Hes·so // VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Domaine Sciences de l'ingénieur
Rte du Rawyl 47
CH- 1950 Sion 2
Tél. +41 27 606 85 11
Fax +41 27 606 85 75
info@hevs.ch

www.hevs.ch

Filière Systèmes industriels

Orientation Power and Control

Diplôme 2008

Tiago Alexandre Nabais Lima

Commande et contrôle d'injection des moteurs à essence

Professeur Fariba Bützberger

Expert Cédric Overmeer

Sion, le 21 novembre 2008

Données du travail de diplôme
Daten der Diplomarbeit

Confidentiel / *Vertraulich* ☐ oui / ja ☒ non / nein

☒ FSI ☐ FTV	Année académique / *Studienjahr* **2007/2008**	No PS / *Nr. PS* **pc/2008/36**
Mandant / *Auftraggeber* ☒ HES—SO Valais ☐ Industrie ☐ Ecole hôte	Etudiant / *Student* **Tiago Alexandre Nabais Lima**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Ecole hôte
Professeur / *Dozent* **Fariba Bützberger**	Expert / *Experte* *(données complètes)* **Cédric Overmeer**	

Titre / *Titel*

Commande et contrôle d'injection des moteurs à essence

Description et Objectifs / *Beschreibung und Ziele*

Le débit d'injection des moteurs à essence est calculé en fonction de différents paramètres comme la vitesse, la charge, la température et la pression d'admission d'air du moteur, la composition des gaz d'échappement, etc.

Le but de ce projet est l'étude, la conception et le développement d'un système de commande et de contrôle de débit d'injection des moteurs à essence, selon les spécifications de son cahier des charges.

Une carte électronique embarquée permettant l'acquisition des mesures provenant des différents capteurs, ainsi que la commande et la régulation du débit d'injection avec un microcontrôleur a été conçue et réalisée pendant le projet de semestre. Les objectifs du projet de diplôme sont :

— Vérifications et tests hardware / software de la carte électronique sur le banc de test moteur du centre CFP de Sion.

— Analyse du fonctionnement du micro-contrôleur avec les programmes existants sur Internet.

— Optimisation des programmes pour obtenir une commande d'injection séquentielle ou semi-séquentielle, ainsi qu'un calibrage automatique des différentes grandeurs lors de démarrage. Tests et modifications des différents paramètres et programmes seront en accord avec l'entreprise Falko à Bex.

Signature ou visa / *Unterschrift oder Visum*	Délais / *Termine*
Resp. de la filière *Leiter des Studieng.:*	Attribution du thème / *Ausgabe des Auftrags:* 01.09.2008
Etudiant/*Student*:	Remise du rapport / *Abgabe des Schlussberichts:* 21.11.2008, 12:00
	Exposition publique / *Ausstellung Diplomarbeiten:* 28.11.2008
	Défense orale / *Mündliche Verfechtung:* semaine/Woche 49

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

Nabais Lima, Tiago Alexandre Sion, le 21 novembre 2008

Commande et contrôle d'injection des moteurs à essence

Steuerung und Kontrolle der Injektion von Benzinmotoren

Objectif

Le but de ce travail de diplôme est l'étude, la conception et le développement d'un système de commande et de contrôle de débit d'injection des moteurs à essence. L'objectif est tout d'abord de réaliser un système permettant une injection d'essence séquentielle sur n'importe quel type de moteur essence 4 temps. Le second objectif est de réaliser un calibrage automatique de la table d'injection donc des paramètres moteur.

Résultats

L'injection séquentielle fonctionne correctement. Elle a été testée en laboratoire pour tous type de moteur allant jusqu'à un douze cylindres. Le calibrage automatique a été testé sur un banc de test moteur. Celui-ci a été concluant du fait que le calibrage automatique a pu configurer une table d'injection en fonction des paramètre que j'ai choisis.

Mots-clés

Automobile, injection séquentielle, moteur à essence, tables d'injection, sonde lambda, gaz d'échappement

Ziel

Das Ziel dieser Diplomarbeit ist die Studie, das Design und die Entwicklung eines Systems zur Steuerung und Kontrolle des Durchflusses der Injektion von Benzinmotoren. Das Ziel ist zunächst ein System, dass eine sequenzielle Benzineinspritzung für irgendeinen 4-Takt-Benzin-Motor. Das zweite Ziel ist die Erstellung einer Tabelle zur automatischen Eichung der Einspritzung, also der Motor-Parameter.

Resultate

Die sequentielle Injektion funktioniert ordnungsgemäß. Sie wurde im Labor für jeden Motortyp mit bis zu zwölf Zylindern getestet. Die automatische Eichung wurde auf einem Prüfstand getestet. Dieser Test war erfolgreich, die automatische Eichung konnte eine Injektions-Tabelle generieren.

Schlüsselwörter

Autos, sequentielle Einspritzung, Benzin-Motor, Einspritzungs-Tabellen, Lamda-Sonde, Abgas.

Tables des matières

1 Contexte

Le principal objectif d'une personne possédant une automobile de nos jours est de réduire au maximum sa consommation d'essence ainsi que les rejets polluants de celle-ci. Bien entendu, le tout à un prix abordable. L'industrie automobile s'efforce de concevoir et produire ce genre de véhicules qui ne l'oublions pas sont de nos jours à la pointe de la technologie.

Le débit d'injection des moteurs à essence est calculé en fonction de différents paramètres comme la vitesse, la charge, la température et la pression d'admission d'air du moteur, la composition des gaz d'échappement, etc.
Le but de ce travail de diplôme est l'étude et la conception d'un système universel de commande et de contrôle de débit d'injection des moteurs à essence permettant :

- Le gain en consommation de carburant
- La diminution des rejets polluants
- Eventuellement le gain de puissance

Comme dit plus haut, l'automobile est de nos jours à la pointe de la technologie. C'est pourquoi ce système s'adresserait plus particulièrement aux véhicules possédant déjà quelques années (5 ans et plus). La plupart des gens utilisent quotidiennement ce genre de véhicules afin d'aller travailler par exemple. Le but de ce système est de pouvoir à moindre cout modifier l'injection des ces automobiles afin de réaliser des économies.

2 Cahier des charges

Le cahier des charge se trouve en seconde page du rapport. Il s'agit en fait de la donnée du travail de diplôme (Echéancier).

3 Planning

Afin de réaliser au mieux ce travail de diplôme, il a fallu dès le départ réaliser un planning général me permettant d'avoir un fil conducteur tout au long de ces trois mois.

Planning en Annexe 18.1.

4 Le moteur thermique 4 temps

Le moteur à explosion (dit aussi moteur thermique) est conçu pour transformer une énergie thermique (l'explosion d'un mélange air-essence) en énergie mécanique d'abord linéaire (bielle) puis rotative (vilebrequin).

4.1 *La composition du moteur "4 Temps"*

Pour comprendre le fonctionnement d'un moteur "4 temps" il faut connaître les pièces qui le compose.

1. CAME :(Rouge)
Monté sur un arbre, cette pièce non circulaire sert à transformer un mouvement rotatif en mouvement de poussé.
2. SOUPAPE: :(Orange)
Obturateur mobile maintenu en position fermée par un ressort. Elle s'ouvre momentanément sous la pression de la came.
3. BOUGIE :(Jaune)
Elle fait jaillir une étincelle qui met le feu au mélange air/essence, créant une explosion.
4. PISTON :(Bleu)
Pièce cylindrique mobile, qui sert à comprimer les gaz en vue d'une explosion, et qui après l'explosion transforme un énergie thermique en énergie mécanique.
5. BIELLE :(Turquoise)
Tige rigide, articulée à ses deux extrémité. Elle transforme un mouvement linéaire en mouvement rotatif.
6. VILEBREQUIN :(Vert)
Arbre articulé en plusieurs paliers excentrés. Transmet indirectement l'énergie mécanique à la boîte.
7. DISTRIBUTION :(Violet)
Mécanisme de régulation d'entré et de sortie des gaz à travers la chambre de combustion. Créant un parfaite coordination entre les arbre à came et le vilebrequin.
8. CHAMBRE DE COMBUSTION :(Grise)
Chambre hermétique où est injecté le mélange air/essence pour y être comprimé, enflammé, et créer un énergie mécanique.

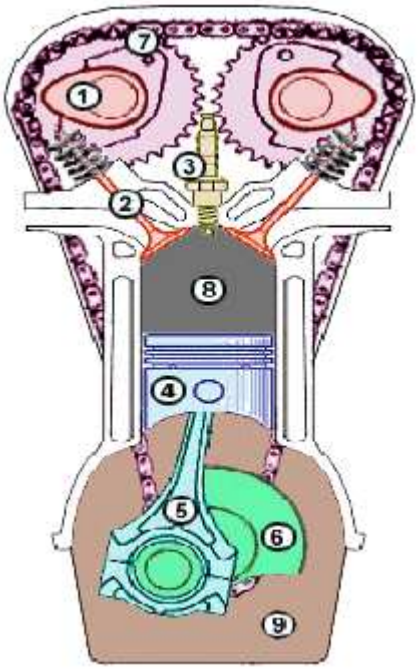


Figure 4.1-1 Moteur 4 temps

9. LUBRIFICATION:(Marron)
Les pièces situées sous le piston baignent dans l'huile. Cette huile n'est jamais en contact avec le dessus du piston. Elle lubrifie: Vilebrequin, Bielle, Piston, et parfois c'est la même qui lubrifie la boîte de vitesse. (A la différence des deux temps, ou la boite est séparé du moteur)

4.2 *Le cycle 4 temps*

Un piston décrit un mouvement de bas en haut dans le but de comprimer de l'essence pour créer une explosion.
Le **PMB** ou **Point Mort Bas**, position minimale du piston. (4.2-1)
Le **PMH** ou **Point Mort Haut**, position maximale du piston. (4.2-2)

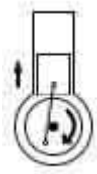


Figure 4.2-1

Figure 4.2-2

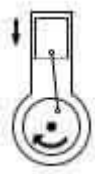


Figure 4.2-3

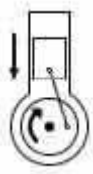


Figure 4.2-4

Un moteur à explosion utilise un gaz inflammable (essence + air). Ce gaz en explosant libère une énergie qui pousse le piston vers le bas, entraînant un ensemble de pièces mobiles qui feront avancer la voiture. On appel "4 temps", le cycle de quatre étapes auquel sont soumis les gaz pour créer cette explosion. Soit deux montées, et deux descentes.

L'admission:
Premier temps, Le piston descend créant une dépression (PMH vers PMB) qui aspire les gaz par la soupape d'admission dans la chambre de combustion. La soupape d'échappement reste fermée.

Compression:
Second temps, Le piston remonte, (PMB vers PMH) comprimant les gaz enfermés dans la chambre de combustion. La soupape d'admission et la soupape d'échappement sont fermées.

Explosion: (ou détente) :
Troisième temps, la bougie crée une étincelle qui enflamme les gaz comprimés, l'explosion pousse le piston vers le bas (PMH vers PMB)La soupape d'admission et la soupape d'échappement sont fermées.
N.B: C'est le seul temps moteur qui crée assez d'énergie pour d'une part relancer un cycle de 4 temps, et pour d'autre part faire avancer la voiture.

Echappement:
Quatrième temps, La soupape d'échappement s'ouvre, le piston remonte poussant les gaz brûlés vers le conduit d'échappement. La soupape d'admission reste fermée.

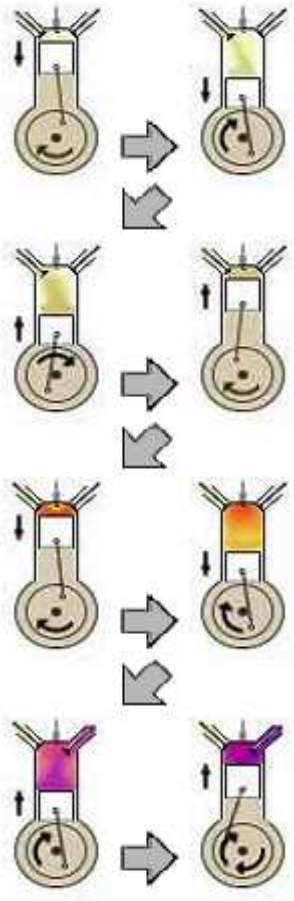


Figure 4.2-5 Le cycle 4 temps

5 Injection d'essence

Les véhicules automobiles utilisent généralement un carburant liquide. Comme la combustion n'est possible que lorsque le carburant est à l'état gazeux et en présence d'oxygène, il faut préparer ce mélange. Le système de carburation doit fournir au moteur un mélange d'air et de carburant finement pulvérisé, homogène et correctement proportionné, afin que la combustion de ce mélange développe le maximum d'énergie. Le mélange air-carburant est aussi appelé « mélange carburé »

5.1 *Dosage du mélange*

Mélange air-carburant :

Le fonctionnement d'un moteur à essence nécessite l'apport d'un mélange air-carburant dans une proportion bien déterminée. La combustion idéale, théorique, complète est obtenue pour un rapport air/carburant de 14,7/1. Ce paramètre est aussi appelé « rapport stœchiométrique ». Il faut distinguer le dosage théorique et le dosage réel du mélange.
Le dosage réel spécifique d'un moteur à essence dépend essentiellement du rapport du mélange air-carburant. En théorie, il serait nécessaire d'avoir un excédent d'air pour assurer une combustion complète et obtenir donc une économie de carburant. En réalité, des limites bien définies doivent être respectées à cause de l'inflammabilité du mélange et de la durée disponible de combustion.
Les moteurs présentent une consommation minimale lorsque le rapport air/carburant est d'environ 15 à 18kg d'air pour 1kg de carburant (mélange pauvre). Cela signifie, en clair, que la combustion d'un litre d'essence nécessite l'apport d'environ 10'000 l d'air. Les moteurs fonctionnant la plupart du temps à charge partielle, leur conception est réalisée de manière à obtenir le minimum de consommation à ce régime. Aux autres états de fonctionnement, tels que ralenti et pleine charge, il est préférable d'obtenir une composition du mélange plus riche en carburant. Le rôle des carburateurs ou des systèmes d'injection est de préparer un mélange air-carburant le plus favorable en fonction des différentes condition de fonctionnement du moteur.

Coefficient d'air :

Le coefficient d'air λ (lambda) a été retenu pour caractériser le rapport entre le mélange air-carburant réellement disponible et la valeur théorique nécessaire (14,7/1) :

$$\lambda = \frac{masse\_d\_air\_admise}{besoin\_en\_air\_pour\_combustion\_stoechiométrique}$$

λ = 1 : La masse d'air admise correspond à la valeur théorique nécessaire
λ < 1 : Concrétise un déficit d'air ou un mélange riche. Une augmentation de puissance apparaît pour λ = 0,85 à 0,95 ainsi qu'une modification du niveau de pollution
λ > 1 : Concrétise un excédent d'air ou un mélange pauvre dans la plage λ = 1,05 à 1,3. Ce coefficient d'air entraîne une réduction de la consommation et de la puissance, une modification du niveau de pollution et une augmentation de la température du moteur en raison de la combustion plus lente.

5.2 *Injection d'essence à commande électronique*

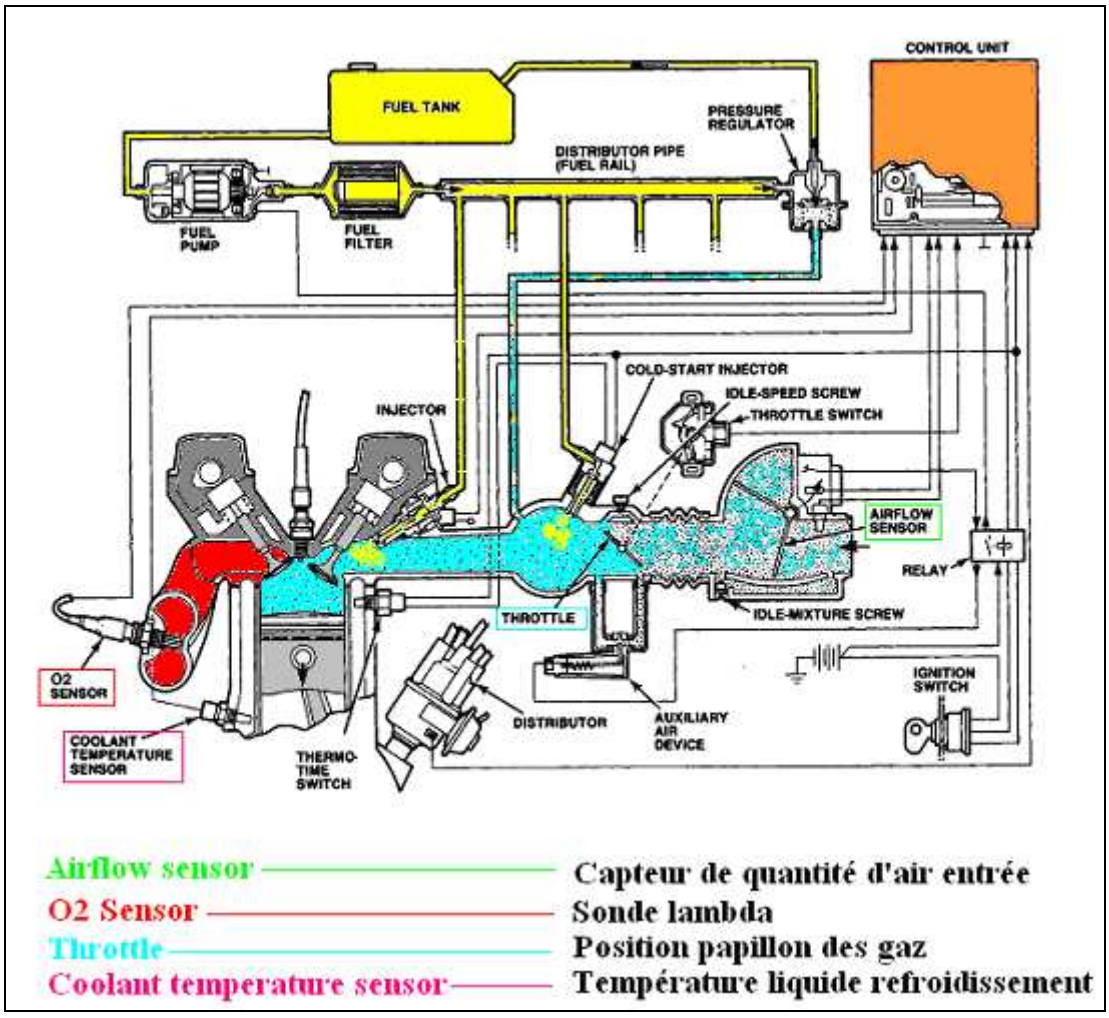


Figure 5.2-1

Les différents systèmes d'injection à commande électronique se différencient, du point de vue principe de fonctionnement, principalement par le système de mesure de la quantité d'air aspiré. Comme le traitement des différents signaux nécessaires à déterminer la quantité d'essence à injecter se fait toujours par un calculateur électronique, chaque système va posséder, dans son exécution de base, le même nombre de capteurs et sondes. Tous les système ont en commun les capteurs et sondes suivants :

- Capteurs nécessaires à la mesure du débit d'air
- Capteur régime moteur
- Contacteurs ou potentiomètres de papillon
- Sonde de température moteur

Selon le degré d'optimisation du système d'injection on trouvera en plus :

- Sonde de température d'air d'admission
- Sonde de détection d'oxygène (sonde lambda)
- Borne signal de démarrage ou thermo contact temporisé
- Correction en fonction de la tension batterie (rapidité réponse injecteurs)

Les perfectionnements suivants sont généralement intégrés :

- Régulation électronique du régime ralenti
- Limitation du régime maximum par coupure d'injection
- Coupure de l'injection en décélération
- Auto diagnostic
- Système d'auto adaptation qui compense les tolérances des composants et des variations de l'état mécanique du moteur

5.3 *Types d'injection*

Une injection d'essence peut s'effectuer de différentes manières. Celles-ci possède leurs avantages et inconvénients. On distingue en général trois types d'injections :

1. Injection Mono point
2. Injection Multipoint
3. Injection Directe

5.3.1 Injection Monopoint

L'injection mono point est une injection commandé électro magnétiquement pour l'ouverture de son aiguille. Il n'y a qu'un seul injecteur qui prend place dans le boîtier papillon à la place du carburateur. Ce système d'injection est assez précis mais trop éloigner du cylindre, il tend donc à disparaître.

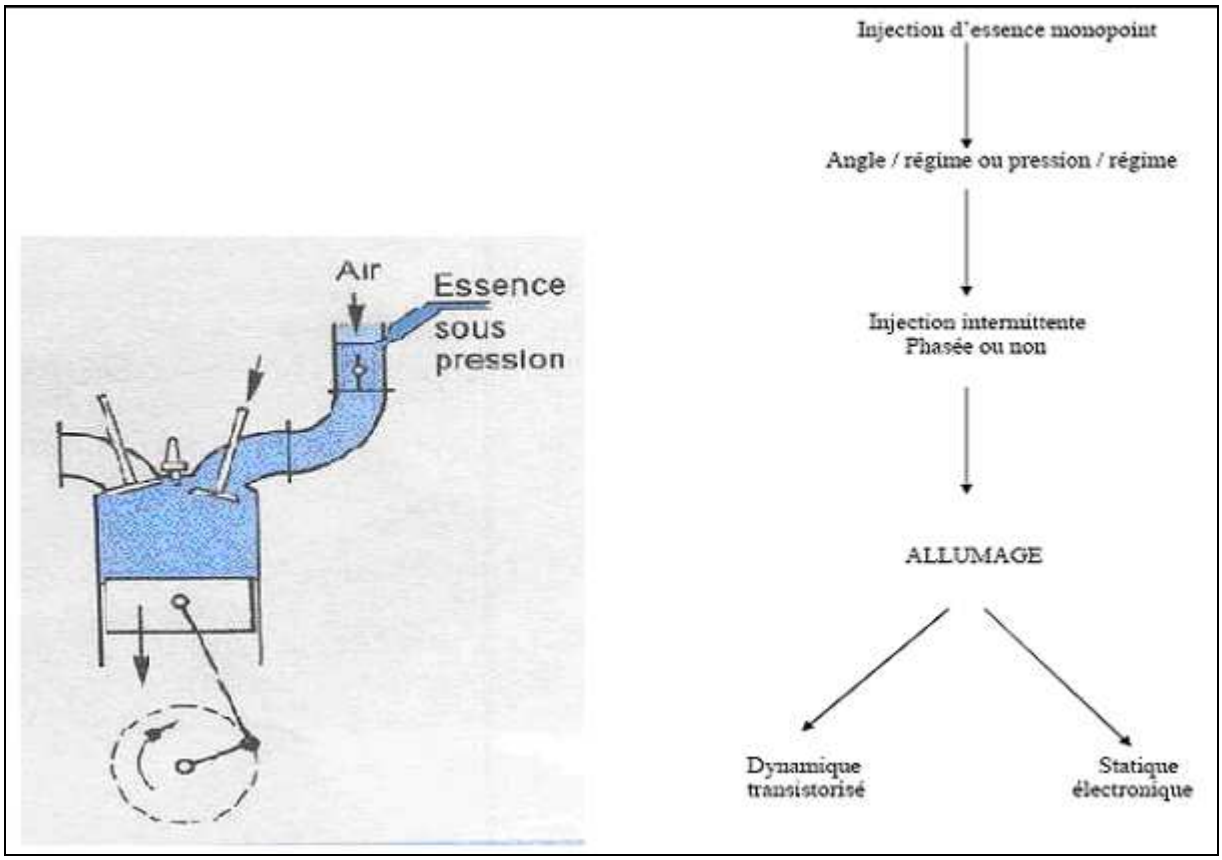


Figure 5.3-1 Injection d'essence Monopoint

5.3.2 Injection Multipoint

L'injection multipoints est une injection où il y a un injecteur par cylindre, il sont commandé soit électroniquement, soit mécaniquement. L'injecteur ce situe en amont de la soupape d'admission.

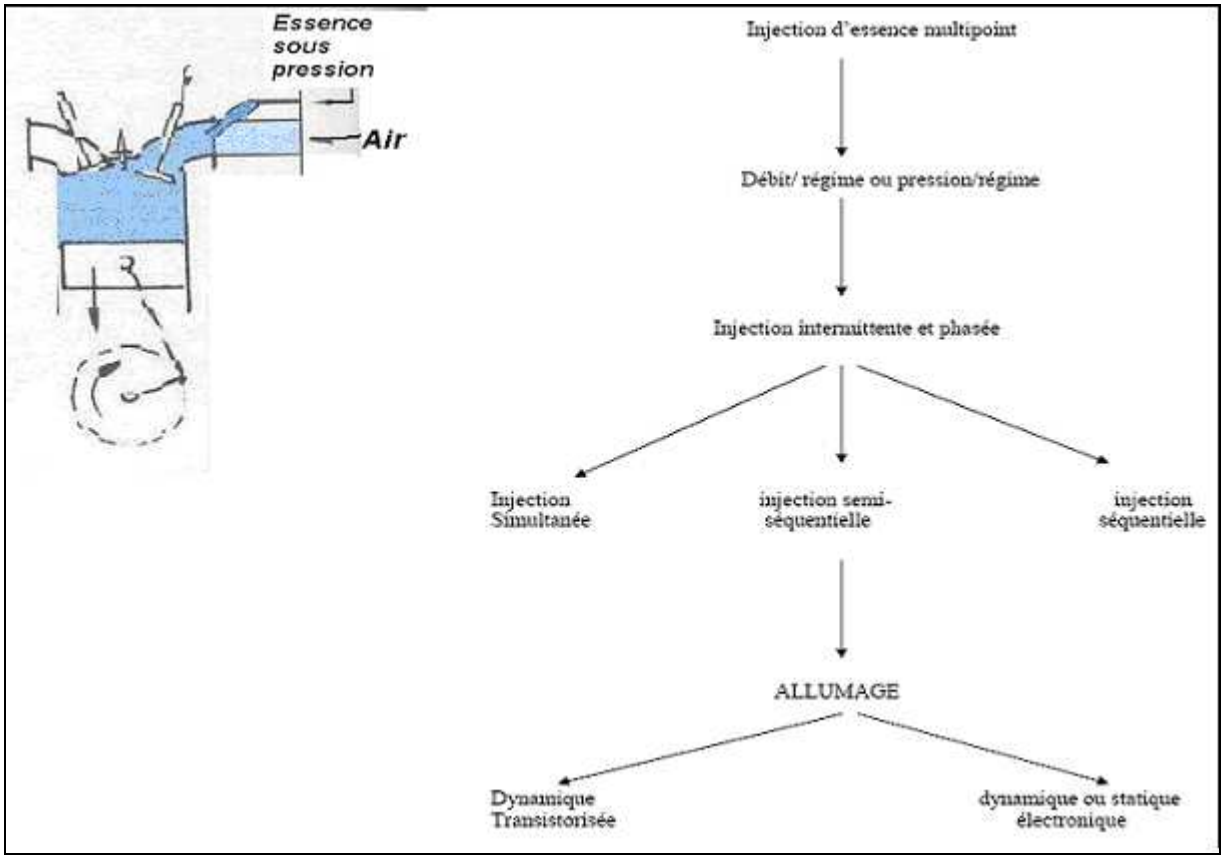


Figure 5.3-2 Injection d'essence Multipoint

L'injection Multipoint comporte aussi trois sous-catégories. Celles-ci se différencient de par la commande des injecteurs et sont:

- **Injection simultanée**: Tous les injecteurs s'enclenchent au même temps

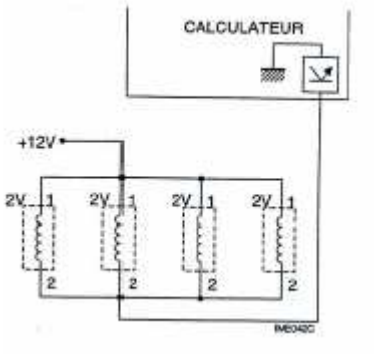


Figure 5.3-3 Injection simultané

- **Injection semi-séquentielle**: Les injecteurs s'enclenchent par banques

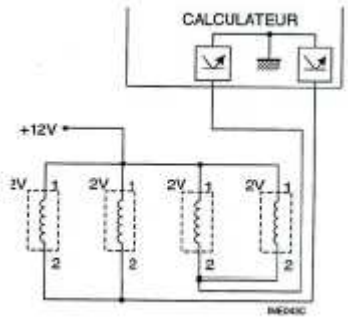


Figure 5.3-4 Injection semi-séquentielle

- **Injection séquentielle**: Chaque injecteur est commandé séparément

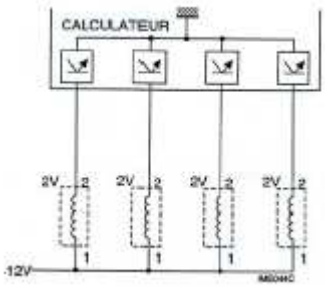


Figure 5.3-5 Injection séquentielle

5.3.3 Injection directe

L'injection directe est une technologie utilisée dans les moteurs à combustion interne. Elle consiste à diffuser le carburant directement dans la chambre de combustion plutôt qu'en amont dans la tubulure d'admission pour les moteurs à allumage commandé, ou dans une préchambre pour les moteurs diesel.

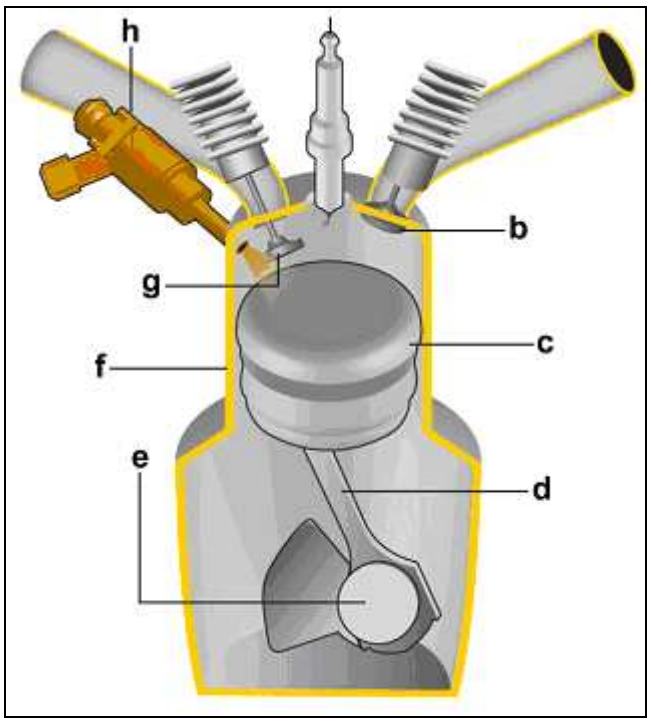
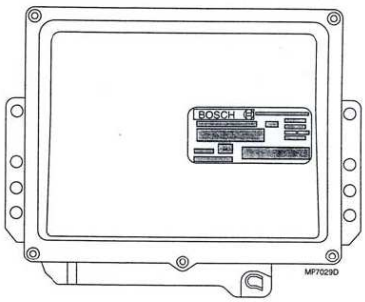


Figure 5.3-6 Injection directe

5.4 *Le calculateur*

Voici une photo montrant l'allure générale d'un calculateur utilisé dans l'automobile. Celui-ci est un calculateur Bosch. Le calculateur doit être capable de réaliser les injections, allumages vu dans le précédant chapitre.



