------------------*/
380     case VARDEBUG_READ_PARAM: {
381
382         Uint16 index;       // of parameter, for 0 to 128
383         Uint32 value;       // of parameter
384
385         // get actual index of parameter and postincrement ptr for
386         // extraction of next index
387         index = *vardebug.request.data_ptr++ & 0xFF;
388
389         // get parameter
390         value = param.all[index];
391
392         // fill reply data array with parameter, splitted in 4 bytes
393         *vardebug.reply.data_ptr++ = value & 0xFF;          // LSB
394         *vardebug.reply.data_ptr++ = (value >> 8) & 0xFF;   // 2nd byte
395         *vardebug.reply.data_ptr++ = (value >> 16) & 0xFF;  // 3rd byte
396         *vardebug.reply.data_ptr++ = (value >> 24) & 0xFF;  // MSB
397
398         // decrement request frame: 1 data cell for 1 parameter index
399         // increment reply frame:   4 data cells for 1 parameter (Uint32)
400         vardebug.reply.length += 4;     // increment reply length of 4 bytes
401         vardebug.request.length -= 1;   // decrement request data of 1 byte
402
403         // check if all parameters extracted (multiple parameter acquisitions)
404         vardebug.reply_ready = (vardebug.request.length == 0);
405
406     }
407         break;
408
409         /*---------------------------------------------------------------------*/
410 //  case VARDEBUG_WRITE_MEM: {
411 //   Uint16 ptr_tmp_calc;
412 //   Uint16* ptr;
413 //   volatile Uint16 value;
414 //
415 //   /* calc pointer to data */
416 //  ptr_tmp_calc = *vardebug.request.data_ptr++ & 0xFF;
417 //  ptr_tmp_calc |= *vardebug.request.data_ptr++ << 8;
418 //  ptr = (Uint16*) ptr_tmp_calc;
419 //
420 //  /* calc value */
421 //  value = *vardebug.request.data_ptr++ & 0xFF;
422 //  value |= (*vardebug.request.data_ptr++ << 8);
423 //
424 //  /* write to memory */
425 //  *ptr = value;
426 //
```

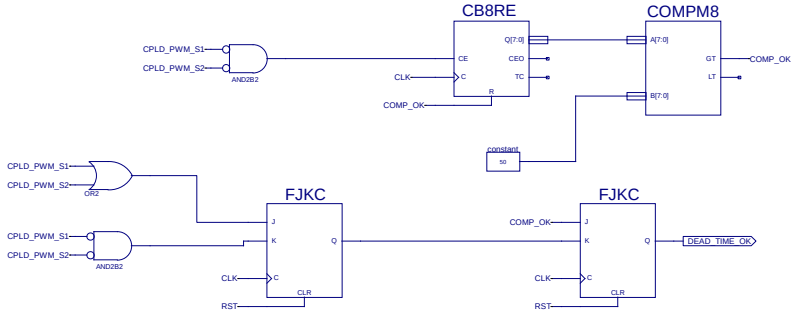
```
427 //  vardebug.request.length -= 4;
428 //
429 //  vardebug.reply_ready = (vardebug.request.length == 0);
430 //
431 //  }
432 //  break;
433         /*-------------------------------------------------------------------------*/
434 //  case VARDEBUG_READ_MEM: {
435 //      Uint16 ptr_tmp_calc;
436 //      Uint16* ptr;
437 //      volatile Uint16 value;
438 //
439 //      /* calc pointer to data */
440 //      ptr_tmp_calc = *vardebug.request.data_ptr++ & 0xFF;
441 //      ptr_tmp_calc |= *vardebug.request.data_ptr++ << 8;
442 //      ptr = (Uint16*) ptr_tmp_calc;
443 //
444 //      /* read data */
445 //      value = *ptr;
446 //
447 //      /* send LSB */
448 //      *vardebug.reply.data_ptr++ = value & 0xFF;
449 //
450 //      /* send MSB */
451 //      *vardebug.reply.data_ptr++ = (value >> 8) & 0xFF;
452 //
453 //      vardebug.reply.length += 2;
454 //      vardebug.request.length -= 2;
455 //
456 //      vardebug.reply_ready = (vardebug.request.length == 0);
457 //
458 //  }
459 //  break;
460         /*-------------------------------------------------------------------------*/
461     case VARDEBUG_RESET_DSP: {
462         /* send ack without data */
463         vardebug.reply_ready = TRUE;
464
465         /* reset after ack */
466         vardebug.make_reset = TRUE;
467     }
468         break;
469
470         /*-------------------------------------------------------------------------*/
471         /*case VARDEBUG_READ_FIRMWARE_INFO: {
472          const Uint16 info_index = vardebug.request.data[0] & 0xFF;
473
474          vardebug.reply.data[0] = firmware_info.all[info_index] & 0xFF;
475          vardebug.reply.data[1] = firmware_info.all[info_index] >> 8;
476          vardebug.reply.length = 2;
477
478          vardebug.reply_ready = TRUE;
479          }
480          break;
481          */
482         /*-------------------------------------------------------------------------*/
483         /*case VARDEBUG_READ_HARDWARE_INFO: {
484          const Uint16 info_index = vardebug.request.data[0] & 0xFF;
485
486          vardebug.reply.data[0] = hardware_info.all[info_index] & 0xFF;
487          vardebug.reply.data[1] = hardware_info.all[info_index] >> 8;
```

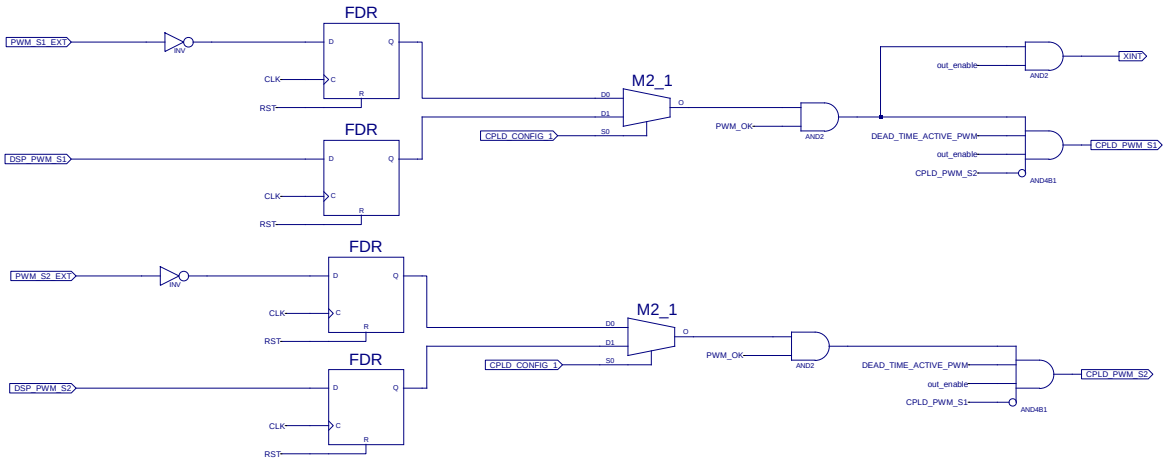
```
488          vardebug.reply.length = 2;
489
490          vardebug.reply_ready = TRUE;
491          }
492          break;
493
494          #if defined(BUILDED_FOR_RAM) || defined (BUILDED_FOR_FIRMWARE)
495          */
496         /*-------------------------------------------------------------------------*/
497         /*case VARDEBUG_READ_BOOTLOADER_INFO: {
498          const Uint16 info_index = vardebug.request.data[0] & 0xFF;
499
500          vardebug.reply.data[0] = bootloader_info.all[info_index] & 0xFF;
501          vardebug.reply.data[1] = bootloader_info.all[info_index] >> 8;
502          vardebug.reply.length = 2;
503
504          vardebug.reply_ready = TRUE;
505          }
506          break;
507
508          # endif
509          */
510         /*-------------------------------------------------------------------------*/
511     default: {
512         /* send nack */
513         vardebug.timeout_counter = 0xFFF0; /* create timeout to send nack immediately */
514         vardebug_change_state(VARDEBUG_DROP_STATE);
515     }
516
517     } /* end of switch */
518 }
519
520
```

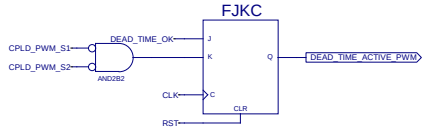
```
1 /*****************************************************************/
2 /*   Module de Conversion Statique              remote_ctrl.h    */
3 /*                                                               */
4 /*   Travail de Bachelor                                         */
5 /*                                                               */
6 /*   HES-SO VALAIS / WALLIS                        13.07.2016    */
7 /*                                                               */
8 /*   AUTHOR : DUBOSSON MAXIME                                    */
9 /*   PROFESSOR : BARRADE PHILIPPE                                */
10 /*                                                              */
11 /*****************************************************************/
12
13 struct param_fields_t {
14     uint32_t vcc;
15     uint32_t vee;
16     uint32_t ualim;
17     uint32_t test1;
18     uint32_t test2;
19     uint32_t pwm;
20     uint32_t pwm_value;
21 };
22
23 union param_t {
24
25     struct param_fields_t fields;
26     Uint32 all[32];
27 };
28 extern union param_t param;
29
```

ANNEXE 5

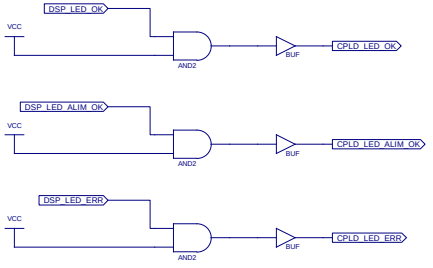

CPLD_OL_S1_N
INV
PWM_1
AND2
AND2B1
RESET_OL
FJKC
J
K
Q
C
CLR
CLK
RST
INV
DSP_OL_S1_N
DSP_DEFAUT
AND3B3
PWM_OK
CPLD_OL_S2_N
INV
PWM_2
AND2
AND2B1
RESET_OL
FJKC
J
K
Q
C
CLR
CLK
RST
INV
DSP_OL_S2_N

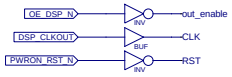

CPLD_PWM_S1
CPLD_PWM_S2
AND2B2
CB8RE
CE
CEO
TC
C
R
Q[7:0]
CLK
COMP_OK
COMPM8
A[7:0]
B[7:0]
GT
LT
COMP_OK
constant
50
CPLD_PWM_S1
CPLD_PWM_S2
OR2
CPLD_PWM_S1
CPLD_PWM_S2
AND2B2
FJKC
J
K
Q
C
CLR
CLK
RST
FJKC
COMP_OK
J
K
Q
C
CLR
CLK
RST
DEAD_TIME_OK


PWM_S1_EXT
INV
FDR
D
Q
C
R
CLK
RST
DSP_PWM_S1
FDR
D
Q
C
R
CLK
RST
M2_1
D0
D1
S0
O
CPLD_CONFIG_1
PWM_OK
AND2
out_enable
AND2
XINT
DEAD_TIME_ACTIVE_PWM
out_enable
CPLD_PWM_S2
AND4B1
CPLD_PWM_S1
PWM_S2_EXT
INV
FDR
D
Q
C
R
CLK
RST
DSP_PWM_S2
FDR
D
Q
C
R
CLK
RST
M2_1
D0
D1
S0
O
CPLD_CONFIG_1
PWM_OK
AND2
DEAD_TIME_ACTIVE_PWM
out_enable
CPLD_PWM_S1
AND4B1
CPLD_PWM_S2


CPLD_PWM_S1
CPLD_PWM_S2
AND2B2
DEAD_TIME_OK
FJKC
J
K
Q
C
CLR
CLK
RST
DEAD_TIME_ACTIVE_PWM


DSP_PWM_VENTIL
IBUF
CPLD_PWM_VENTIL


DSP_LED_OK
VCC
AND2
BUF
CPLD_LED_OK
DSP_LED_ALIM_OK
VCC
AND2
BUF
CPLD_LED_ALIM_OK
DSP_LED_ERR
VCC
AND2
BUF
CPLD_LED_ERR


OE_DSP_N
INV
out_enable
DSP_CLKOUT
BUF
CLK
PWRON_RST_N
INV
RST


DSP_TACH_VENTIL
IBUF
CPLD_TACH_VENTIL

