INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

freescale™

Bureau d'étude Electronique Automobile

http://www.alexandre-boyer.fr

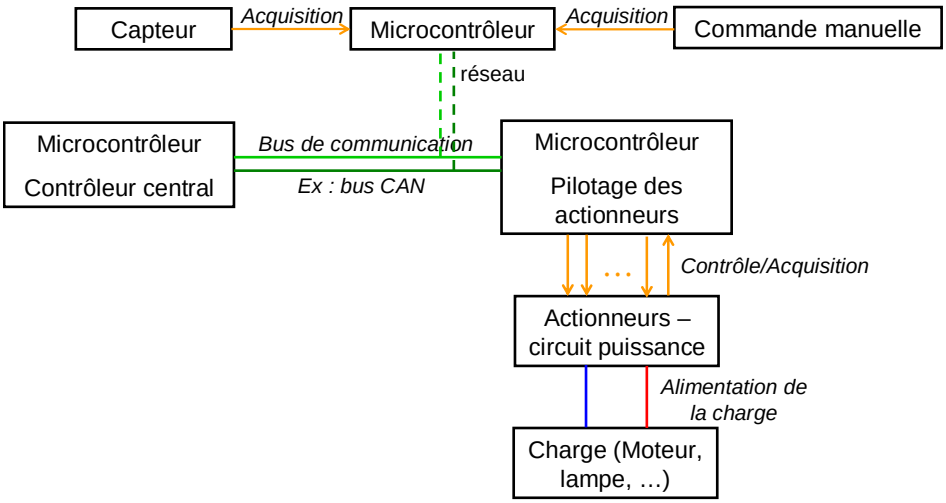
Alexandre Boyer
Patrick Tounsi

5e année ESE
Novembre 2014

Contenu

I - Contexte

Dans les véhicules actuels sont embarqués de nombreux composants électroniques (microcontrôleur, capteur, actionneur de puissance, …) permettant d'améliorer la fiabilité du système, la sécurité et le confort des passagers et le rendement énergétique. La mise en place et l'optimisation de tous ces organes électroniques à l'intérieur d'un véhicule nécessite un savoir faire large en électronique (analogique, numérique, puissance) et en informatique matérielle. La figure ci-dessous décrit l'architecture typique d'une application automobile.



Une charge (lampe, moteur) est pilotée par un actionneur capable de délivrer le courant nominal absorbé par la charge. Ce circuit de puissance est lui-même commandé par un microcontrôleur. Les circuits de puissance utilisés dans l'industrie automobile ne sont pas de simples commutateurs de puissances puisqu'ils intègrent aussi toute une partie logique de contrôle, disposent de plusieurs modes de fonctionnement et renvoient de nombreuses informations comme la température ou le courant débité. L'actionnement des charges dépend des informations acquises et envoyées par un ensemble de capteurs aux organes de contrôle. Tous ces organes de contrôle forment un réseau interconnecté par un ou plusieurs bus, comme le bus standard CAN.

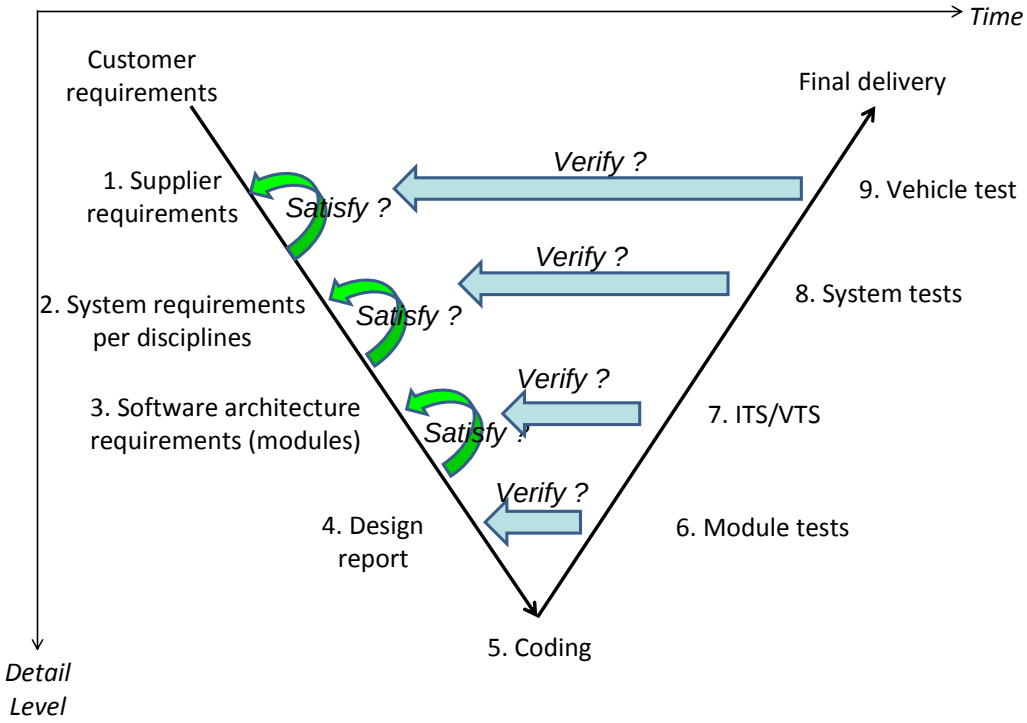
En raison des exigences économiques, de sûreté de fonctionnement et de robustesse, le processus de conception d'une application automobile suit souvent un cycle particulier, appelé cycle en V, décrit à la figure ci-dessous.

Ce cycle de conception a pour objectifs de faciliter l'élaboration d'un produit final à partir de spécifications client, vérifier la cohérence et le respect des spécifications client à chaque étape et le « debug » des problèmes. Dans le cadre du développement d'une application électronique automobile, les différentes étapes sont :

1. Exigences équipementiers : A partir des exigences ou spécifications client, l'équipementier définit son propre cahier des charges. Le cahier des charges contient des exigences fonctionnelles et des contraintes auxquelles doit répondre l'application.

2. Exigences systèmes par discipline : les exigences en différentes disciplines : logicielle (code embarqué), électronique (conception carte et choix composants) et mécanique. Il est nécessaire de s'assurer que les exigences systèmes doivent se conformer aux exigences de la phase 1. Dans le cadre de ce BE, nous mettrons l'accent sur les

solutions matérielles et logicielles pour garantir la robustesse et la qualité énergétique du système.



3. Exigences d'architecture logicielle : l'analyse des exigences logicielles permet de proposer l'architecture de l'application logicielle, décomposée en modules. Les exigences logicielles sont décrites dans un rapport. Il est nécessaire de s'assurer que les exigences systèmes se conforment aux exigences de la phase précédente.

4. Exigence design : Chaque module est décomposé en fonctions. Il est nécessaire de s'assurer que les exigences systèmes se conforment aux exigences de la phase 3. Les tests modulaires peuvent être définis à cette étape (pas leur mise en œuvre).

5. Codage : Il s'agit de l'écriture proprement dite du code permettant de satisfaire aux exigences de design. Le codage est soumis à un ensemble de recommandations et de contraintes afin d'en améliorer la portabilité, la lisibilité, la robustesse …

6. Tests modulaires : Chaque fonction doit être testée et validée. Des vecteurs de tests sont choisis afin d'obtenir une couverture de test = 100 % et garantir le bon fonctionnement de l'application finale. Un rapport est créé qui détaille le test des fonctions. On s'assure que toutes les exigences définies en 4 sont respectées.

7. Integration Test Specification (ITS) / Verification Test Specification (VTS) : Le test ITS permet de s'assurer la bonne compatibilité entre les modules, leur initialisation correcte, leur bonne communication, la bonne récurrence des fonctions dans le temps. Le test VTS permet de s'assurer que les exigences sont respectées. On s'assure qu'on respecte toutes les exigences définies en 3.

8. Tests système : On vérifie sur l'équipement l'ensemble du code. On s'assure qu'on respecte toutes les exigences définies en 2.

9. Tests véhicule : il s'agit du test final avant le livrable pour le client.

L'organisation de ce bureau d'étude suivra ce type de cycle de conception, dans la mesure du temps et du matériel disponible. Le travail réalisé durant les 9 séances de BE s'inscrit dans les étapes 3, 4, 5, 6 et 7 de ce cycle de conception.