TECHNOLOGIE « FLASH EPROM » : possibilité de mise à jour sans dépose du calculateur, par téléchargement à partir de l'outil via la prise diagnostic du programme du calculateur dans sa mémoire

A – ROLE

Recevoir les informations suivantes des différents capteurs et sondes

- Tension batterie
- + après contact
- + démarreur
- régime et position moteur
- référence cylindre
- température d'eau
- température d'air
- quantité d'air aspiré
- position papillon
- vitesse véhicule
- richesse
- détection cliquetis
- réfrigération BVA
- ADC
- diagnostic

CALCULATEUR

assurer les fonctions suivantes

Injection

Allumage

Figure 5.4-1 Calculateur

On peut voir sur ce schéma bloc une vue générale des informations reçues par les capteurs. Bien entendu ceci n'est qu'à titre d'exemple. Chaque calculateur possède ses propres spécifications.

5.5 *Les capteurs*

Capteur de régime et de position moteur : Il permet de déterminer le régime de rotation du moteur ainsi que la position du vilebrequin. Les informations fournies sont transmises au calculateur afin d'assurer les fonctions avance à l'allumage, charge bobine, quantité d'essence à injecter, régulation du régime de ralenti, et de déterminer une cadence d'injection.

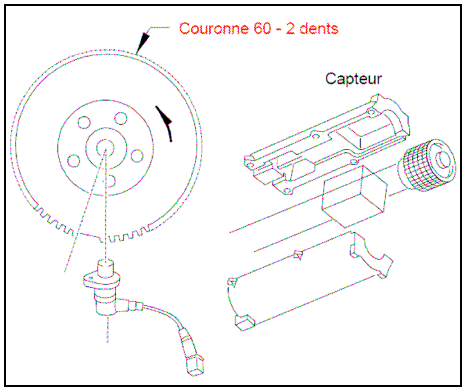


Figure 5.5-1 Capteur vilebrequin

Capteur de référence AAC: Le calculateur a besoin d'une référence de cylindre afin de pouvoir phaser les commandes des bobines d'allumage et des injecteurs en mode séquentiel (cylindre par cylindre dans l'ordre d'allumage 1 - 3 - 4 - 2). Pour cela, il reconnaît le PMH en allumage du cylindre n° 1.

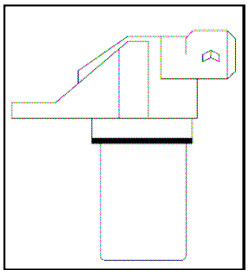


Figure 5.5-2 Capteur arbre à cames

Sonde de température d'air: La densité de l'air varie avec la température, si bien que l'information "quantité d'air aspirée" se trouve faussée pour des variations de températures importantes. Elle informe donc le calculateur de la température de l'air admis afin que celui-ci corrige le temps d'excitation des injecteurs. Lorsque la température de l'air baisse, sa densité augmente et le calculateur accroît la quantité d'essence injectée pour rétablir le rapport air/essence prévu. Elle est implantée sur le circuit d'air. C'est une thermistance de type CTN (résistance à coefficient de température négatif), ce qui signifie que lorsque la température de l'air admis diminue, la valeur de résistance augmente, et inversement. Le circuit de la sonde est alimenté sous cinq volts continu. Le calculateur mesure la tension aux bornes de la sonde, qui varie en fonction de la résistance de celle-ci.

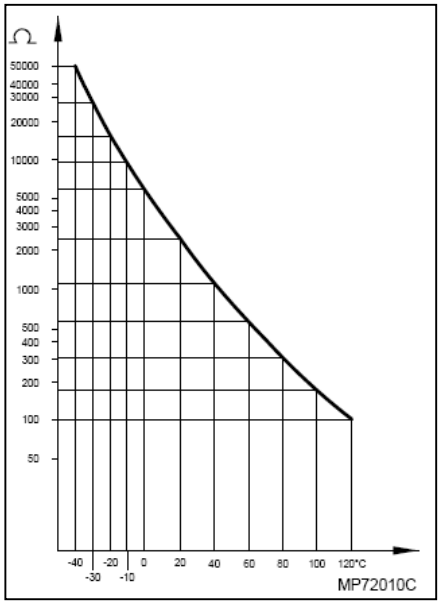


Figure 5.5-3 Capteur T° air

Sonde de température d'eau: Elle informe le calculateur de la température du liquide de refroidissement moteur. Elle lui permet d'apporter des corrections au niveau de l'injection et de l'allumage.

Capteur de pression: Il donne au calculateur l'information "charge" afin que celui-ci puisse déterminer la quantité d'essence optimale en fonction du remplissage et de la richesse souhaitée, ainsi que le point d'avance à l'allumage approprié aux conditions de fonctionnement du moteur. C'est un capteur de pression absolue de type piézorésistif se composant principalement de jauges de contraintes reliées à un pont de mesure. Ces jauges de contraintes se déforment sous l'action de la pression et il en résulte un signal de tension proportionnel à cette pression.

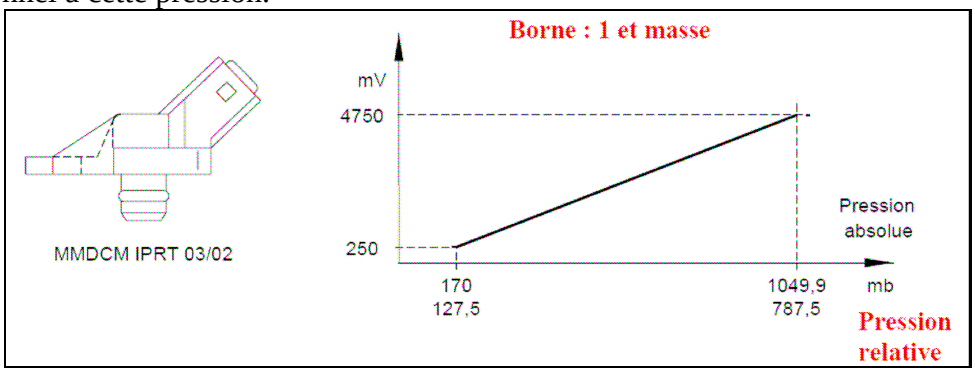


Figure 5.5-4 Capteur pression

Potentiomètre papillon: Fixé sur le boîtier papillon, il informe le calculateur de la position angulaire du papillon. Cette information est utilisée pour la reconnaissance des positions "pied levé", "pied à fond" et "transitoires". En fonction de ces données, le calculateur peut reconnaître le mode de fonctionnement et appliquer les stratégies d'avance et d'injection. De plus, il permet au calculateur de calculer un temps d'injection en fonction de la position du papillon pour assurer un mode secours en cas d'une défaillance du capteur de pression.

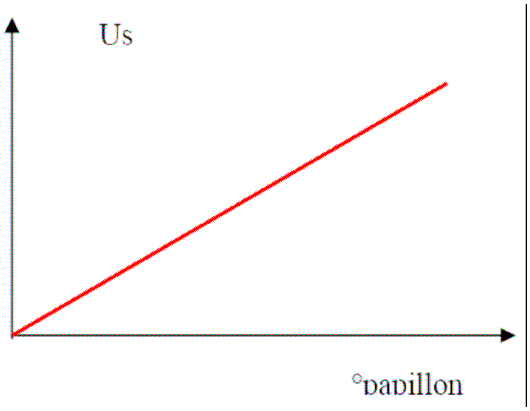


Figure 5.5-5 Capteur papillon des gaz

Capteur de cliquetis : La tendance des motoristes est actuellement d'accroître le rapport volumétrique pour réduire la consommation et accroître le couple moteur. L'augmentation de ce rapport risque toutefois de provoquer une combustion détonante du mélange air/carburant et le cliquetis du moteur. L'emploi d'un tel capteur permet une détection du phénomène et grâce au traitement électronique de l'avance à l'allumage, une correction rapide et efficace.

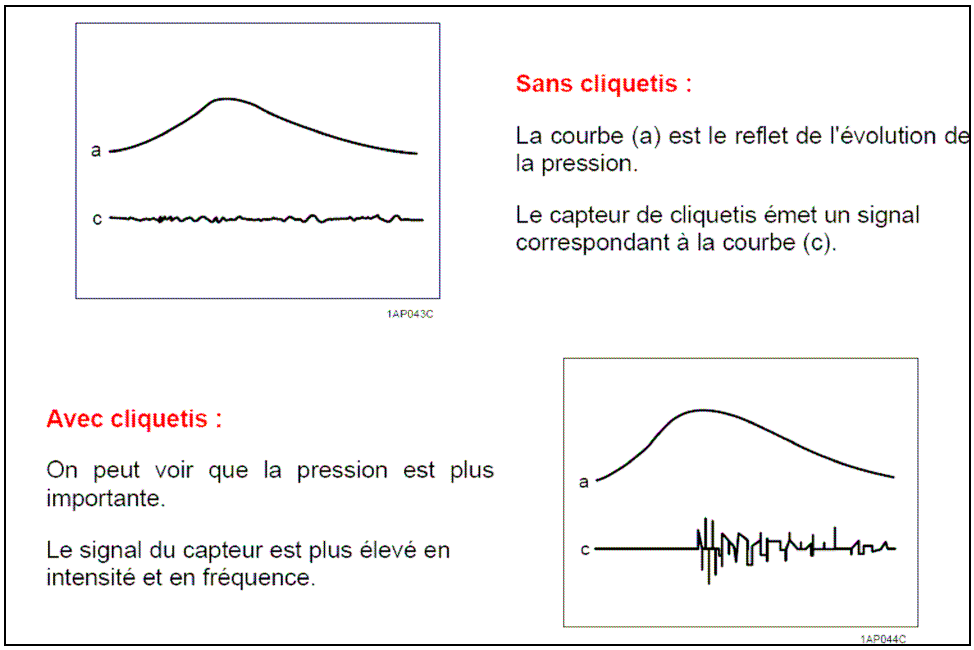


Figure 5.5-6 Capteur cliquetis

Sonde lambda: La sonde lambda est généralement placée sur le pot d'échappement entre le collecteur et le catalyseur. Ses mesures donnent la possibilité au calculateur d'injection de déterminer la proportion du mélange air-carburant pour laquelle l'efficacité du catalyseur sera optimale. Cela permet un faible niveau de rejets polluants et éventuellement une consommation réduite.

Figure 5.5-7 Sonde lambda

6 Conception carte « commande d'injection séquentielle »

La carte de commande d'injection réalisée pendant le projet de semestre possède deux sorties de puissance afin de commander les injecteurs. Celle-ci ne peut être utilisée que pour réaliser une commande semi-séquentielle, c'est-à-dire par banque d'injecteurs.

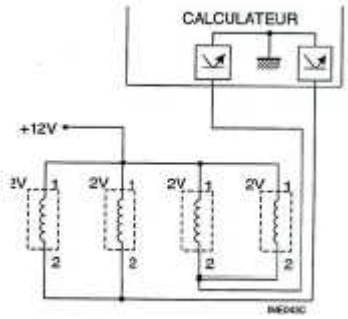


Figure 5.5-1 Injection semi-séquentielle

Elle nous est utile dans ce travail de diplôme afin de pouvoir réaliser une première approche sur le banc de tests du CFP pour pouvoir par la suite concevoir au mieux la carte d'injection séquentielle.

Le cahier des charges demande la réalisation d'une commande d'injection séquentielle. C'est-à-dire la commande séparée des injecteurs. La carte doit être capable de commander douze injecteurs séparément.

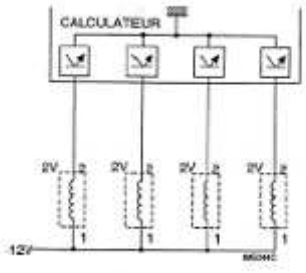


Figure 5.5-2 Injection séquentielle

Pour réaliser la commande séquentielle, il est également nécessaire de rajouter une entrée capable de traiter le signal provenant d'un capteur d'arbre à cames. Celui-ci est important afin que le microcontrôleur sache à quel moment le 1er cylindre est au point mort haut et entre en phase d'explosion.

6.1 *Schéma bloc*

Voici le schéma bloc de la carte à concevoir. On peut distinguer les différents blocs à réaliser ainsi que la direction du flux des données. Tout les blocs sont expliqués de façon précise dans les chapitres suivants.

La schématique, le routage, la commande des composants ainsi que le mode d'emploi de la carte se trouvent dans le CD-ROM (Annexe 18.6).

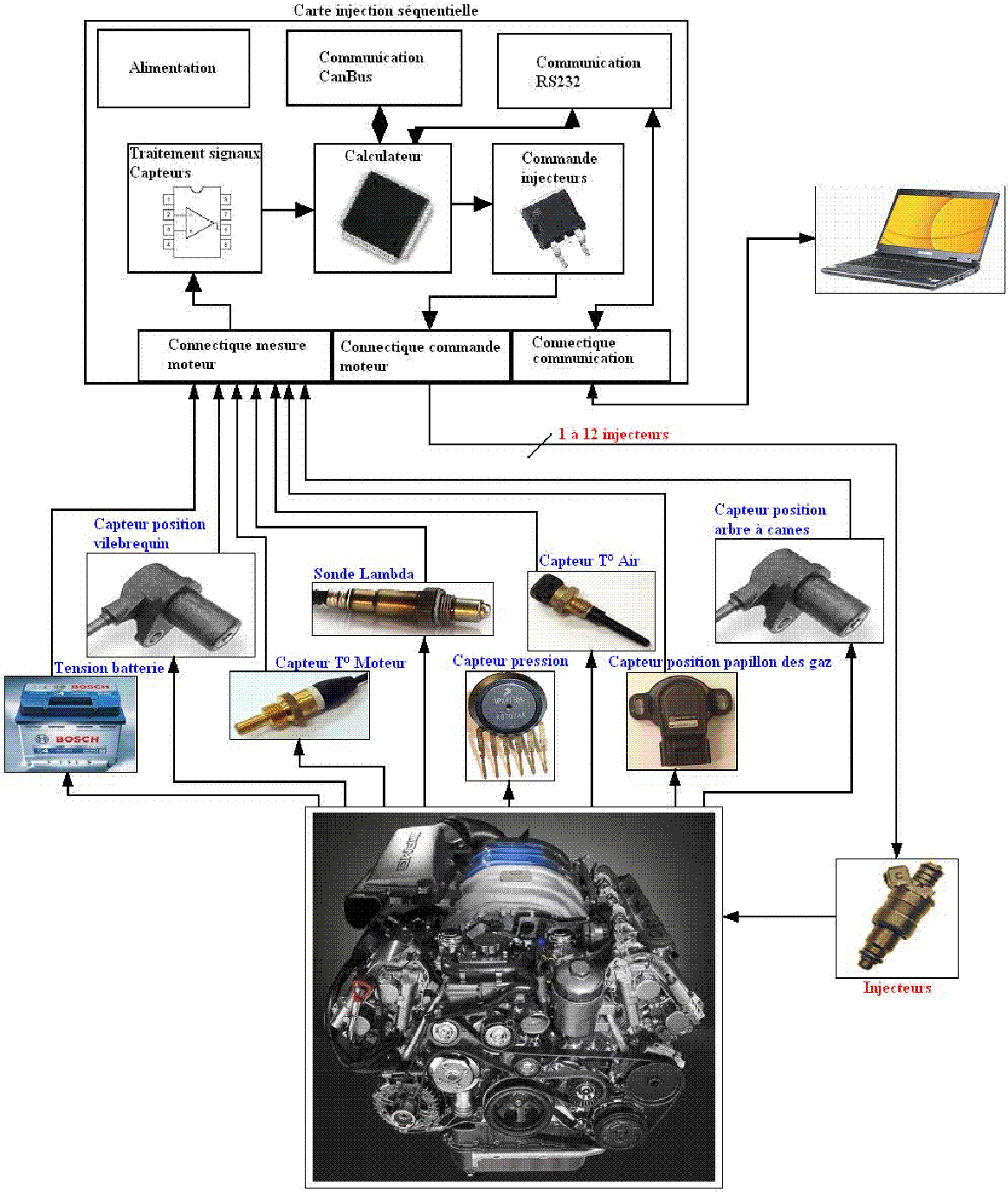
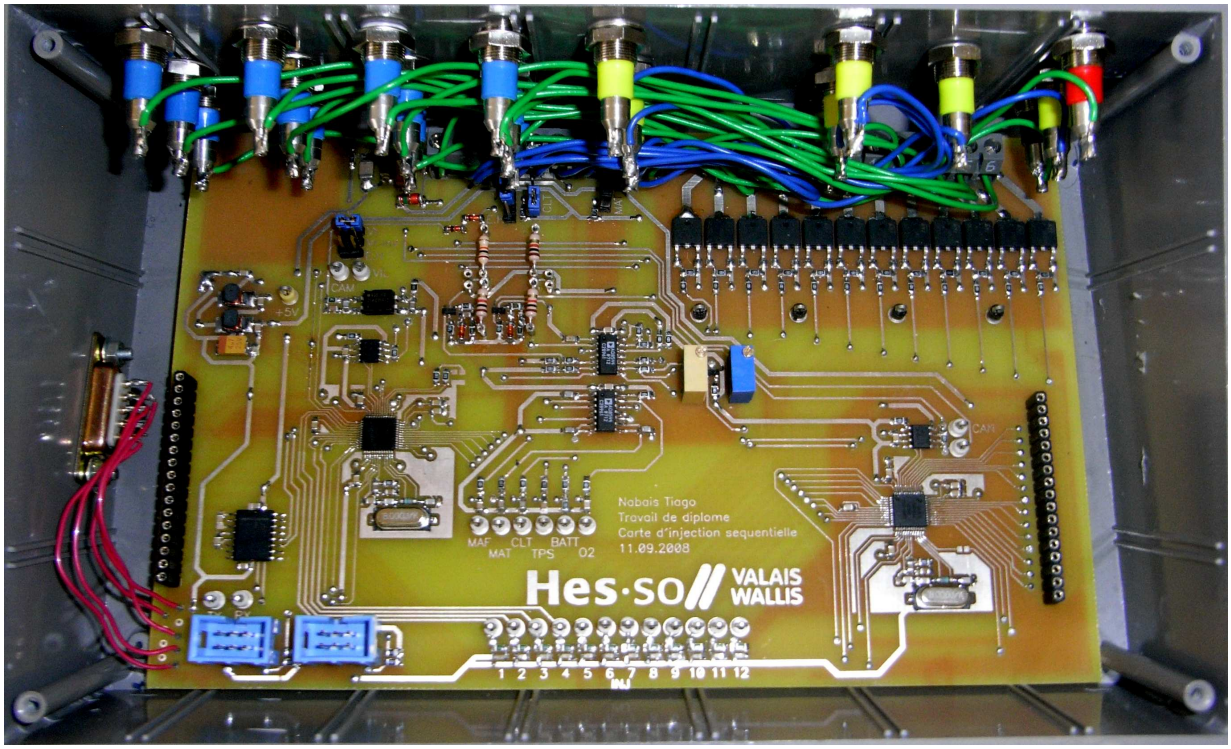


Figure 6.1-1 Schéma bloc carte injection séquentielle

Voici une figure de la carte d'injection séquentielle :



Dans le CD-ROM (Annexe 18.6) se trouve la description de la mise en service de la carte comprenant également un explicatif de la connectique d'entrées-sorties.

6.2 *Microcontrôleur*

Le microcontrôleur utilisé doit être de la même famille que celui utilisé pendant le projet de semestre afin de conserver les mêmes outils informatiques nécessaires à la programmation. Celui-ci est un MC9S12C96 provenant de la famille HCS12 du fabricant Freescale.

Caractéristiques :

Le MC9S12C96 utilisé est un 48LQFP. Celui-ci possède une unité centrale de traitement de 16 bits, 96K octets de Flash EEPROM, 4K octets de RAM, une interface série asynchrone (SCI), une interface périphérique série (SPI), un module Timer de 16 bits et 8 canaux (TIM), un module PWM de 8 bits et 6 canaux (PWM), un module de conversion analogique-digitale de 10 bits et 8 canaux (ADC), une interface CAN (MSCAN).
Afin de commander les injecteurs le plus précisément possible, il est nécessaire d'utiliser les sorties Timer du microcontrôleur. En effet celles-ci permettent un contrôle précis du temps d'injection. Le problème est que ce microcontrôleur ne possède que 8 sorties compteur et il nous en faut 12 pour commander tous les injecteurs. Il existe trois solutions afin de résoudre ce problème :

Solution 1 : Trouver un microcontrôleur avec plus de sorties compteur
Solution 2 : Réaliser deux cartes avec chacune un microcontrôleur indépendant
Solution 3 : Réaliser une carte avec deux microcontrôleurs communiquant entre eux par Can Bus

Solution 1:

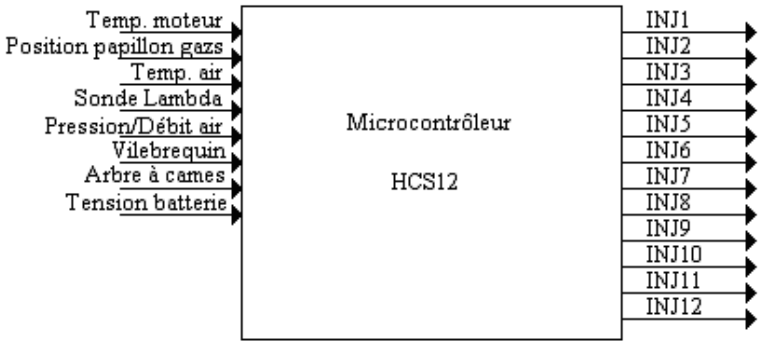


Figure 6.2-1: Structure solution 1

Aucun microcontrôleur de la famille HCS12 n'a été trouvé avec autant de sorties compteur. Etant donné que nous ne voulons pas changer de famille afin de conserver une bonne partie du code étudié pendant le projet de semestre, cette solution n'est donc pas valable.

Points positifs	Points négatifs
- Utilisation d'un seul microcontrôleur	- Non-utilisation des sorties Timer => Perte de précision

Tableau 1: Points positifs/négatifs solution 1

Solution 2:

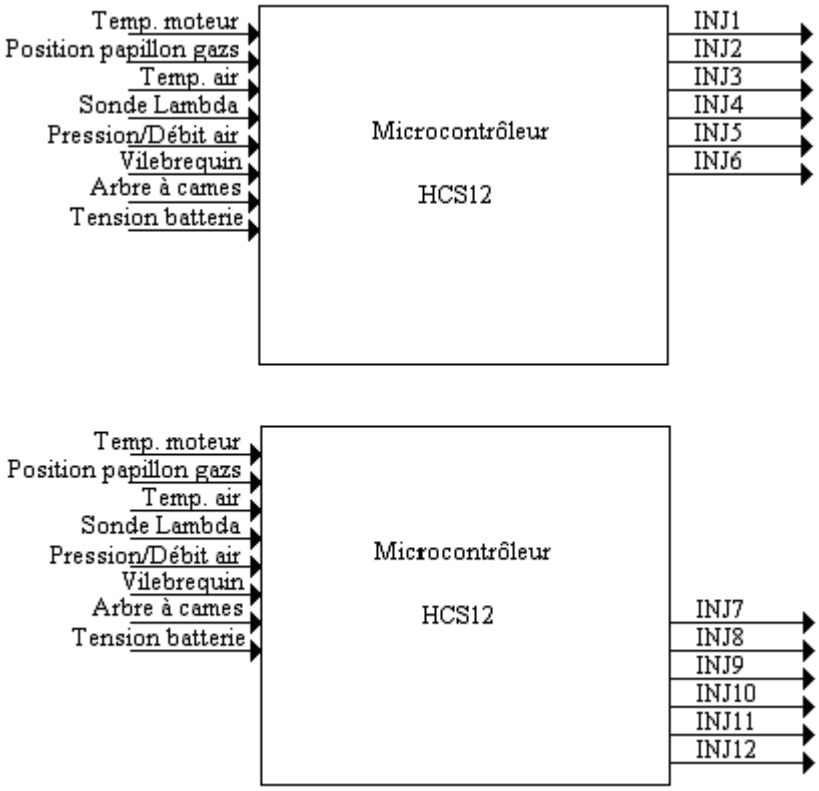


Figure 6.2-2: Structure solution 2

Le principe serait de relier les mesures faites sur le moteur aux deux microcontrôleurs et leurs implanter le même code. Il faudrait uniquement réaliser une communication entre les deux permettant de les synchroniser.

Points positifs	Points négatifs
- Peu de modifications du code existant	- Doubler les mesures => Perte inutile d'I/O
	- Calibrage du moteur pour les deux microcontrôleurs
	- Programmation des tables d'injection pour les 2 microcontrôleurs
	- Communication RS232 avec le PC en temps réel difficile avec les deux microcontrôleurs en même temps

Tableau 2: Points positifs/négatifs solution 2

Solution 3:

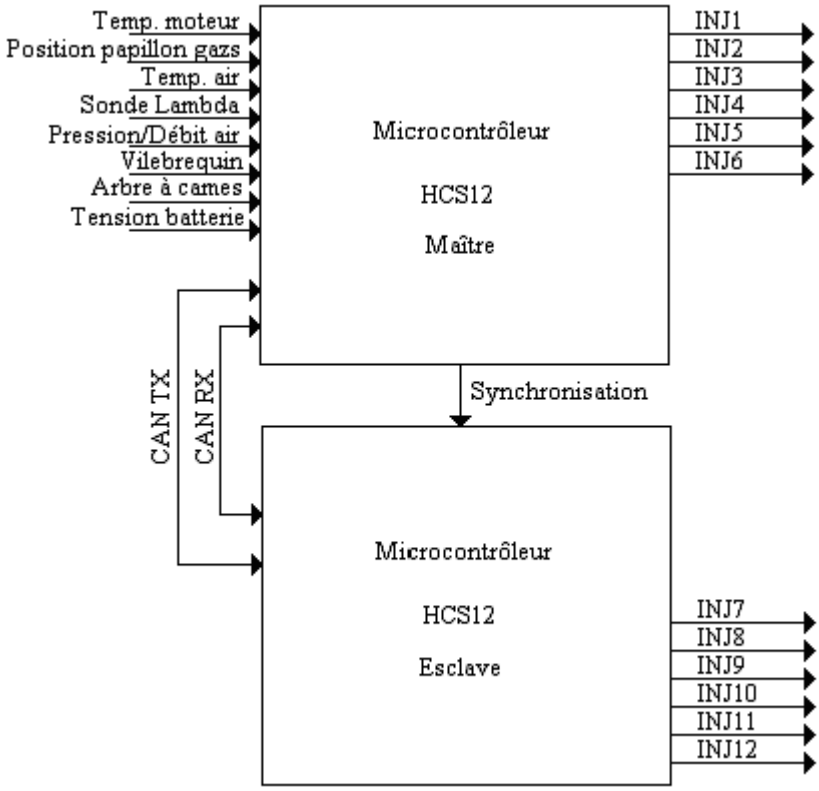


Figure 6.2-3: Structure solution 3

Voici la solution que j'ai choisie d'utiliser: Celle-ci consiste à utiliser à nouveau deux microcontrôleurs. Il y en aura un maître et un second esclave. Les données provenant du calibrage du moteur seront enregistrées dans le maître. Celui-ci les transmettra à l'esclave une à une avant chaque injection à partir du Can Bus. La communication avec le PC se ferait uniquement avec le master.

Points positifs	Points négatifs
- Peu de modifications du code existant	- Communication difficile à réaliser entre les 2 uC.
- Economies d'I/O	
- Permet de laisser le projet ouvert	
- Possibilité de rajouter d'autres fonctions par la suite	

Tableau 3: Points positifs/négatifs solution 3

6.2.1 Entrées-Sorties uC

On peut voir sur cette image la structure des entrées-sortie du microcontrôleur utilisé. J'ai utilisé 6 entrées A/D du uC maître pour le traitement du signal des capteurs, 4 sorties compteur pour les injecteurs. Concernant le uC 2, j'ai utilisé 8 sorties compteur pour les injecteurs. **L'affectation des ces entrées-sorties pour le uC maître et esclave se trouve en détail à l' annexe 18.2.1 et 18.2.2.**

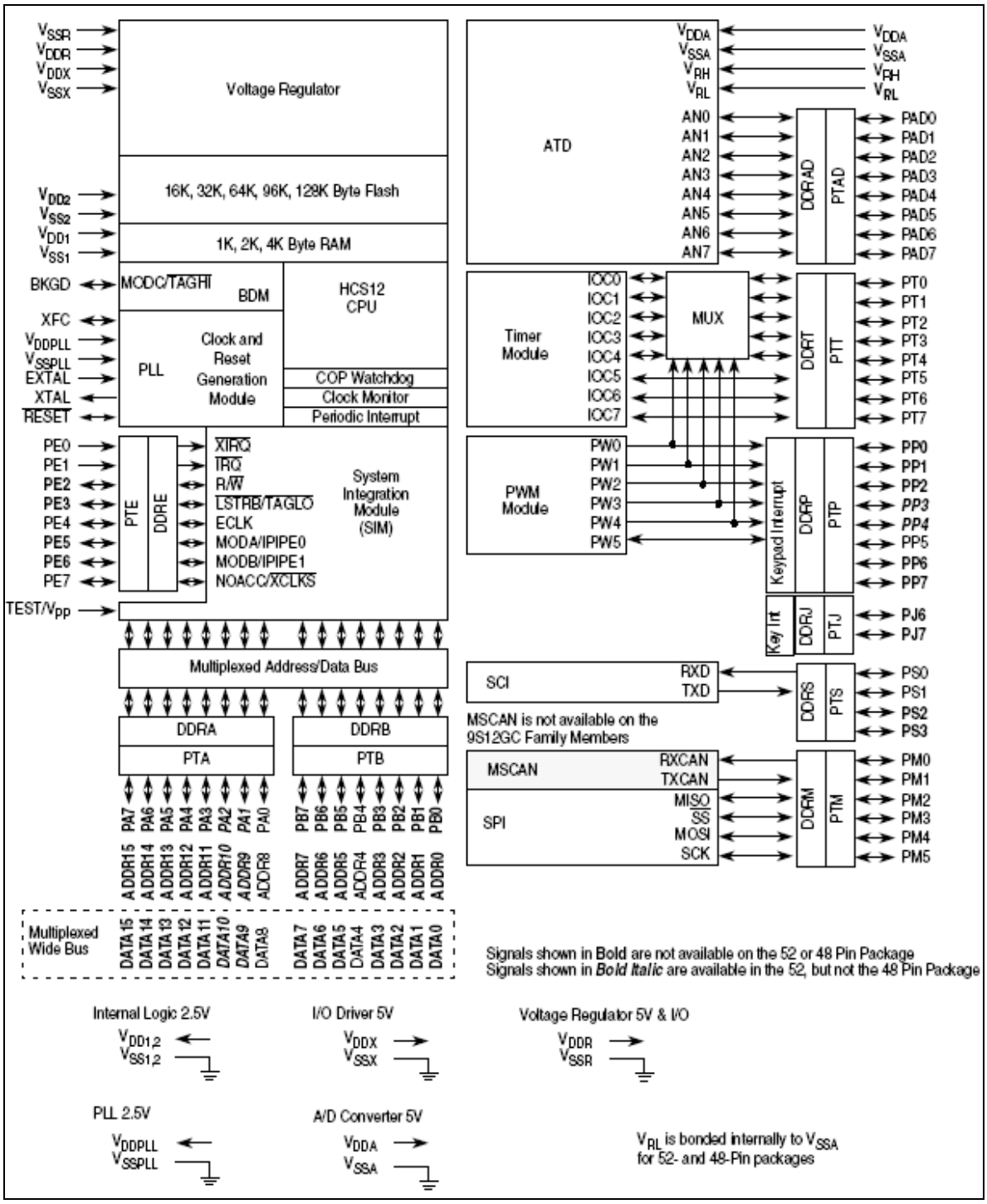


Figure 6.2-4 Structure uC MC9S12C

6.2.2 Clock

Ce bloc crée un signal d'horloge qui va cadencer les opérations du uC maître. Le même circuit est utilisé pour le uC esclave. Il s'agit d'un oscillateur Pierce. Le signal d' horloge a une fréquence de 8MHz.