```
# Constraint:   UCF file for sc_charger, rev 1.0
# Rel. prj.:    sc_charger, rev 1.0
# Device:       XC2C62A
# Package:      VQ44
# Author:       GEA, 17.11.2015
##########################################
# possible options for IO:
#
# NET "" LOC = "P" | FLOAT ;
# NET "" LOC = "P" | PULLUP ;
# NET "" LOC = "P" | KEEPER ;
# NET "" LOC = "P" | OPEN_DRAIN;
# NET "" LOC = "P" | SCHMITT_TRIGGER ;
# NET "" LOC = "P" | SLEW = SLOW ;
#
# or combined:
# NET ""  LOC = "P" | PULLUP  | SCHMITT_TRIGGER ;
# NET "signal_name" LOC = "Pinumber";

# NET "dsp_pb1"      LOC = "P31" ;
# NET "dsp_pb2"      LOC = "P30" ;
# NET "dsp_pb3"      LOC = "P29" ;
# NET "test"      LOC = "P28" ;

####################################
# PULLUP
####################################
# GENERAL
#

# EXTERNE

####################################
# SCHMITT_TRIGGER
####################################
#NET "watchdog_clk_ok" SCHMITT_TRIGGER;
####################################
# TIMING CONTRAINTS
####################################
NET "DSP_CLKOUT" TNM_NET = "DSP_CLKOUT";

#PACE: Start of Constraints generated by PACE
#PACE: Start of PACE I/O Pin Assignments


#  SET AS   :  OUTPUT
NET "DSP_OL_S1_N"               LOC = "P6"        ;

#  SET AS   :  OUTPUT
NET "DSP_OL_S2_N"               LOC = "P8"        ;

#  SET AS   :  INPUT
NET "DSP_DEFAUT"                LOC = "P38"       ;

#  SET AS   :  INPUT
NET "PWM_S2_EXT"                LOC = "P33"       ;

```

```
58   #  SET AS   :  INPUT
59   NET "PWM_S1_EXT"                LOC = "P32"       ;
60
61   #  SET AS   :  OUTPUT
62   NET "DSP_TACH_VENTIL"           LOC = "P3"        ;
63
64   #  SET AS   :  INPUT
65   NET "DSP_LED_OK"                LOC = "P12"       ;
66
67   #  SET AS   :  INPUT
68   NET "DSP_LED_ALIM_OK"           LOC = "P13"       ;
69
70   # SET AS :  INPUT
71   NET "DSP_LED_ERR"               LOC = "P14"       ;
72
73   #  SET AS   :  INPUT
74   NET "OE_DSP_N"                  LOC = "P5"        ;
75
76   #  SET AS   :  INPUT
77   NET "DSP_PWM_VENTIL"            LOC = "P43"       ;
78
79   # SET AS :  INPUT
80   NET "DSP_PWM_S2"                LOC = "P40"       ;
81
82   #  SET AS   :  OUTPUT
83   NET "DEAD_TIME_OK"              LOC = "P39"       ;
84
85   #  SET AS   :  INPUT
86   NET "DSP_PWM_S1"                LOC = "P42"       ;
87
88   #  SET AS   :  INPUT
89   #NET "PWM_RES_1"                   LOC = "P41"       ;
90
91   #  SET AS   :  INPUT
92   NET "CPLD_TACH_VENTIL"          LOC = "P29"       ;
93
94   # SET AS :  INPUT
95   #NET "CPLD_CONFIG_3"             LOC = "P23"       ;
96
97   #  SET AS   :  INPUT
98   #NET "CPLD_CONFIG_2"             LOC = "P27"       ;
99
100  #  SET AS   :  INPUT
101  NET "CPLD_CONFIG_1"             LOC = "P28"       ;
102
103  # SET AS :  INPUT
104  NET "PWRON_RST_N"               LOC = "P30"       ;
105
106  #  SET AS   :  INPUT
107  NET "RESET_OL"                  LOC = "P1"        ;
108
109  #  SET AS   :  INPUT
110  NET "CPLD_OL_S2_N"              LOC = "P19"       ;
111
112  #  SET AS   :  INPUT
113  NET "CPLD_OL_S1_N"              LOC = "P21"       ;
114
```

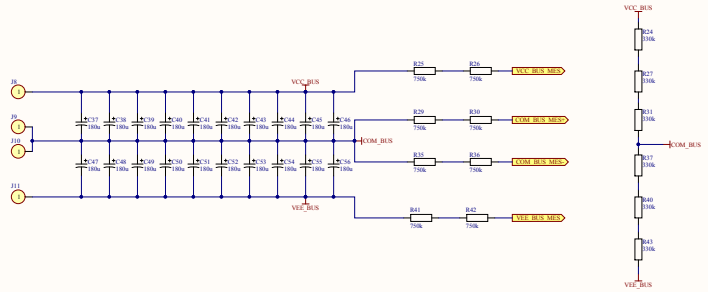
```
115   #  SET AS   :  OUTPUT
116   NET "CPLD_PWM_VENTIL"            LOC = "P31"       ;
117
118   #  SET AS   :  OUTPUT
119   NET "CPLD_LED_ERR"               LOC = "P37"       ;
120
121   #  SET AS   :  OUTPUT
122   NET "CPLD_LED_ALIM_OK"           LOC = "P36"       ;
123
124   #  SET AS   :  OUTPUT
125   NET "CPLD_LED_OK"                LOC = "P34"       ;
126
127   #  SET AS   :  OUTPUT
128   NET "CPLD_PWM_S2"                LOC = "P20"       ;
129
130   #  SET AS   :  OUTPUT
131   #NET "DEAD_TIME_OK"                 LOC = "P16"       ;
132
133   #  SET AS   :  OUTPUT
134   NET "CPLD_PWM_S1"                LOC = "P22"       ;
135
136   #  SET AS   :  OUTPUT
137   #NET "CPLD_PWM_RES_1"             LOC = "P18"       ;
138
139   # SET AS :  INPUT
140   NET "XINT"                       LOC = "P2"        ;
141
142   #SET AS  :  INPUT
143   NET "DSP_CLKOUT"                 LOC = "P44"       ;
144
145
146   #CPLD Name  :  P4_GND
147   #CPLD Name  :  P7_3V3
148   #CPLD Name  :  P9_CPLD_TDI
149   #CPLD Name  :  P10_CPLD_TMS
150   #CPLD Name  :  P11_CPLD_TCK
151   #CPLD Name  :  P15_1V8
152   #CPLD Name  :  P17_GND
153   #CPLD Name  :  P24_CPLD_TDO
154   #CPLD Name  :  P25_GND
155   #CPLD Name  :  P26_3V3
156   #CPLD Name  :  P35_3V3
157   #PACE: Start of PACE Area Constraints
158   #PACE: Start of PACE Prohibit Constraints
159   #PACE: End of Constraints generated by PACE
160   TIMESPEC TS_CLK = PERIOD "DSP_CLKOUT" 10 ns HIGH 50%;
161
```

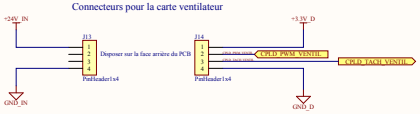
ANNEXE 6

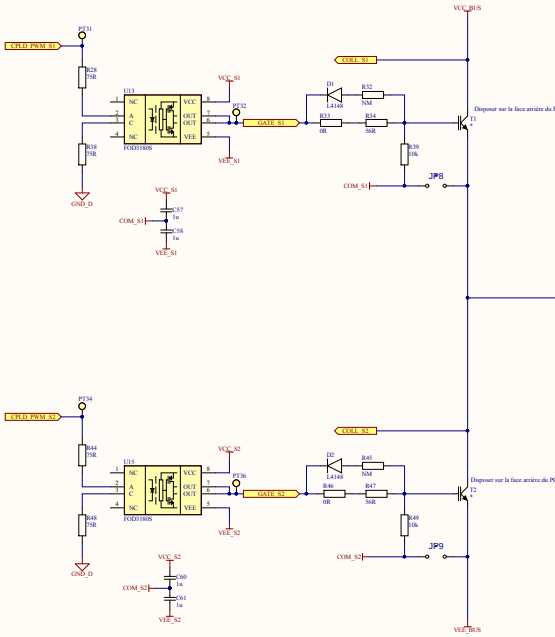
```
<?xml version="1.0" encoding="ISO-8859-1"?>