II - Objectifs du bureau d'étude

Le but de ce bureau d'étude est de réaliser une application automobile basé sur l'architecture précédente en suivant un processus de développement industriel. L'application à réaliser est spécifiée par un cahier des charges définissant non seulement les exigences fonctionnelles, mais aussi les exigences en termes de gestion d'énergie, de sûreté (safety), de robustesse du système aux erreurs et aux pannes, de compatibilité électromagnétique (EMC).
Pour cela, vous disposez d'un ensemble de composants électroniques dédiés aux applications automobiles, fournis par la société Freescale Semiconductor, et des logiciels de programmation associés (Freescale CodeWarrior).

Les objectifs du bureau d'étude sont les suivants :

- mettre en place une architecture typique d'une application automobile (microcontrôleur, actionneur, charge, bus)
- développer un projet complet : mise en place du cahier des charges, développement logiciel et mise en œuvre pratique à l'aide de composants électroniques
- proposer des solutions logicielles et matérielles améliorant la sécurité, le confort du conducteur et des passagers, réduisant la consommation d'énergie, le risque d'erreurs matérielles et logicielles et permettant le diagnostic du système
- se familiariser aux composants industriels automobiles (microcontrôleur, transceiver CAN, high side switch, superviseur d'alimentation, System Basis Chip) et aux contraintes de ce domaine
- mettre en réseau l'ensemble des composants en utilisant un bus de terrain tel que le bus CAN (Controller Area Network)

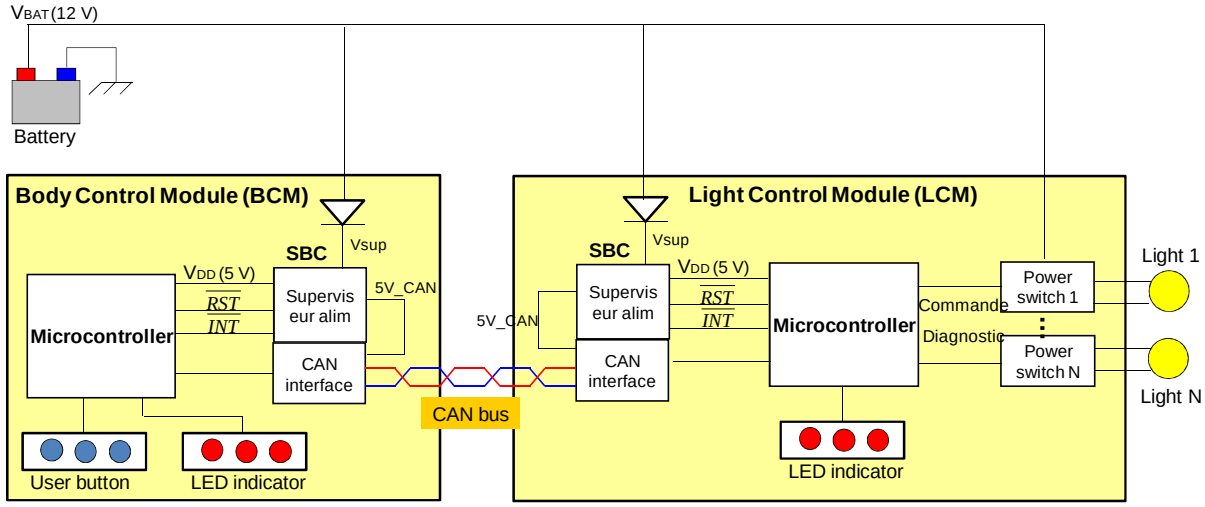
III - Enoncé du BE – Cahier des charges

1. Présentation générale du projet

Votre équipe est en charge du développement du système d'éclairage avant du véhicule. Cette application est répartie entre plusieurs ECU (Electronic Control Unit) :

- Le contrôleur habitacle ou Body Controller Module (BCM), qui gère l'interface avec l'utilisateur et reçoit les informations de diagnostic d'autres ECU tels que le Light Control Module.
- Le contrôleur phare ou Light Control Module (LCM), qui gère l'allumage des phares et clignotants, ainsi que la détection de pannes éventuelles.

Ces différents modules communiquent sur un réseau CAN (Controller Area Network) – ISO11898. La figure ci-dessous décrit l'architecture matérielle de l'application et l'interconnexion entre les différents équipements.



Votre travail consiste à développer l'application gestion fonctionnelle et de diagnostic en temps réel, embarquée dans le BCM, le DCM et le LCM. Cette application doit respecter les exigences définies dans le cahier des charges ci-dessous.
Il s'agira de reprendre un travail déjà commencé l'an dernier pour lequel il des y aura des améliorations à apporter (avec esprit critique). La mission consistera à finaliser l'application et ses tests. L'ensemble des travaux effectués l'an dernier (rapport de conception, codes sources, avancement du projet) sont disponibles sur le site http://www.alexandre-boyer.fr/enseignements.

Votre travail se décompose en différentes parties :

1. Reprendre la spécification de l'architecture matérielle et logicielle de l'application faite l'an dernier (voir annexe 1 pour le contenu)
2. Répartition des tâches après désignation du chef de projet
3. Reprendre et poursuivre le travail de développement
4. Définir des vecteurs de tests, tester et valider chaque module (voir annexe 1 pour le contenu du rapport d'avancement et de test)
5. Tester et valider l'application complète
6. Délivrable final : Rapport détaillé et codes sources

Vous développerez et testerez vos modules logiciels sur des kits de développement Freescale (voir annexes 2 et 3). Les tests pourront être effectués sur des kits de développement différents.

Remarque : Même s'il s'agit d'un travail individuel, le travail sur la spécification de l'application et le test final est un travail de groupe. Une proposition de découpage du travail entre les élèves de façon équilibrée est indispensable.

2. Cahier des charges

a. Exigences fonctionnelles

Description de la fonction phare et clignotant
La commande l'allumage/extinction des phares et des clignotants est assurée par :

- Les commodos situés sur le tableau de bord (on simulera au début ces commodos les interrupteurs et boutons poussoirs disponibles sur la carte de développement TRK-MPC5604B), actionnés par le conducteur. L'action est détectée par le BCM.
- Si le capteur de luminosité du véhicule détecte une obscurité trop importante, les phares sont mis en marche automatiquement. Ils s'éteignent automatiquement dès qu'une luminosité suffisante est revenue, sauf si le conducteur a donné l'ordre d'activation des phares. Vous pourrez par exemple simuler le capteur de luminosité par le photorécepteur disponible sur la carte de développement TRK-MPC5604B. Le capteur est connecté au BCM.

La commande des phares est assurée par le high side switch MC33984. Le réglage de l'intensité des phares se fera par une commande PWM. Vous pourrez limiter votre travail à un seul bloc phare + clignotant.
Le choix des entrées-sorties utilisé sur les microcontrôleurs du BCM et LCM est libre mais devra être clairement spécifiée dans le rapport de spécifications.

b. Exigences sur le BCM

Trois modes de fonctionnement sont proposés pour le BCM :

- Mode de fonctionnement nominal :

La fréquence d'horloge du bus système du BCM est fixée à 64 MHz. Elle est fournie par le module FM-PLL à partir d'un oscillateur à quartz externe de 8 MHz. La tolérance sur la fréquence du quartz est inférieure à 0.5 %. La modulation de fréquence sera activée (cf. Exigences CEM).

- Mode de fonctionnement dégradé :

Le microcontrôleur rentrera dans ce mode lorsque des problèmes seront détectés sur la tension d'alimentation du module BCM (cf. Exigences sur la surveillance de l'alimentation et Exigences safety).

- Mode basse consommation :

Cf. Exigences basse consommation.

c. Exigences sur le LCM

Trois modes de fonctionnement sont proposés pour le LCM :

- Mode de fonctionnement nominal :

La fréquence d'horloge du bus système du BCM est fixée à 34 MHz. Elle est fournie par le module FM-PLL à partir d'un oscillateur à quartz externe de 8 MHz. La tolérance sur la fréquence du quartz est inférieure à 0.5 %. La modulation de fréquence sera activée (cf. Exigences CEM).

- Mode de fonctionnement dégradé :

Le microcontrôleur rentrera dans ce mode lorsque des problèmes seront détectés sur la tension d'alimentation du module LCM (cf. Exigences sur la surveillance d'alimentation et Exigences safety).