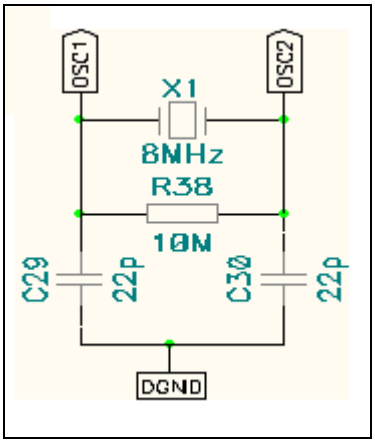


Figure 6.2-5: Schéma oscillateur Pierce

6.2.3 BDM

BDM est en fait un connecteur qui permet de reprogrammer le uC via le port USB du PC. BDM signifie en anglais 'Background debug mode'. Ceci veut dire que via ce connecteur il également possible de débuguer le code implémenté. Il en existe un par microcontrôleur.

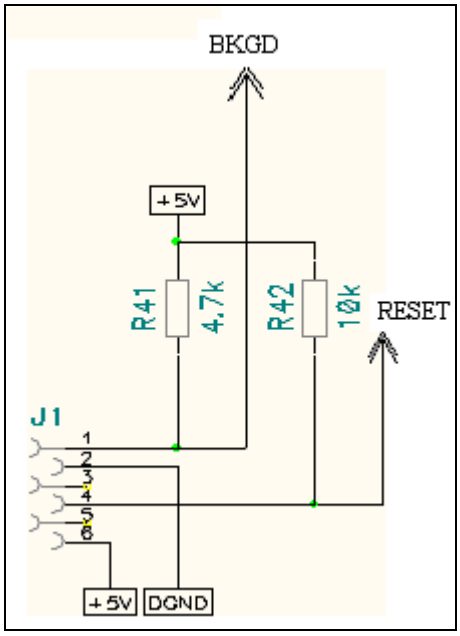


Figure 6.2-6: Schéma BDM

6.3 *Communication RS-232*

Ce bloc sert d'interface pour la communication série de la carte avec le PC. Il s'agit tout simplement d'un chip qui converti les signaux de tension Carte (Tx et Rx) =>PC (Port série). Le PC peut uniquement communiquer avec le uC maître.

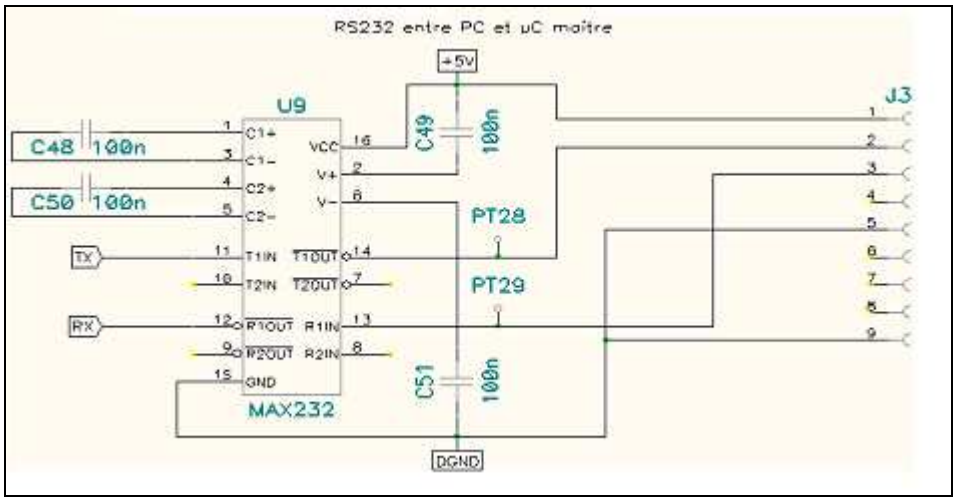


Figure 6.3-1: Schéma communication RS232

6.4 *Communication Can Bus*

Le protocole CAN (Control Area Network) est un protocole de communication série qui supporte des systèmes temps réel avec un haut niveau de fiabilité. Ses domaines d'application s'étendent des réseaux moyens débits aux réseaux de multiplexages faibles coûts. Il est avant tout à classer dans la catégorie des réseaux de terrain utilisés dans l'industrie pour remplacer la boucle analogique. Celui-ci est utile dans ce projet afin de faire communiquer les deux uC ensemble. L'avantage de ce protocole est que nous pouvons par la suite raccorder notre carte à d'autres modules simplement par deux fils.

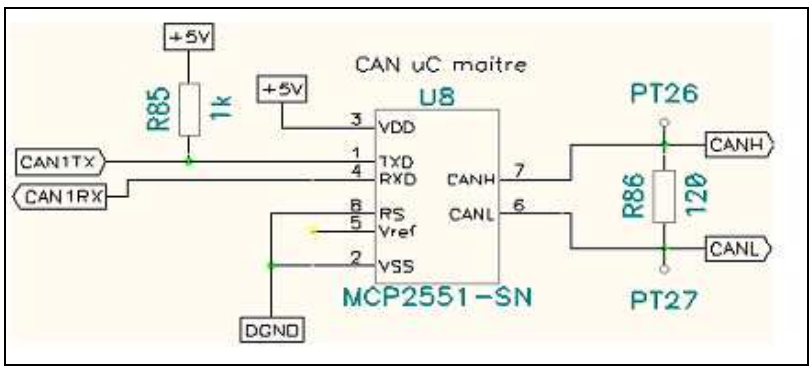


Figure 3.4-6.4-1: Schéma communication Can Bus

Le MCP2551 sert à l'interface entre le microcontrôleur et le bus physique. Ce circuit est également utilisé pour le uC esclave.

6.5 *Commande des injecteurs*

La commande des injecteurs sert à activer ou désactiver un MOSFET. Celui-ci laissera passer ou bloquera le courant passant à travers l'injecteur. Le signal de commande vient des uCs. Il s'agit d'un signal 0-5V. En cas de court-circuit du MOSFET en entrée, la résistance de 1k est là afin de limiter le courant sortant des uCs à 5mA. Les leds de visualisation sont des leds basse consommation. Le courant passant à travers celle-ci est limité à 1mA avec la résistance de 3.3k. L'injecteur est commandé par sa masse.

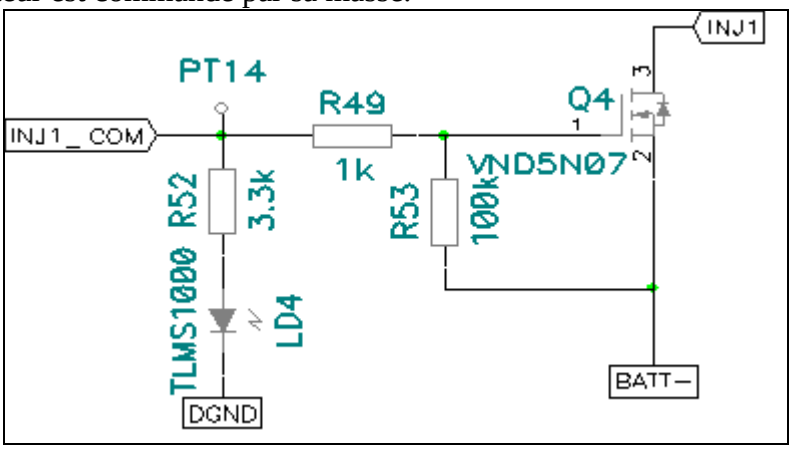


Figure 6.5-1 Schéma commande d'un injecteur

Le MOSFET utilisé est un VND 5N07. Il s'agit en fait d'un OMNIFET. Celui-ci possède une limitation de courant interne à 5A ainsi qu'une limitation de tension VDS à 70V. Ce qui permet de commander des injecteurs basse impédance sans utiliser de PWM pour limiter le courant. La tension VDS étant limitée en interne, l'utilisation d'une diode roue libre n'est plus indiquée. Pour chaque injecteur une commande de ce type a été réalisée.

Refroidissement OMNIFET :

Il est important que les OMNIFETs montés sur le PCB soit bien refroidis. Ceux-ci possèdent une limitation de température interne de 150°C. Si cette valeur est dépassée, l'OMNIFET interrompt la circulation du courant ce qui implique la fermeture de l'injecteur.

6.6 *Capteurs*

Les mesures provenant des différents capteurs du moteur doivent être adaptées afin de pouvoir être traitées par le uC maître. Il est également important que les entrées des mesures soient de hautes impédances afin de ne pas modifier la mesure. C'est pourquoi des suiveurs de tension ont été utilisés pour les mesures analogiques 0-5V. Les suiveurs sont alimentés en 5V. Ceci afin de limiter à 5V les signaux allant des suiveurs au uC maître. Il est également important d'utiliser des ampli OP Rail to Rail pour pouvoir utiliser toute la plage d'alimentation.

6.6.1 Pression d'air

Pour avoir une information de la quantité d'air entrant dans le moteur, on peut utiliser un capteur de pression et un capteur de T° air. Le capteur de pression fournis une tension de sortie allant de 0 à 5V en fonction de la pression d'air se trouvant à l'admission.

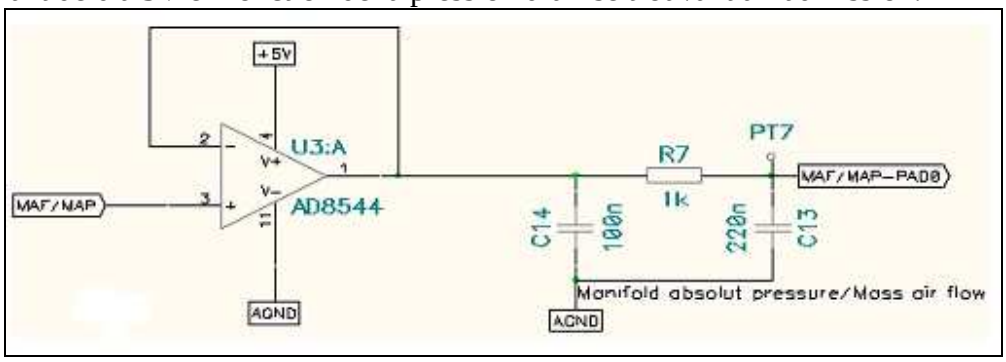


Figure 6.6.1-6.6-1 Schéma mesure capteur de pression/débit d'air

6.6.2 T°Moteur

La température du moteur est mesurée par une résistance NTC (résistance à coefficient de température négatif) immergée dans le liquide de refroidissement. Cette information est utilisée par le uC maître afin de modifier la quantité d'essence à injecter en fonction de la température du moteur.

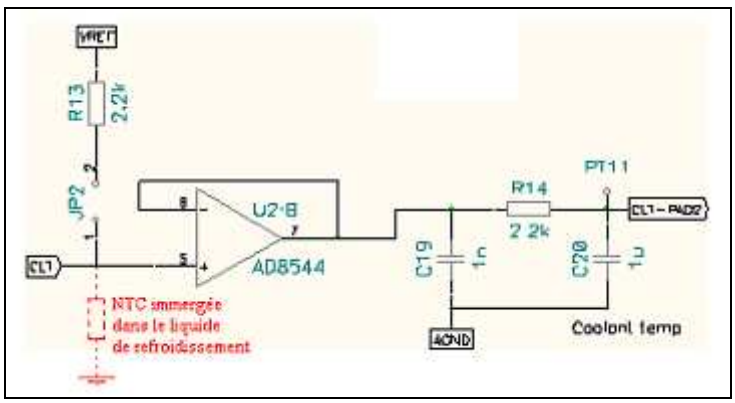


Figure 6.6.2-1 Schéma mesure T° Moteur

Etant donné que la mesure de température est faite par une résistance, il faut alimenter celle-ci. Nous savons que la valeur de la résistance de mesure est de 2.2 kOhms à 20°C. C'est pourquoi il faut rajouter une résistance en série pour avoir un diviseur de tension. La valeur de cette résistance est également de 2.2 kOhms. Ceci nous permet d'avoir comme tension 2.5V allant au uC maître pour une température de 20°C. Si cette mesure vient d'un module donnant directement une tension en fonction de la température, il est nécessaire d'enlever le JUMPER JP2.

6.6.3 T° Air

En ayant la pression de l'air entrant ainsi que sa température, le uC maître peut savoir combien d'air est entré dans le moteur. Le principe de fonctionnement est le même que pour le capteur de T° Moteur.

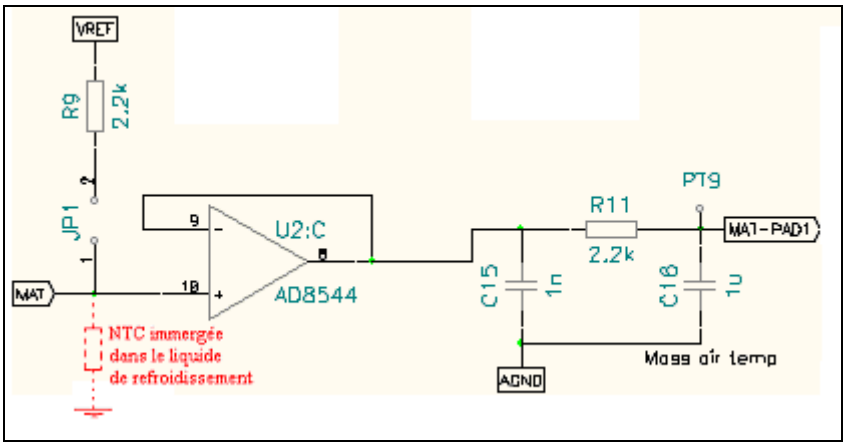


Figure 6.6.3-1 Schéma mesure T° Air

6.6.4 Sonde Lambda

La sonde Lamda est un capteur permettant de connaître le taux d'essence contenu dans l'air à la sortie du moteur. Celle-ci fourni une tension allant de 0V à 5V en sortie en fonction de la tenaur en essence contenue dans l'air qui sort des gazs d'échappement.

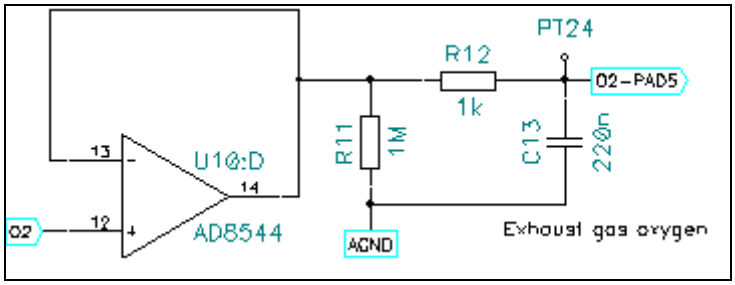


Figure 6.6.4-1 Schéma mesure sonde lambda

Cette valeur est utilisée par le uC maître afin de corriger la quantité d'essence à injecter ainsi que pour le calibrage automatique de la table d'injection.

6.6.5 Vilebrequin

Il est important de connaître la position du capteur de villbrequin. Celui-ci nous permet de savoir à quelle moment précis il faut injecter l'essence. Il permet également de savoir à combien de tours par minute le moteur tourne.

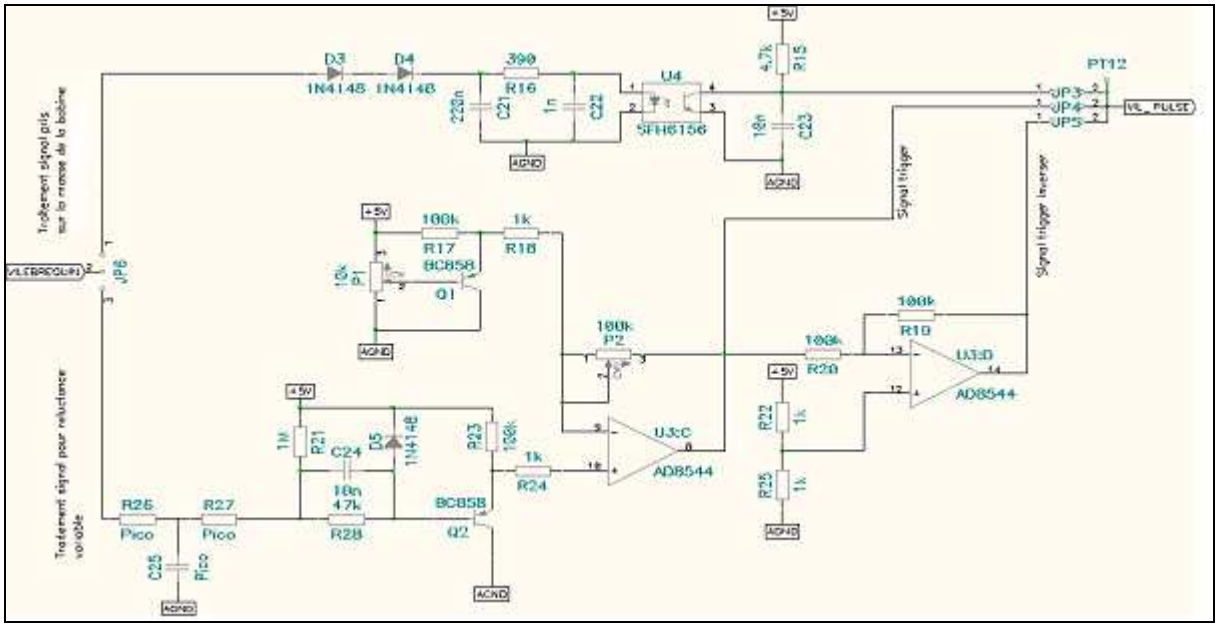


Figure 6.6.5-1: Schéma mesure capteur vilebrequin

Il existe principalement deux types de mesures. La première consiste à mesurer la tension sur la borne négative de la bobine d'allumage. Sur les vieux moteurs, l'allumage n'était pas électronique. Il s'agissait de contacts mécaniques qui forçaient la pin négative de la bobine à la masse à un moment précis. C'est pourquoi nous pouvons mesurer la tension sur la pin négative de la bobine.

La seconde contient les capteur à réluctance magnétique ainsi que les capteurs effet Hall. Ceux-ci nous permettent de connaître exactement la position du vilbrequin. Le jumper JP6 permet de sélectionner laquelle de ces deux familles nous voulons utiliser.

JP6: Position 1-2 => Borne négative de la bobine
JP6: Position 2-3 => Capteurs à réluctance magnétique ou effet Hall

Mesure de la tension sur la borne négative de la bobine

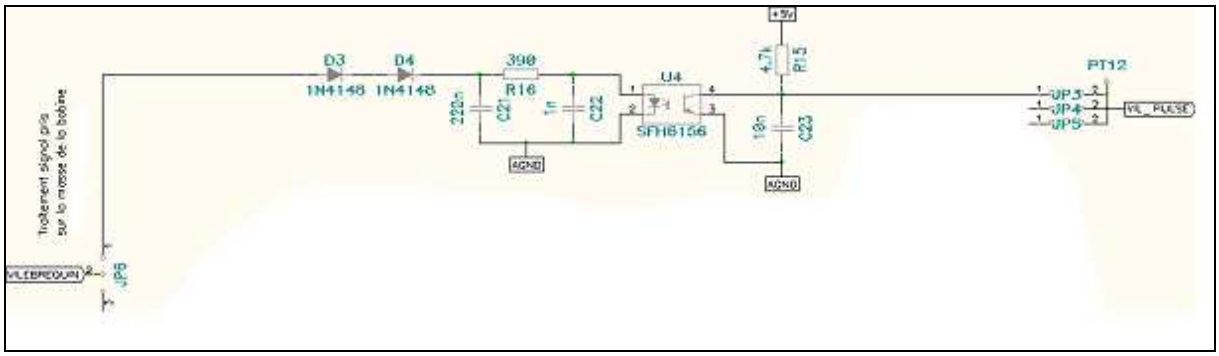


Figure 6.6.5-2 Schéma mesure capteur vilebrequin 1

En mesurant sur la borne négative de la bobine, le signal sera high(tension de batterie) quand la bobine ne conduit pas et low(0V) quand celle-ci conduit. Les diodes D4 et D5 ne servent qu'à abaisser la tension (D3+D4+Vf_opto_diode =2.6V). R16 sert à limiter le courant traversant la diode de l'optocoupleur. Le courant If pouvant traverser la diode de celui-ci est de 60mA.

$Vin\_\max = R*I+0.7+0.7+1.2=390*60*10_{-3}+2.6=26V$: Tension maximale pouvant être appliquée en entrée.

L'optocoupleur sert à séparer le potentiel provenant de la bobine à celui de la carte. Celui-ci est un open-collector. C'est pourquoi il a fallu rajouter R15 avec l'alimentation. En résumé, si la bobine conduit, il y aura la tension VCC en sortie et inversement.

Pour utiliser ce circuit, le JUMPER JP3 doit être mis.

Capteurs à réluctance magnétique

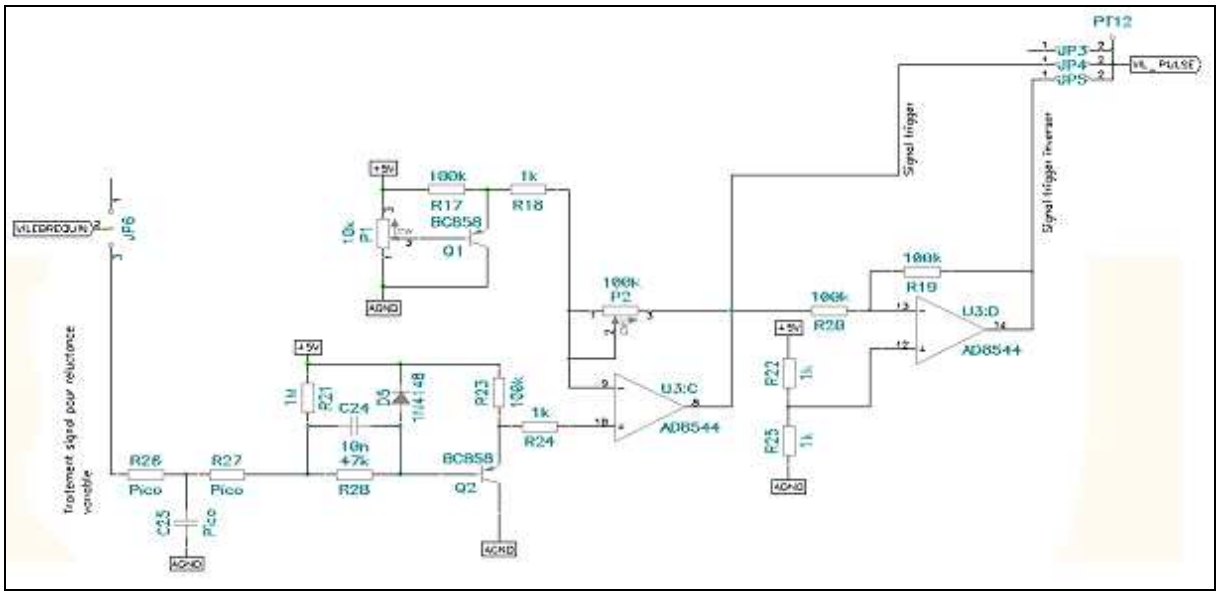


Figure 6.6.5-3 Schéma de mesure capteur vilebrequin 2

La tension de sortie d'un capteur à réluctance magnétique varie en fonction de la vitesse de rotation du moteur. Plus celle-ci est élevée plus la tension de sortie du capteur le sera également.

Voici l'allure de la tension provenant du capteur à réluctance magnétique :

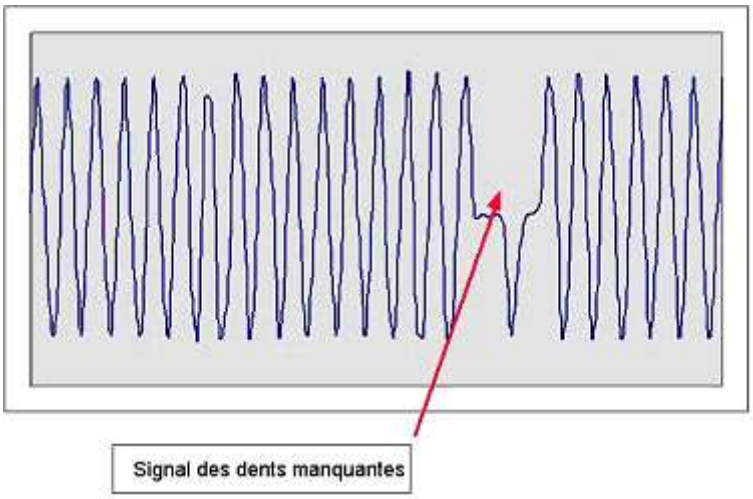


Figure 6.6.5-4 Allure signal capteur à réluctance magnétique

Détection du zéro:

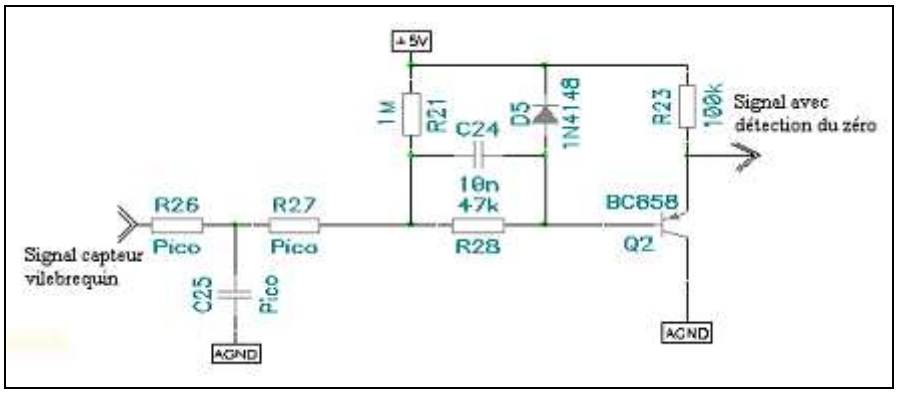


Figure 6.6.5-5 Détection du zéro capteur vilebrequin

Ce circuit permet la détection du zéro de la tension d'entrée. A l'aide de celui-ci, les signaux en dessous de 0V ne passent pas. La tension de sortie qui se trouve sur l'emetteur de Q2 est limitée à 5.7V grâce à la diode D5. En effet quand la tension tend à être plus grande que cette valeur, la diode devient conductrice. Ce qui fait que le surplus de tension se retrouve aux bornes de R28.

Les résistances d'entrées R26 et R27 sont de 10k. Elles sont placées sur Pico afin de pouvoir les remplacer en cas de changement de capteur. Le condensateur C25 n'est pour l'instant pas monté. Si par la suite le signal provenant du capteur vilebrequin est trop bruyant, le condensateur adéquat pourra être monté afin de filtrer le bruit.

La résistance R21 et le condensateur C24 sont utilisés pour qu'en sortie, le signal ne possède pas de flancs raides. L'explication sur l'utilité de telles flancs est expliquée dans la section ajustage.

Mesure:

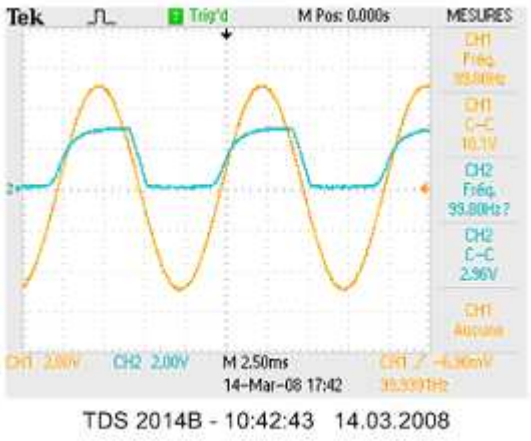


Figure 6.6.5-6 Mesure bloc détection du zéro

On peut voir sur cette mesure le signal orange (signal d'entrée) et le signal bleu (signal de sortie). On constate que ce circuit élimine bien l'alternance négative du signal d'entrée.

Ajustage du zéro et de l'hystérésis:

Si on utilise un capteur à réluctance variable, le signal n'est pas forcément symétrique. De plus, le signal provenant du bloc de détection du zéro n'atteint jamais 0V à cause de la diode du transistor Q2. C'est pourquoi il est nécessaire de régler le zéro ainsi que l'hystérèse. Il est important d'avoir en sortie un signal symétrique, ceci à cause de l'algorithme implémenté dans le uC maître.

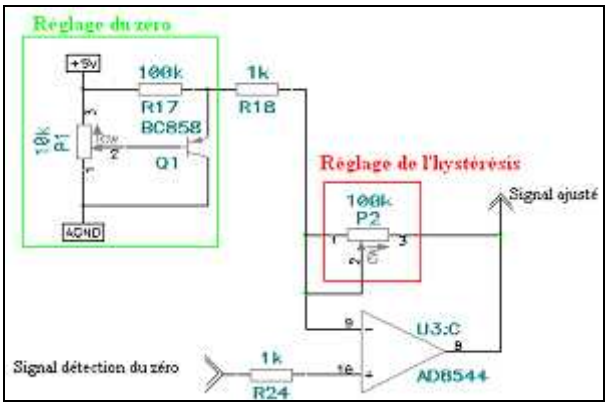


Figure 6.6.5-7 Ajustage du zéro et de l'hystérésis

Le signal provenant de la détection du zéro arrive sur R44. Le LM2904 est utilisé comme trigger de schmidt inverseur.

➢ Ajustage du zéro

Le potentiomètre P1 est utilisé pour réaliser un offset. On rajoute cet offset afin de régler le point inférieur de basculement.

➢ Réglage de l'hysérésis

Le potentiomètre P2 permet de régler le point supérieur de basculement.

Mesure:

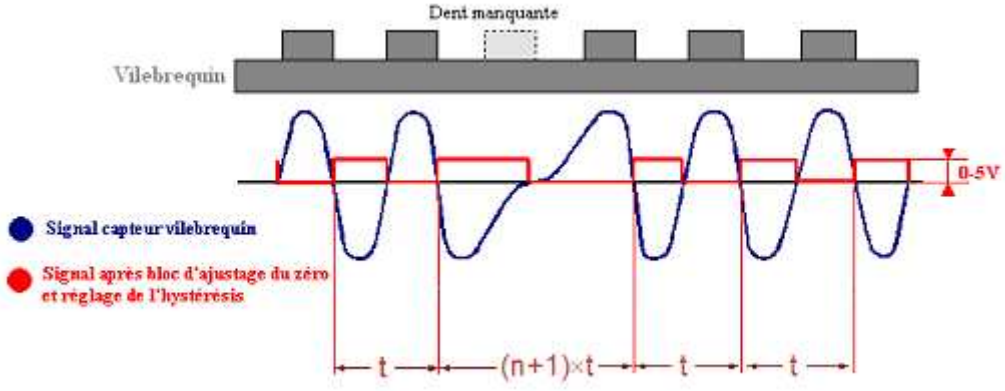


Figure 6.6.5-8 Signal en sortie du bloc d'ajustage du zéro et réglage de l'hystérésis

Voici le signal devant rentrée dans le uC maître pour que l'algorithme d'injection fonctionne correctement. On constate que le signal de sortie est parfaitement symétrique.

6.6.6 Arbre à cames

Le capteur d'arbre à cames est important afin que le microcontrôleur sache à quel moment le 1er cylindre est au point mort haut et entre en phase d'explosion. Ces capteurs sont soit à réluctance magnétique soit à effet Hall. Le circuit de traitement du signal est quasiment le même que celui s'occupant du capteur vilebrequin.