<visualisation processor="msp430fxxxx" >
  <friendlyname>Module de conversion statique </friendlyname>

    <view orientation="vertical" update-interval="50">
        <friendlyname>PWM Control</friendlyname>

        <section orientation="horizontal">
            <control component="Standard LCD Display" type="parameter" id="1"
            datatype="Uint32" >
                <friendlyname>VCC [V]</friendlyname>
                  <option name="fixed-width">150</option>
                  <option name="font-size">20</option>
                  <option name="fixed-height">100</option>
                  <option name="min">0</option>
                  <option name="max">1000</option>
            </control>

            <control component="Standard LCD Display" type="parameter" id="2"
            datatype="Uint32">
                <friendlyname>VEE [V]</friendlyname>
                  <option name="fixed-width">150</option>
                  <option name="font-size">20</option>
                  <option name="fixed-height">100</option>
                  <option name="min">0</option>
                  <option name="max">1000</option>
            </control>

            <control component="Standard LCD Display" type="parameter" id="3"
            datatype="Uint32">
                <friendlyname>Iout [A]</friendlyname>
                  <option name="fixed-width">150</option>
                  <option name="font-size">20</option>
                  <option name="fixed-height">100</option>
            </control>


            <control component="Standard LCD Display" type="parameter" id="4"
            datatype="Uint32">
                <friendlyname>Ualim [V]</friendlyname>
                  <option name="fixed-width">150</option>
                  <option name="font-size">20</option>
                  <option name="fixed-height">100</option>
            </control>

        </section>

        <section orientation="horizontal">



            <control component="Standard Bar Display" type="parameter" id="5"
            datatype="Uint32" >
              <friendlyname>IGBT Temp. [°C]</friendlyname>

              <option name="fixed-width">200</option>
```

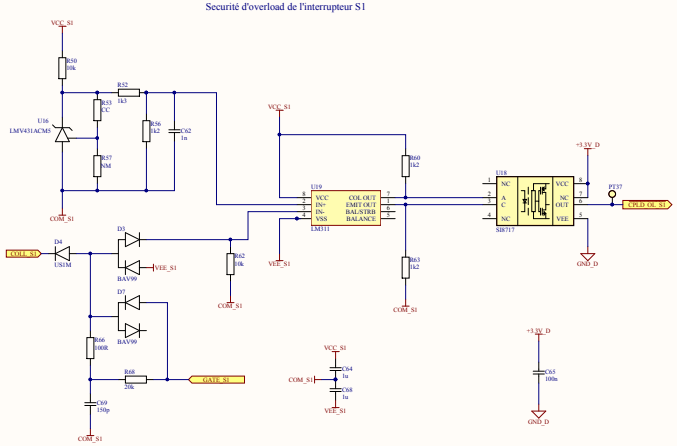
```
            <option name="min">-20</option>
            <option name="max">80</option>
            <option name="orientation">vertical</option>
            <option name="scale-position">left</option>
            <option name="fill-color">#00FF00</option>
            <option name="alarm-color">#FF0000</option>
            <option name="alarm-enabled">true</option>
            <option name="alarm-level">70</option>
            <option name="bar-width">50</option>

          </control>


    <control component="Standard Slider Control" type="parameter" id="6" datatype="Uint32">
      <friendlyname> Set PWM </friendlyname>
      <option name="min">10.0</option>
      <option name="max">90.0</option>
      <option name="step">1.0</option>
      <option name="realtime">true</option>

    </control>

    <control component="Standard LCD Display" type="parameter" id="7" datatype="Uint32" >
              <friendlyname>PWM VALUE</friendlyname>
                <option name="fixed-width">150</option>
                <option name="font-size">20</option>
                <option name="fixed-height">100</option>
                <option name="min">0</option>
                <option name="max">100</option>
          </control>