- Mode basse consommation :

Cf. Exigences basse consommation.

d. Exigences sur la communication CAN

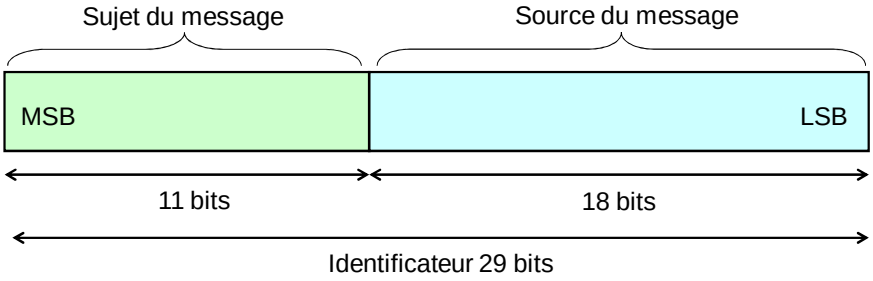
Le module DCM communique avec le module BCM par un bus CAN. La longueur maximale du bus CAN entre le DCM et le BCM est de 5 mètres. La liaison CAN est assurée par une paire bifilaire. Le temps de propagation le long de la paire est de 5 ns/m. Le retard maximal introduit par un contrôleur CAN ou un transceiver CAN est estimé à 25 ns.

Exigences sur le débit binaire

Le bus CAN fonctionne selon la spécification CAN 2.0B. Le débit binaire doit être supérieur à 1 Mbits/s en mode de fonctionnement nominal. Il sera ramené à 125 Kbits/s en mode dégradé.

Exigences sur le format des trames CAN

Le Standard CAN 2.0B est utilisé. Seules des trames de données sont transmises. Les *remote frames* ne sont pas utilisées. Les données sont transmises par paquet de 8 octets. Le choix des identifiants doit se conformer au format suivant :



L'adresse (source du message) donnée au BCM sera '0x00000'. Celle pour le DCM sera '0x20000' et celle pour le LCM sera '0x10000'.

Gestion de la consommation d'énergie

Cf. Exigences basse consommation. Lorsque les microcontrôleurs entreront en mode basse consommation, les contrôleurs CAN entreront en mode Disable. Les interfaces CAN entreront en mode Sleep. Ceux-ci sortiront des modes basse consommation soit par demande du microcontrôleur, soit par requête sur le bus CAN. Un pattern correspondant à 3 impulsions à

l'état dominant sera utilisé pour réveiller les interfaces CAN. Une impulsion sur la ligne INT indiquera la sortie du mode Sleep de l'interface CAN.

Gestion des erreurs de transmission et de réception sur le bus CAN
Dans le cas d'erreurs de transmission ou de réception, seules les entrées et les sorties de l'état Bus Off sont repérées. Pour le BCM, la sortie de l'état Bus Off se fait par une requête du contrôleur, après l'entrée dans l'état Bus Off. Après 5 entrées consécutives dans l'état Bus Off, une erreur est affichée sur le tableau de bord et la demande en cours est abandonnée.
Pour le DCM et le LCM, la sortie de l'état Bus Off est automatique (selon la spécification du protocole CAN 2.0B : 128 occurrences de 29 bits récessifs).

Inactivité sur le bus CAN
Si un contrôleur CAN ne transmet rien pendant plus de 2 secondes, celui-ci devra passer en mode Listen Only. L'interface CAN passera en mode Receive Only.

Gestion des problèmes matérielles sur le bus CAN
Le System Basis Chip (SBC) intègre une interface CAN capable de détecter des problèmes électriques sur le bus CAN, sur les broches Rx et Tx et une surtempétaure de l'interface CAN. En cas de problème matériel, l'interface CAN enverra une impulsion sur la ligne INT. Le microcontrôleur détectera l'origine du problème et notifiera la présence d'une erreur en allumant un indicateur lumineux. L'interface CAN sera mise en mode Receive Only, tant que le problème n'a pas disparu. Le microcontrôleur cherchera régulièrement à faire revenir l'interface CAN en mode d'émission/réception normal.
En cas de problème relevé sur l'alimentation 5V_CAN, la même procédure sera appliquée mais l'interface CAN passera en mode Sleep.

Remarque : le SBC pourra être configuré en mode debug pour faciliter la tâche de développement et de test.

e. Exigences Basse consommation

Modes de consommation des microcontrôleurs
Trois modes de consommation d'énergie sont définies pour les microcontrôleurs :

- Mode Run0 : fonctionnement normal.
- Mode dégradé : en cas d'apparition de problème (voir exigences Safety)
- Mode basse consommation STOP0 : voir détails ci-dessous.

Le BCM passe en mode STOP0 si sa période d'inactivité dépasse 10 secondes. Il sort de ce mode dans les cas suivants :

- Un message lui est adressé sur le bus CAN
- L'utilisateur appuie sur une commande d'allumage de phare
- Un problème matériel est détecté par le superviseur d'alimentation ou l'interface CAN (signal INT)
- Une condition de faible luminosité est détectée

Avant d'entrer en mode STOP0, le microcontrôleur s'assurer que l'interface CAN est entrée en mode Sleep et que le contrôleur CAN est entré en mode Disable.

Le LCM passe en mode STOP0 si sa période d'inactivité dépasse 10 secondes et si aucun phare n'est allumé. Il sort de ce mode dans les cas suivants :

- Un message lui est adressé sur le bus CAN
- Un problème matériel est détecté par le superviseur d'alimentation, l'interface CAN (signal INT) ou un des drivers de charge

Avant d'entrer en mode STOP0, le microcontrôleur s'assurer que l'interface CAN est entrée en mode Sleep, que le contrôleur CAN est entré en mode Disable et que les drivers de charge (inactifs) sont entrés en mode Sleep.

f. Exigences sur la surveillance de l'alimentation

L'alimentation du BCM et du LCM est issue de la batterie Vbat. Le superviseur d'alimentation (SBC) fournit une alimentation 5 V (Vdd) régulée au microcontrôleur. Le SBC surveille la qualité de ces différentes alimentations (surtension, soustension, présence d'impulsions parasites, load dump, crank phase). En fonction de la tension mesurée par le SBC, plusieurs scénarios sont prévus (voir datasheet SBC MC33905 pour plus d'informations sur les différentes tensions mesurées par le SBC) :

- Si la tension Vsense devient inférieure à 8.6 V ou si la tension Vsup devient inférieure à 6 V, le SBC envoie une impulsion sur la ligne INT. Le microcontrôleur vérifie la valeur de la tension batterie (Vsense) et notifie ou non le risque de tension de batterie faible en allumant un indicateur lumineux. Les microcontrôleurs des BCM et LCM passent ensuite en mode de fonctionnement dégradé.
- Si la tension d'alimentation Vdd du microcontrôleur est comprise entre 3.5 et 4.5 V, le SBC envoie une impulsion sur la ligne INT. Le microcontrôleur notifie ou non le risque de tension de batterie faible en allumant un indicateur lumineux et les microcontrôleurs des BCM et LCM passent en mode de fonctionnement dégradé.
- Si la tension d'alimentation Vdd du microcontrôleur devient inférieure à 3.5 V, les microcontrôleurs sont mis en reset.
- Si la tension Vsup est inférieure à 4 V (crank phase), la tension Vdd délivrée par le SBC est désactivée.
- Si un problème est relevé sur la tension d'alimentation 5V_CAN de l'interface CAN (overcurrent, underevoltage, overtemperature), celle-ci passe en mode Sleep jusqu'à ce que le problème disparaisse.

g. Exigences en terme de diagnostic des charges

Le module LCM vérifie l'état des switches de puissance et fait remonter toute défaillance au BCM (overtemperature, overcurrent, overvoltage, open load).
La détection d'un surcourant est déterminée par un seuil bas fixé à 20 A et un seuil haut fixé à 75 A. Lorsque le courant dépasse le seuil bas pendant plus de 10 ms, la sortie de puissance correspondante est inhibée.
Dès qu'un problème est signalé sur un switch, un indicateur lumineux indique la nature du problème et la commande du switch est coupée jusqu'à ce que le problème disparaisse. Lorsque le problème a disparu, le DCM ou le LCM transmet au BCM un message indiquant le recouvrement de la panne.
Le BCM peut demander des diagnostics sur les switches de puissance contrôlés par le DCM, ainsi qu'un relevé du courant moyen consommé pendant une période donnée.

h. Exigences CEM

Afin de réduire les émissions électromagnétiques parasites, la modulation FM de la PLL de chaque microcontrôleur sera activée. Celle-ci sera réglée afin de minimiser l'émission électromagnétique et ne pas provoquer de problème de synchronisation du bus CAN.
Toutes les sorties digitales du microcontrôleur, les sorties CAN_H et CAN_L des interfaces CAN et les sorties de puissance seront configurés en mode « slow slew rate ».

i. Exigences Safety

Lorsque des problèmes d'alimentation seront rencontrés (cf. Exigences sur la surveillance de l'alimentation), les microcontrôleurs passeront en mode dégradé, afin de limiter leur consommation en courant. Les PLL seront désactivés et l'horloge système sera directement fournie par l'oscillateur à quartz externe. Le bus CAN fonctionnera à 125 kbits/s. Un indicateur lumineux signalera l'entrée en mode de fonctionnement dégradé.
Un watchdog interne aux microcontrôleurs assurera un reset en moins de 50 ms en cas de problème logiciel.