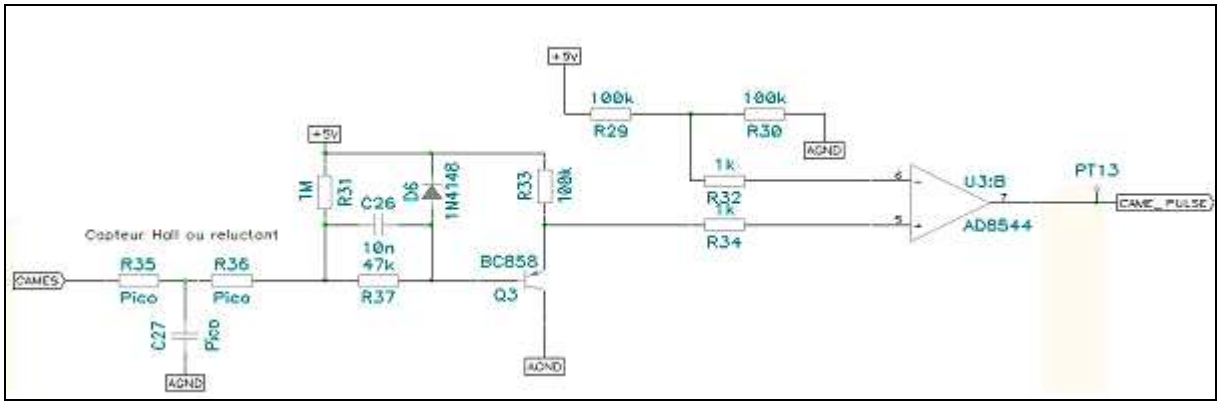


Figure 6.6.6-1 Schéma mesure capteur arbre à cames

Le circuit de détection du zéro est le même que celui du capteur vilebrequin. Le second circuit permet d'avoir un point de basculement supérieur et inférieur fixe à 2.5V.

6.6.7 Papillon des gaz

La position du papillon des gaz est mesurée par le capteur appelé TPS. Celui-ci est alimenté par la tension VREF et fourni une tension en sortie proportionnelle à la position du papillon allant de 0V à VREF.

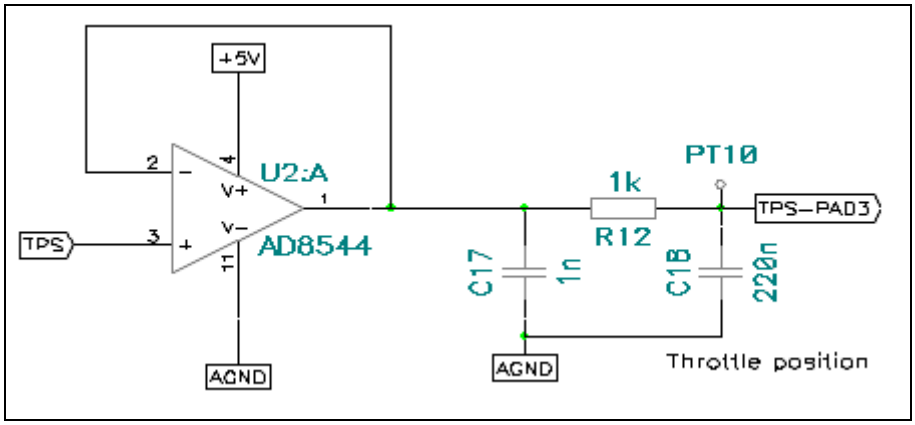


Figure 6.6.7-1 Schéma de mesure papillon des gaz

Le microcontrôleur utilise cette valeur afin de savoir si le conducteur est en accélération ou en décélaration pour pouvoir ensuite enrichir ou appauvrir le mélange air/essence.

6.7 *Alimentation*

Le bloc d'alimentation sert à adapter le niveau de tension provenant de la batterie de la voiture aux différents composants utilisés sur cette carte. Les tensions d'alimentations utilisées sont la tension de batterie ainsi que le 5V.

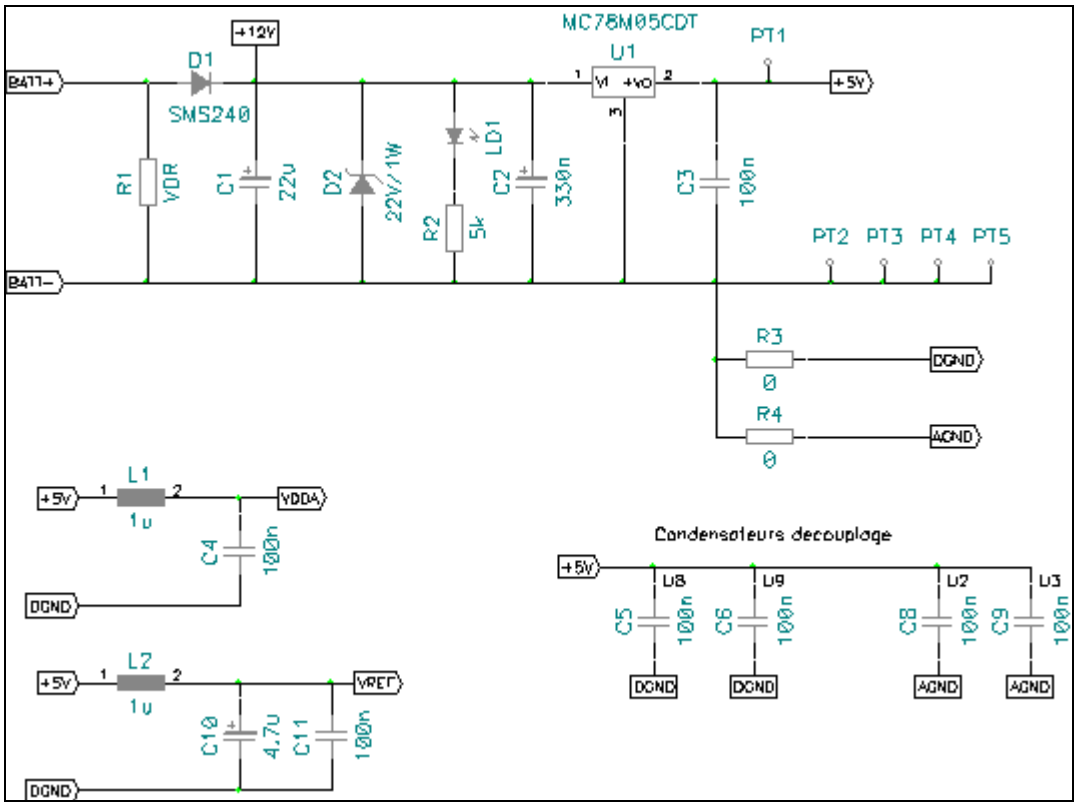


Figure 6.7-1: Schéma du bloc d'alimentation

Nous avons en entrée la tension de la batterie. La varistance R1 sert à absorber les pics de tension. Le condensateur C1 sert à compenser les demandes en courant du circuit. Pour utiliser au mieux celui-ci, il a fallu placer la diode D1 afin que le condensateur ne fournisse du courant qu'au circuit et non pas à la batterie. La diode ZENER D2 sert à limiter la tension sur le régulateur U1 à 22V au maximum.

Le 12V est ensuite converti en 5V à l'aide du LM293ET-5.0. Celui-ci permet de délivrer un maximum de 500mA. La tension VSYN(5V) est utilisée pour alimenter uniquement le microcontrôleur. La tension VREF(5V)est utilisée comme tension de référence pour les capteurs. Ces tensions sont séparées de l'alimentation générale en 5V afin qu'elles soient propres et stables. Il est à noter que les résistances R3 et R4 de 0Ω sont là uniquement pour séparer les masses sur la carte.

Refroidissement régulateur 5V :

Il est important que le régulateur 5V monté sur le PCB soit bien refroidi. Celui-ci possède une limitation de température interne de 150°C. Si cette valeur est dépassée, le régulateur interrompt la circulation du courant ce qui implique une chute de la tension 5V.

6.8 *Refroidissement composants :*

Après tests de la carte d'injection séquentielle, j'ai constaté que le refroidissement réaliser par plan de masse sur le PCB n'était pas concluant. En effet les Omnifets coupent si on dépasse 1A. C'est pourquoi j'ai décidé d'attendre la conception du PCB final qui sera introduit dans un boitier métallique. Les composants à refroidir seront alors collés sur ce même boîtier. L'utilisation d'un refroidisseur ne serait pas pratique du fait qu'il devrait être beaucoup trop grand par rapport au boîtier que je veux utiliser.

7 Code injection séquentielle

Le but de cette partie est de modifier le code d'injection semi-séquentielle étudié pendant le projet de semestre en injection séquentielle. Deux algorithmes doivent être conçus, un pour le uC maître qui commande quatre injecteurs et un autre pour le uC esclave qui commande les huit autres. Dans ce chapitre vous trouverez :

- Les notions fondamentales nécessaires à comprendre les codes implémentés
- Un descriptif de chaque fonction du uC maître avec structogramme
- Un descriptif de chaque fonction du uC esclave avec structogramme

Les codes uC maître et esclave se trouvent dans le CD-ROM (Annexe 18.6)

Les langages de programmation sont le C et l'assembleur. Le software utilisé pour le développement du code, son implantation dans les uCs ainsi que pour le debug est le CodeWarrior IDE 5.7.0 de chez Freescale. Il s'agit d'une version special edition. Celle-ci limite la compilation à des codes entre 0 et 32K bytes.

Figure 4-6.8-1: CodeWarrior Development Studio

L'interface entre le software et les uCs est un P&E USB Multilink BDM.

Figure 4-6.8-2: Interface Software-uCs

7.1 *Notions fondamentales*

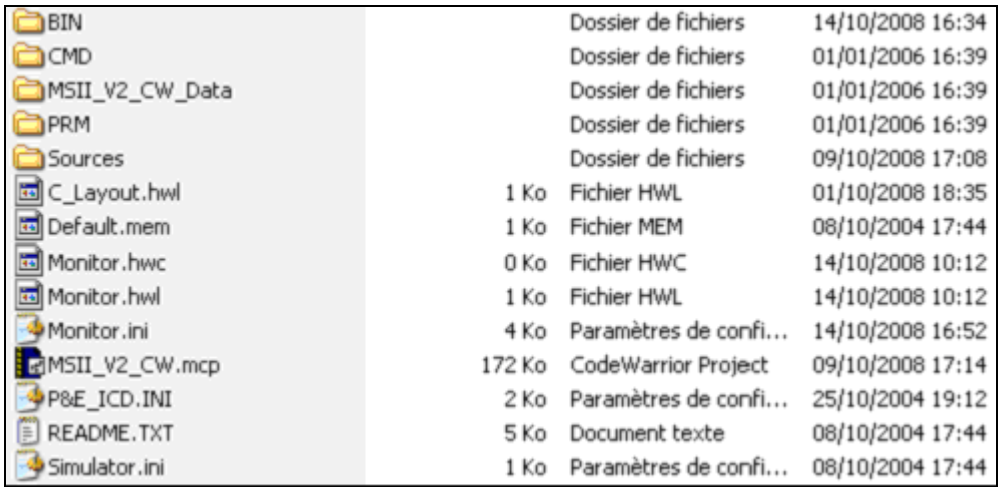
Afin de pouvoir reprendre plus facilement ce projet, voici quelques notions fondamentales à connaître sur l'environnement de développement ainsi que sur les codes.

7.1.1 CodeWarrior

Afin de reprendre les codes d'injection, il vous faut copier le dossier entier Codes. A l'intérieur de celui-ci se trouvent les dossiers **MSIIC_Proj_uC_maître** et **MSIIC_Proj_uC_esclave**. Ceux-ci contiennent respectivement les projets, codes pour le uC maître et uC esclave. Il est important de reprendre ces projets tels quels afin de garder les différents réglages effectués sur le compilateur et autres.

Exemple pour uC maître :

Voici ce que contient le dossier MSIIC_Proj_uC maître :



BIN		Dossier de fichiers	14/10/2008 16:34
CMD		Dossier de fichiers	01/01/2006 16:39
MSII_V2_CW_Data		Dossier de fichiers	01/01/2006 16:39
PRM		Dossier de fichiers	01/01/2006 16:39
Sources		Dossier de fichiers	09/10/2008 17:08
C_Layout.hwl	1 Ko	Fichier HWL	01/10/2008 18:35
Default.mem	1 Ko	Fichier MEM	08/10/2004 17:44
Monitor.hwc	0 Ko	Fichier HWC	14/10/2008 10:12
Monitor.hwl	1 Ko	Fichier HWL	14/10/2008 10:12
Monitor.ini	4 Ko	Paramètres de confi...	14/10/2008 16:52
MSII_V2_CW.mcp	172 Ko	CodeWarrior Project	09/10/2008 17:14
P&E_ICD.INI	2 Ko	Paramètres de confi...	25/10/2004 19:12
README.TXT	5 Ko	Document texte	08/10/2004 17:44
Simulator.ini	1 Ko	Paramètres de confi...	08/10/2004 17:44

Figure 7.1-1 Dossier MSIIC_Proj_uC maître

Le fichier projet à ouvrir est le**.mcp**. En l'ouvrant, l'environnement de développement sera directement lancé avec tous les réglages compilateurs et autres corrects. Le dossier source contient les différents fichiers écrits en code C et ASM inclus dans le projet.

7.1.2 Tach Pulse

La TACH pulse est une pulse indiquant le moment précis où un des pistons se trouve au point mort haut en phase d'explosion. C'est à partir de ces pulses que l'algorithme calcul l'avance à l'allumage, le moment d'injection etc. Si la mesure du capteur vilebrequin est faite sur la borne négative de la bobine, chaque pulse reçue est une TACH pulse. Par contre si on mesure à l'aide d'un capteur réluctant ou effet Hall sur une roue dentée, la pulse reçue n'est pas forcément une TACH pulse. Voici un exemple permettant de visualiser ceci :

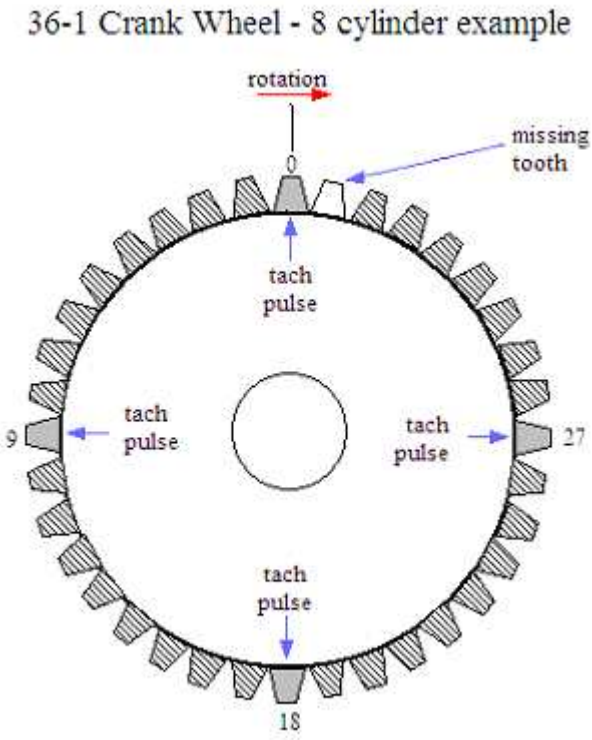


Figure 7.1-2: Exemple de TACH pulse

Ceci est un exemple de roue dentée avec 36 dents dont une manquante. On constate que pour un moteur de 8 cylindres, nous avons 4 TACH pulses par tour.

Delay Teeth: Pour des raisons de bruit, les dents manquantes ne sont pas toujours alignées exactement sur le point mort haut du cylindre 1. Delay Teeth est en fait le nombre de dents de décalage entre les dents manquantes et le capteur vilebrequin au point mort haut.

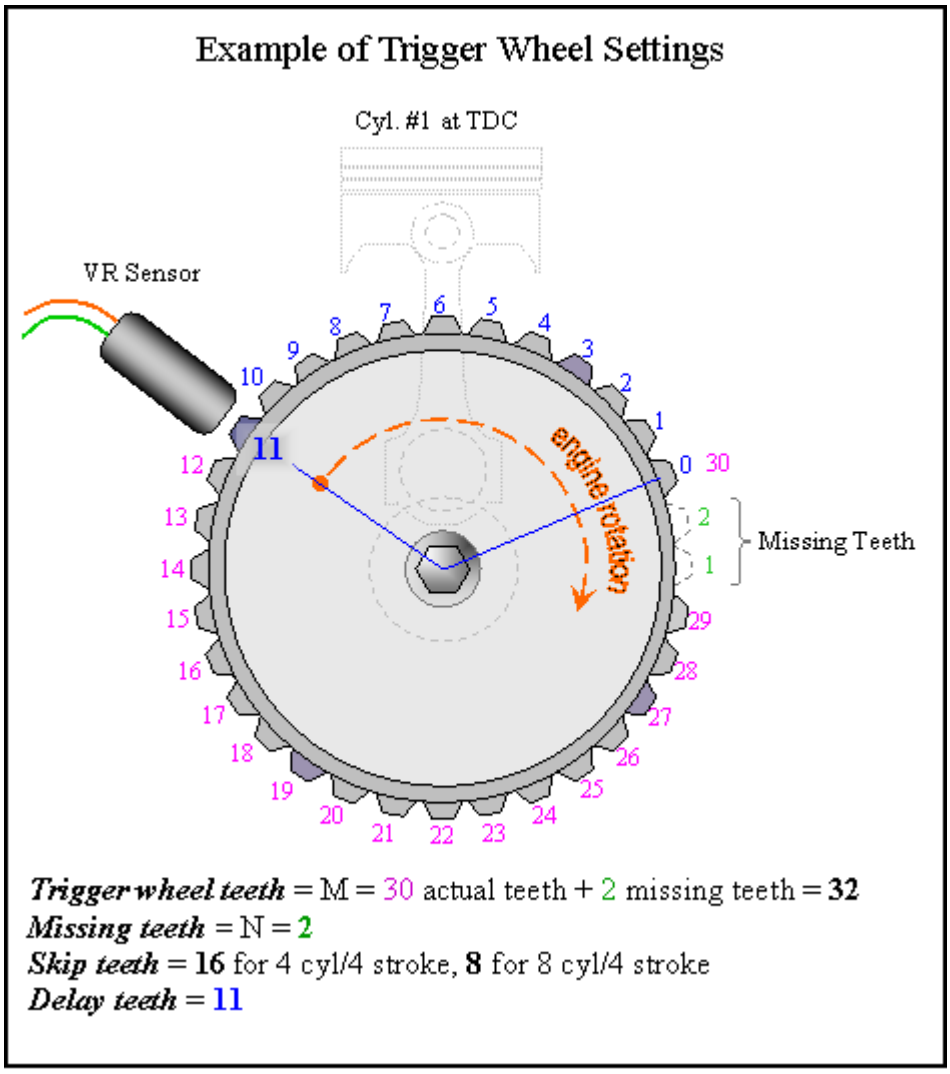


Figure 7.1-3: Trigger Wheel Settings

7.1.3 Injection

Avant de commencer à étudier le code d'injection séquentielle, il est important de savoir à quel moment quel cylindre doit injecter.

En annexe 18.3 se trouve un diagramme permettant de visualiser les injections en fonction de la valeur des variables utilisées par le code.

7.1.4 Overflow

La notion d'overflow signifie dépassement. En effet les compteurs sont de 16bits. Ceux-ci permettent de compter jusqu'à 65535. Passé cette valeur ils recommencent à zéro.

Exemple :

Compter jusqu'à 60000 = 0xEA60

Variable 16 bits => Var = 0xEA60 : Pas de dépassement

Compter jusqu'à 70000 = 0x11170

Variable 16 bits => Elle va uniquement enregistrer 0x1170 => le compteur doit faire un tour avant d'arriver à cette valeur => dépassement de un tour.

Compter jusqu'à 200'000 = 0x30D40

Variable 16 bits => Elle va uniquement enregistrer 0x0D40 => le compteur doit faire trois tours avant d'arriver à cette valeur => dépassement de trois tour.

Comment le uC fait il pour savoir ce nombre de tours ? Dans le code, il existe une interruption s'enclenchant à chaque fois qu'un compteur fait un tour. A chaque fois que le code entre dans cette interruption, il va décrémenter une variable possédant le nombre de tour de dépassement des compteur. Et quand cette variable égale à zéro, ceci veut dire que le compteur à fait les tours voulu. C'est à ce moment la qu'on peut recharger le compteur avec la variable de 16bits.

7.2 *Code uC maître*

Dans ce chapitre, je vais vous expliquer en quelques lignes chaque fonction du uC maître étant liée à l'injection d'essence. **Les structogrammes détaillés se trouvent en annexe 18.4.** Vous pourrez également trouver un grand nombre d'explications dans le code sous forme de commentaires.

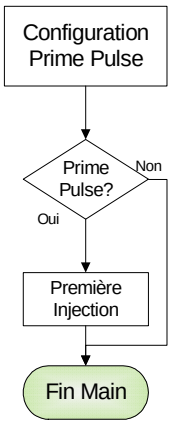
Le code passe en premier dans la routine Startup puis dans le Main. Pour finir, celui-ci va tourner en boucle dans la routine Main Loop. Le code possède 7 interruptions implémentées :

- **ISR_Injx_TimerOut**: Enclenchement déclenchement injecteurs
- **ISR_Timer_Clock**: Base de temps
- **ISR_Ign_TimerIn**: Réception d'une pulse du capteur vilebrequin ou arbre à cames
- **ISR_Ign_TimerOut**: Déclenchement de la bobine d'allumage
- **ISR_TimerOverflow**: Dépassement d'un compteur
- **ISR_SCI_Comm** : Communication RS232
- **CanTxIsr** : Transmission trame CAN
- **CanRxIsr** : Réception trame CAN

A chaque fois qu'un flag lié à une de ces interruptions passe à High, le code va sortir de la Main Loop, exécuter l'interruption puis revenir dans la Main Loop.

7.2.1 Main

Dans cette routine on trouve la configuration des modules du MC9S12C96 et de ces ports. On y trouve également la configuration de certaines variables ainsi que l'initialisation de celles-ci. A la fin de la routine, on peut voir une section appelée Prime Pulse :



Qu'est-ce que la Prime Pulse ? Il s'agit en fait de la première injection. Celle-ci est réalisée directement après avoir mis le contact. Elle sert à pouvoir démarrer plus facilement le moteur. En effet si cette opération n'est pas effectuée, il faut attendre que le code se synchronise avec le capteur vilebrequin avant d'avoir une injection. Ce qui peut durer environ 2s sachant que le moteur est entraîné à 50tr/min par le démarreur. La largeur de cette pulse dépend de la température du moteur ainsi que de la pression atmosphérique. Celle-ci est calculée dans « Configuration Prime Pulse ».

Figure 7.2-1 Configuration Prime Pulse

7.2.2 Main loop

Le programme va tourner en boucle dans la fonction Main loop. Dans cette routine se trouvent tous les calculs afin de déterminer le temps précis d'injection.

7.2.3 Ign_reset

La fonction Ign_reset est utilisée à l'initialisation ainsi qu'au moment où le code n'est plus synchronisé avec l'allumage. L'allumage non synchronisé veut dire que le processeur ne sait pas ou plus où le moteur se trouve. Cette fonction réinitialise les variables d'injection et d'allumage puis reconfigure les Timers.

7.2.4 ISR_Timer_Clock

Cette fonction est une interruption. Celle-ci est appelée chaque 0.128ms. Elle va générer les différentes variables de temps utilisées dans tous le code.

7.2.5 ISR_Ign_TimerOut

Le code va entrer dans cette interruption quand le flag OC5 passe High. Ce flag indique que l'allumage doit se terminer. Cette interruption ne contient aucune configuration pour l'injection d'essence.

7.2.6 Unimplemented ISR

Cette routine est uniquement là au cas ou un flag d'interruption non-configurée passerait involontairement High. Elle permet au uC de ne pas se perdre suite à une action de ce genre. Aucune ligne de code n'est implémentée dans cette fonction.

7.2.7 ISR_SCI_Comm

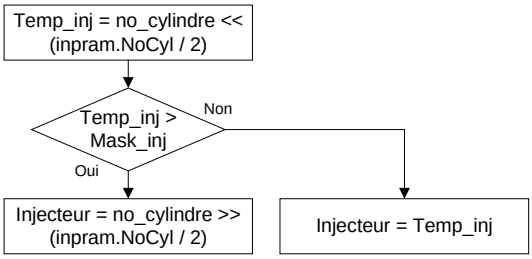
Cette fonction est une interruption utilisée pour la communication RS232 avec le PC. Cette interruption ne contient aucune configuration pour l'injection d'essence.

7.2.8 ISR_Ign_TimerIn

Le code entre dans cette interruption quand les flags IC0 ou IC1 passent à High. C'est-à-dire qu'une pulse provient du capteur vilebrequin ou du capteur d'arbre à cames. Dans cette routine, on trouve la synchronisation avec le capteur vilebrequin, le calcul des variables de temps concernant l'allumage et l'injection, le paramétrage de l'allumage ainsi que la configuration de l'injection d'essence.

Configuration de l'injection séquentielle :

Pour utiliser l'injection séquentielle, il est nécessaire de réaliser des calculs permettant de trouver dans quel cylindre il faut injecter :



Il faut injecter 360° plus loin par rapport au point d'allumage. Voici la formule permettant de calculer ceci :

$$Temp\_inj = no\_cylindre << \frac{Nombre\_cylindre}{2}$$

Le signe "<<" permet de décaler les bits de no_cylindre vers la gauche.

Exemple pour un 4 cylindre :

No de cylindre entrant en explosion	variable no_cylindre	Temp_inj	Injecteur
1	**0b0000'0001**	**0b0000'0100**	**0b0000'0100**
2	**0b0000'0010**	**0b0000'1000**	**0b0000'1000**
3	**0b0000'0100**	**0b0001'0000**	**0b0000'0001**
4	**0b0000'1000**	**0b0010'0000**	**0b0000'0010**

Avance/Retard à l'injection:

Ce code permet également de régler l'avance/retard à l'injection. L'utilisateur rentre une valeur comprise entre –360° et 360° dans Megatune. Cette valeur correspond pour le signe négatif à de l'avance et inversement pour le retard. Megatune va envoyer cette information à notre code en lui rajoutant un offset de 360. Ceci permet de ne pas avoir à utiliser de signe négatif dans le code.
Celui-ci doit tout d'abord calculer le temps que met le moteur pour effectuer un tour. Ce calcul est primordiale afin de pouvoir convertir l'avance/retard en une valeur de temps.

$$TimeOneRotation = \frac{1'000'000}{\frac{RPM}{60}}$$

TimeOneRotation : Temps pour effectuer un tour en [us]
1'000'000 : Facteur pour obtenir une valeur en us
RPM : Variable indiquant la vitesse de rotation [tr/min]
60 : Facteur pour normaliser RPM en [tr/s]

Angle_retard = inpram.Injection_degree – 360°
Angle_avance = 360° - inpram.Injection_degree

Retard :

Le retard à l'injection signifie que la pulse d'injection va se décaler vers la droite par rapport à la Tach Pulse (image ci-dessous). Le temps de retard est donné par la variable TimeInjWait. Dans un quatre cylindres, 180° de retard signifie que nous avons un cylindre de retard(Inj_missed).

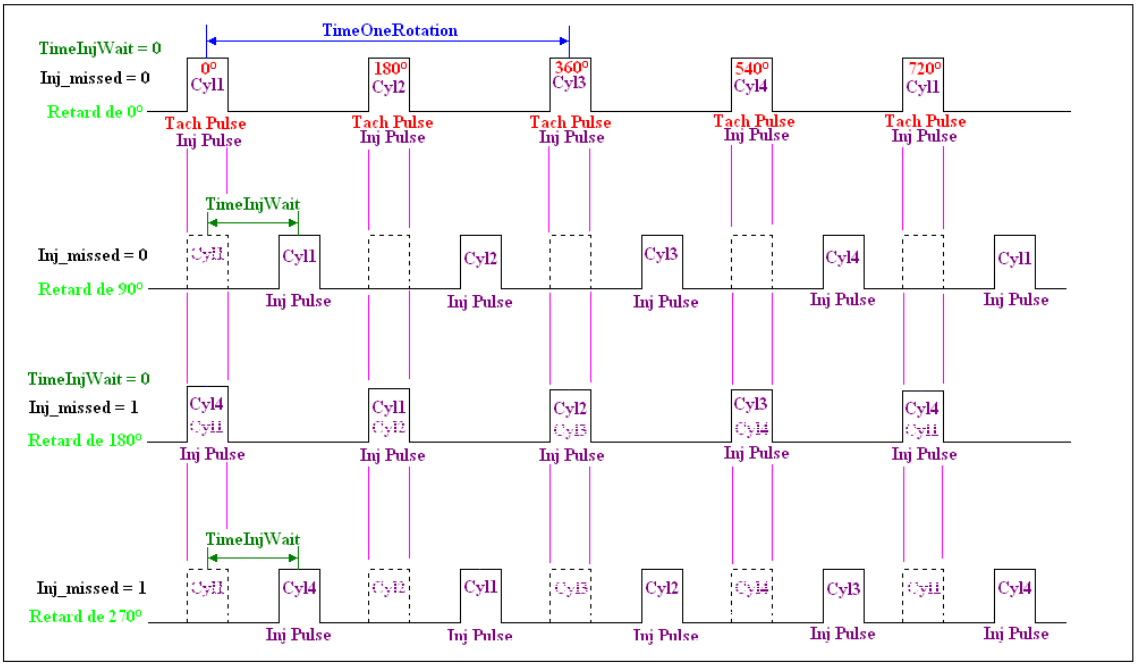


Figure 7.2-2 Croquis retard à l'injection

Exemple calcul pour 4 cylindres, 270° de retard et 10ms/tour :

Calcul de Inj_missed : $$Inj\_missed = \frac{Angle\_retard}{\frac{720}{Nb\_cylindre}} = \frac{Angle\_retard}{\frac{720}{4}} = \frac{Angle\_retard}{180°}$$

=> Inj_missed = 270°/180° = 1,….=> 1

Degré d'attente après Tach_Pulse: $$Degré\_attente = Angle\_retard - \frac{720}{Nb\_cylindre} * Inj\_missed$$

=> Degré_attente = 270°-180°*1 = 90°

Calcul de TimeInjWait : $$TimeInjWait = TimeOneRotation * \frac{degré\_attente}{360°}$$

TimeInjWait = 10ms*(90°/360°)=2.5ms

Nous avons au final un décalage de 1 cylindre vers la droite ainsi qu'une attente de 2.5ms.

Avance :

L'avance à l'injection signifie que la pulse d'injection va se décaler vers la gauche par rapport à la Tach Pulse (image ci-dessous). Le temps d'avance est donné par la variable TimeInjWait. Dans un quatre cylindres, 180° d'avance signifie que nous avons un cylindre d'avance (Inj_missed).