      </section>

  </view>


</visualisation>
```

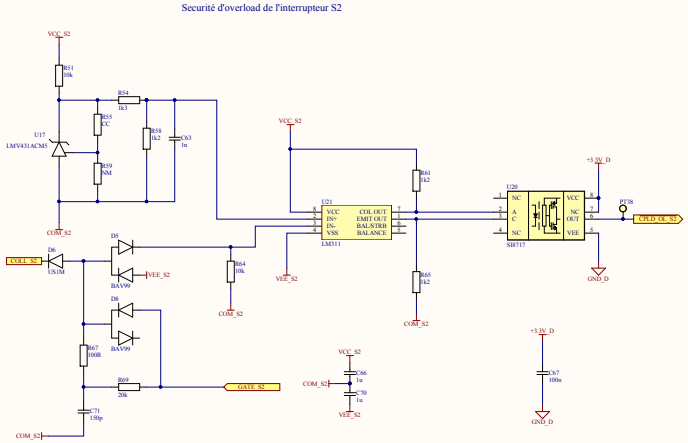
ANNEXE 7



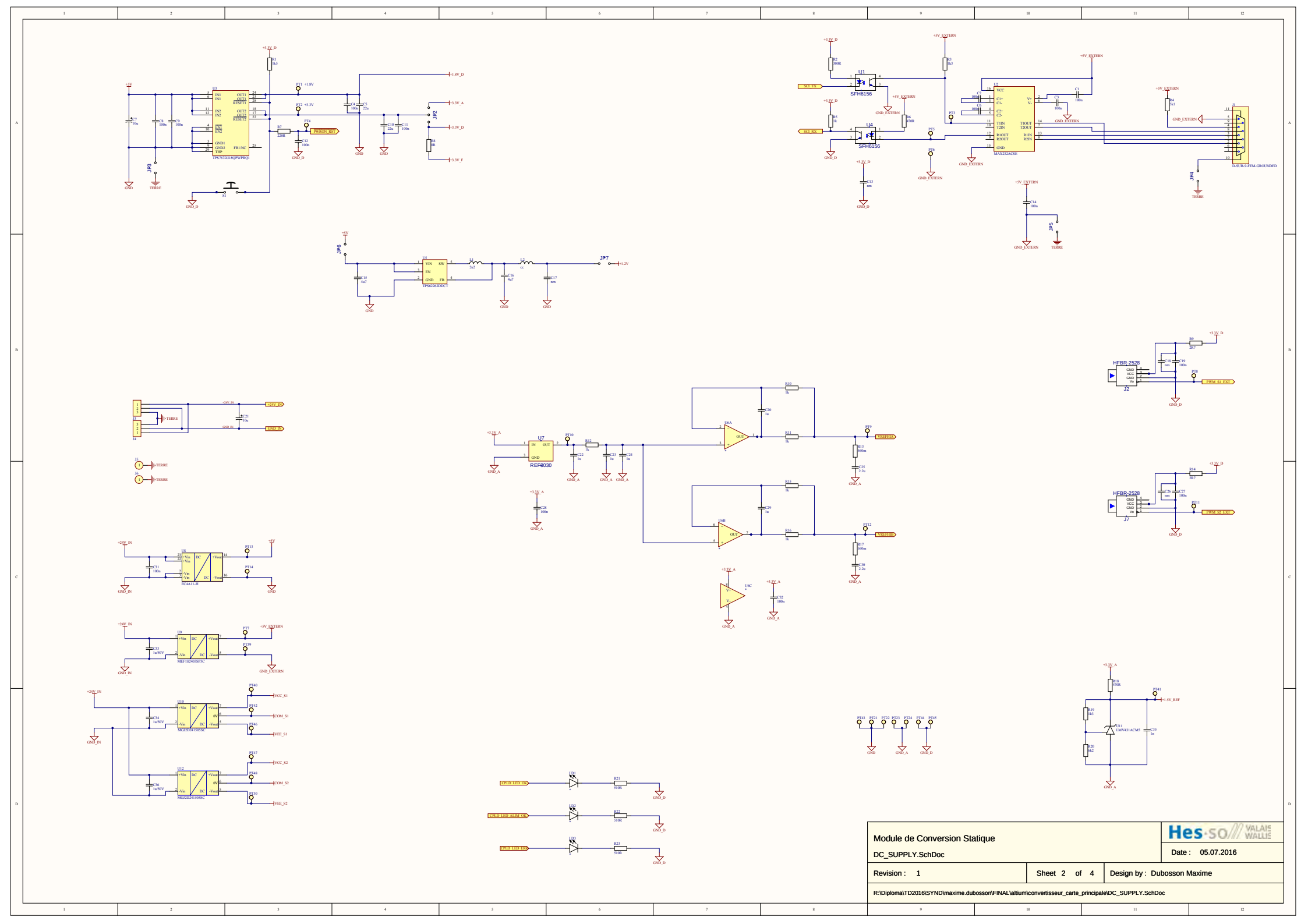


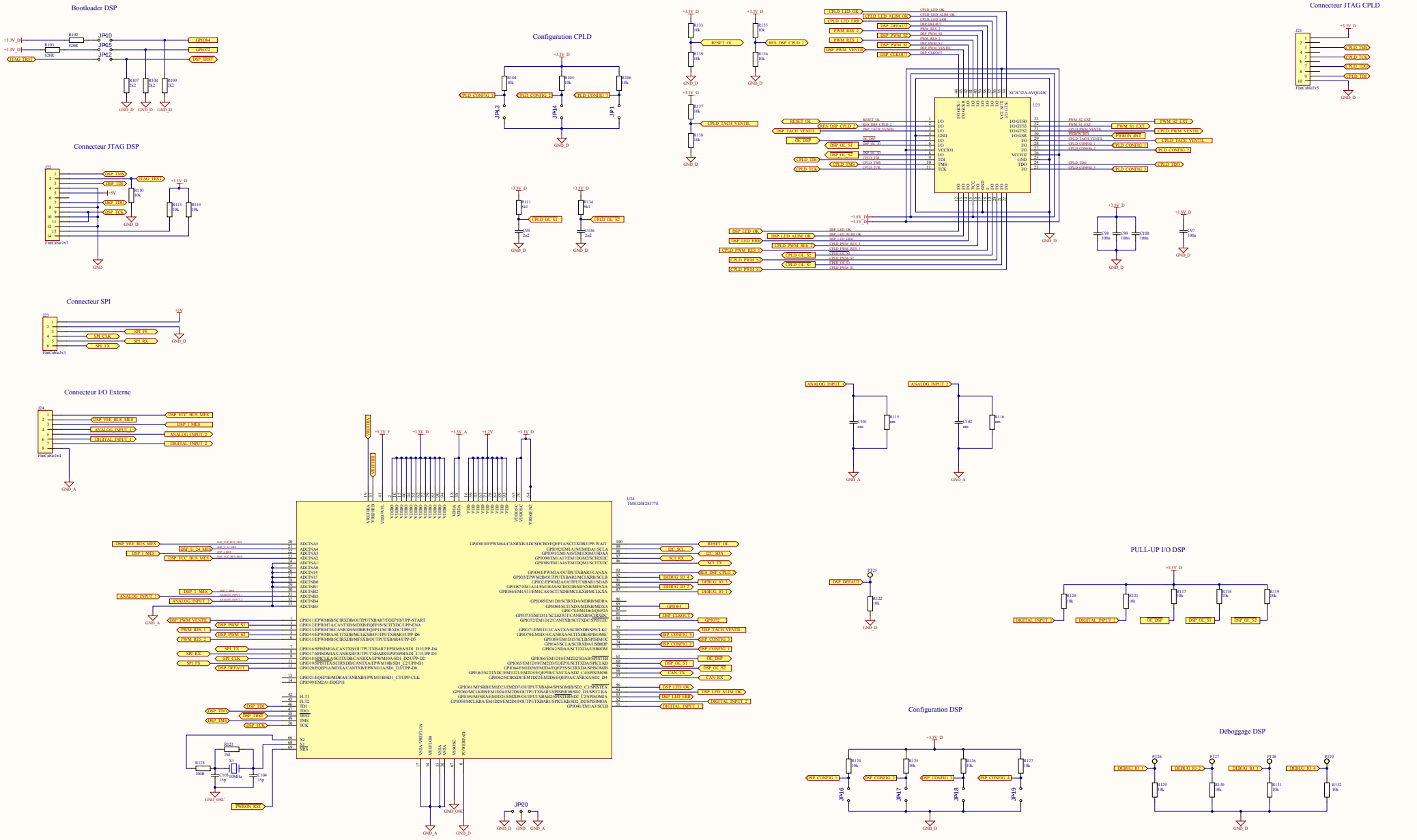






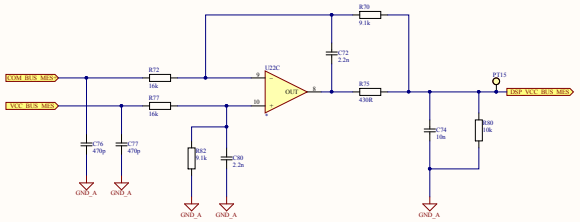
Module de Conversion Statique POWER.SchDoc		Hes·so VALAIS WALLIS Date : 05.07.2016
Revision : 1	Sheet 1 of 4	Design by : Dubosson Maxime
R:\Diploma\TD2016\SYND\maxime.dubosson\FINAL\altium\convertisseur_carte_principale\POWER.SchDoc		


Module de Conversion Statique
DC_SUPPLY.SchDoc
Date : 05.07.2016
Revision : 1
Sheet 2 of 4
Design by : Dubosson Maxime
R:\Diploma\TD2016\SYND\maxime.dubosson\FINAL\altium\convertisseur_carte_principale\DC_SUPPLY.SchDoc
Hes·so VALAIS WALLIS

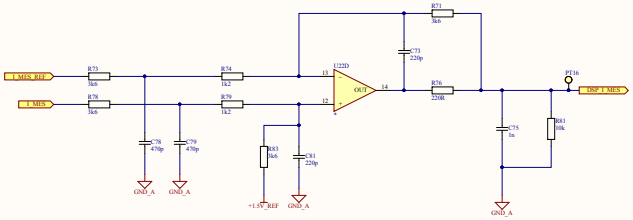


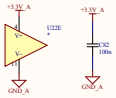
Module de Conversion Statique DSP_CPLD.SchDoc		Hes·so VALAIS WALLIS Date : 05.07.2016
Revision : 1	Sheet 3 of 4	Design by : Dubosson Maxime
R:\Diploma\TD2016\SYND\maxime.dubosson\FINAL\altium\convertisseur_carte_principale\DSP_CPLD.SchDoc		

Mesure de la tension positive du bus DC

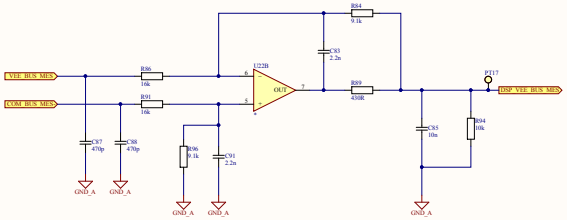


Mesure du courant de sortie

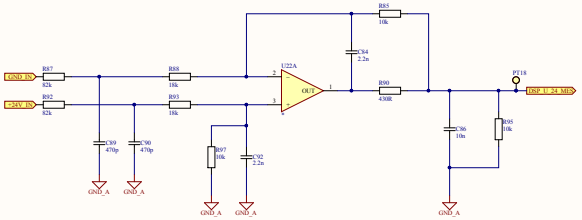




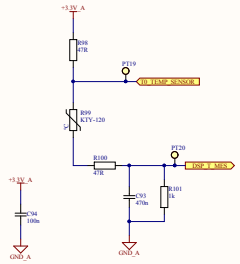
Mesure de la tension négative du bus DC



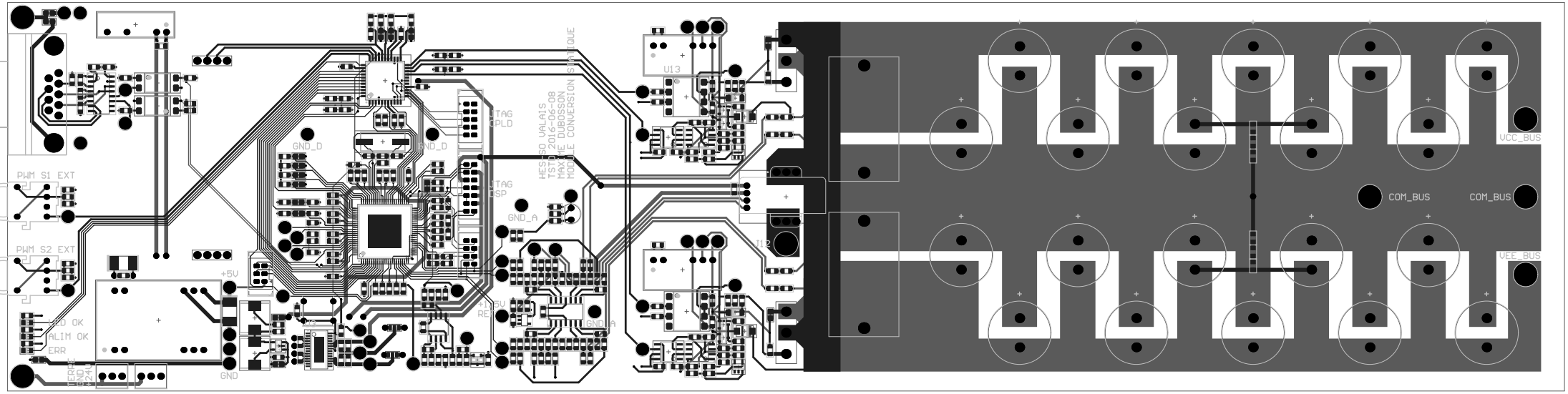
Mesure de la tension d'alimentation 24VDC



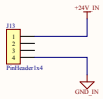
Mesure de la température

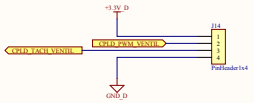


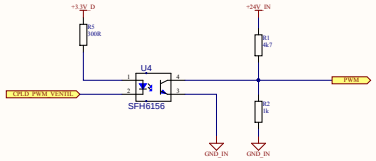
Module de Conversion Statique DSP_CPLD_MES.SchDoc		Hes·so VALAIS WALLIS Date : 05.07.2016
Revision : 1	Sheet 4 of 4	Design by : Dubosson Maxime
R:\Diploma\TD2016\SYND\maxime.dubosson\FINAL\altium\convertisseur_carte_principale\DSP_CPLD_MES.SchDoc		

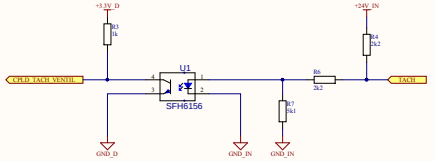

PWM S1 EXT
PWM S2 EXT
PWR OK
ALIM OK
ERR
+5V
GND
GND_D
GND_A
TAG
FPGA
DSP
U13
J12
COM_BUS
COM_BUS
VCC_BUS