IV - Organisation et Planning

Un groupe de TP travaille ensemble au développement de l'application. Le groupe travaille à la spécification logicielle de l'application et à sa validation. Un chef de projet est désigné pour le groupe et les tâches sont réparties sur l'ensemble des élèves. Chaque élève doit mener à bien des tâches bien définies.. Le travail de spécification, de codage et de test de chaque module de l'architecture logicielle est réparti entre les différents élèves. La répartition de ce travail est libre mais doit rester équilibrée. La notation par élève dépendra du travail effectué.
La date limite pour fournir le rapport final détaillé est le : Mardi 27 Janvier 2015.

Pour toute question technique portant sur les composants Freecale, envoyez un e-mail simultanément aux encadrants de TP : alexandre.boyer@insa-toulouse.fr et patrick.tounsi@insa-toulouse.fr.

V - Notation

La notation portera sur les rapports techniques rédigés par groupe et par binôme, ainsi que sur l'évaluation orale. La répartition de la note est la suivante :

- 1/2 Evaluation continue
- 1/2 pour le rapport final détaillé

Conseils pour les rapports de spécifications, de codage et de tests :

- Présenter le projet de manière synthétique et claire. Ce n'est pas la quantité de pages qui compte.
- Utiliser des outils graphiques adaptés, facilitant la compréhension de vos algorithmes et le codage des applications (voir annexe 1).
- Décrire l'architecture matérielle en faisant apparaître les branchements entre les différentes cartes, les entrées-sorties utilisées, les types de signaux, les fréquences, les niveaux de tension, les débits binaires, les options d'entrée-sortie (pull-up, pull-down …). Toute entrée-sortie dont le statut n'est pas détaillé rend la spécification matérielle imprécise (voir annexe 1).
- Décrire les solutions apportées pour respecter les contraintes fonctionnelles, de sécurité, de dimensionnement du bus CAN et de gestion de l'énergie.
- Les codes sources doivent être suffisamment et correctement documentés. le code source, des commentaires doivent être associés à chaque fonction et indiquer : le rôle de la fonction, les variables d'entrées et de sorties.
- A l'issue du projet, un rapport d'avancement et de test vous est demandé. Celui-ci doit montrer quelles exigences ont pu être pris en compte durant votre travail et si elles ont été respectées (voir annexe 1).

VI - Annexe 1 – Format des documents à rendre

Un des objectifs derrière les différents rapports qui vous sont demandés est l'écriture de rapports techniques professionnels. Ces rapports ont un but précis : spécification matérielle, logicielle et test. Quelques conseils pour l'écriture des rapports :

- Le rapport doit être synthétique et précis. Ce n'est pas la quantité de pages qui compte, mais la rigueur du document.
- N'hésitez pas à synthétiser les informations par des tableaux et des schémas clairs et adaptés.
- Utilisez un format unique et lisible pour vos rapports.

1. Spécification matérielle

Fournir un schéma de câblage électrique des différents ECU, faisant apparaître l'ensemble des broches utilisées sur les différents microcontrôleurs et switches de puissance.
Lister dans un tableau les caractéristiques des entrées-sorties utilisées sur les microcontrôleurs, avec leurs caractéristiques :

- Type : entrée ou sortie digitale, PWM, entrée analogique, CAN TX ou RX, entrée quartz …
- Option : pull-up, pull-down, open drain…
- Détaillez la source des signaux d'horloge système et leur fréquence
- Donnez pour les entrées-analogiques la résolution de la conversion, la fréquence de conversion
- Pour les bus, donnez le débit binaire
- Pour les sorties PWM, donnez les caractéristiques du signal
- Tensions d'alimentation des différents composants
- …

2. Spécification logicielle

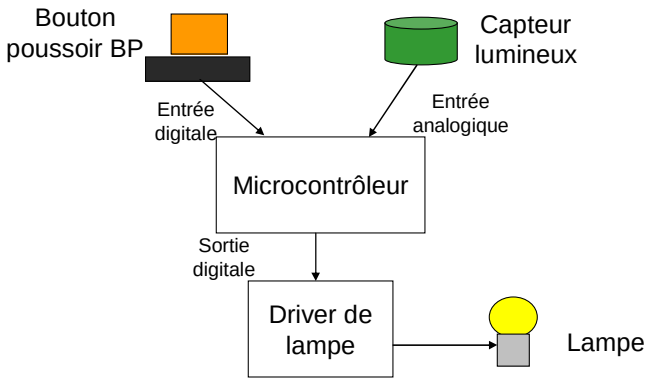
L'application que vous allez coder provient du cahier des charges du client. Avant de coder une application, il convient de traduire les spécifications du client en un algorithme et de s'assurer que cet algorithme répond aux spécifications du client.
La spécification logicielle de l'application que vous allez coder doit se faire à 2 niveaux :

- la spécification de l'architecture logicielle, qui fait apparaître les différents modules composant l'application et leurs interactions
- la spécification de chaque module, qui détaille les fonctions de chaque module

Cette spécification vous permettra de définir votre algorithme. Pour réaliser cette spécification, différents formats peuvent être utilisés : texte, schéma-bloc, diagramme d'état, flowchart …
Pour bien comprendre ces deux niveaux de spécifications et les différentes descriptions pouvant être employées, nous pouvons illustrer par l'exemple de spécification suivant :

« *Un client souhaite réaliser une application d'allumage d'une lampe par l'appui sur un bouton et en fonction de la luminosité ambiante. La figure ci-dessous illustre l'architecture matérielle de l'application. La lampe est commandé par un driver de puissance, la commande est de type PWM, la fréquence de la PWM est de 100 Hz, le rapport cyclique de 50 %. La*

lampe s'allume lorsqu'on appuie sur le bouton poussoir ou si la tension analogique aux bornes du capteur lumineux est inférieure à Vref. »



A partir de la spécification client, il est possible de décomposer l'application en plusieurs modules fonctionnels, que l'on peut décrire sous forme de texte (la fonction du module, les entrées-sorties, les contraintes …) :

- Module « Détection_Appui_BP » : ce module détecte l'appui sur le bouton poussoir et transmet l'état du bouton poussoir.
- Module « Détection_obscurité » : ce module détecte si la luminosité ambiante passe sous le seuil (Vcapteur < Vref) et transmet l'état de lumineux ambiant.
- Module « Commande_lampe » : ce module décide de l'allumage de la lampe, en autorisant la commande PWM.
- Module « PWM_lampe » : ce module génère la commande PWM de la lampe à partir des paramètres de fréquence et de rapport cyclique, si le module Commande_lampe a donné son autorisation.

Chacun de ces modules peut aussi être décrit de manière un peu plus précise. La description d'une application ou d'un module sous forme de texte est certes générale, mais elle reste surtout adaptée à une description fonctionnelle. Elle n'est pas la plus adaptée à la description de l'architecture en module ou du séquencement des actions. D'autres outils sont plus adaptés :

Schéma-bloc ou schéma fonctionnel :

Il s'agit d'une représentation graphique d'un processus complexe présentant plusieurs unités. Le schéma fait apparaître les différents blocs du système, leurs entrées-sorties et les lignes d'action. La figure ci-dessous propose un schéma-bloc de l'application précédente et fait apparaître les 4 modules et leurs interactions. Ces différents modules pourront être traduits par une ou plusieurs fonctions.