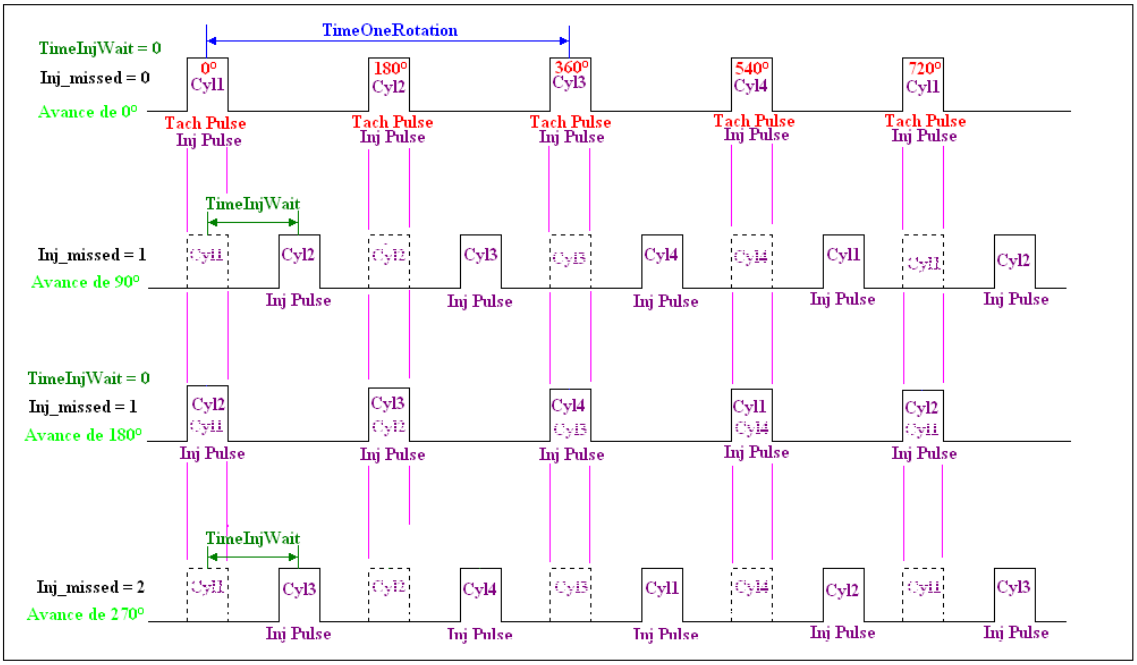


Figure 7.2-3 Croquis avance à l'injection

Exemple calcul pour 4 cylindres, 270° d'avance et 10ms/tour :

Calcul de Inj_missed : $$Inj\_missed = 1 + \frac{Angle\_avance}{\frac{720}{Nb\_cylindre}} = 1 + \frac{Angle\_avance}{\frac{720}{4}} = 1 + \frac{Angle\_avance}{180°}$$

=> Inj_missed = 1+270°/180° = 2.5=> 2

Degré d'attente après Tach_Pulse: $$Degré\_attente = \frac{720}{Nb\_cylindre} * Inj\_missed - Angle\_avance$$

=> Degré_attente = 180°*2-240 = 90°

Calcul de TimeInjWait : $$TimeInjWait = TimeOneRotation * \frac{\deg ré\_attente}{360°}$$

TimeInjWait = 10ms*(90°/360°)=2.5ms

Nous avons au final un décalage de 2 cylindre vers la gauche ainsi qu'une attente de 2.5ms.

Configuration des interruptions pour enclenchement injecteurs :

Après avoir calculé quel injecteur il faut enclencher ainsi que son moment d'enclenchement, le code va maintenant paramétrer l'interruption qui va enclencher les injecteurs en temps voulu. Le code va tout d'abord tester la variable injecteur pour savoir s'il faut injecter dans le cylindre No x. Si c'est le cas, il va configurer l'enclenchement de l'injecteur x. Si ce n'est pas le cas, il réaliser les mêmes tests pour les injecteurs y,z et w. Le reste des injecteur est contrôlé par le uC esclave.

InjecteurX :

On va tout d'abord tester si l'injecteur est encore enclenché. Si c'est le cas, le code va uniquement mettre le flag_depassementx à 1 et le flag_injecteurx à 0 puis mémoriser l'état du compteur. Ceci afin de ne pas couper l'injection en cours. L'état du compteur est mémorisé afin de pouvoir calculer dans l'interruption le temps d'avance/retard restant avant le prochain enclenchement.

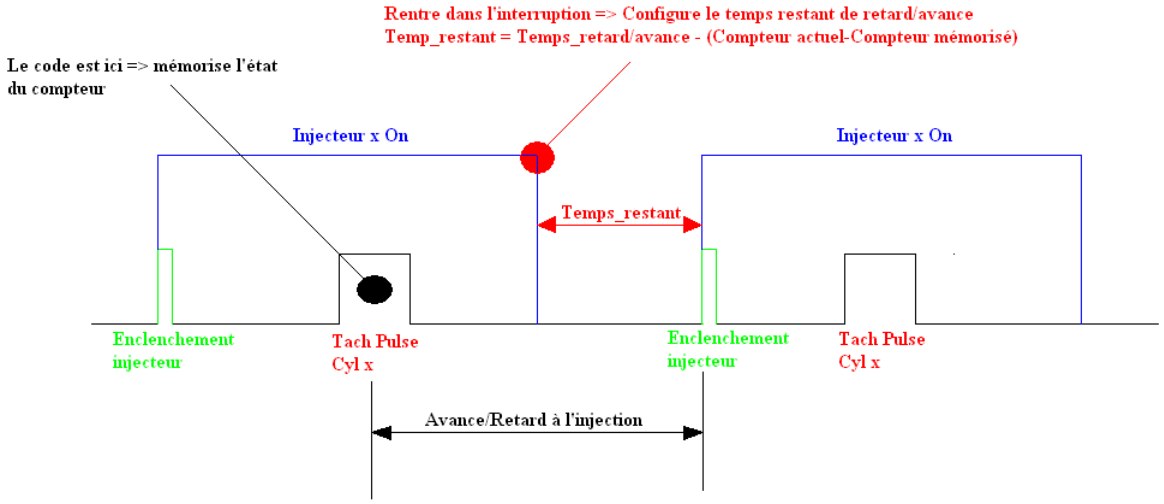


Figure 7.2-4 Croquis enclenchement injecteur

Si l'injecteur n'est pas enclenché alors le code va mettre le flag_depassementX à 0 et le flag_injecteurX à 1. Il va ensuite configurer le compteur afin que l'injecteur s'enclenche tout de suite après que le temps de retard/avance soit passé.

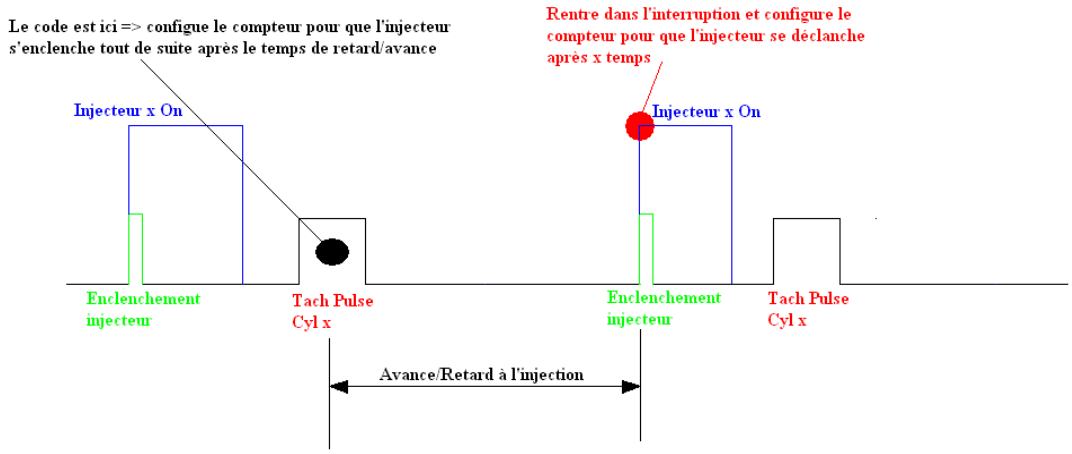


Figure 7.2-5 Croquis 2 enclenchement injecteur

7.2.9 ISR_TimerOverflow

Le code rentre dans cette interruption à chaque fois que le compteur TCNT fait un tour. C'est-à-dire environ chaque 43.7ms.

$$TCNT = 0à65535$$

$$Temps = TCNT * \frac{1}{ClockTCNT} = 65535 * \frac{1}{1.5MHz} = 43.7ms$$

Dans cette interruption, le code va tester si un compteur d'injection doit faire plusieurs tours. Si c'est le cas, il va décrémenter chaque tour. Quand le compteur d'injection a fait le nombre de tours voulu, alors l'interruption d'injection pourra être activée.

7.2.10 ISR_Injx_TimerOut

Le code va entrer dans cette interruption à chaque enclenchement ou déclenchement de l'injecteur X. En entrant dans l'interruption, l'injecteur sera automatiquement enclenché ou déclenché suivant la configuration des registres TCTL1/TCTL2. Tout d'abord, le code va tester s'il faut configurer l'injecteur pour l'enclencher, le déclencher ou ne rien faire.

- Configurer le déclenchement

Quand le programme entre dans cette portion de code, c'est que l'injecteur est enclenché et qu'il faut configurer le compteur afin de déclenché celui-ci. Il faut tout d'abord mettre le flag_injecteurX à 0. Ceci afin de faire savoir que l'injecteur n'a plus besoin d'être activé vu qu'il l'est déjà. Puis le code va charger le compteur avec la valeur du temps d'injection calculée dans la Main Loop.

- Configurer l'enclenchement

Si le programme entre dans cette portion de code, c'est que le flag de dépassement est à 1. C'est-à-dire qu'au moment de la Tach pulse, l'injecteur était toujours ON. Le code va alors mettre flag_depassementX à 0 puis activer le flag_injecteurX. Il va ensuite déterminer le temps qu'il reste avant le début de la prochaine injection. Si ce temps est plus petit que 0, c'est que l'injecteur ne doit jamais se désactiver(Pleine charge). Le code va alors uniquement configurer le compteur pour le déclenchement futur. Par contre, si le temps n'est pas plus petit que 0, alors le code va charger le compteur avec le temps qu'il reste avant l'enclenchement.

- Ne rien faire

Si le code entre dans cette portion de code, ceci veut dire que l'injecteur est déjà OFF. Alors celui-ci va désactiver le flag_injecteurX ainsi que l'interruption qui ne pourra être activée qu'après le passage de la Tach Pulse.

Vous trouverez ci-dessous trois figures permettant de visualiser ces différents cas.

Pas de dépassement :

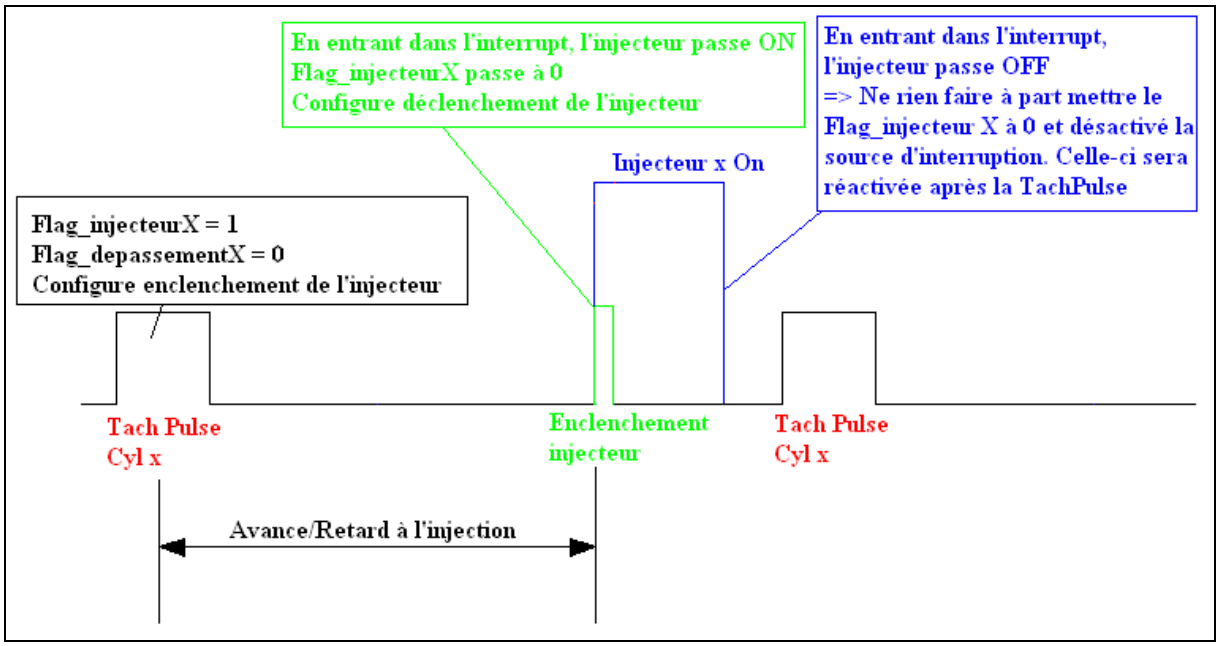


Figure 7.2-6 Croquis enclenchement injecteur sans dépassement

Dépassement :

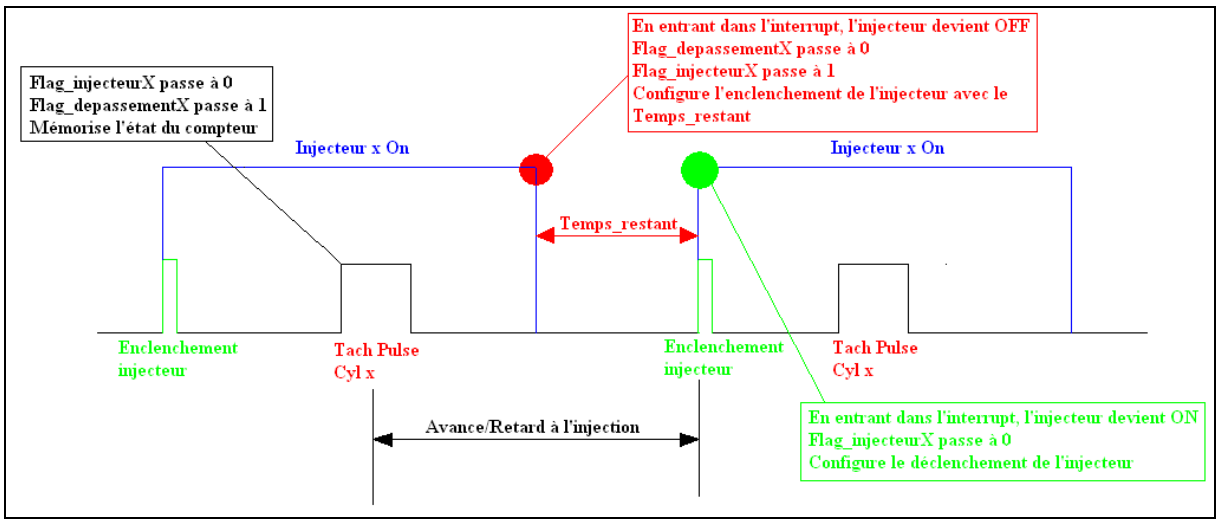


Figure 7.2-7 Croquis enclenchement injecteur avec dépassement

Injecteur toujours enclenché :

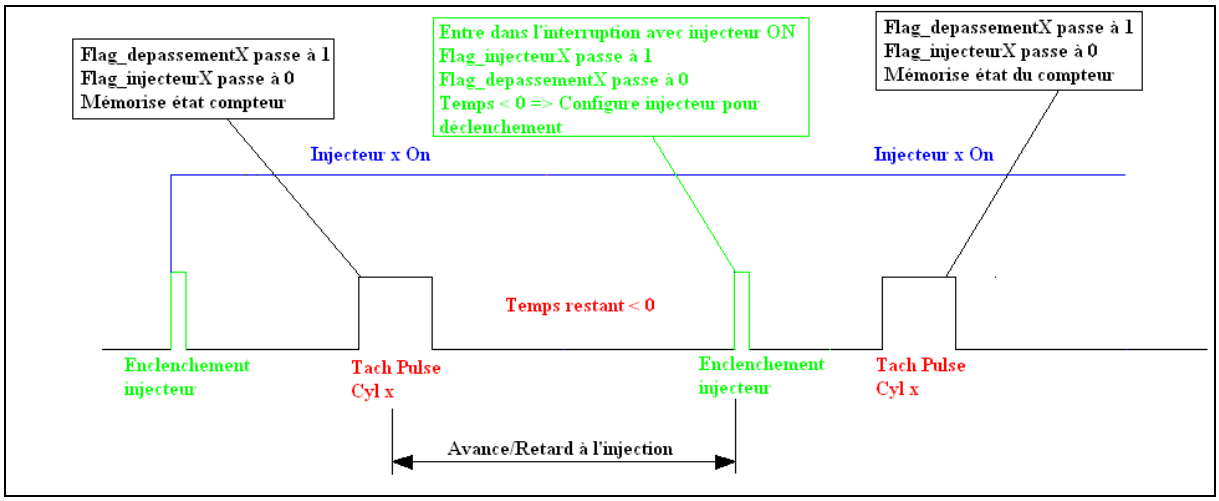


Figure 7.2-8 Croquis injecteur toujours enclenché

7.2.11 CanTxIsr

Le code va entrer dans cette interruption à chaque fois que les injecteurs 5,6,7,8,9,10,11 ou 12 doivent être enclenchés. Les informations envoyés sont :

- Les injecteurs à enclencher
- Le temps d'injection
- Le temps avant de commencer à injecter

7.3 *Fonctions uC esclave*

Les fonctions utilisées pour le uC esclave sont les même que celles utilisées pour le maître à une exception prêt. Dans le maître nous configurions les injecteurs dans l'interruption ISR_Ign_TimerIn. Dans l'esclave, nous n'utilisons pas cette interruption. Les injecteur sont configurés dans la routine CanRxISR.

8 Softwares PC (Megatune & Configurator)

Megatune2.25 est un programme permettant de visualiser/modifier les paramètre d'injection/allumage etc. se trouvant dans le uC maître de la carte d'injection séquentielle. Le programme envoie et récolte les données du uC maître via RS232.

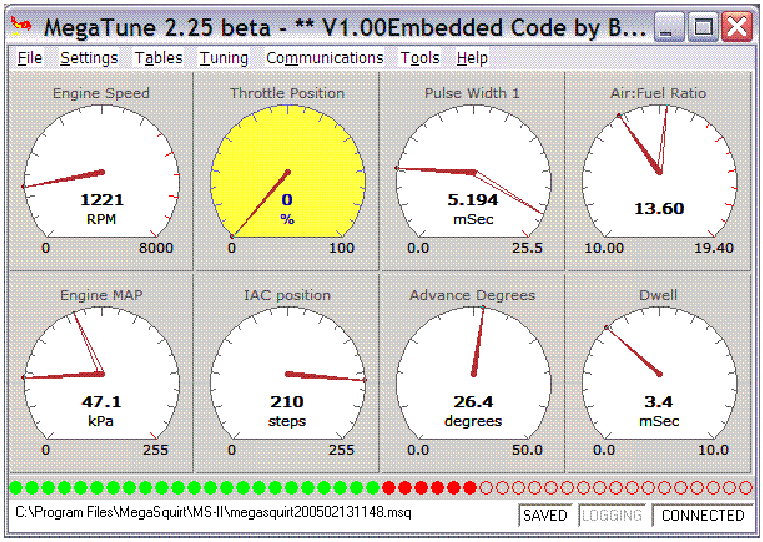


Figure 7.3-1 Ecran de garde Megatune2.25

Megatune Configurator permet de configurer le logiciel Megatune2.25 en fonction du code implémenté dans le uC maître.

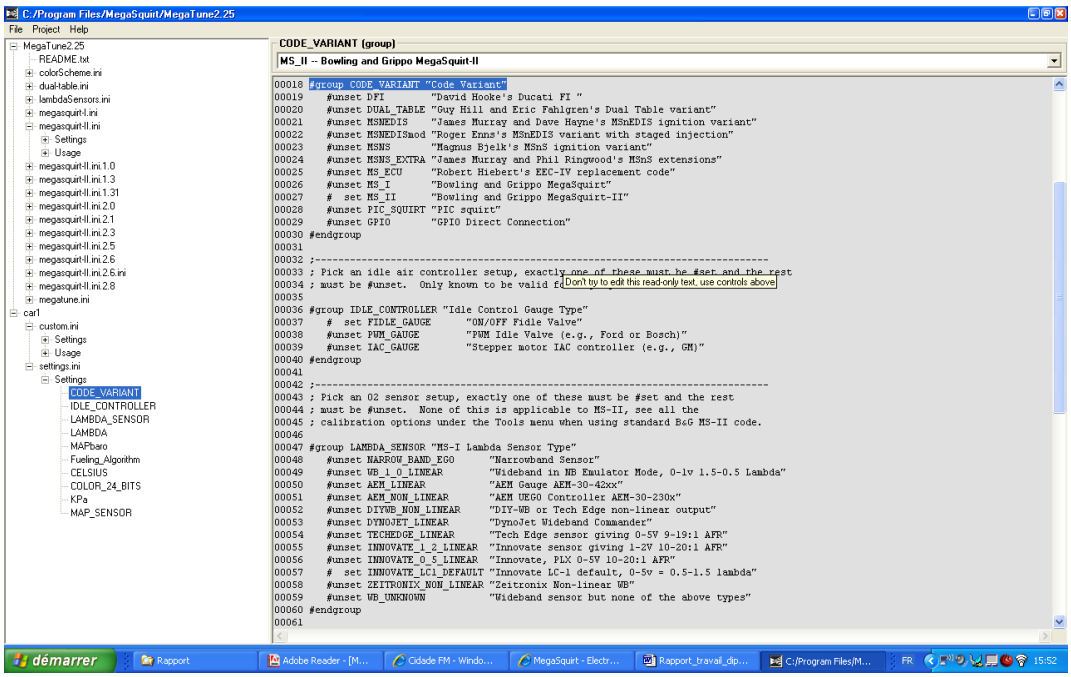


Figure 7.3-2 Ecran de garde Configurator

8.1 *Mode d'emploi Megatune2.25 & Configurator*

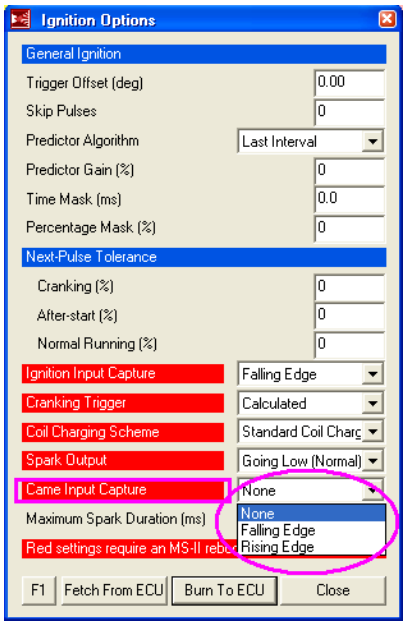
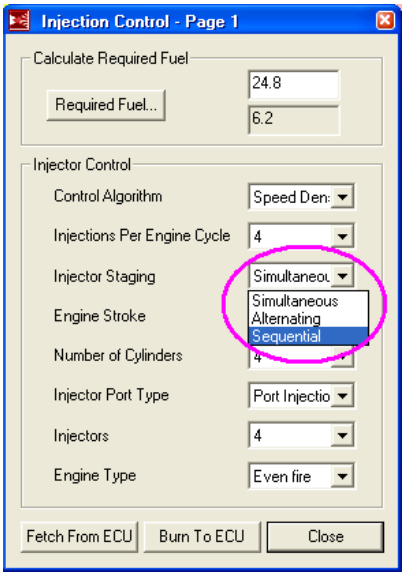
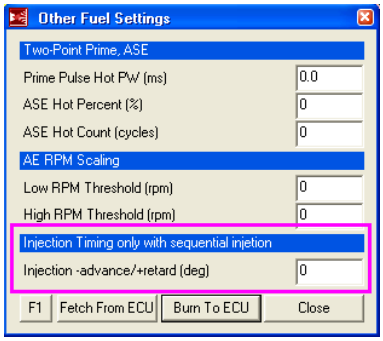
Le programme Megatune peut communiquer avec plusieurs versions de code implémentées dans le uC maître. Chaque versions possède ses paramètres d'injection et d'allumage. Certaines version contiennent par exemple un tel menu qu'une autre n'aura pas. C'est pour cette raison que le programme utilise des fichier de configuration .INI permettant de régler le programme en fonction de la version. C'est fichiers ont également été prévu afin que chaque utilisateur puisse y apporter sa touche personnelle. Le logiciel Configurator permet de modifier ces fichier INI tout simplement en cochant des options. Ce qui évite à tout utilisateur novice de devoir étudier tout le fichier INI pour modifier un paramètre.

Le mode d'emploi des deux logiciels se trouve dans le CD-ROM (Annexe 18.6).

9 Modifications Megatune2.25

Comme dit plus haut, le logiciel Megatune2.25 utilise des fichiers INI contenant les paramètres en fonction des versions de code implémentées dans le uC maître. Le code d'injection séquentielle a été implémenté dans la version 2.6 reprise depuis Internet.

Dans ce code, j'ai rajouté l'éventuelle utilisation d'un capteur d'arbre à cames, l'injection séquentielle ainsi que l'avance/retard à l'injection. Afin de pouvoir configurer ses paramètres, j'ai du rajouter deux menus plus modifier l'algorithme de calcul d'injection dans Megatune2.25. Ces modifications ont été apportées dans le fichier **MegasquirtII2.6.INI.**

Menu pour l'utilisation du capteur vilebrequin	Menu pour utilisation injection séquentielle	Menu pour l'utilisation du retard/avance à l'injection
		

Au cours de ce travail de diplôme, j'ai également du rajouter l'option de configuration automatique des tables d'injection. En fait, j'ai constaté que ce software permettais déjà ceci après modifications de quelques ligne du fichier **custom.INI**

La structure du fichier MegasquirtII2.6.INI est commentée ci-dessous :

1.[Constants] : Dans cette partie, on retrouve l'image mémoire des variables se trouvant dans le uC maître. On y trouve les types inpram ainsi que inpram2.

2.[Menu]: Cette partie contient la déclaration des menus que Megatune possède.

3.[UserDefined]: On y trouve la déclaration des sous-menus pour la configuration de valeurs.

4.[CurveEditor]: Section traitant la création des courbes.

5.[TableEditor]: Section traitant la création des tables.

6.[GaugesConfiguration]: Section traitant de la configuration des gauges de visualisation.

7.[FrontPage]: Partie traitant de la configuration de la page d'accueil de Megatune2.25.

8.[Runtime]: Cette partie contient la configuration du menu « Realtime display ».

9.[Tuning]: Cette partie contient la configuration des pages Tuning Table XXX.

10.[AccelerationWizzard]: Cette section contient la configuration du menu TriggerWizzard.

11.[BurstMode]: On y trouve la configuration du menu Burst Mode.

12.[OutputChannels] : Cette partie contient la déclaration des variables ne se trouvant pas dans le code du uC maître.

13.[DataLog] : Configure les variables que l'on veut retrouver dans le fichier datalog Excel généré.

Tout le programme n'est pas configuré dans le fichier .INI. On y trouve uniquement certaines configurations spécialement celles d'affichages et autres.

9.1 *Menu pour l'utilisation du capteur d'arbre à cames*

Pour pouvoir utiliser convenablement le capteur d'arbre à cames, il faut savoir quel type de flanc utiliser. C'est pourquoi j'ai décidé de réaliser un menu permettant le choix de trois configurations :

1. **None** : Pas d'utilisation de capteur d'arbre à came

Nous devons utiliser cette option quand l'injection séquentielle n'est pas utilisée. Cette option désactive alors dans le uC maître la source d'interruption du capteur d'arbre à cames.

2. **Rising Edge** : Détection du flanc montant du signal

Cette option est à utiliser quand le capteur d'arbre à cames fourni en entrée du uC maître un signal de ce type :

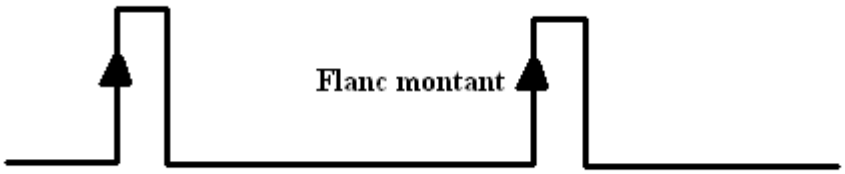


Figure 9.1-1 Signal capteur arbre à came flanc montant

Cette option va configurer le uC maître afin d'utiliser les flancs montants du signal.

3. **Falling Edge**: Détection du flanc descendant du signal

Cette option est à utiliser quand le capteur d'arbre à cames fourni en entrée du uC maître un signal de ce type :

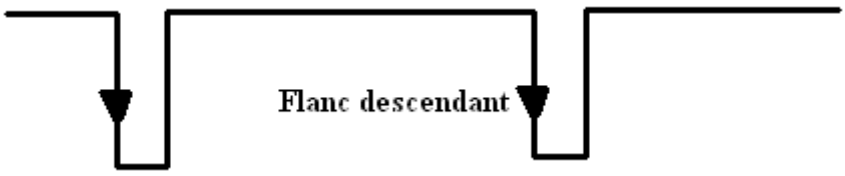


Figure 9.1-2 Signal capteur arbre à came flanc descendant

Cette option va configurer le uC maître afin d'utiliser les flancs descendants du signal.

Modification du Fichier .INI :

Pour réaliser ce menu, j'ai du rajouter dans la section **[Constants]** la variables utilisée dans le code uC maître. Il faut la placer exactement comme dans le code. C'est-à-dire après la variable injTestOffTime.

```
injTestSqrts     = scalar,  U16,    982,            "squirts",  1,         0,          0,    65000,      0 ;   (  2 bytes)
injTestPW        = scalar,  U16,    984,            "us",       1,         0,          0,    65000,      0 ;   (  2 bytes)
injTestOffTime   = scalar,  U16,    986,            "ms",       0.1,       0,          0,     6500,      1 ;   (  2 bytes)

ICCamCapture     = bits   ,  U08,    988,      [0:1], "None", "Falling Edge", "Rising Edge", "INVALID"       ; * (  1 byte )
Injection_delay  = scalar,  S16,    989,            "deg",      1.0,           -360,  -360,      360,      0 ; * (  2 bytes)
pageSize =  991
```

La variable s'appelle **ICCamCapture**. **Bits** signifie que la variable est utilisé bit à bit.

=> Le bit 0 est High pour None, le bit 1 est High pour Falling Edge etc.

La variable est de type U08 => unsigned short. Elle est enregistrée à la position 988.
[0 :1] signifie que nous allons uniquement utilisé le byte jusqu'au bit 1.

Bit 1;0	Configuration	
00	None	
01	Falling Edge	
10	Rising Edge	
11	INVALID	=> Non utilisé

*(1 byte) est uniquement un commentaire montrant la taille de la variable. Ne pas oublier d'incrémenter la variable **pageSize**.

Il faut maintenant aller dans la partie **[UserDefined]** afin de réaliser le menu. On veut ajouter ce sous-menu dans le menu Ignition options. C'est pourquoi on rajoute la ligne ci-dessous (Encadré rouge). **'' !Came Input Capture''** est le nom du sous-menu. Le symbole''!'' signifie que nous voulons que le titre apparaisse avec un fond rouge. Puis il suffit de rajoutant le nom de la variable gérant le sous-menu. Il s'agit de **ICCamCapture**. Le dernier '' !'' signifie que le sous-menu est de type déroulant.