VEE_BUS
HES SO VALAIS
TSTE
2016-06-08
MAXIME DUBOSSON
MODULE CONVERSION SPATIALE

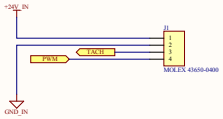
ANNEXE 8



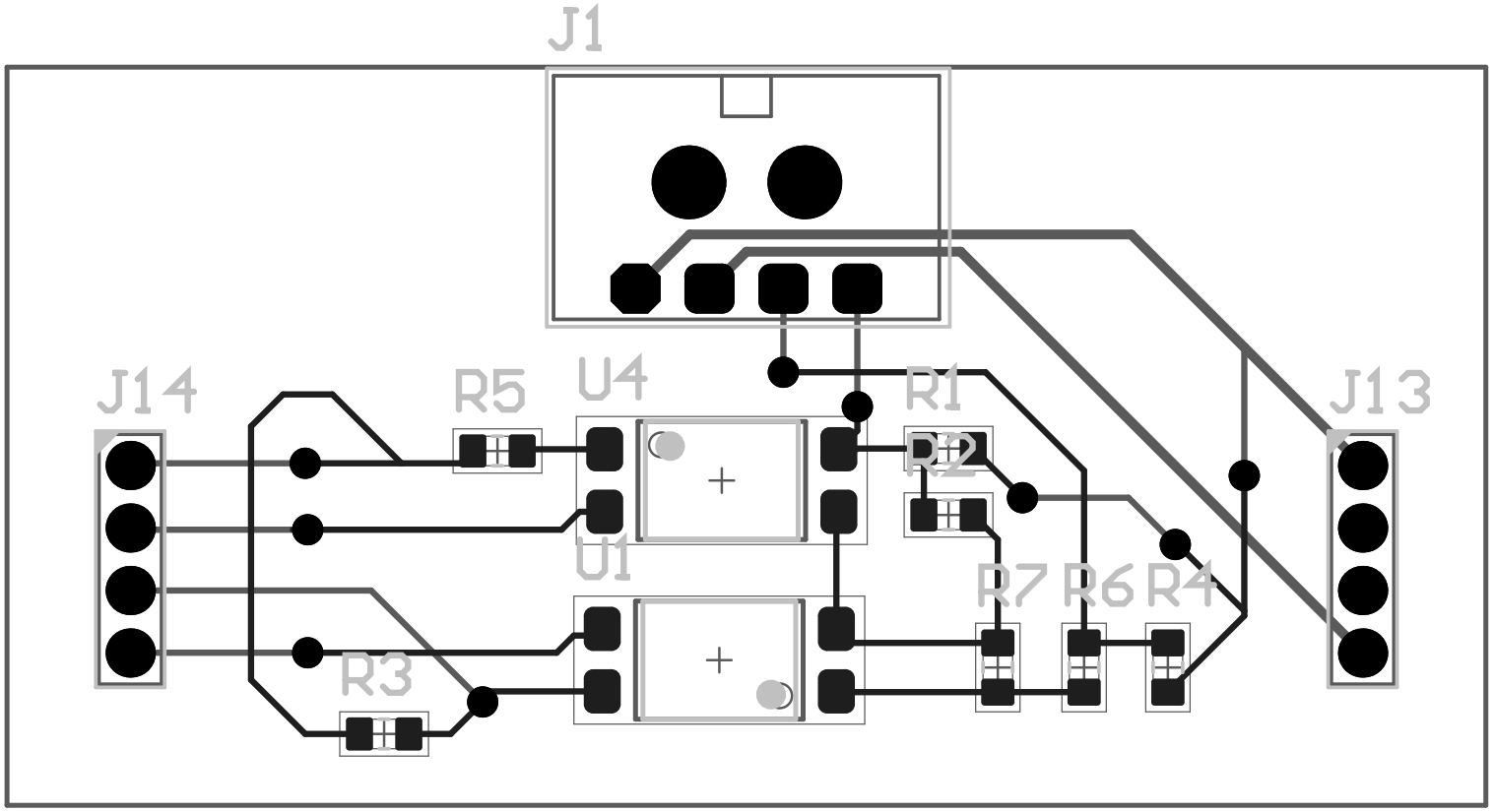




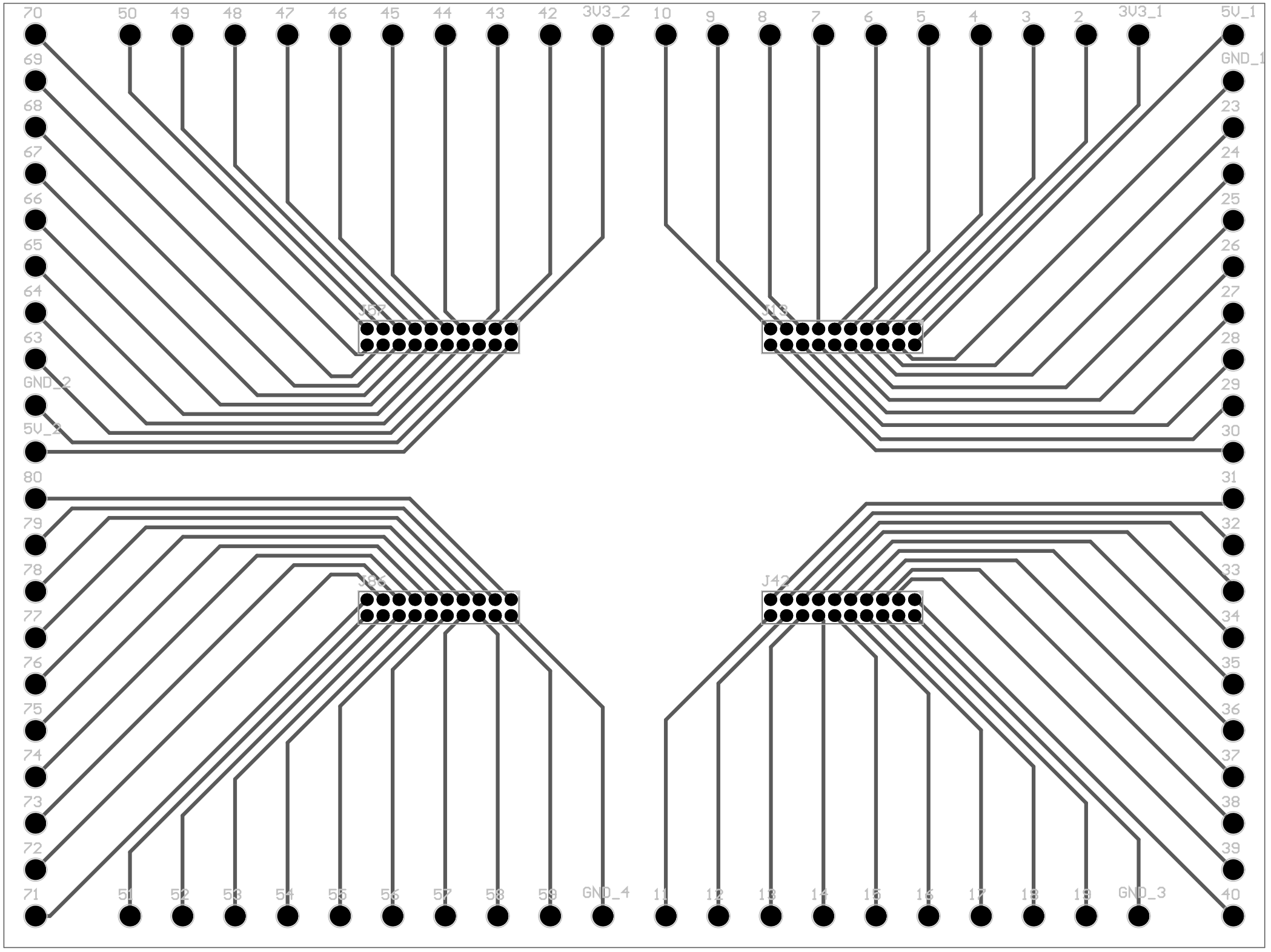




module_ventilateur.PrjPcb module.SchDoc		Connet opers file P:\Library\Altium\Lib\Templates\SchematicTemplates\1_HES_SO_VS_CMJN.png Date : DD.MM.YYYY
Revision : 1	Sheet 1 of 1	Design by : Dubosson Maxime
R:\Diploma\TD2016\SYND\maxime.dubosson\FINAL\altium\module_ventilateur\module.SchDoc		


J1
J14
R5
U4
R1
R2
J13
U1
R7
R6
R4
R3

ANNEXE 9


70
50
49
48
47
46
45
44
43
42
3V3_2
10
9
8
7
6
5
4
3
2
3V3_1
5V_1
69
GND_1
68
23
67
24
66
25
65
26
64
27
J57
J3
63
28
GND_2
29
5V_2
30
80
31
79
32
78
33
J36
J42
77
34
76
35
75
36
74
37
73
38
72
39
71
51
52
53
54
55
56
57
58
59
GND_4
11
12
13
14
15
16
17
18
19
GND_3
40

ANNEXE 10

Protocole de mesures et de tests

Description	Effectué le	Par
Contrôle visuel		
Contrôle visuel du PCB	21.06.2016	
Montage		
Montage des composants	22.06.2016	
Alimentations		
Mesure d'un court-circuit franc entre les bornes +24Vdc et 0Vdc	22.06.2016	
Mesure d'un court-circuit franc sur les sorties des alimentations	"	
Mesures des tensions d'entrées	"	
Mesures des tensions de sorties	"	
Mesure d'un court-circuit pour les tensions 3.3Vdc - 1.8Vdc - 1.2Vdc	"	
Mesures des tensions de sorties	"	
Mesure du Power Good et du bouton de Reset	"	
Mesure des deux tensions de référence VREFHIA et VREFHIB	"	
Mesure de la tension 1.5V_REF	"	
Puissances		
Mesure d'un court-circuit sur le bus DC	23.06.2016	
Mesure du circuit de puissance	"	
Mesure et test de commutation	"	
Mesure du circuit des condensateurs	"	
Microprocesseur et CPLD		
Mesure de leurs alimentations	23.06.2016	
Mesure de la fréquence d'oscillations	"	
Mesure des signaux avec pull-up	"	
Test des pins de config	"	
Mesure des signaux avec pull-down	"	
Signaux de mesures		
Mesures et tests des overload	24.06.2016	