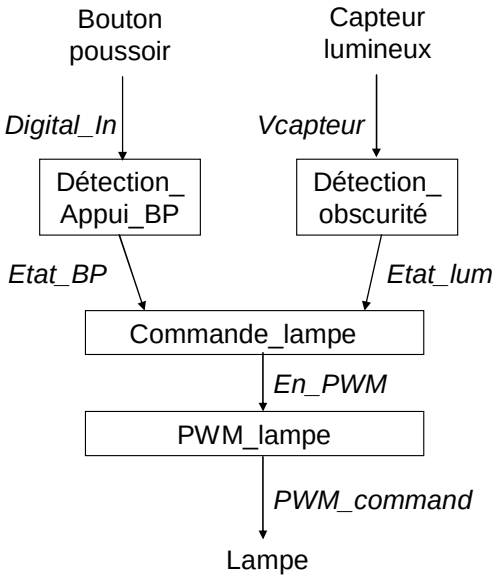
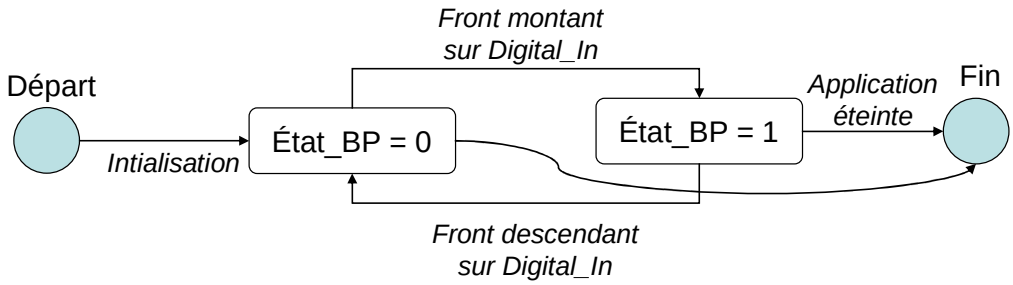


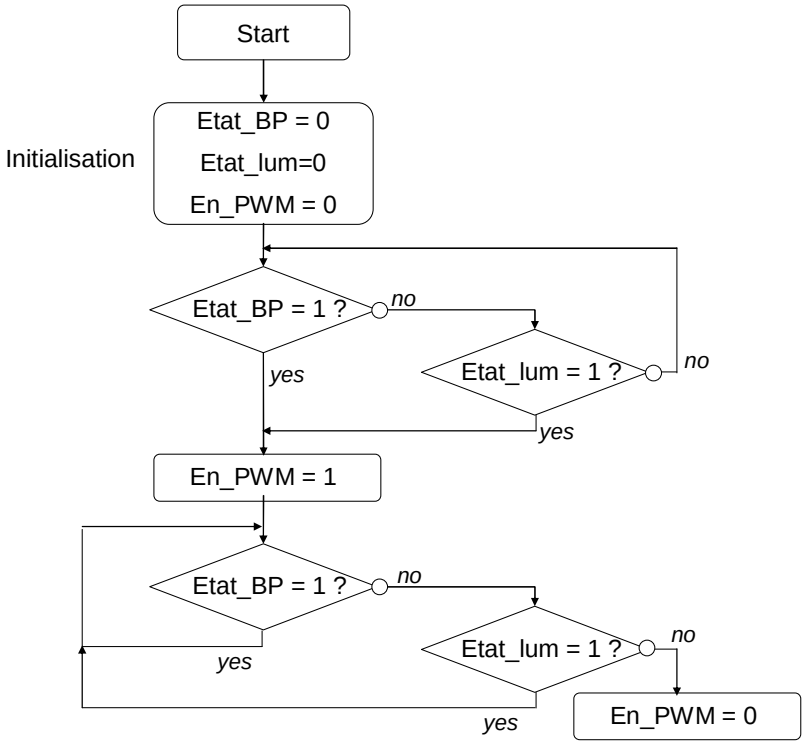
Diagramme d'état :

Le diagramme d'état permet de représenter les différents états pris par une machine à états finis et les conditions à remplir pour passer d'un état à un autre (transition). Un diagramme d'état présente l'avantage d'être directement traduisible sous la forme d'un algorithme. Par exemple, voici le diagramme d'état du module Détection_Appui_BP :



Flowchart ou algorigramme :

Le flowchart permet de représenter graphiquement l'enchainement des opérations. Comme le diagramme d'état, il est traduisible sous la forme d'un algorithme. La figure ci-dessous présente le flowchart de l'application :



Dans le cadre de ce BE, il n'y a pas d'obligation à utiliser ces différentes formes de description pour spécifier l'architecture logicielle que vous allez spécifier. Néanmoins, elles vous permettront de faciliter la création de vos algorithmes, la validation de vos spécifications par rapport à la spécification du client, le debug, la collaboration entre les différents groupes de travail.

3. Rapport final ou d'avancement

Le but de ce rapport final est de permettre l'évaluation de l'avancement du projet (qu'est-ce qui a été fait ? qu'est-ce qui a été validé ? qu'est-ce qui est en cours ? quelles sont les preuves de la validation ?) et l'adéquation avec le cahier des charges initial. Celui-ci est écrit par chaque membre de l'équipe et coordonné par le chef de projet. C'est ce dernier qui le rend.
Le rapport d'avancement peut inclure un tableau (Excel par exemple) listant l'ensemble des spécifications et des modules fonctionnels. A chaque ligne on trouvera :

- une description succincte d'une spécification ou d'un module fonctionnel (exemple : Configuration horloge système : activation d'une PLL à 64 MHz)

- un statut donnant l'avancement de la tâche de développement d'un module fonctionnel ou de vérification d'une spécification : Fait, non fait, en cours, abandonné…

- état de la validation : test conforme, test non conforme, test non effectué. Le résultat de tout test devra être présenté dans un document annexe. Indiquez comment retrouver le résultat de test (page d'un document, n° de figure, nom de fichiers…).

La validation de tout module fonctionnel ou toute spécification fera appel à un ou plusieurs tests matériels (mesure à l'oscilloscope, multimètre …) ou logiciels (changement d'état binaire dans un registre, mesure de temps de cycle …). Ceux-ci devront être détaillés dans un

rapport annexe au rapport d'avancement. Le résultat des tests devra être présenté et commenté de manière succincte (le résultat attendu, le résultat obtenu). Le format de ce document est libre. Celui-ci doit rester synthétique.

VII - Annexe 2 – Présentation du matériel

Dans ce BE, nous disposons de plusieurs maquettes, de composants dédiés automobile fournis par la société Freescale Semiconductor, et de cartes d'interface. Les notes d'application des différents composants vous sont fournies.
L'ensemble des documents sont disponibles sur le site http://www.alexandre-boyer.fr.

1. Les maquettes

Six maquettes dédiées à une application automobile sont proposées. Elles sont présentées ci-dessous accompagnées de leurs fiches signalétiques.

a. Module phare directionnel

2. Les cartes électroniques

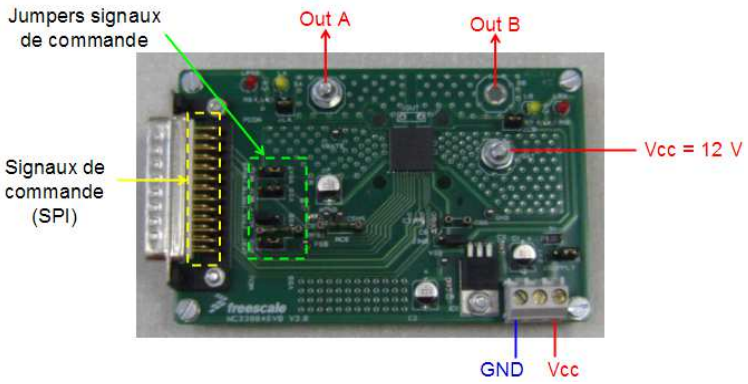
Plusieurs cartes d'évaluation sont fournies par la société Freescale Semiconductor dans le cadre du BE automobile. L'interfaçage entre les différentes cartes sera assuré par une carte spécifique.

a. MC33984 – Dual High side switch

Ce composant intègre 2 commutateurs de puissance de type high side switchs et leurs diodes de protection, ainsi que toute la partie logique d'interfaçage avec un contrôleur externe. Ce composant peut être piloté à l'aide du protocole SPI ou directement en PWM.

Remarque : A noter que la communication via SPI permet d'avoir accès à un plus grand nombre d'informations sur le statut du composant.

Ce composant peut faire passer un fort courant (jusqu'à 30 A en DC !). Il propose en plus une sortie analogique (CSNS) renseignant sur le courant de sortie permettant ainsi de mettre en place une boucle de courant. La présence d'une résistance entre CSNS et la masse est nécessaire pour assurer une conversion courant – tension.
Le composant propose aussi une sortie logique Fault Status indiquant des conditions de fonctionnement anormales (sous tension, sur courant, surchauffe). Comme la plupart des composants automobiles, il possède plusieurs modes de fonctionnement permettant de réduire la consommation en énergie : mode normal ou mode sleep. Il est monté sur une carte de développement KIT33981BPNAEVB, présentée sur la figure ci-dessous.



Fiche signalétique :

- Alimentation : 6 – 27 V (on travaillera sous 12 V)
- Courant DC max : 30 A (attention au cas du rotor bloqué)
- Fréquence max PWM : 300 Hz
- Signaux de commandes compatibles TTL/CMOS 5 V

Pour plus de détails, se reporter à la datasheet du composant et de la carte de développement KIT33984PNAEVB (http://www.alexandre-boyer.fr).