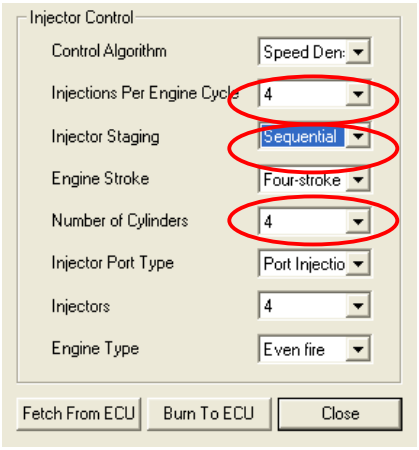
```
dialog = ignitionOptions, "Ignition Options"
   topicHelp = "http://www.megasquirt.info/megatune.htm#si"
   field = "#General Ignition"
   field = "Trigger Offset",                triggerOffset
   field = "Skip Pulses",                   no_skip_pulses
   field = "Predictor Algorithm",           PredOpt
   field = "Predictor Gain",                Dtpred_Gain
   field = "Time Mask",                     ICISR_tmask
   field = "Percentage Mask",               ICISR_pmask
   field = "#Next-Pulse Tolerance"
   field = "    Cranking",                  crankTolerance
   field = "    After-start",               asTolerance
   field = "    Normal Running",            pulseTolerance
   field = "!Ignition Input Capture",       ICIgnCapture,   { ICIgnOption != 2 && ICIgnOption != 3 } ; !
   field = "!Cranking Trigger",             ICCrankTrigger, { ICIgnOption != 2 && ICIgnOption != 3 } ; !
   field = "!Coil Charging Scheme",         ICIgnOption  ; !
   field = "!Spark Output",                 spkout_hi_lo ; !
   field = "!Came Input Capture",           ICCamCapture ; !
   field = "Maximum Spark Duration",        max_spk_dur
   field = "!Red settings require an MS-II reboot!"
```

9.2 *Menu pour utilisation injection séquentielle*

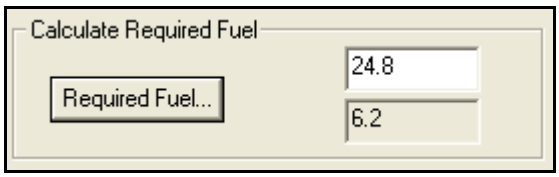
Il existe déjà un menu qui permet de sélectionner entre l'injection simultanée et alternée. Il suffit maintenant de retrouver où ce menu est créé puis lui ajouter l'injection séquentielle.

```
      fastIdleT       = scalar,  S16,    602,              "°F",       0.10000,    0.00000,-40.00,  300.00,      1
      egoTemp         = scalar,  S16,    604,              "°F",       0.10000,    0.00000,-40.00,  300.00,      1
#endif
      egoRPM          = scalar,  S16,    606,              "RPM",      1.00000,    0.00000,  0.00, 15000.0,      0 ; * (  2 bytes)
      reqFuel         = scalar,  U16,    608,              "ms",       0.00100,    0.00000,  0.00,  65.536,      2 ; * (  2 bytes)
      divider         = scalar,  U08,    610,              "",         1.00000,    0.00000,  0.00,    255,       0 ; * (  1 byte)
      alternate       = bits,    U08,    611,     [0:1],   "Simultaneous", "Alternating", "Sequential", "INVALID" ; * (  1 byte)
      injOpen         = scalar,  U08,    612,              "ms",       0.10000,    0.00000,  0.00,  25.50,       1 ; * (  1 byte)
      injPwmT         = scalar,  U08,    613,              "ms",       0.10000,    0.00000,  0.00,  25.50,       1 ; * (  1 byte)
      injPwmPd        = scalar,  U08,    614,              "us",       1.00000,    0.00000, 40.00,  100.00,      0 ; * (  1 byte)
      injPwmP         = scalar,  U08,    615,              "%",        1.00000,    0.00000,  0.00,  100.00,      0 ; * (  1 byte)
      battFac         = scalar,  U08,    616,              "ms/v",  0.0166667,       0.0,    0.0,    1.0,       2 ; * (  1 byte)
```

On trouve la variable traitant de ce menu dans la partie **[Constants].** Il a suffit de modifier **[0:0]** en **[0:1]** puis de rajouter les options **''Sequential'' et ''Invalid''**. Ceci nous permet maintenant de sélectionner le menu séquentiel. Afin d'utiliser ce type d'injection correctement, il faut configurer autant d'injections par cycle que ce qu'il y a de cylindres.



Le problème vient maintenant du calcul du ReqFuel. On constate dans l'image ci-dessous que si nous choisissons la configuration pour l'injection séquentielle, le temps de base de la Pulse est erroné. Le temps pour chaque Pulse est ici de 6.2ms alors qu'il devrait être de 24.8ms.



Pour remédier à ce problème, il a fallu modifier le code source de Megatune2.25. Celui-ci peut-être modifié avec le logiciel Visual Studio 6.0. Le fichier à modifier s'appelle **InjControl.cpp.**

Voici les modifications apportées :

InjControl ::fieldsFromDb()

```
void injControl::fieldsFromDb()
{
   int savePage = mdb.setPageNo(pageNo);
      mdb.getConst();
      _fld.setFld();

      int alternating = _fld.value(S_alternate);
      int divider     = _divider ? _divider->valueUser() : 1;
      int nSquirts    = int(0.00001 + double(_fld.value(S_nCylinders)) / divider);

      double rfqnum   = (_fld.value(S_nInjectors)) / (double(divider) * double(alternating + 1));
      if(alternating==2)
         rfqnum          =  (_fld.value(S_nInjectors)) / (_fld.value(S_nCylinders));
      double reqFuel  = _fld.value(S_reqFuel) * rfqnum + 0.0001;

      m_nSquirts.SetCurSel(nSquirts-1); // Number of squirts.

      setFld(&m_reqFuel, reqFuel, 1.0);
   mdb.setPageNo(savePage);
}

//---------------------------------------------------------------------------
```

if(alternating = = 2) : Le code test s'il s'agit de l'injection séquentielle. Si c'est bien le cas, celui-ci passe à la ligne du dessous. Si ce n'est pas le cas, alors il saute la ligne du dessous.

Calcul de la longueur de Pulse pour injection sequentiel :

$$Pulse = \frac{Nb\_injecteurs}{Nb\_cylindres} * \mathrm{Re}\,qFuel$$

injControl ::dbFromFields()

```
void injControl::dbFromFields()
{
   int savePage = mdb.setPageNo(pageNo);
      _fld.getFld();

      UINT rpmk = (_fld.value(S_twoStroke) ? 6000 : 12000) / (_fld.value(S_nCylinders));
      if (_rpmk) _rpmk->storeValue(rpmk);

      int    nCylinders  = _fld.value(S_nCylinders);
      int    alternating = _fld.value(S_alternate);
      int    divider     = m_nSquirts.GetCurSel() + 1; // User-view number of squirts per cycle.
      double intCheck    = (0.0001 + double(nCylinders) / double(divider));
      BYTE   dividerdl   = intCheck;

      if (::fabs(double(dividerdl - intCheck)) > 0.1)
         MessageBox("Number of Squirts not an integer divisor of cylinder count.");
      else if (alternating==1 && !(dividerdl < nCylinders && ::fmod(nCylinders, dividerdl*2.0) == 0.0))
         MessageBox("Cannot Alternate this Number of Squirts with this Cylinder Count.");
      else if (alternating==2 && nCylinders != divider)
         MessageBox("With sequential injection please set <<Injections Per Engine Cycle = Number of Cylinders>>.");
      else {
         if (_divider) _divider->storeValue(dividerdl);
      }
```

Dans cette partie de code, il fallu rajouter la condition que pour avoir une injection séquentielle, le nombre d'injection par cycle doit être égal au nombre de cylindres.

9.3 *Menu pour l'utilisation du retard/avance à l'injection*

Le but de concevoir ce menu est que l'utilisateur puisse en temps réel modifier l'avance/retard à l'injection. Ceci afin de pouvoir visualiser les conséquences du choix de l'angle sur la consommation, la puissance ainsi que la pollution. L'utilisateur peut choisir un angle compris entre –360° et 360° avec une précision au degré près.

Modification du Fichier .INI :

Pour réaliser ce menu, j'ai du rajouter dans la section **[Constants]** la variables utilisée dans le code uC maître. Il faut la placer exactement comme dans le code. C'est-à-dire après la variable ICCamCapture.

```
injTestSqrts    = scalar,  U16,    982,            "squirts",  1,          0,          0,    65000,      0 ;   (  2 bytes)
injTestPW       = scalar,  U16,    984,            "us",       1,          0,          0,    65000,      0 ;   (  2 bytes)
injTestOffTime  = scalar,  U16,    986,            "ms",       0.1,        0,          0,     6500,      1 ;   (  2 bytes)

ICCamCapture    = bits  ,  U08,    988,      [0:1], "None", "Falling Edge", "Rising Edge", "INVALID"       ; * (  1 byte )
Injection_delay = scalar,  S16,    989,            "deg",      1.0,            -360,  -360,      360,      0 ; * (  2 bytes)
pageSize = 991
```

La variable s'appelle **Injection_delay**. **scalar** signifie que la variable est utilisé comme nombre entier.

La variable est de type **S16** => signed INT. Elle est enregistrée à la position 989. ''**deg**'' est l'unité de la valeur rentrée. Le fichier .INI permet également la conversion d'une valeur provenant du code du uC maître ou allant vers celui-ci. Voici la conversion possible :

Valeur **allant** vers le uC maître : $$uC1\_value = \frac{UserValue}{scale} - translate$$

Valeur **provenant** du uC maître : $$UserValue = (uC1\_value + translate) * scale$$

1.0 est la valeur **scale.** Le premier -360 signifie la valeur de **translate**. Le second est la valeur **minimum** que peut rentrer l'utilisateur. Le 360 est la valeur **maximum**.

*(2 byte) est uniquement un commentaire montrant la taille de la variable. Ne pas oublier d'incrémenter la variable **pageSize**.

Il faut maintenant aller dans la partie **[UserDefined]** afin de réaliser le menu. On veut ajouter ce sous-menu dans le menu Other Fuel Settings. C'est pourquoi on rajoute les deux lignes ci-dessous (Encadré rouge). **''Injection –advance/+retard''** est le nom du sous-menu. Le nom de la variable utilisateur est ajoutée après le titre du sous-menu. Afin que ce menu ne soit accessible uniquement en injection séquentielle, il a fallu rajouter ceci :

{ alternate = = 2}

La première ligne est uniquement l'affichage du texte permettant de comprendre le sous-menu.

Le symbole''#'' signifie que nous voulons que le titre apparaisse avec un fond bleu.

```
dialog = otherFuel, "Other Fuel Settings"
   topicHelp = "http://www.megasquirt.info/megatune.htm#so"
   field = "#Two-Point Prime, ASE"
   field = "Prime Pulse Hot PW",       primePulseHot, { CWOption == 0 }
   field = "ASE Hot Percent",          asePctHot,     { CWOption == 0 && xTauOption < 2 }
   field = "ASE Hot Count",            aseCountHot,   { CWOption == 0 && xTauOption < 2 }
   field = "#AE RPM Scaling"
   field = "Low RPM Threshold",        ae_lorpm,      { xTauOption < 1 }
   field = "High RPM Threshold",       ae_hirpm,      { xTauOption < 1 }
   field = "#Injection Timing only with sequential injetion",
   field = "Injection -advance/+retard" Injection_delay,{ alternate == 2}
```

9.3.1 Calibrage automatique des tables d'injection

Comme déjà mentionné, les calculateurs n'effectuent pas un calcul en temps réel de l'essence à injecter en fonction de l'air entrant. Le temps d'injection est prédéterminé dans une table avec comme axe x la pression de l'air entrant et comme axe y la rotation du moteur. Nous retrouvons en z le temps d'injection.

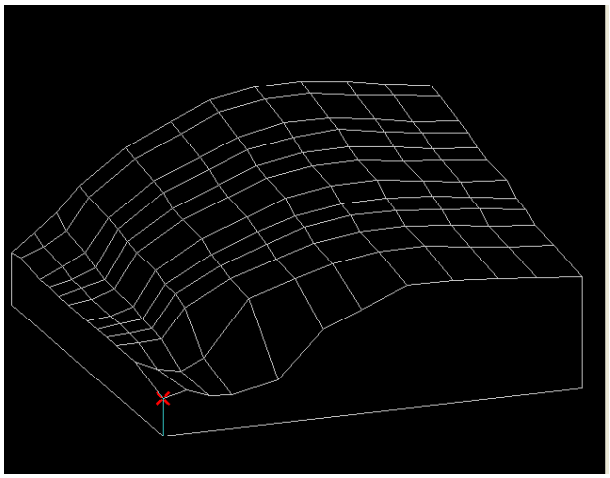


Figure 9.3-1: Table d'injection

Modification du Fichier .INI :

Comme mentionné plus haut, le logiciel Megatune2.25 permettait déjà un calibrage automatique. Il suffisait de modifier le fichier custom.INI afin de faire fonctionner celui-ci. Il faut aller dans la partie **[AutoTune]** du fichier.

```
      ; Controller parameters
      initialStartupInterval = 1.0 ; Seconds before first adjustment
      updateInterval         = 1.0 ; Seconds between each consecutive adjustment.
      proportionalGain       = 0.5 ; Proportion of (100-corrector) to use for adjustment.
      lumpiness              = 5   ; Maximum percent adjustment above or below neighboring VE points.

#elif MS_II
   table = veTable1Map
      allowAutoTune          = on
      corrector              = egoCorrection1
      xLimits                = 0, 8000
      yLimits                =   0,   100
      zLimits                =   0,  200
      xRadius                =  200
      yRadius                =    7
      initialStartupInterval =  1.0
      updateInterval         =  1.0
      proportionalGain       =  0.5
      lumpiness              =    5

   table = veTable2Map
      allowAutoTune          = on
      corrector              = egoCorrection2
      xLimits                = 0, 4000
      yLimits                =   0,   90
      zLimits                =  10,  200
      xRadius                =  200
      yRadius                =    7
      initialStartupInterval =  1.0
      updateInterval         =  1.0
      proportionalGain       =  0.5
      lumpiness              =    5
```

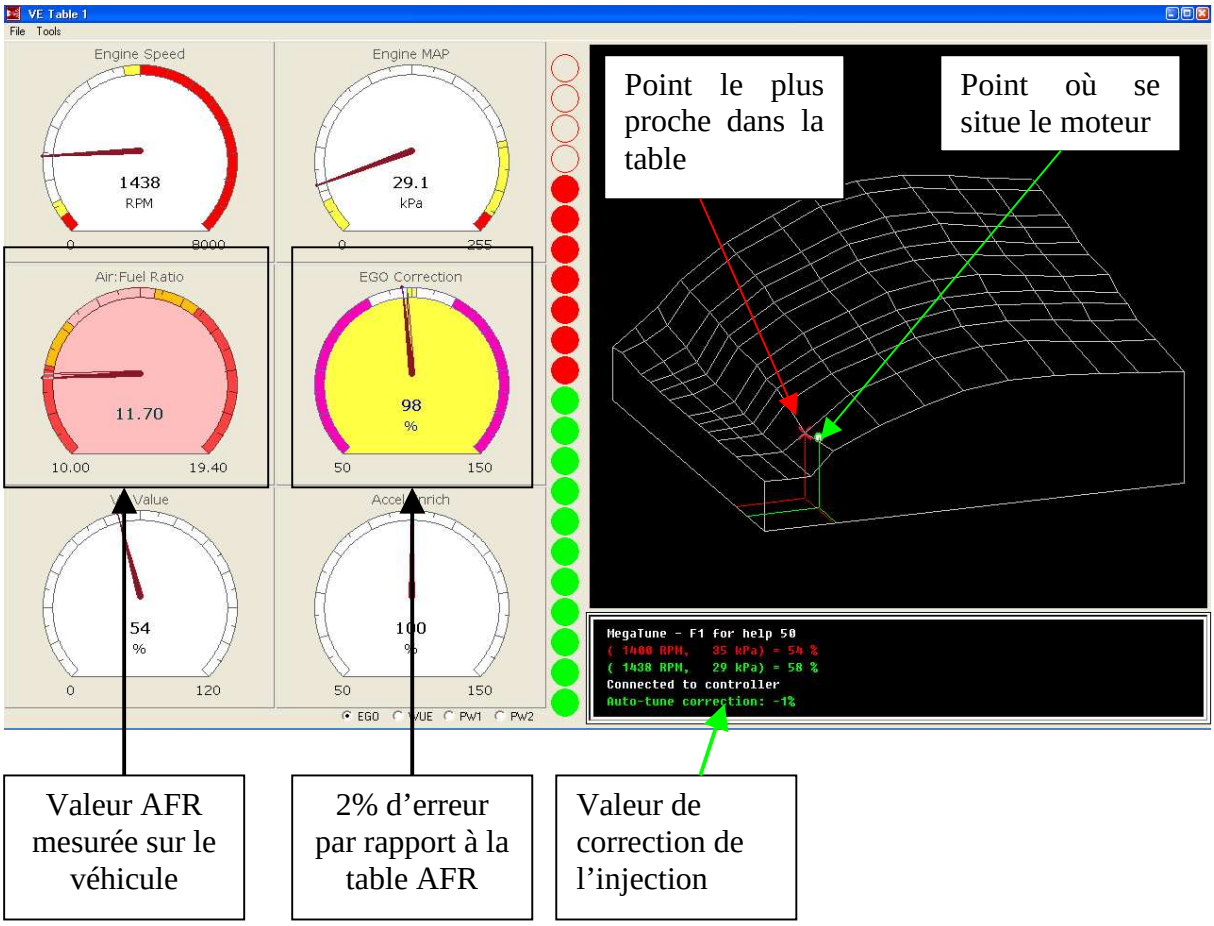
Utilisation de MegasquirtII ?

Voici la définition des paramètres à configurer dans le fichier :

AllowAutoTune : Autorisation d'utiliser le calibrage automatique ? on => oui, off => non
Corrector : Variable de correction. C'est en fonction de cette variable que le régulateur va fonctionner.
xLimits : Définit dans quelle portion de l'axe x nous voulons autoriser le calibrage automatique
yLimits : Définit dans quelle portion de l'axe y nous voulons autoriser le calibrage automatique.
zLimits : Définit les valeurs maximum et minimum d'injection.
xRadius : Définit la distance maximum entre le points mesuré et le point le plus près se situant dans la table afin d'autorisé le calibrage automatique. Exemple : Nous sommes à 4000tr/min. Le point de la table le plus proche se trouve à 4500tr/min => Nous somme à 500tr/min du point le plus proche. Comme xRadius = 200, le calibrage automatique ne sera pas autorisé.
YRadius : De même que xRadius mais pour l'axe y.
initialStartupInterval : Temps en seconde avant d'autoriser le premier calibrage
updateInterval : Temps entre chaque calibrage
proportionalGain : Définit le gain qui sera appliqué sur la variable egoCorrection.
Lumpiness : Définit le pourcentage maximal qu'AutoTune peut modifier.

La variable egoCorrection est calculée à partir de la table AFR. Donc, l'injection va se calibrer en fonction de la table AFR.

Voici la page de garde utilisée pendant le calibrage automatique :



Valeur AFR mesurée sur le véhicule

2% d'erreur par rapport à la table AFR

Valeur de correction de l'injection

10 Simulation des capteur du moteur

Afin de pouvoir tester la carte d'injection séquentielle, le code implémenté dans le uC maître ainsi que les modifications apportées à Megatune2.25 sans avoir à risquer de casser un moteur, il est nécessaire de pouvoir simuler les valeurs provenant des différents capteurs de celui-ci. C'est uniquement après avoir constaté que le tout fonctionne correctement que les tests pratiques peuvent être effectués. J'ai donc réaliser une carte permettant à l'aide potentiomètre de simuler la valeurs de ces capteurs.

10.1 *Carte de simulation*

La carte de simulation permet de simuler ces différents capteurs :

- Papillon des gaz
- Sonde lambda
- Capteur T° air
- Capteur T° moteur

Vous trouverez dans le CD-ROM (Annexe 18.6) la schématique ainsi que le routage.

10.1.1 Schéma bloc

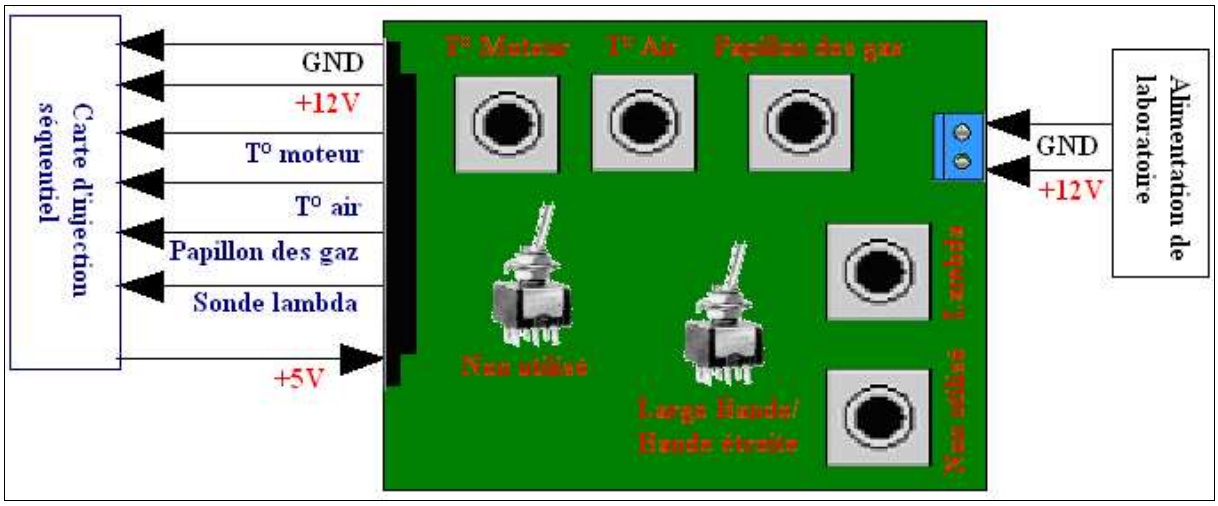


Figure 10.1-1 Schéma bloc carte de simulation

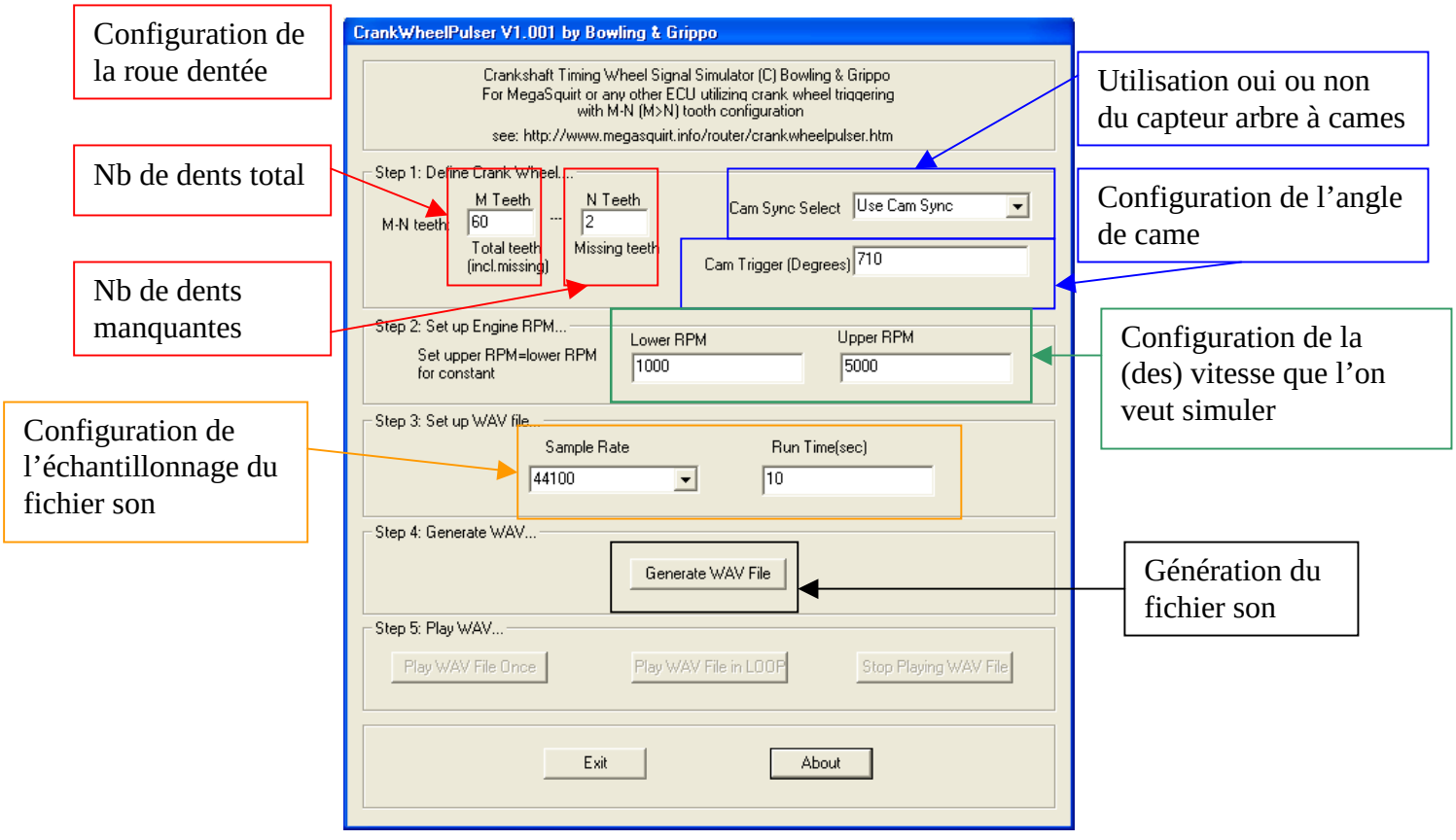
On peut voir sur ce schéma bloc les potentiomètres utiles à simuler la valeur des capteurs. On constate également qu'il y a un interrupteur permettant de simuler soit une sonde lambda large bande soit une bande étroite.

10.2 *Simulation capteur vilebrequin(roue dentée) et capteur arbre à cames*

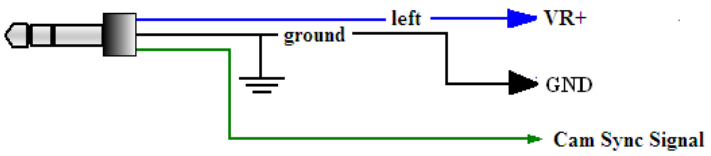
Comme vu précédemment, le signal provenant du capteur vilebrequin avec utilisation d'une roue dentée ne possède pas une forme standard(sinus, carré, linéaire). C'est pourquoi il est difficile de simuler ceci. Sur internet, j'ai trouver un logiciel permettant de réaliser ce signal.

Suivant les paramètres que nous entrons dans le logiciel, celui-ci va nous créer un fichier son qui fournit un signal similaire à celui du capteur vilebrequin. Ce signal peut être repris à travers la prise casque de votre PC. Nous savons que la prise casque permet de restituer deux signaux(stéréo). C'est pourquoi nous pouvons également configurer l'utilisation d'un capteur d'arbre à cames. Le logiciel se trouve dans le CD-ROM (Annexe 18.6).

Configuration logiciel :



Voici le schéma de câblage du signal son :



11 Banc de test CFP

Afin de pouvoir tester la carte d'injection séquentielle réalisée pendant le travail de diplôme, le centre professionnel de Sion nous a mis à disposition un banc de test moteur. Il s'agit d'un moteur Opel Kadett GSI avec installation électrique Motronic 2.5 de chez Bosch.

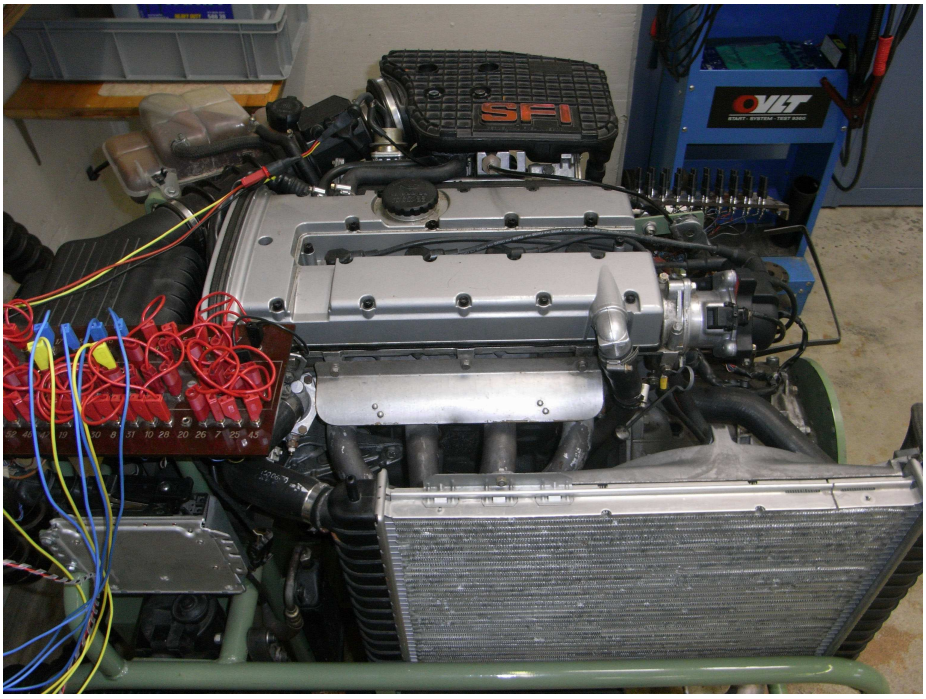


Figure 10.2-1: Banc moteur CFP

11.1 *Caractéristiques mécaniques*

Il s'agit d'un moteur 4 cylindres en ligne de 1998 cm^3 possédant 16 soupapes. Celui-ci développe une puissance maximum de 156 ch. (115 kW) à 6000 tr/min. Le régime maximum est de 6800 tr/min. Le couple maximum est de 203 Nm à 4800 tr/min.

11.2 *Caractéristiques électriques*

L'installation électrique est réalisée par Bosch. Il s'agit du système Motronic 2.5. **Vous trouverez dans le CD-ROM le schéma d'entrées-sorties du boitier électronique ainsi qu'un explicatif général de ce système d'injection Bosch.** Les injecteurs sont des haute impédance d'environ 16 [Ω] chacun, ce qui fait un courant maximum de 12V/16Ω = 750mA.

11.3 *Types de capteurs et réglages/modifications*

Il est important de connaître les types de capteurs montés sur ce moteur afin de pouvoir régler au mieux le software d'injection séquentielle. Nous devrons également effectuer quelques modifications afin d'adapter notre carte à ce moteur.