Mesures et test des entrées analogiques	"	
Mesures et test de la température	"	
Connectivité		
Mesures et test de la communication UART	—	
Mesure et test des capteurs optiques	24.06.2016	
Mesure des bornes du ventilateur	"	
Divers		
Test du Boot Loader	"	
Test des LEDs de signalisations	"	
Global		
Essai du module complet	30.06.2016	
Final		
Module testé et approuvé	12.07.2016	

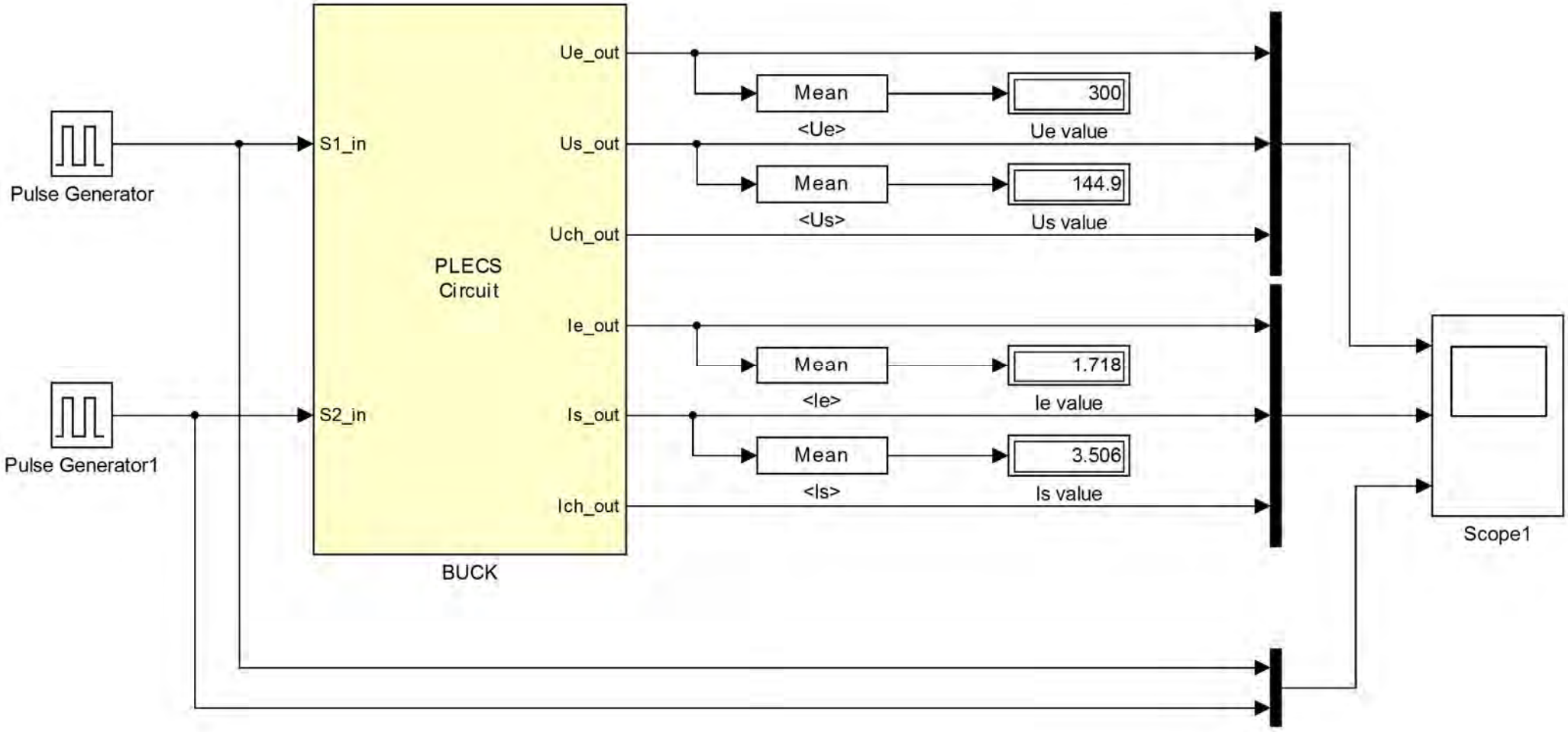
ANNEXE 11

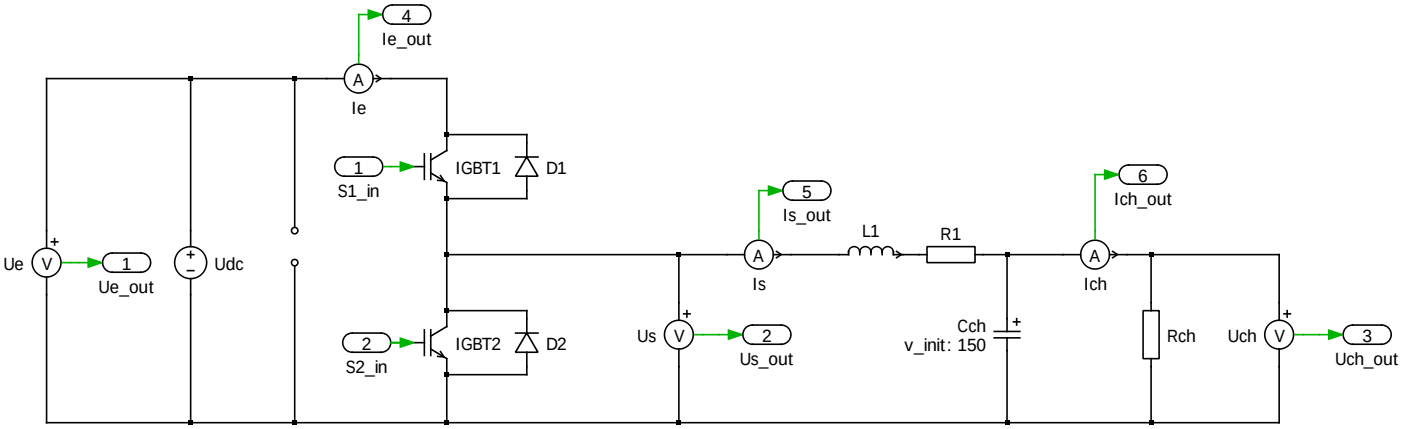
Protocole de test (DSP)

Description	Effectué le	Par
Fonctionnalités		
Initialisation des valeurs	06.07.2016	DP
Initialisation des Interruptions	"	"
Initialisation des ADCs	"	"
Initialisation des GPIOs	"	"
Initialisation des PWMs	"	"
Temps maximal entre chaque mesure de courant : 30s	"	"
Fréquence de mesures des autres canaux ADCs : 5kHz	"	"
Erreur en cas d'overload	"	"
Erreur en cas de courant trop important : 50 à 80 A	"	"
Erreur en cas de tension 24V trop faible : 22V	"	"
Erreur en cas de tension VCC et VEE trop importante : 450	"	"
Erreur en cas de tension VCC et VEE trop petite : —	"	"
Erreur en cas de température trop élevée : 80°C	"	"
Erreur si le ventilateur ne tourne pas et que la pwm est non nulle	"	"
Activation du bit de défaut en cas d'erreur	"	"
Activation de la led d'erreur en cas d'erreur	"	"
Quittance d'erreur possible que si aucune erreur n'est présente	"	"
Activation de la led verte si tout est ok	"	"
Contrôle du signal PWM pour la régulation du ventilateur	"	"
Régulateur à 5 sorties (0, 25, 50, 75, 100%) en fonction de la température : < 30°C, 35-40, 40-55, 55-70, > 70	"	"
Actualisation des valeurs mesurées cycliquement : 10ms	"	"
Remarque : Ces valeurs correspondent aux derniers parametres implementé dans le programme du processeur.		
Ces-ci peuvent être modifiés par la suite.		

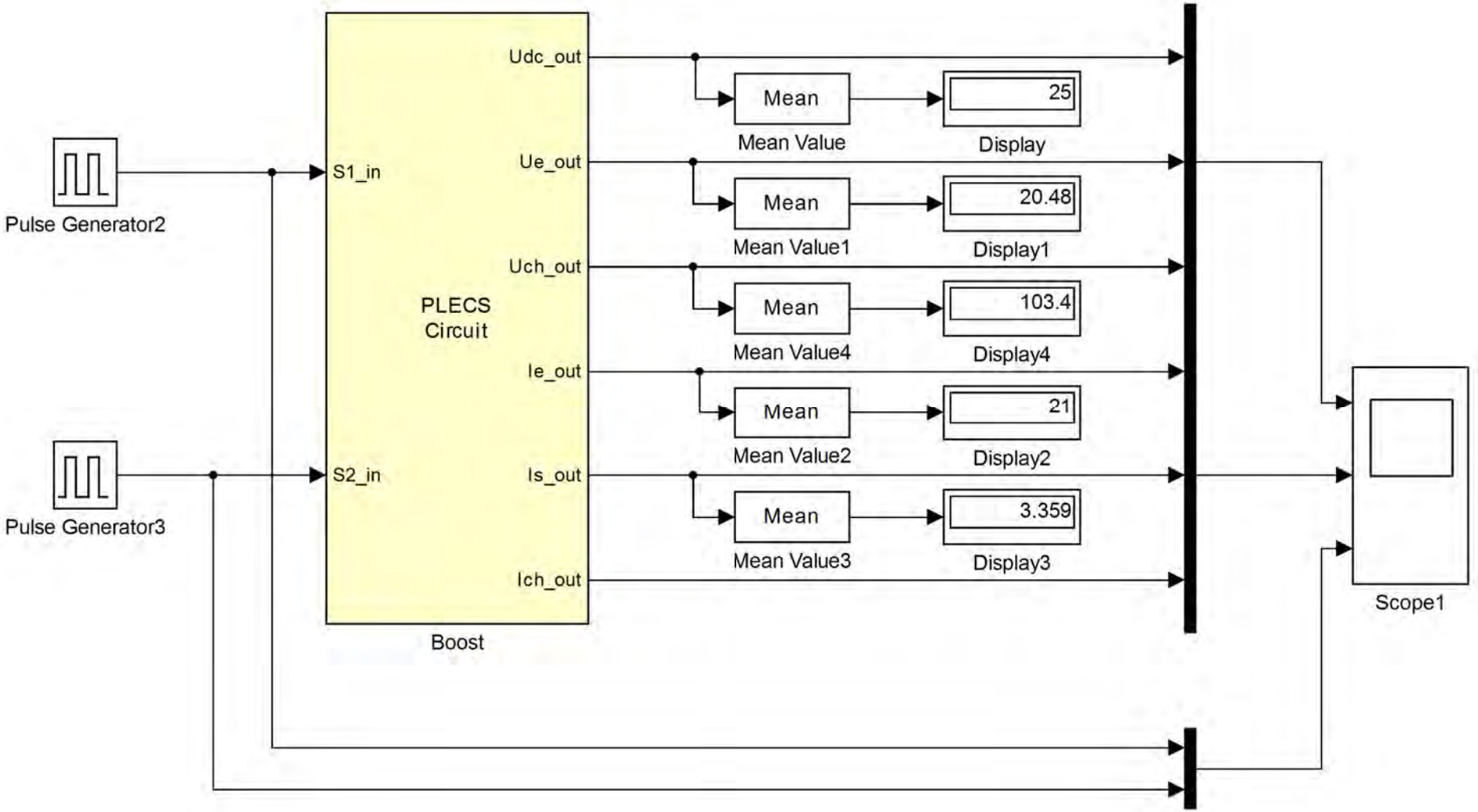
ANNEXE 12

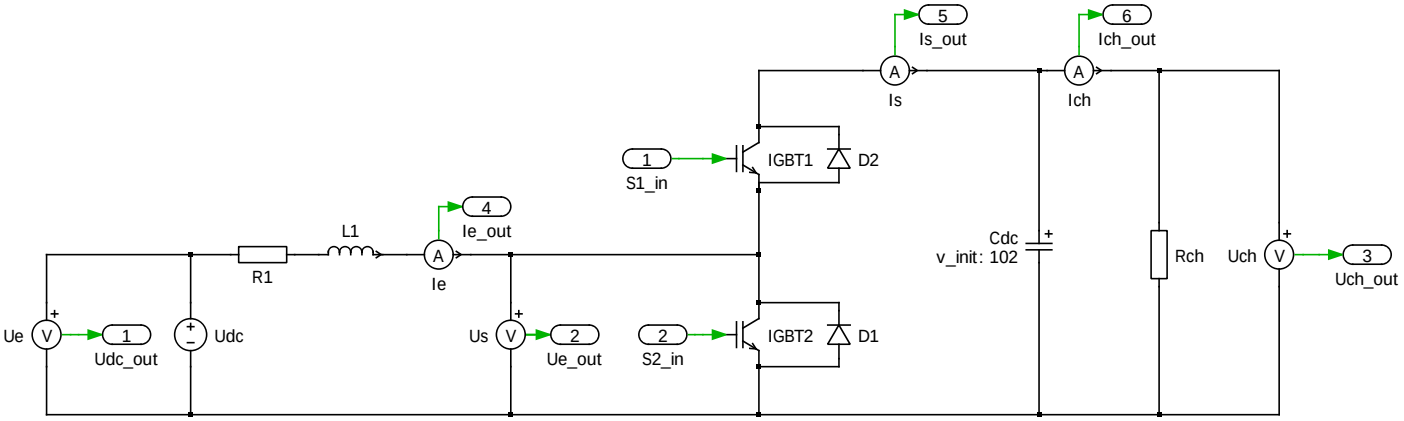
```
clear all;