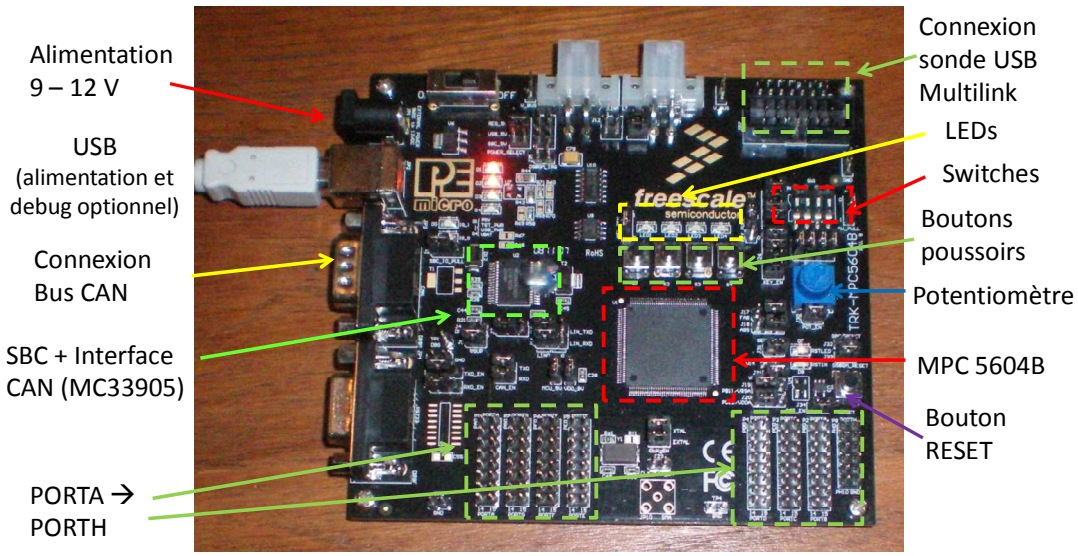
b. Kit de développement TRKMPC5604B - Sonde USB Multilink Universal

Le kit de développement propose le microcontrôleur MPC5604B (Bolero) dans sa version LQ monté dans un boîtier LQFP 144. Cette carte peut être alimenté soit par port USB, soit par une alimentation externe DC 9-12 V. L'alimentation on-board peut être régulée par un régulateur 5 V ou par un superviseur d'alimentation de type System Basis Chip (SBC) (MC33905). Ce composant gère l'alimentation de la carte de développement et intègre une interface CAN. La sélection de l'alimentation de la carte de développement se fait par le jumper J1.

Un quartz de 8 MHz fournit la référence d'horloge au composant. Sur cette carte, on trouve :

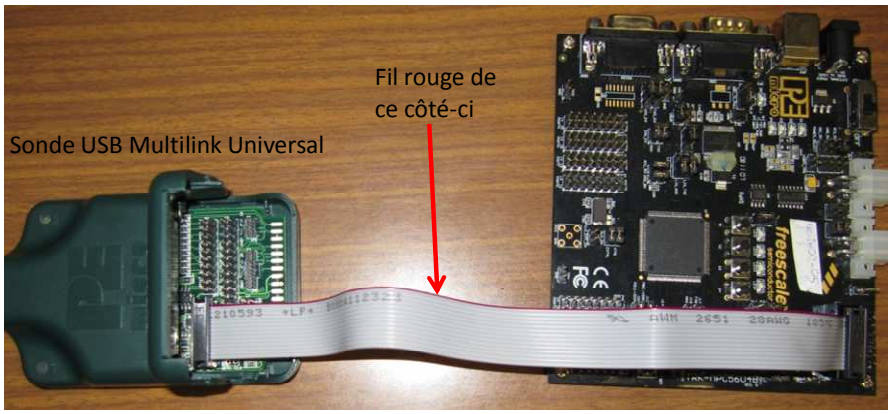
- 4 LEDs connectées aux pins PE4 à PE7
- 4 switchs connectés aux pins PG6 à PG9
- 4 boutons poussoirs connectés aux pins PE0 et PE3
- 1 bouton poussoir de reset
- un potentiomètre connecté à l'entrée de conversion analogique numérique ANP0 (pin PB4)
- une cellule photovoltaïque connectée à l'entrée de conversion analogique numérique ANP1 (pin PB5)
- un connecteur Sub-D9 (JP3) connecté à une interface CAN
- Toutes les broches des ports du microcontrôleur (PORTA → PORTH) sont accessibles à travers les connecteurs P1 à P8.

Vous pouvez vous reporter au document TRK_MPC5604B_Rev_B_Schematic_Layout.pdf pour avoir la schématique détaillée de la carte de développement, et au document TRKMPC5604BEVBUM.pdf pour plus de détails sur les positions des jumpers.



La programmation et le debug in-situ du microcontrôleur se fera par l'intermédiaire des sondes P&E Micro USB Multilink Universal. Celle-ci se connecte sur un port USB du PC et se branche sur cible alimentée. Deux LED indiquent le statut de l'interface. La LED bleue indique que l'interface est correctement reconnue par le PC, la LED jaune indique que la cible est alimentée.

Attention : l'embase de connexion pour la sonde USB Multilink sur le kit TRK_MPC5604B n'a pas de détrompeur. Des erreurs de connexion sont à craindre ! La photo ci-dessous vous indique le sens correct de connexion de la sonde.



3. Carte d'interface

La carte TRKMPC5604B représente l'élément central de chaque application. Afin de la connecter aux différentes cartes de puissance et aux autres cartes microcontrôleur par bus CAN, une carte d'interface a été développée. Celle-ci est présentée ci-dessous.

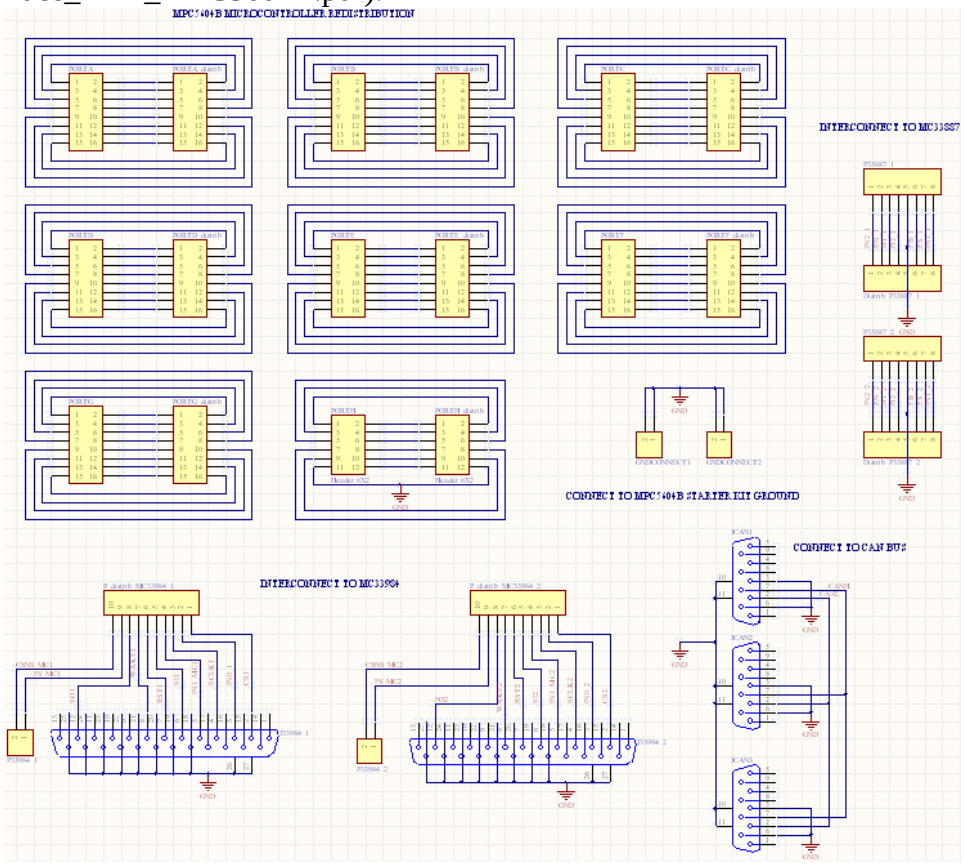
Les signaux provenant de la carte TRKMPC5604B proviennent des 8 connecteurs 2*8 ou 2*6, appelés PORTA → PORTH. Différents types de connecteurs sont utilisés pour se connecter aux cartes de puissance. En face de chaque broche des différents connecteurs de la carte sont placés des connecteurs tulipe. Ainsi, les interconnections entre cartes peuvent être réalisées manuellement en plaçant des fils entre les connecteurs tulipes. L'assignation des broches du MPC5604B est donc laissée libre.