Capteur vilebrequin (Pin 47 Motronic) :

Le capteur vilebrequin est une inductance magnétique posée en face d'une roue dentée. Cette roue dentée possède 60 dents dont 2 manquante au point mort haut. Il est important de bien régler les 2 potentiomètres sur la carte d'injection séquentielle ainsi que de placer le JUMPER JP5 afin d'avoir le signal ci-dessous :

Signal provenant du capteur vilebrequin

Signal allant à l'entrée du uC maître

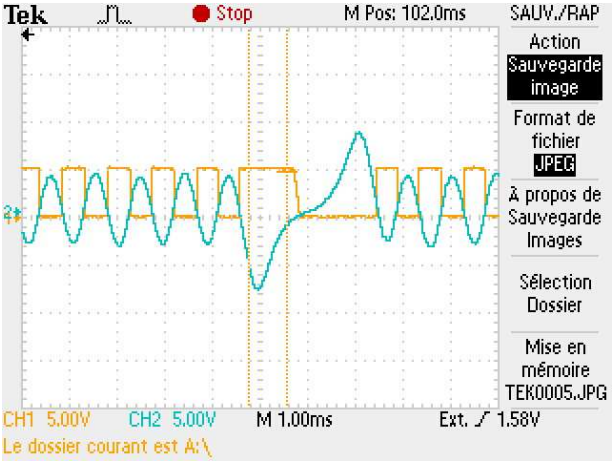


Figure 11.3-1 Mesure capteur vilebrequin

Voici les valeurs à entrer dans Megatune2.25 afin de calibrer correctement la mesure de vitesse :

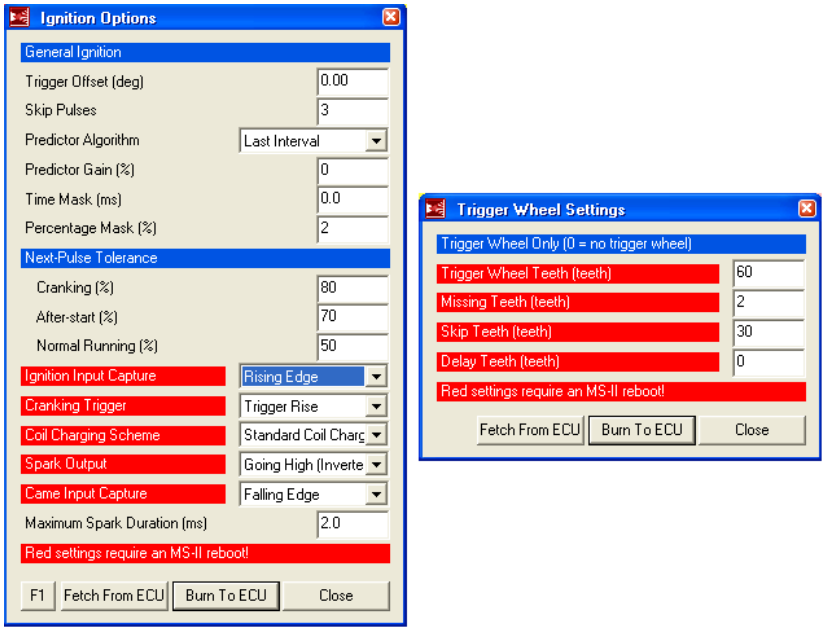


Figure 11.3-2 Configuration dans Megatune2.25 du capteur vilebrequin

Capteur de pression (Externe) :

Ce moteur possède un débitmètre d'air permettant au boitier de savoir combien d'air entre dans les cylindres. Le problème est que notre software d'injection séquentielle n'est pas prévu pour ce type de capteurs. Celui-ci est prévu pour utiliser un capteur de pression lié à un capteur de température d'air. En effet avec ces deux valeurs, il est également possible de connaître la quantité d'air entrée dans le cylindre. Il a fallu rajouter un capteur de pression à l'admission du moteur. Pour les tests, il n'est pas nécessaire de rajouter le capteur de température car nous travaillons dans un milieu à température ambiante de 20°C.

Figure 11.3-3 Montage capteur de pression

Le capteur de pression utilisé est un MPX4115AS. Celui-ci permet de mesurer une pression absolue allant jusqu'à 115 kPa. Ce qui convient parfaitement pour la mesure de pression sur moteur atmosphérique. Sur un moteur turbo, il faudrait utiliser un capteur permettant de monter plus haut en pression. Voici les valeurs de calibration à entrer dans Megatune2.25 en fonction du capteur utilisé :

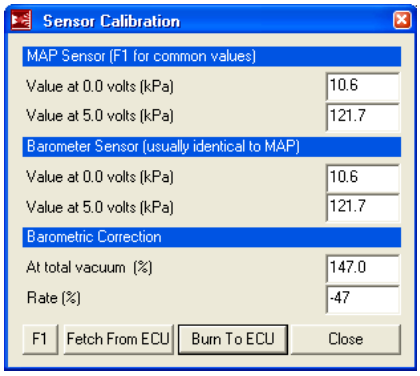
Capteur	Value at 0.0 volts (kPa)	Value at 5.0 volts (kPa)	vLo [V]	pLo [kPa]]	vHi[V]	pHi [kPa]	Vref [V]
MPX4115	**10.6**	**121.7**	0.204	15	4.794	115	5.1
MPX4250	10	260	0.204	20	4.896	250	5.1
MPXH6300	1.1	315.5	0.306	20	4.913	304	5.1
GM 3-BAR	1.1	315.5	0.631	40	4.914	304	5.1
MPXH6400	3.5	416.5	0.200	20	4.800	400	5

Voici les formules utilisée pour retrouver ces valeurs:

$$m = \frac{(pHi - pLo)}{vHi - vLo}$$

$$pv1 = pLo - m * vLo$$

$$pv2 = pv1 + m * v\,\mathrm{Re}\, f$$



Température moteur (Pin 45 Motronic) :

Le capteur de température moteur est une résistance NTC immergée dans le liquide de refroidissement moteur. La résistance est alimentée par le boitier Motronic, ce qui veut dire que la référence de tension est celle du Motronic. C'est pourquoi il ne faut pas utiliser la référence de notre carte. Le JUMPER JP2 doit être enlevé.

Voici les valeurs à entrer dans Megatune2.25 afin de calibrer notre carte pour cette sonde :

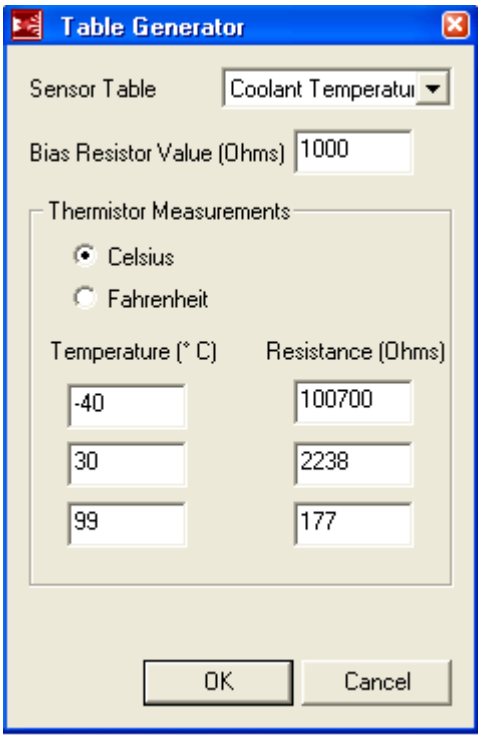


Figure 11.3-4 Configuration des capteurs T° dans Megatune2.25

Sonde Lambda (Externe) :

Le système Motronic travail avec une sonde lambda à bande étroite. Pour le calibrage des paramètres du moteur cette sonde n'est pas du tout adaptée. C'est pourquoi il a fallu rajouter une sonde lambda large bande sur le moteur qui elle fournit à notre carte la qualité exacte du mélange air/essence. **Dans le chapitre 12**, vous trouverez un explicatif sur l'utilisation de la sonde.

Voici le schéma de câblage de la sonde :

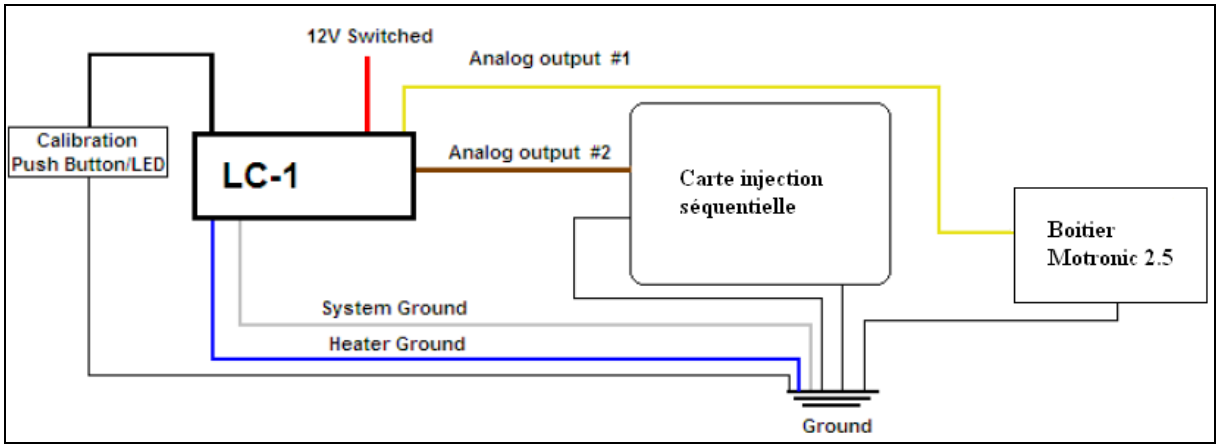
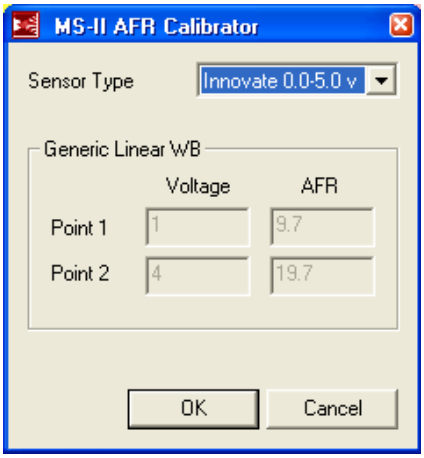


Figure 11.3-5 Schéma de câblage de la sonde lambda sur le moteur

Le fil **rouge** doit être relié au 12V. Il faut faire attention à bien brancher sur du 12V possédant un fusible de 5A. En effet la sonde consomme à froid un courant de 2.5A.
Le fil **noir** est le fil de calibration. Sur celui-ci est câblé un interrupteur qui permet à tout moment de faire la calibration de la sonde.
Les fils **blanc** et **bleu** sont des fils de masse.
Le fil **jaune** est le signal de sortie analogique numéro 1. Celui-ci a été configuré pour recréer le signal d'une sonde lambda à bande étroite. J'ai décidé de le câblé sur le boitier Motronic afin que celui-ci ne se mette pas en mode d'erreur en croyant qu'il y a une sonde défectueuse.
Le fil **brun** est le signal de sortie analogique numéro 2. Celui-ci a été configuré pour avoir en sortie un signal de 0V pour un rapport lambda de 7.35 et 5V pour un rapport de 22.39.

Configuration Megatune2.25 :



Capteur papillon des gaz (Pin 53 et 52 Motronic) :

Le capteur de position du papillon des gaz de ce moteur ne fournit uniquement les informations suivantes : ralenti, moyenne charge, pleine charge. Ces informations sont codées sur deux signaux. Ce qui n'est pas compatible avec notre carte. Nous savons que ce capteur est utile uniquement pour le calcul de l'enrichissement du mélange en cas de forte accélération et pour le réglage du ralenti. Celui-ci n'est pas utilisé pendant le calibrage des paramètres. C'est pourquoi il n'est pas câblé. Si par la suite on voudrait connaître cette information, il faudrait rajouter sur le papillon des gaz un potentiomètre avec une résistance en série le tout alimenté par Vref qui est de 5V.

12 Lambda large bande et Innovation LC-1 Kit

Voici le chapitre traitant la sonde lambda, plus particulièrement de la sonde lambda large bande. Pourquoi montrer autant d'intérêt pour ce capteur et moins pour les autres ? Tout simplement car sans celui-ci, il est impossible de réaliser une table d'injection précise et encore moins de réaliser un calibrage automatique.

12.1 *Sonde large bande*

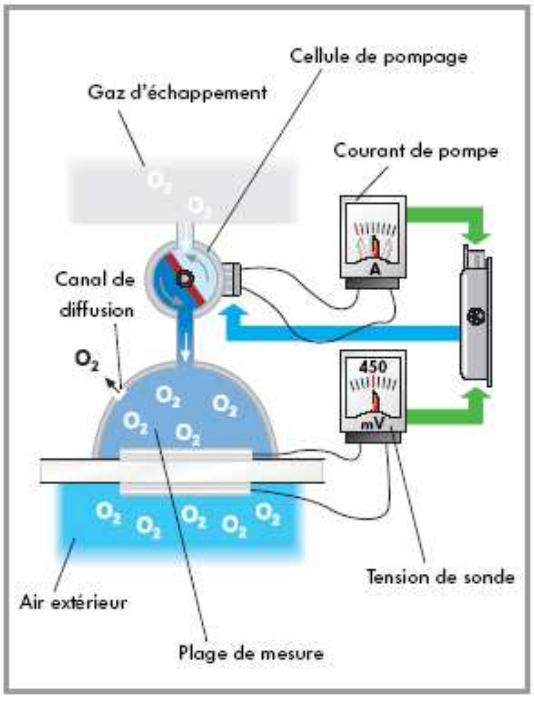


Figure 12.1-1 Schéma bloc sonde lambda

Cette sonde produit une tension par le biais de deux électrodes, qui résulte de la différence de teneur en oxygène. La différence par rapport à la sonde lambda bande étroite réside dans la tension constante des électrodes. Ce processus est obtenu grâce à une cellule de pompage (pompe miniature) qui alimente l' électrode côté échappement avec une quantité d' oxygène permettant de maintenir la tension à une valeur constante de 450 mV. L' appareil de commande du moteur convertit la consommation électrique de la pompe en une valeur lambda.

Exemple de fonctionnement :

Le mélange air-carburant s' appauvrit, ce qui signifie que la teneur en oxygène des gaz d' échappement augmente et qu' à puissance égale de la pompe, la cellule pompe davantage d' oxygène dans la plage de mesure qu.il ne peut s' échapper du canal de diffusion. Ce processus a pour effet de modifier la proportion d' oxygène par rapport à l.air extérieur et d' entraîner une baisse de la tension entre les électrodes.

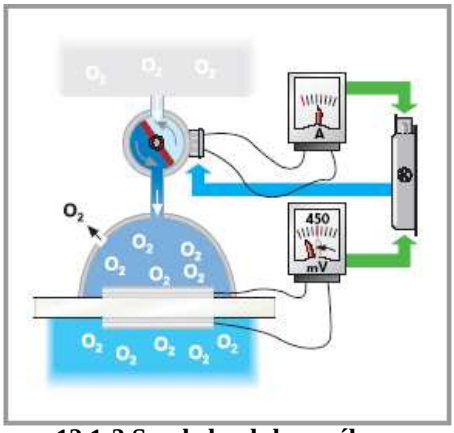


Figure 12.1-3 Sonde lambda - mélange pauvre

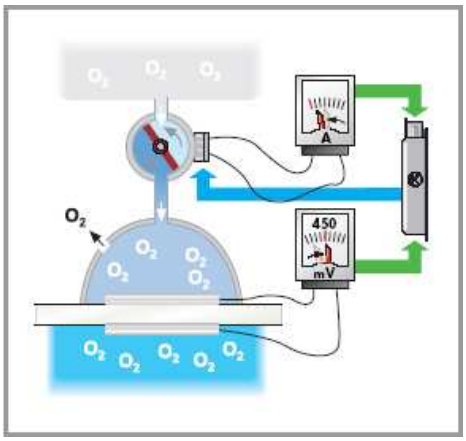


Figure 12.1-4 Sonde lambda - mélange normal

Afin qu.une tension de 450 mV puisse de nouveau être atteinte entre les électrodes, il est nécessaire de réduire la teneur en oxygène côté échappement. Pour ce faire, la cellule doit pomper une quantité moindre d'oxygène dans la plage de mesure. La puissance de la pompe est ainsi réduite jusqu'à ce qu.une tension de 450 mV soit de nouveau atteinte. L'appareil de commande du moteur convertit la consommation électrique de la pompe miniature en une valeur de régulation lambda et modifie en conséquence la composition du mélange.

Lorsque le mélange air/carburant devient trop riche, la teneur en oxygène des gaz d' échappement diminue. A puissance égale de la pompe, la cellule achemine ainsi une quantité moindre d' oxygène dans la plage de mesure et la tension entre les électrodes augmente. Dans ce cas, il s' échappe à travers le canal de diffusion une quantité d' oxygène plus importante que celle refoulée par la cellule de pompage.

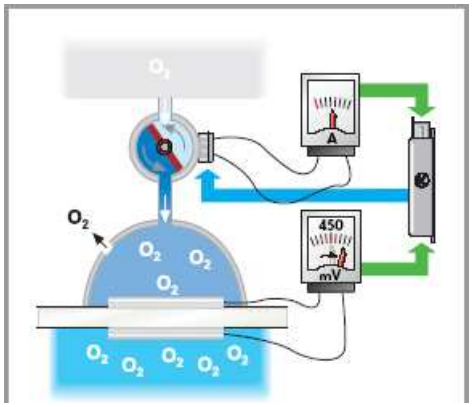


Figure 12.1-5 Sonde lambda - mélange

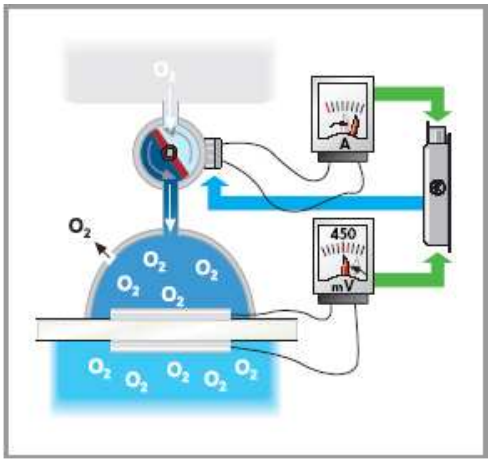


Figure 12.1-2 Sonde lambda - mélange normal

La puissance de la cellule de pompage doit être augmentée de manière à accroître la teneur en oxygène dans la plage de mesure. La tension au niveau des électrodes est ainsi de nouveau réglée sur une valeur de 450 mV et l' appareil de commande du moteur convertit la consommation électrique de la cellule de pompage en une valeur de régulation lambda.

12.2 *Innovation LC-1 Kit*

Comme vu plus haut, une sonde lambda large bande n'est pas si simple d'utilisation. Il faut un régulateur en amont afin de régler un courant. C'est grâce à ce courant que la valeur lambda pourra être connue. Il est important pour la conception de tables d'injection que la sonde lambda soit parfaitement calibrée. C'est pourquoi j'ai décidé de commander un Kit possédant le régulateur ainsi que la sonde lambda. Ce kit permet de contrôler le chauffage de la sonde, le calibrage de la sonde ainsi que la régulation en courant. Ce kit possède deux signaux de sorties analogiques permettant d'avoir la valeur lambda. La réponse en sortie en fonction de la valeur lambda peut être modifiée au gré de l'utilisateur par le PC.

Figure 12.2-1 Kit LC-1

13 Tests

Voici la partie tests sur le banc moteur. Nous allons tout d'abord tester l'injection séquentielle à l'aide de la carte de simulation. Avec ce test, on pourra confirmer que notre carte injecte au bon moment ainsi que dans le bon cylindre. Le second test est celui de l'avance/retard à l'injection. Ensuite la partie calibrage automatique des paramètres d'injection pourra être testée. En dernier point, je vais mesurer les rejets polluants sortant du moteur pour différentes configurations de celui-ci. **Vous trouverez dans le CD-ROM (Annexe 18.6) un complément de tests effectués.**

13.1 *Injection séquentielle*

Voici la partie tests de l'injection séquentielle : Les couleurs des mesures ont été gardées pour tous les tests. C'est pourquoi la description de celles-ci ne se fait que sur la première image.

Tests sans retard/avance à 800tr/min pour moteur 4 cylindres :

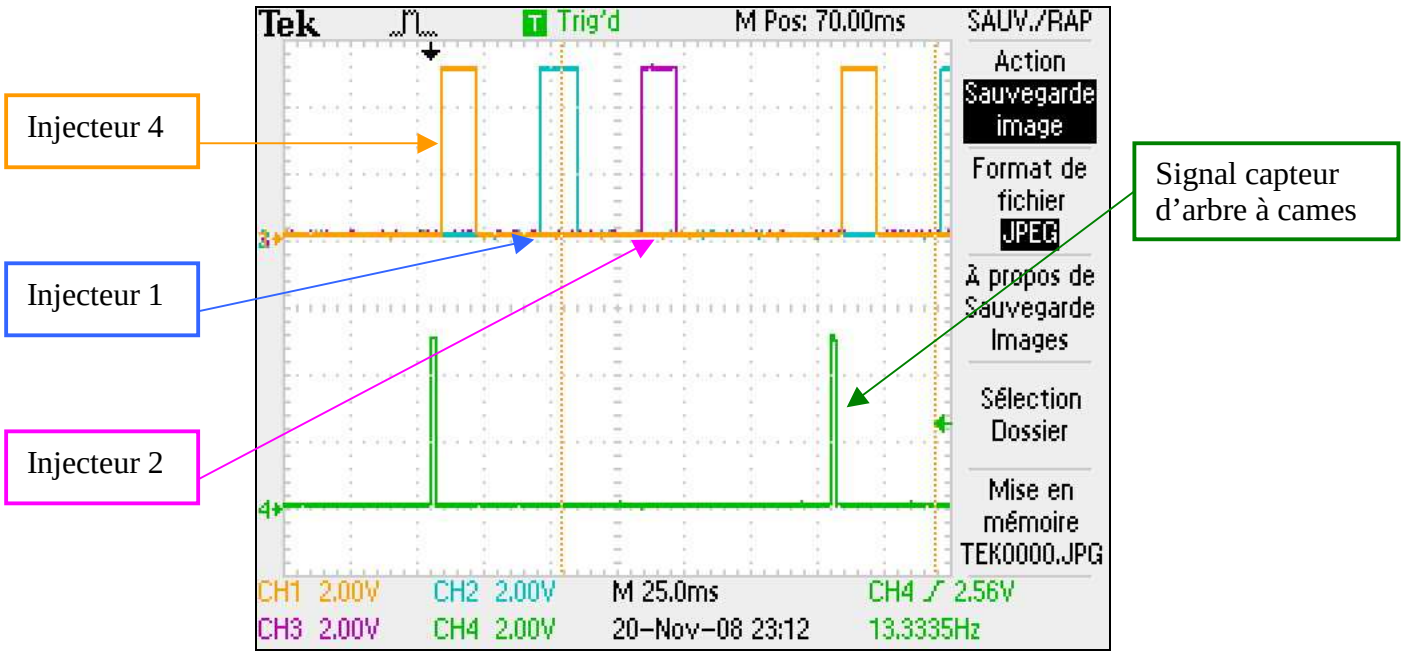


Figure 13.1-1: Test injection séquentielle à 800tr/min

Sur ce test on peut constater que l'injection séquentielle fonctionne correctement. On peut voir que tout de suite après avoir reçu le signal du capteur arbre à cames il se produit une injection dans le cylindre 4(Voir Timing d'injection pour moteur 4 cylindres dans le CD-ROM). Puis 180° plus loin il y a une injection dans le cylindre 1 et ainsi de suite.

Tests sans retard/avance à 3000tr/min pour moteur 4 cylindres :

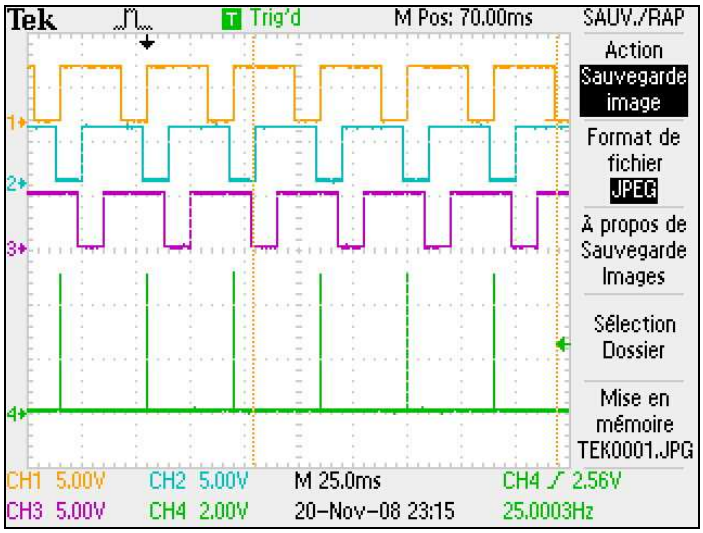


Figure 13.1-2: Test injection séquentielle à 3000tr/min

On constate que l'injection séquentielle fonctionne toujours correctement à cette rotation. On peut voir que maintenant les injections se chevauchent correctement. On remarque également que le temps d'injection a augmenté ce qui cause le chevauchement des injections. L'ordre d'injection est toujours respecté.

Tests sans retard/avance à 6000tr/min pour moteur 4 cylindres :

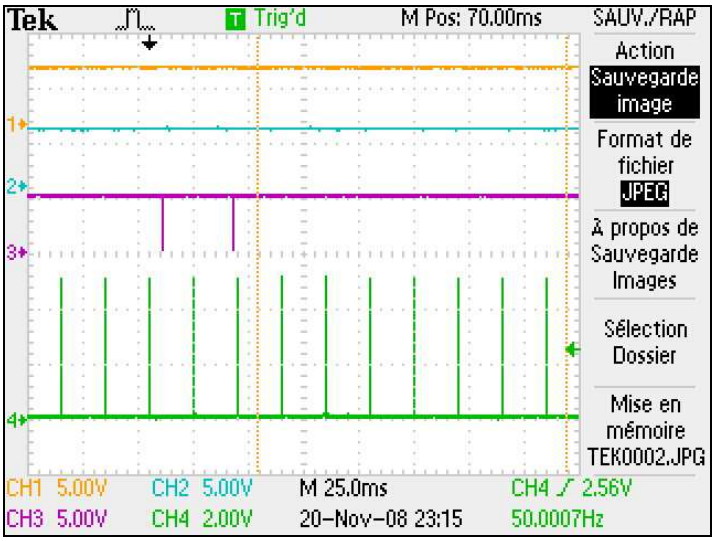


Figure 13.1-3: Test injection séquentielle à 6000tr/min

Sur cette mesure on peut voir que le moteur tourne à pleine charge. C'est pour cela que les injecteurs sont continuellement en train d'injecter.

13.2 *Avance/retard à l'injection*

Voici la partie tests de l'avance/retard à l'injection. Je n'ai pas pris des images pour toutes les possibilités d'angle mais uniquement pour les plus critiques. C'est angle critiques sont les angles où des Tach Pulse interviennent.

Test avec avance de 360°à 800tr/min pour moteur 4 cylindres :

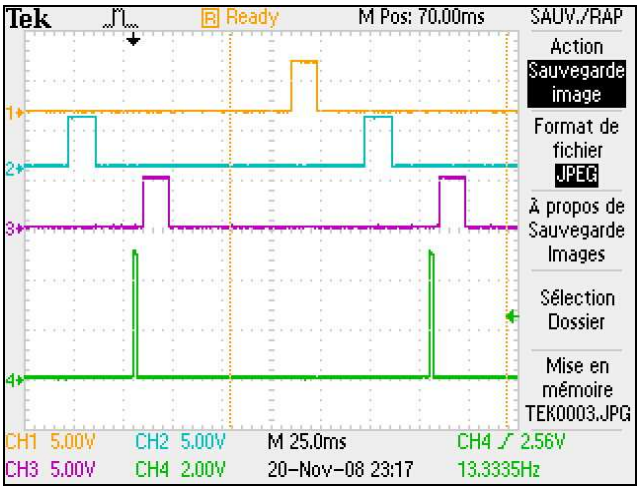


Figure 13.2-1: Test avance de 360° avec injection séquentielle

On constate sur cette mesure que l'enclenchement des injecteurs s'est bien décalé de 360° vers la gauche par rapport à la mesure faite sans avance ni retard (Figure 13.1-1).

Test avec avance de 180°à 800tr/min pour moteur 4 cylindres :

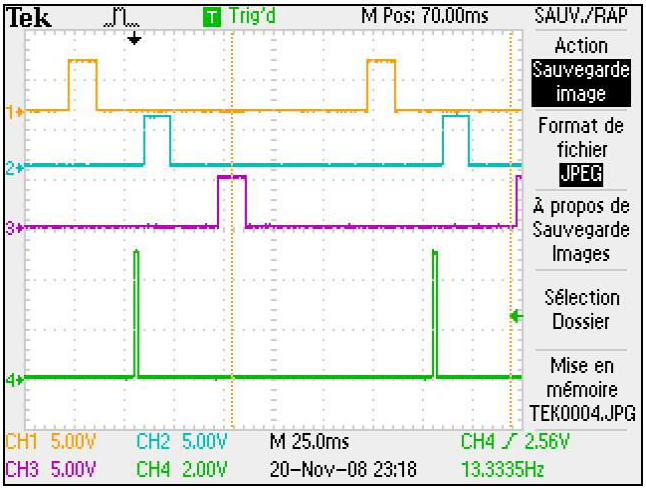


Figure 13.2-2: Test avance de 180° avec injection séquentielle

On constate sur cette mesure que l'enclenchement des injecteurs s'est bien décalé de 180° vers la gauche par rapport à la mesure faite sans avance ni retard (Figure 13.1-1).

Test avec retard de 180°à 800tr/min pour moteur 4 cylindres :

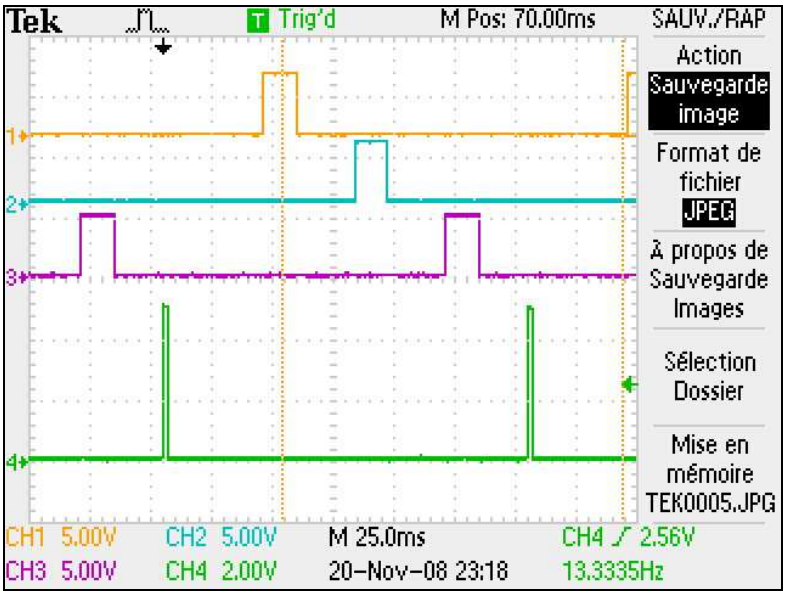


Figure 13.2-3: Test retard de 180° avec injection séquentielle

On constate sur cette mesure que l'enclenchement des injecteurs s'est bien décalé de 180° vers la droite par rapport à la mesure faite sans avance ni retard (Figure 13.1-1).