%%Paramètres de simulation
TIME = 0.01;

%%Paramètres du convertisseur BUCK
Udc = 300;
Cdc = 1.8e-3;
Ls = 1.5e-3;
Rs = 0.2;
Cch = 2.2e-6;
Rch = 41;

Vf_igbt = 2;
Ron_igbt = 0.1;
Vf_diode = 1.7;
Ron_diode = 0.068;


%%Parmètres du signal PWM
F = 20000;
T = 1/F;
T_dead = 1;     %unité us
gain = F/20000;
duty = 50;
Amp = 10;


% Calcul du signal PWM
PWM_DEAD = T_dead*gain;
PWM1 = duty-PWM_DEAD;
PWM2 = (100-duty)-PWM_DEAD;
delay = T*PWM1/100+T_dead/2000000;
buck;
```


Pulse Generator
Pulse Generator1
S1_in
S2_in
PLECS
Circuit
BUCK
Ue_out
Us_out
Uch_out
Ie_out
Is_out
Ich_out
Mean
<Ue>
300
Ue value
Mean
<Us>
144.9
Us value
Mean
<Ie>
1.718
Ie value
Mean
<Is>
3.506
Is value
Scope1

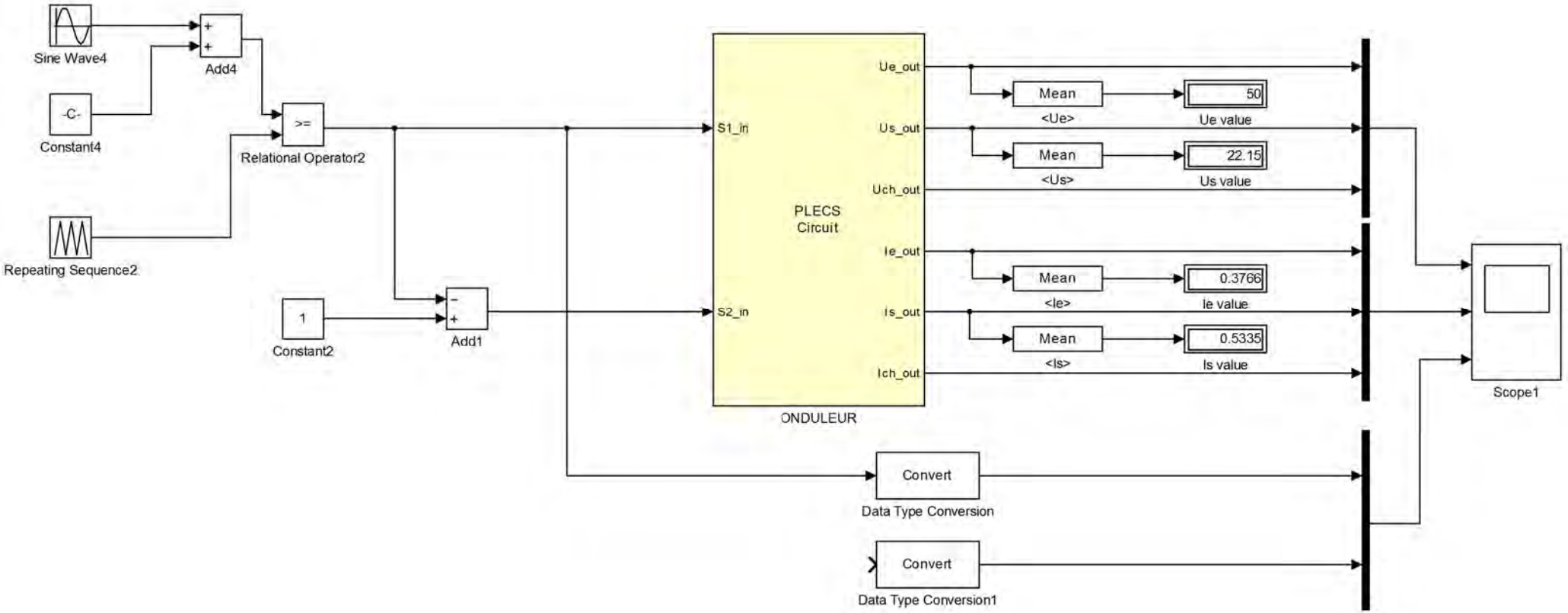

4
Ie_out
A
Ie
1
S1_in
IGBT1
D1
5
Is_out
L1
R1
6
Ich_out
Ue
V
1
Ue_out
Udc
A
Is
A
Ich
2
S2_in
IGBT2
D2
Us
V
2
Us_out
Cch
v_init: 150
Rch
Uch
V
3
Uch_out

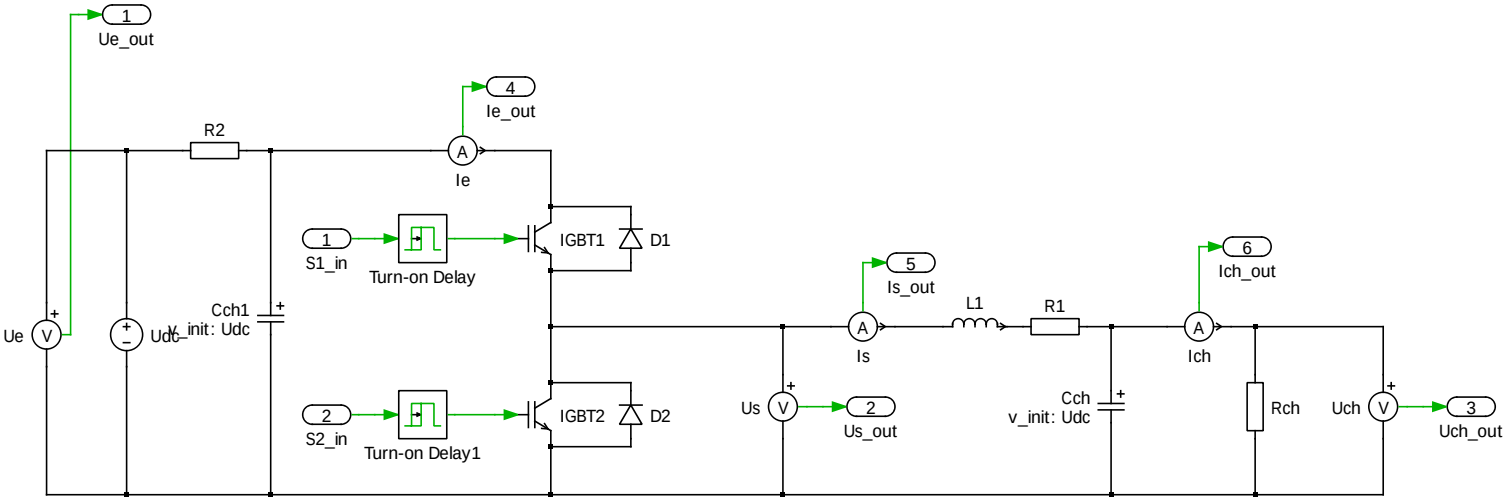
```
clear all;