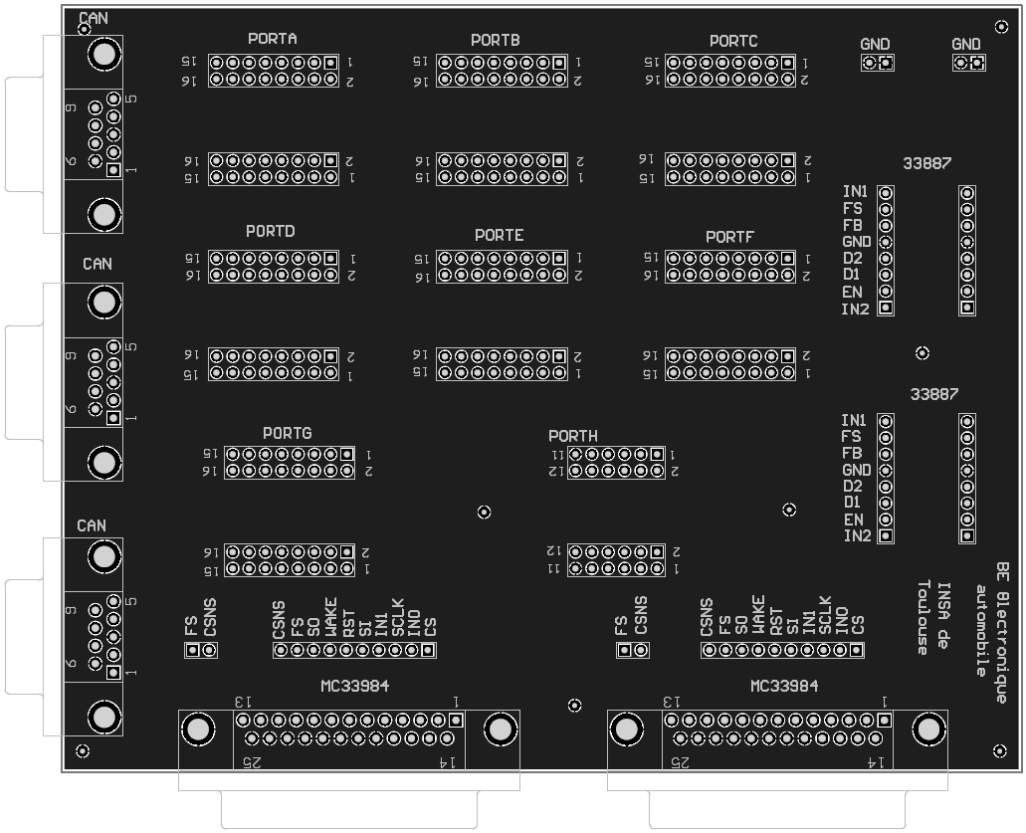
Les indications sur les connecteurs et le sens de connexion sont reportées sur les cartes.

Remarque 1 : il est de votre responsabilité de veiller à ne pas faire de mauvaise connexion (exemple : court-circuiter une alimentation à la masse, connecter une sortie de puissance 12 V sur une entrée digitale).

Remarque 2 : si vous connectez le starter kit TRKMPC5604B à la carte d'interface, vous devez impérativement relier leurs masses. Hormis sur le connecteur PORTH, aucune pin des connecteurs PORTA → PORTG n'est connectée au plan de masse du kit TRKMPC5604B. Plusieurs pins sont disponibles sur le TRKMPC5604B et sur la carte d'interface permettant de relier les masses ensembles.

Ci-dessous, la vue schématique de la carte d'interface (Schematic_interface_TRK_MPC5604B.pdf) et la vue top layer (Top_interface_TRK_MPC5604B.pdf).




CAN
CAN
CAN
PORTA
PORTB
PORTC
PORTD
PORTE
PORTF
PORTG
PORTH
GND
GND
33887
33887
IN1
FS
FB
GND
D2
D1
EN
IN2
FS
CSNS
CSNS
FS
SO
WAKE
RST
SI
IN1
SCLK
IND
CS
MC33984
MC33984
INSA de Toulouse
BE électronique automobile

VIII - Annexe 3 - Prise en main du matériel

1. Prise en main de la carte MC33984

Il est possible de prendre en main cette carte par l'intermédiaire du logiciel SPIGEN. Cependant, nous préférons utiliser une approche manuelle, sachant qu'il est possible de fixer l'état des différents signaux logiques directement sur la carte. La charge utilisée est une lampe.

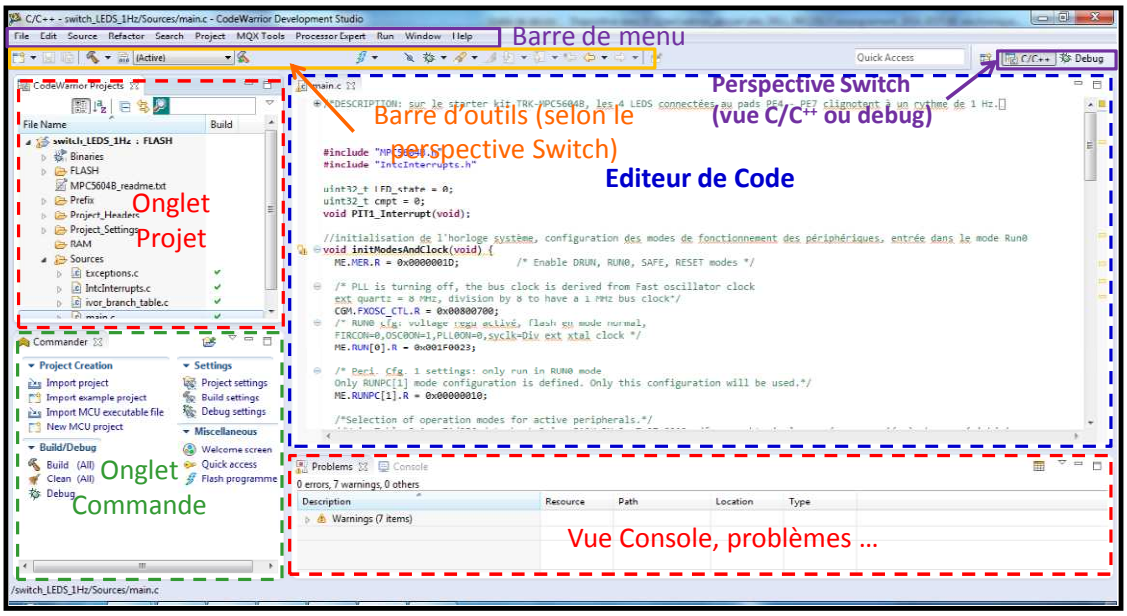
- Connecter la lampe sur une des sorties (SA ou SB)
- Connecter l'alimentation sans l'allumer (tension 12 V)
- A l'aide de la datasheet de la carte, mettre à '0' ou à '1' les différents signaux logiques de commande. Pour cela, placer correctement les jumpers présents sur la carte
- Mettre sous tension, plusieurs LEDs sur la carte vous indiquent la mise sous tension du composant et l'état de la sortie

2. Prise en main de l'outil de programmation Freescale CodeWarrior Development Studio for Micro v10.5

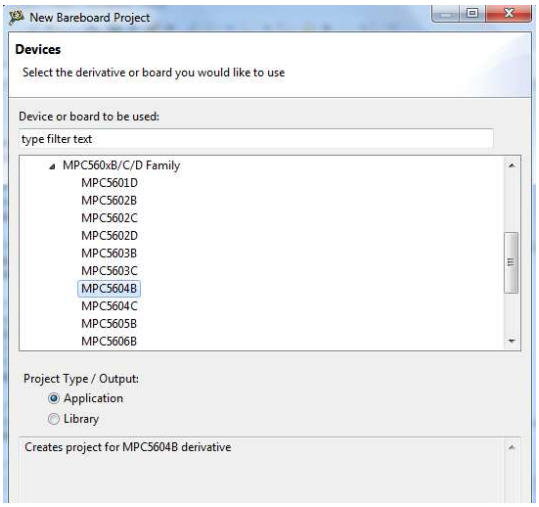
L'outil Freescale CodeWarrior Development Studio for Micro v10.5 permet de développer, compiler et debugger in-situ des projets pour les différentes familles de microcontrôleurs développés par Freescale sous environnement Eclipse.
Dans cette partie, nous allons brièvement décrire les principales étapes décrire permettant de lancer l'environnement, de créer un projet pour un microcontrôleur de la famille Qorivva, de charger l'exécutable sur un microcontrôleur et de faire un debug in-situ.

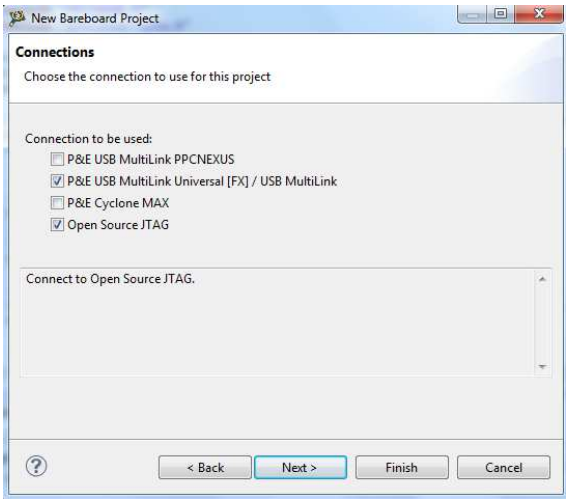
- Lancer Freescale CodeWarrior Development Studio for Micro v10. La fenêtre de dialogue **Workspace Launcher** s'ouvre. Vous pouvez sélectionner le chemin d'accès de votre dossier de travail.
- Au premier démarrage, la fenêtre Welcome s'affiche. Vous pouvez directement cliquer sur le bouton **Go to workbench**.
- La fenêtre ci-dessous, appelée **Workbench**, s'affiche (vide). L'ajustement des multiples fenêtres est configurable à partir du menu Window dans la barre de menus.

Fenêtre Workbench



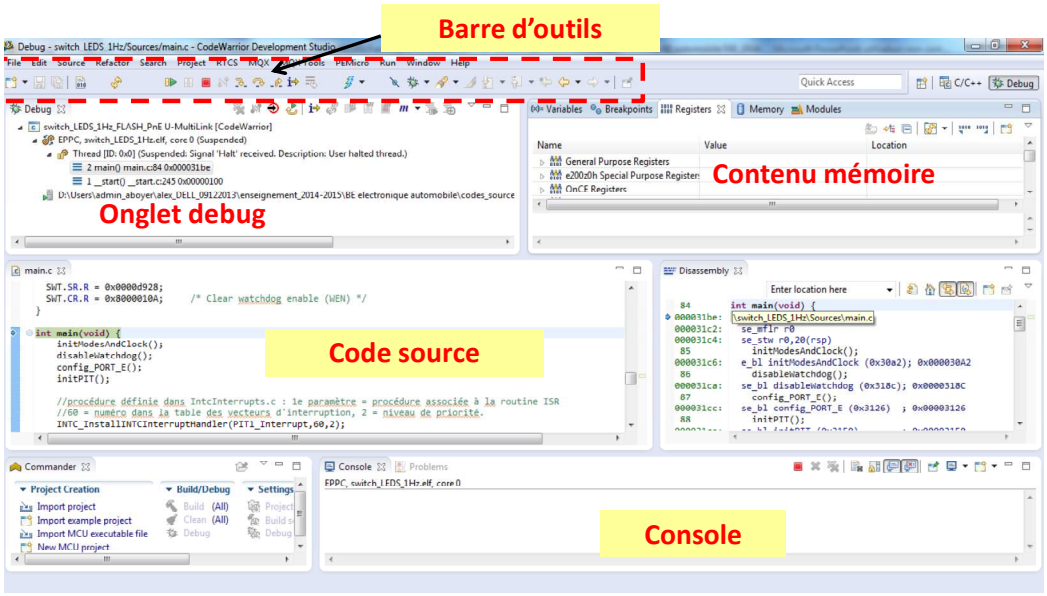
- Pour construire un nouveau projet, plusieurs méthodes sont possibles. Dans l'onglet de commande, cliquez sur le bouton **New MCU project**, ou dans le menu **File > New > Bareboard Project**.
- Nommez votre projet, spécifiez le chemin d'accès des fichiers puis cliquez **Next**.
- Sélectionnez ensuite le composant cible. Dans la liste, sélectionnez : **Qorivva > MPC560xB/C/D Family > MPC5604B**. Cliquez sur **Next**.
- Sélectionnez le ou les modes de connexion avec la cible : P&E USB MultiLink Universal et Open Source JTAG. Cliquez sur Next.
- Indiquez ensuite le langage (langage C), les options d'instruction (VLE) et le format en virgule flottante (None par défaut). Terminez en cliquant sur **Finish**. Le projet apparaît dans l'onglet Projet, le Perspective Switch est en mode C/C++. En face du nom du projet apparaît la zone mémoire où le code exécutable sera stockée (en RAM ou en Flash). Sélectionnez Flash si vous voulez que celui-ci soit stocké dans une mémoire non volatile.





- Vous pouvez écrire le code source dans les différents fichiers de votre projet. Vous pourrez utiliser les exemples de codes sources qui vous sont fournis pour prendre en main l'outil.

- Une fois le code source écrit, il est nécessaire de le compiler et de construire le projet. Pour cela, dans l'onglet de commande, cliquez sur le bouton **Build** ou dans le menu **Project > Build**.
- Pour lancer le chargement de l'exécutable dans la mémoire du microcontrôleur et lancer le debug in-situ, cliquez sur le bouton **Debug** dans l'onglet de commande. Si la carte démo est correctement alimentée et connectée, le programme charge l'exécutable et le Perspective Switch est en mode Debug.
- La fenêtre ci-dessous s'ouvre.



- La barre d'outils permet de réaliser différents modes d'exécution (run, pas à pas), de suspendre et d'arrêter l'exécution. Plusieurs points d'arrêt peuvent être inclus par un double clic dans la fenêtre Code Source au niveau de l'instruction où l'on souhaite arrêter l'exécution.
- L'onglet Contenu mémoire permet d'avoir accès au contenu des variables, registres internes et de manière générale de toute la mémoire.

- Une fois l'opération de Debug effectué, pour revenir au développement du code source, vous pouvez basculer sur la vue C/C^{++} dans le Perspective Switch.
- Pour fermer un projet, cliquez dans la barre de menus sur **Project > Close Project**. Pour le supprimer de l'onglet projet, clic droit sur le nom du projet puis **Delete**.
- Pour ouvrir un projet existant, vous pouvez cliquer sur **File > Import** ou le bouton **Import project**.

3. *Exemples de codes sources - Librairie*

Afin de prendre en main le microcontrôleur, vous pouvez télécharger la note d'application AN2864 – « MPC5500 & 56000 - Simple Cookbook » sur le site de Freescale. Ce document vous propose plusieurs exemples de codes sources pour les microcontrôleurs de la famille MPC5500 et 5600.

Sur le site http://www.alexandre-boyer.fr/enseignements.htm vous trouverez aussi quelques exemples de codes sources adaptés au starter kit TRK-MPC5604B (exemples_codes_sources_MPC5604B.zip). Ci-dessous, une liste des programmes fournis dans ce fichiers.

exemples_codes_sources_MPC5604B.zip :

- ADC_CTU_PIT3 : conversion analogique-numérique du signal délivré par le photorécepteur du starter kit TRK-MPC5604B. L'acquisition est cadencée via le CTU, par le débordement du timer PIT3.
- CAN_receive : programme basique de configuration du module FlexCAN en réception (CAN1). Le SBC est placé en mode Debug et l'interface CAN est activée via une commande SPI.
- CAN_transmit : programme basique de configuration du module FlexCAN en émission (CAN1). L'émission sur le bus CAN est cadencée par le timer1. Le SBC est placé en mode Debug et l'interface CAN est activée via une commande SPI.
- DSPI_command_send : envoi d'une commande sur le bus SPI (DSPI1).
- EIRQ_light_LED : allumage des 4 indicateurs lumineux du starter kit TRK-MPC5604B lors d'un appui sur l'interrupteur SW1-3, connectée à une entrée EIRQ (external interrupt request).
- OPWM_ch22_ch23 : configuration de 2 sorties PWM (module eMIOS). L'horloge système est délivrée par la FM-PLL et sa fréquence est de 45 MHz. Les 2 canaux PWM commutent à 1 KHz avec un rapport cyclique de 50 et 25 % respectivement.
- switch_LEDS_1Hz : clignotement toutes les secondes des 4 indicateurs lumineux du starter kit TRK-MPC5604B. L'opération est synchronisée par le débordement du timer PIT1. L'horloge système est délivrée par l'oscillateur à quartz (fréquence bus = 8 MHz).
- switch_LEDS_1Hz_PLL : clignotement toutes les secondes des 4 indicateurs lumineux du starter kit TRK-MPC5604B. L'opération est synchronisée par le débordement du timer PIT1. L'horloge système est délivrée par la FM-PLL (fréquence bus = 45 MHz).