%%Paramètres de simulation
TIME = 0.1;

%%Paramètres du convertisseur BUCK
Udc = 25;
Cdc = 0.82e-3;
Ls = 1.5e-3;
Rs = 0.2;
Rch = 31;

Vf_igbt = 2;
Ron_igbt = 0.1;
Vf_diode = 1.7;
Ron_diode = 0.068;

%%Parmètres du signal PWM
F = 20000;
T = 1/F;
T_dead = 1;     %unité us
gain = F/20000;
duty = 15;
Amp = 10;


% Calcul du signal PWM
PWM_DEAD = T_dead*gain;
PWM1 = duty-PWM_DEAD;
PWM2 = (100-duty)-PWM_DEAD;
delay = T*PWM1/100+T_dead/2000000;
boost;
```


Udc_out
Ue_out
Uch_out
Ie_out
Is_out
Ich_out
S1_in
S2_in
PLECS
Circuit
Boost
Pulse Generator2
Pulse Generator3
Mean
Mean Value
25
Display
Mean
Mean Value1
20.48
Display1
Mean
Mean Value4
103.4
Display4
Mean
Mean Value2
21
Display2
Mean
Mean Value3
3.359
Display3
Scope1


5
Is_out
6
Ich_out
A
Is
A
Ich
1
S1_in
IGBT1
D2
4
Ie_out
L1
R1
A
Ie
Cdc
v_init: 102
Rch
Uch
V
3
Uch_out
Ue
V
1
Udc_out
Udc
Us
V
2
Ue_out
2
S2_in
IGBT2
D1

```
clear all;clc;

%%Paramètres de simulation
TIME = 0.02;

%%Paramètres du convertisseur
Udc = 50;
Cdc = 1.8e-3;
Ls = 1.5e-3;
Rs = 0.2;
Cch = 2.2e-6;
Rch = 41;

Vf_igbt = 2;
Ron_igbt = 0.1;
Vf_diode = 1.7;
Ron_diode = 0.068;


%%Parmètres du signal PWM
F = 20000;
f = 50;
T = 1/F;
T_dead = 1;     %unité us
gain = F/20000;
duty = 50;
Amp = 1;


% Calcul du signal PWM
PWM_DEAD = T_dead*gain;
PWM1 = duty-PWM_DEAD;
PWM2 = (100-duty)-PWM_DEAD;
delay = T*PWM1/100+T_dead/2000000;
ond;
```


Sine Wave4
Add4
-C-
Constant4
Relational Operator2
Repeating Sequence2
1
Constant2
Add1
S1_in
S2_in
Ue_out
Us_out
Uch_out
Ie_out
Is_out
Ich_out
PLECS
Circuit
ONDULEUR
Mean
<Ue>
50
Ue value
Mean
<Us>
22.15
Us value
Mean
<Ie>
0.3766
Ie value
Mean
<Is>
0.5335
Is value
Scope1
Convert
Data Type Conversion
Convert
Data Type Conversion1


1
Ue_out
4
Ie_out
R2
A
Ie
1
S1_in
Turn-on Delay
IGBT1
D1
Ue
V
Udc
Cch1
v_init: Udc
2
S2_in
Turn-on Delay1
IGBT2
D2
5
Is_out
L1
R1
6
Ich_out
A
Is
A
Ich
Us
V
2
Us_out
Cch
v_init: Udc
Rch
Uch
V
3
Uch_out

ANNEXE 13

Ampli OP select:	AD864x
Slew Rate [V/s]	11.000E+6

Attenuation Critique:

a1	*0.5098*
b1	*0*
a2	*1.0197*
b2	*0.2599*

ADC FS [V]	3.000E+0
Vpeak	1.500E+0
f max [Hz]	1.167E+6

Cadc [F]	10.000E-9
fg [Hz]	10.000E+3
wg [rad/sec]	62.832E+3

R3_min_calc [Ω]	405.652E+0
R3_Norm [Ω]	430.000E+0

C2_calc [F]	2.137E-9
C2_Norm [F]	2.200E-9

e [%]	*E [Ω]*
2.95	*63.0E-12*

R2_calc [Ω]	6.959E+3
R2_Norm [Ω]	10.000E+3

e [%]	*E [Ω]*
43.70	*3.0E+3*

Gain needed	100.000E-3

[Rht+R1]_max_calc [Ω]	100.000E+3
Rht_Norm [Ω]	82.000E+3

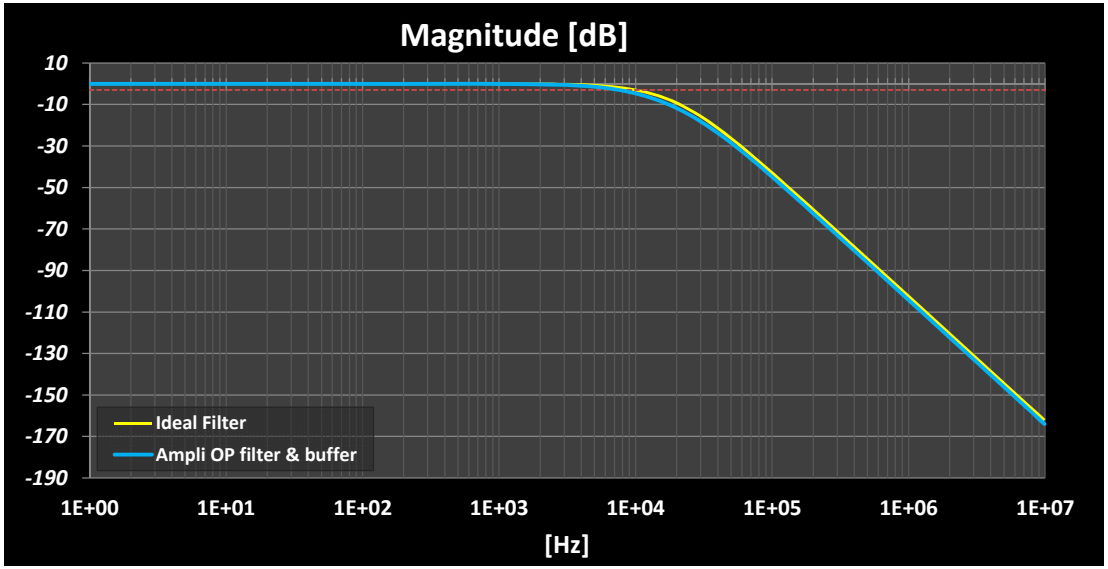
R1_calc [Ω]	18.000E+3
R1_Norm [Ω]	18.000E+3

e [%]	*E [Ω]*
0.00	*000.0E+0*

Normalised Gain	100.000E-3

0.00

Rin [Ω]	14.760E+3
Cf_calc [F]	549.710E-12
Cf_Norm [F]	470.000E-12



Cut off frequency [Hz] :	7.1E+3

Look for Gain @ Freq. [Hz]:	500.0E+3	=>	-84.5094 [dB]