Test avec retard de 360°à 800tr/min pour moteur 4 cylindres :

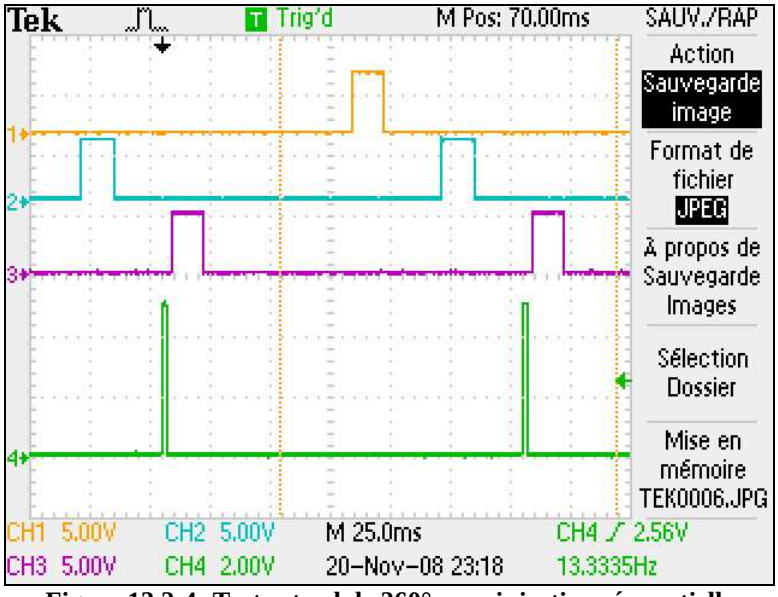
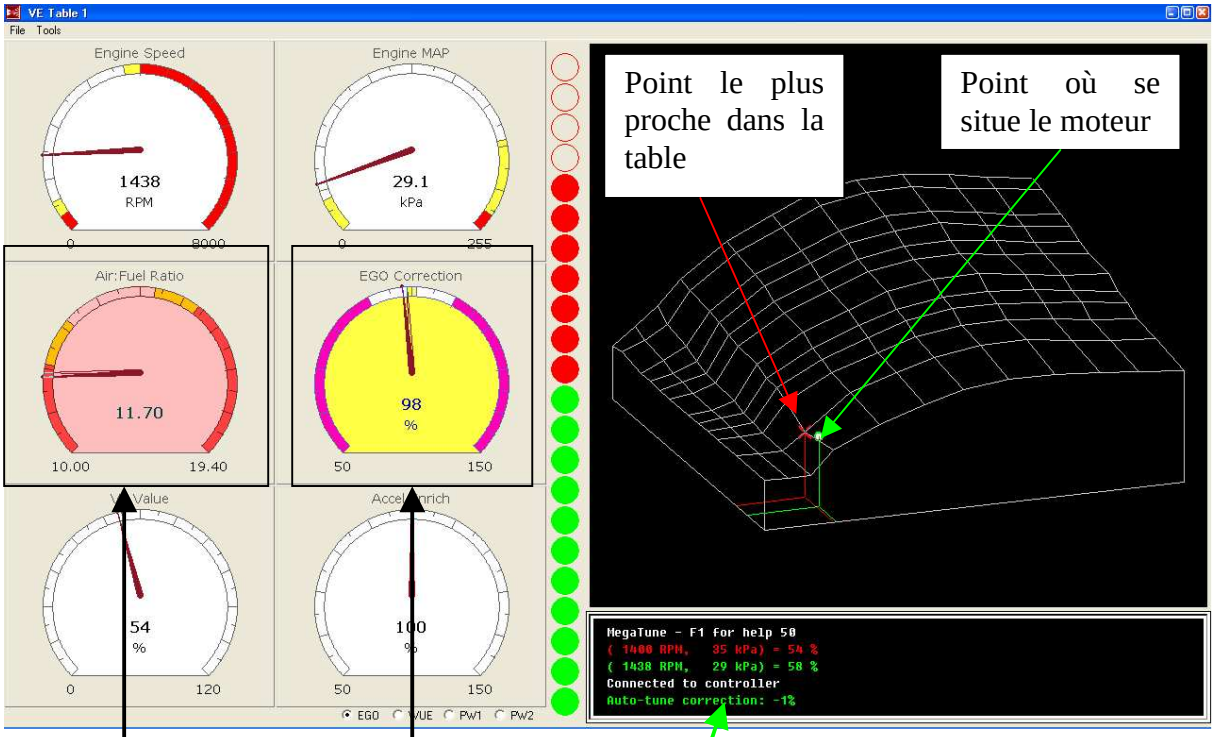


Figure 13.2-4: Test retard de 360° avec injection séquentielle

On constate sur cette mesure que l'enclenchement des injecteurs s'est bien décalé de 360° vers la droite par rapport à la mesure faite sans avance ni retard (Figure 13.1-1).

13.3 *Calibrage automatique*

Voici maintenant la partie de test du calibrage automatique. Il est très compliqué de pouvoir montrer des mesures concrètes à l'aide d'image. Ceci car quand on test sur une auto, le software va injecter plus ou moins en fonction du pourcentage de correction, ce qui va tout de suite faire changer le comportement du moteur. C'est pourquoi vous trouverez ci-dessous uniquement une illustration du calibrage automatique en marche.



Valeur AFR mesurée sur le véhicule

2% d'erreur par rapport à la table AFR

Valeur de correction de l'injection

13.4 *Anti-pollution*

Il est important d'effectuer des mesures d'anti pollution afin de démontrer l'utilisation de notre système. En effet celles-ci pourrons montrer les effets positifs sur l'anti-pollution que peut engendrer l'utilisation de ce système. Malheureusement ces mesures n'ont pas pu êtres effectués avant la rédaction de ce rapport. **C'est pourquoi vous les trouverez dans le CD-ROM (Annexe 18.6).**

13.4.1 Comparaison injection simultanée avec injection séquentielle

CD-ROM (Annexe 18.6).

13.4.2 Effets de l'avance/retard à l'injection

CD-ROM (Annexe 18.6).

13.4.3 Comparaison Motronic2.25 avec Carte d'injection séquentielle

CD-ROM (Annexe 18.6).

14 Améliorations/Suite du projet

Le résultat final de ce travail de diplôme est très concluant. En effet l'injection séquentielle ainsi que le calibrage automatique ont pu être testés avec succès. Les résultat obtenus concordent avec le cahier des charges présenté. Cependant il reste quelques améliorations à ajouter afin d'avoir un système 100% fiable.

Améliorations :

- Calcul du Reqfuel

Si on choisit dans Megatune2.25 l'option séquentielle, le calcul du Reqfuel va être exacte pour autant que le nombre d'injection par cycle soit correct. Si celui-ci est mal choisi, un message d'erreur apparaitra. Jusque là tout est OK. Après que ce message d'erreur disparaisse, le ReqFuel ne va pas prendre la bonne valeur.

- CanBus

Avec la communication CanBus, nous avons un retard d'injection pour les injecteurs du uC esclave de 340us. Pour pallier à ce problème, il faudrait utiliser l'entrée prévue pour la synchronisation des deux uC. Le uC maître enverrai à l'avance les informations concernant l'injection. Puis au moment exact de l'injection, il activerait le bit de synchronisation qui serait directement perçu par le uC esclave. Ce bit déclencherait une interruption lançant immédiatement l'injection. Ceci permettrai également de libérer la bande passante du CanBus.

- Taille du Code uC/Rajout d'options

Comme dit précédemment, le compilateur accepte jusqu'à 32 kbytes de code. Pour arriver à cette limitation, j'ai du placer des options d'injection déjà existantes en commentaires. Le code pourrait être maintenant optimisé au niveau taille. C'est-à-dire que les fonctions répétée pourrait être assemblées en une seule fonction et utilisées à l'aide de pointeurs. En effectuant cette opération il serait possible de récupérer les options d'injection mises en commentaires.

Suite du projet :

Le système est maintenant capable de paramétrer automatiquement la table d'injection en fonction des paramètres utilisateur. Cependant pour que ce calibrage automatique se fasse, l'utilisateur doit contrôler par lui-même la pédale des gaz ainsi que la charge pour passer sur tous les points de la table d'injection. Le but serait maintenant de concevoir un système entièrement automatique. C'est-à-dire que le moteur accélérerait ou décélérerait tout seul et que la charge augmenterait ou diminuerait automatiquement. Ceci afin de réaliser toute la table sans aucune intervention humaine. Il faudrait concevoir un logiciel permettant de commander un moteur pas-à-pas(réglage pédale des gaz) ainsi qu'une charge (Onduleur). La charge se ferait à l'aide d'un moteur électrique qui serait commandé par un onduleur pour freiner le moteur. Ainsi l'énergie de freinage serait convertie en électricité qui pourrait être soit renvoyée dans le réseau soit être dissipée dans des résistances.

15 Remerciements

Atelier mécanique CFP de Sion : Je souhaite tout particulièrement remercier M. Eric Beytrison (Responsable de l'atelier) ainsi que ses collègues pour nous avoir mis à disposition le banc de test moteur ainsi que tout l'outillage et appareils nécessaires afin de réaliser périodiquement les tests de la carte d'injection. Ceux-ci m'ont également beaucoup aidé sur certaines questions liées à l'injection d'essence. Sans ces personnes, mon travail de diplôme n'aurait certainement pas pu aboutir à ce jour.

Je souhaite également remercier toutes les autres personnes non citées dans ce document qui m'ont aidé d'une quelconque façon à réaliser ce travail de diplôme

16 Bibliographies/sources

Sources Internet :

www.megamanual.com: Vous trouverez dans ce site la base de ce travail de diplôme. En effet le travail de diplôme s'est basé sur le système réalisé par les créateurs de ce site. Tout complément d'info non trouvé dans ce rapport ou sur le CD-ROM peuvent se trouver sur ce site. Vous pourrez y trouver les codes sources de bases ainsi que plusieurs informations concernant le système. Ce site possède également un forum dont j'ai eu recourt quelques fois.
http://deacon.chez-alice.fr/inject_ess_p5.html: Ce site contient une explication sur l'injection séquentielle. On peut également y trouver des timing d'injection.
http://www.stealth316.com/2-fuelinjection.htm: Explications sur les systèmes d'injection.
http://www.autoshop101.com/forms/h22.pdf: Explications sur les systèmes d'injection.

Source littéraire :

Présentation de la nouvelle génération de moteurs à 4 cylindres M 271 : Brochure d'introduction pour le Service Après-Vente. DaimlerChrysler AG-Teile-Technik und Technische Information (GSP/SI)-D70546 Stuttgart.

17 Conclusion

Ce travail de diplôme touche maintenant à sa fin. Celui-ci m'a permis de toucher à une variété de domaine électroniques. Ces domaines sont la programmation assembleur, la programmation C, la programmation orienté objet (Visual C++), la réalisation de PCB ainsi que la régulation des moteurs à essence.

En ce qui concerne le système, celui-ci fonctionne bien comme demandé dans le cahier des charges. Les améliorations a apporter sont citées dans le **chapitre 14.** Celles-ci sont minimes. Le système fonctionne correctement sans ces améliorations. Les tests sur le moteur sont concluants. C'est-à-dire qu'à l'aide de mon système j'ai pu démarrer le moteur, le faire tourner avec une injection séquentielle puis faire le calibrage automatique des paramètres. Malheureusement, toute la table n'a pas pu être calibrée. Pourquoi ? Tout simplement car le banc moteur se situe à l'intérieur et n'a pas de système de refroidissement assez puissant. En effet une voiture est refroidie par l'air extérieur qui en roulant refroidit suffisamment le moteur ce qui n'est pas le cas dans un local fermé.

Les points sensibles de ce travail ont été la conception de la carte de façon à pouvoir aller prendre les valeurs de mesures sur le boitier Motronic2.5 du CFP sans pour autant les modifier. Beaucoup de temps a été perdu dans cette partie là. Ce travail contenait beaucoup de programmation à réaliser, domaines peu vu en Power & Control. C'est pourquoi il m'a été difficile au départ de commencer à modifier les codes d'injection pris sur internet. Le dernier problème a été de trouver de la documentation spécifique sur les composants automobiles (exemple : les caractéristiques des injecteurs). En effet sur internet on trouve énormément d'informations mais pas toujours celles dont on a besoin.

Ce travail de diplôme a été réalisé de façon à ne pas restreindre une futur suite. C'est-à-dire que la carte d'injection séquentielle a été conçue de façon à pouvoir lui rajouter d'autres modules par la suite.

Reste maintenant à savoir si l'utilisation de ce système est réellement bénéfique. **Pour démontrer ceci, je dois encore réaliser des tests anti-pollution qui seront joints au CD-ROM.**

18 Annexes

18.1 *Planning*

01/09/2008	Analyse CPU et ses I/O	13/10/2008	Conception d'un menu pour le choix de l'inj. Sequentielle
	Conception nouvelle carte pour inj. Sequentielle		Conception d'un menu pour le choix de l'inj. Sequentielle
	Analyse Banc du CFP		Conception d'un menu pour le choix de l'inj. Sequentielle
	Conception cablage pour banc CFP		Conception d'un menu pour le choix de l'inj. Sequentielle
05/09/2008	1er branchement ancienne carte au CFP	17/10/2008	Test injection sequentielle au CFP
08/09/2008	Conception nouvelle carte pour inj. Sequentielle	20/10/2008	Conception du calibrage automatique
	Conception nouvelle carte pour inj. Sequentielle		Conception du calibrage automatique
	Routage nouvelle carte pour inj. Sequentielle		Conception du calibrage automatique
	Réalisation Cablage pour banc CFP		Conception du calibrage automatique
13/09/2008	Paramétrage ancienne carte au CFP	24/10/2008	Premiers tests du calibrage automatique au CFP
15/09/2008	Routage nouvelle carte pour inj. Sequentielle	27/10/2008	Conception du calibrage automatique
	Routage nouvelle carte pour inj. Sequentielle		Conception du calibrage automatique
	Routage nouvelle carte pour inj. Sequentielle		Conception du calibrage automatique
	Tests nouvelle carte pour inj.Sequentielle		Conception du calibrage automatique
19/09/2008	Paramétrage ancienne carte au CFP	31/10/2008	Tests du calibrage automatique au CFP
22/09/2008	Programmation injection séquentielle	03/11/2008	Conception d'un menu pour le choix de capteur d'arbre a came
	Programmation injection séquentielle		Conception d'un menu pour le choix de capteur d'arbre a came
	Programmation injection séquentielle		Conception d'un menu pour le choix de capteur d'arbre a came
	Programmation injection séquentielle		Conception d'un menu pour le choix de capteur d'arbre a came
26/09/2008	Paramétrage ancienne carte au CFP+ Capteur arbre a came	07/11/2008	Tests du calibrage automatique au CFP
29/09/2008	Programmation injection séquentielle(Capteur arbre a came)	10/11/2008	Améliorations Calibrage automatique
	Programmation injection séquentielle(Capteur arbre a came)		Améliorations Calibrage automatique
	Programmation injection séquentielle(Capteur arbre a came)		Améliorations Calibrage automatique
	Programmation injection séquentielle(Capteur arbre a came)		Améliorations Calibrage automatique
03/10/2008	Test injection sequentielle au CFP	14/11/2008	Tests du calibrage automatique au CFP
06/10/2008	Analyse Software Megatune(Visual Basic)	17/11/2008	Rapport/Améliorations/Mesures finales
	Analyse Software Megatune(Visual Basic)		Rapport/Améliorations/Mesures finales
	Analyse Software Megatune(Visual Basic)		Rapport/Améliorations/Mesures finales
10/10/2008	Analyse Software Megatune(Visual Basic)	21/11/2008	Rapport/Améliorations/Mesures finales

18.2 *Carte d'injection séquentielle*

18.2.1 Attribution des entrées-sorties du uC maître

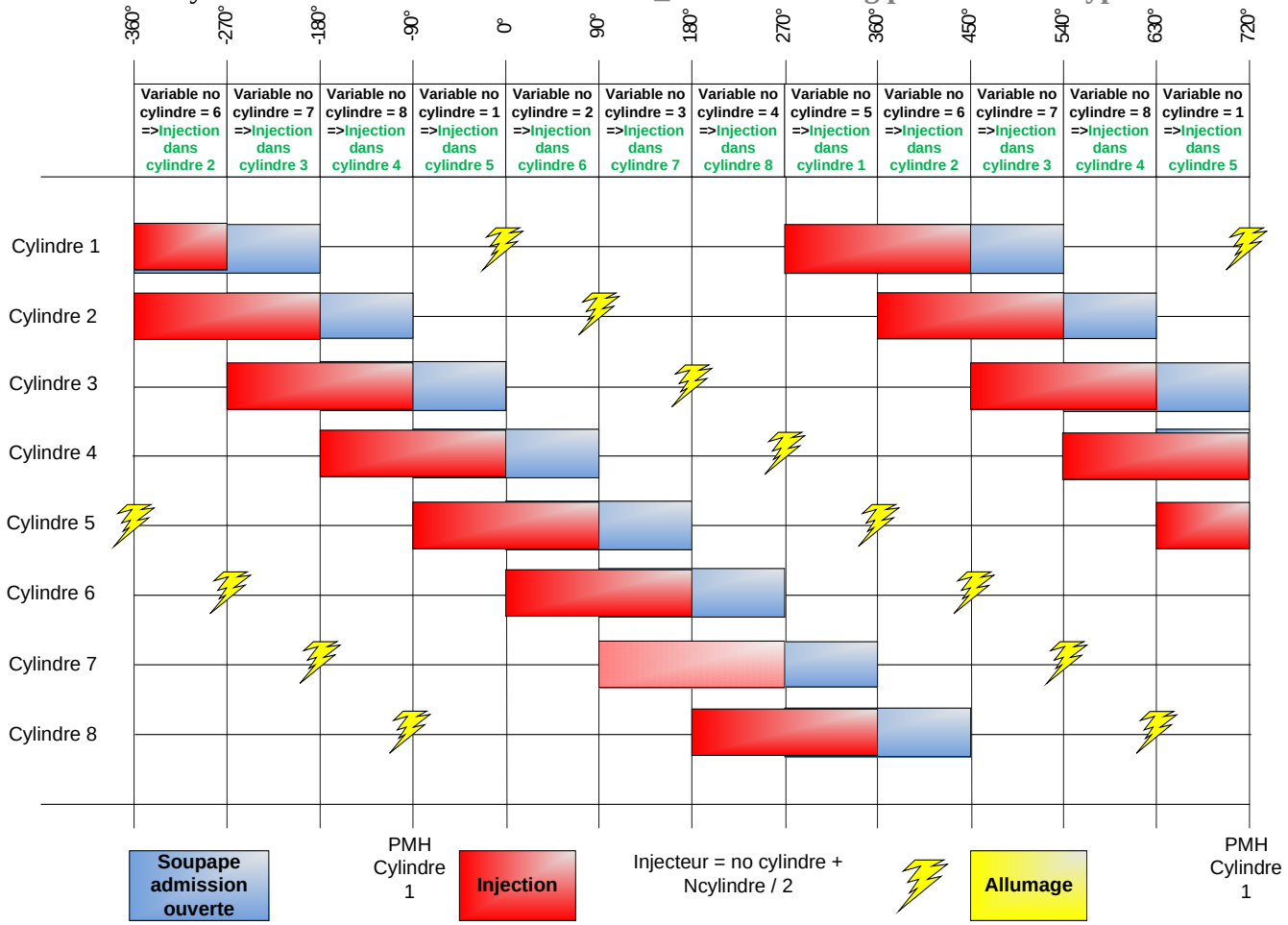
Pin	Nom	Fonction	Label	Fonction
1	PW0/IOC0/PT0	PWM / **TIMER** / Port T I/O pin	VIL_PULSE	Entrée signal vilebrequin
2	PW1/IOC1/PT1	PWM / **TIMER** / Port T I/O pin	CAME_PULSE	Entré signal arbre à came
3	PW2/IOC2/PT2	PWM / **TIMER** / Port T I/O pin	INJ1_COM	Sortie commande injecteur 1
4	PW3/IOC3/PT3	PWM / **TIMER** / Port T I/O pin	INJ2_COM	Sortie commande injecteur 2
5	VDD1	**Power and Ground Pins for I/O Drivers**	-	-
6	VSS1		-	-
7	IOC4/PT4	**TIMER** / Port T I/O pin	INJ3_COM	Sortie commande injecteur 3
8	IOC5/PT5	**TIMER** / Port T I/O pin	IGN	Sortie commande allumage
9	IOC6/PT6	**TIMER** / Port T I/O pin	INJ4_COM	Sortie commande injecteur 4
10	IOC7/PT7	TIMER / Port T I/O pin	SPARE1	**Libre**
11	MODC/BKGD	**Background debug mode pin**	-	BDM
12	PB4	**Port B I/O pin**	SYNCHRO	Synchro injection avec uC2
13	XCLKS/PE7	**Access clock select** / Port E I/O pin	DGND	-
14	ECLK/PE4	Bus clock output / Port E I/O pin	SPARE2	**Libre**
15	VSSR	Power and Ground Pins for I/O Drivers	-	-
16	VDDR	and for Internal Voltage Regulator	-	-
17	RESET	External reset pin	-	BDM
18	VDDPLL	Power Supply Pins for PLL	-	-
19	XFC	PLL loop filter pin	-	-
20	VSSPLL	Power Supply Pins for PLL	-	-
21	EXTAL	Oscillator pin	OSC1	Oscillateur
22	XTAL	Oscillator pin	OSC2	Oscillateur
23	VPP/TEST	Test pin only	-	-
24	IRQ/PE1	External interrupt pin / Port E input	SPARE3	**Libre**
25	XIRQ/PE0	Non-maskable interrupt pin / Port E input	SPARE4	**Libre**
26	PA0	Port A I/O pin	SPARE5	**Libre**
27	PAD00/AN00	Port AD I/O pin / **ATD input**	MAF/MAP	Entrée signal débit/pression air
28	PAD01/AN01	Port AD I/O pin / **ATD input**	MAT	Entrée signal T°air
29	PAD02/AN02	Port AD I/O pin / **ATD input**	CLT	Entrée signal T°moteur
30	PAD03/AN03	Port AD I/O pin / **ATD input**	TPS	Entrée signal papillon des gaz
31	PAD04/AN04	Port AD I/O pin / **ATD input**	BATT	Entrée signal batterie
32	PAD05/AN05	Port AD I/O pin / **ATD input**	O2	Entrée signal sonde lambda
33	PAD06/AN06	Port AD I/O pin / ATD input	SPARE6	**Libre**
34	PAD07/AN07	Port AD I/O pin / ATD input	SPARE7	**Libre**
35	VDDA	Power Supply Pins for ATD and VREG	-	-
36	VRH	ATD Reference Voltage Input Pins	-	-
37	VSSA	Power Supply Pins for ATD and VREG	-	-
38	PS0/RXD	Port S I/O pin / **SCI receive signal**	RX	Entrée comm. Série RX
39	PS1/TXD	Port S I/O pin / **SCI transmit signal**	TX	Sortie comm. Série TX
40	PM5/SCK	**Port M I/O pin** / SPI SCK signal	-	WarmLed
41	PM4/MOSI	**Port M I/O pin** / SPI MOSI signal	-	AccLed
42	PM3/SS	Port M I/O pin / SPI SS signal	SPARE8	**Libre**
43	PM2/MISO	Port M I/O pin / SPI MISO signal	SPARE9	**Libre**
44	PM1/TXCAN	Port M I/O pin / **CAN transmit signal**	CAN1TX	Sortie comm. Can bus transmit
45	PM0/RXCAN	Port M I/O pin / **CAN receive signal**	CAN1RX	Entrée comm. Can bus receive
46	VSSX	Power and Ground Pins for I/O Drivers	-	-
47	VDDX		-	-
48	PP5/KWP5/PW5	Port P I/O pin / keypad wake-up/PWM output	SPARE10	**Libre**

18.2.2 Attribution des entrées-sorties du uC esclave

Pin	Nom	Fonction	Label	Fonction
1	PW0/IOC0/PT0	PWM / **TIMER** / Port T I/O pin	INJ5_COM	Sortie commande injecteur 5
2	PW1/IOC1/PT1	PWM / **TIMER** / Port T I/O pin	INJ6_COM	Sortie commande injecteur 6
3	PW2/IOC2/PT2	PWM / **TIMER** / Port T I/O pin	INJ7_COM	Sortie commande injecteur 7
4	PW3/IOC3/PT3	PWM / **TIMER** / Port T I/O pin	INJ8_COM	Sortie commande injecteur 8
5	VDD1	**Power and Ground Pins for I/O Drivers**	-	-
6	VSS1		-	-
7	IOC4/PT4	**TIMER** / Port T I/O pin	INJ9_COM	Sortie commande injecteur 9
8	IOC5/PT5	**TIMER** / Port T I/O pin	INJ10_COM	Sortie commande injecteur 10
9	IOC6/PT6	**TIMER** / Port T I/O pin	INJ11_COM	Sortie commande injecteur 11
10	IOC7/PT7	**TIMER** / Port T I/O pin	INJ12_COM	Sortie commande injecteur 12
11	MODC/BKGD	**Background debug mode pin**	-	BDM
12	PB4	**Port B I/O pin**	SPARE11	**Libre**
13	XCLKS/PE7	**Access clock select** / Port E I/O pin	DGND	-
14	ECLK/PE4	Bus clock output / Port E I/O pin	SPARE12	**Libre**
15	VSSR	Power and Ground Pins for I/O Drivers	-	-
16	VDDR	and for Internal Voltage Regulator	-	-
17	RESET	External reset pin	-	BDM
18	VDDPLL	Power Supply Pins for PLL	-	-
19	XFC	PLL loop filter pin	-	-
20	VSSPLL	Power Supply Pins for PLL	-	-
21	EXTAL	Oscillator pin	OSC1	Oscillateur
22	XTAL	Oscillator pin	OSC2	Oscillateur
23	VPP/TEST	Test pin only	-	-
24	IRQ/PE1	**External interrupt pin** / Port E input	SYNCHRO	Synchro injection avec uC2
25	XIRQ/PE0	Non-maskable interrupt pin / Port E input	SPARE13	**Libre**
26	PA0	Port A I/O pin	SPARE14	**Libre**
27	PAD00/AN00	Port AD I/O pin / ATD input	SPARE15	**Libre**
28	PAD01/AN01	Port AD I/O pin / ATD input	SPARE16	**Libre**
29	PAD02/AN02	Port AD I/O pin / ATD input	SPARE17	**Libre**
30	PAD03/AN03	Port AD I/O pin / ATD input	SPARE18	**Libre**
31	PAD04/AN04	Port AD I/O pin / ATD input	SPARE19	**Libre**
32	PAD05/AN05	Port AD I/O pin / ATD input	SPARE20	**Libre**
33	PAD06/AN06	Port AD I/O pin / ATD input	SPARE21	**Libre**
34	PAD07/AN07	Port AD I/O pin / ATD input	SPARE22	**Libre**
35	VDDA	Power Supply Pins for ATD and VREG	-	-
36	VRH	ATD Reference Voltage Input Pins	-	-
37	VSSA	Power Supply Pins for ATD and VREG	-	-
38	PS0/RXD	Port S I/O pin / SCI receive signal	SPARE23	**Libre**
39	PS1/TXD	Port S I/O pin / SCI transmit signal	SPARE24	**Libre**
40	PM5/SCK	Port M I/O pin / SPI SCK signal	SPARE25	**Libre**
41	PM4/MOSI	Port M I/O pin / SPI SCK signal	SPARE26	**Libre**
42	PM3/SS	Port M I/O pin / SPI SS signal	SPARE27	**Libre**
43	PM2/MISO	Port M I/O pin / SPI MISO signal	SPARE28	**Libre**
44	PM1/TXCAN	Port M I/O pin / **CAN transmit signal**	CAN2TX	Sortie comm. Can bus transmit
45	PM0/RXCAN	Port M I/O pin / **CAN receive signal**	CAN2RX	Entrée comm. Can bus receive
46	VSSX	Power and Ground Pins for I/O Drivers	-	-
47	VDDX		-	-
48	PP5/KWP5/PW5	Port P I/O pin / keypad wake-up/PWM output	SPARE29	**Libre**

18.3 *Timing d'injection*

Voici les timing pour un moteur 8 cylindres. **Vous trouverez dans le CD_ROM les timing pour d'autres types de moteurs.**



18.4 *Structogrammes code uC maître*

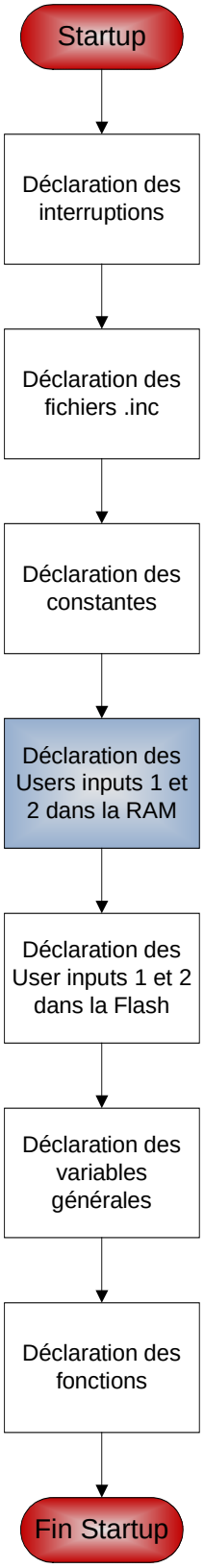
18.4.1 Code Global

* Les fonctions munies de cette étoile ne sont pas commentées car elles n'ont pas été étudiés dans le cas de la modification de l'injection

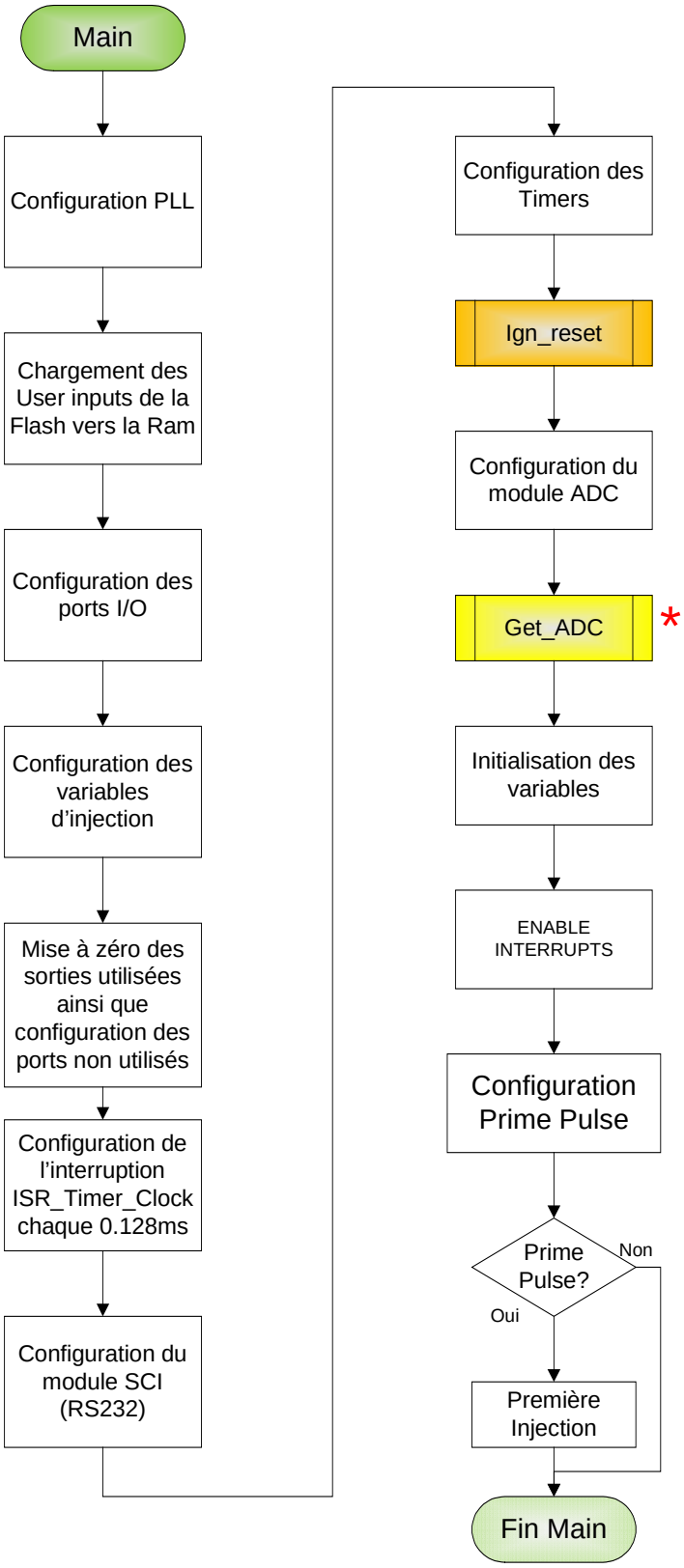
** Les fonctions munies de deux étoiles sont des options d'injections ou d'allumage. Celles-ci ne sont pas commentée car elles n'ont pas été étudiées dans le cas de la modification de l'injection. Par contre, un explicatif des ces options se trouvent dans la partie Software PC

Ces structogrammes sont réalisés de manière à avoir une vue globale du code. Des informations plus détaillées se trouve dans le rapport

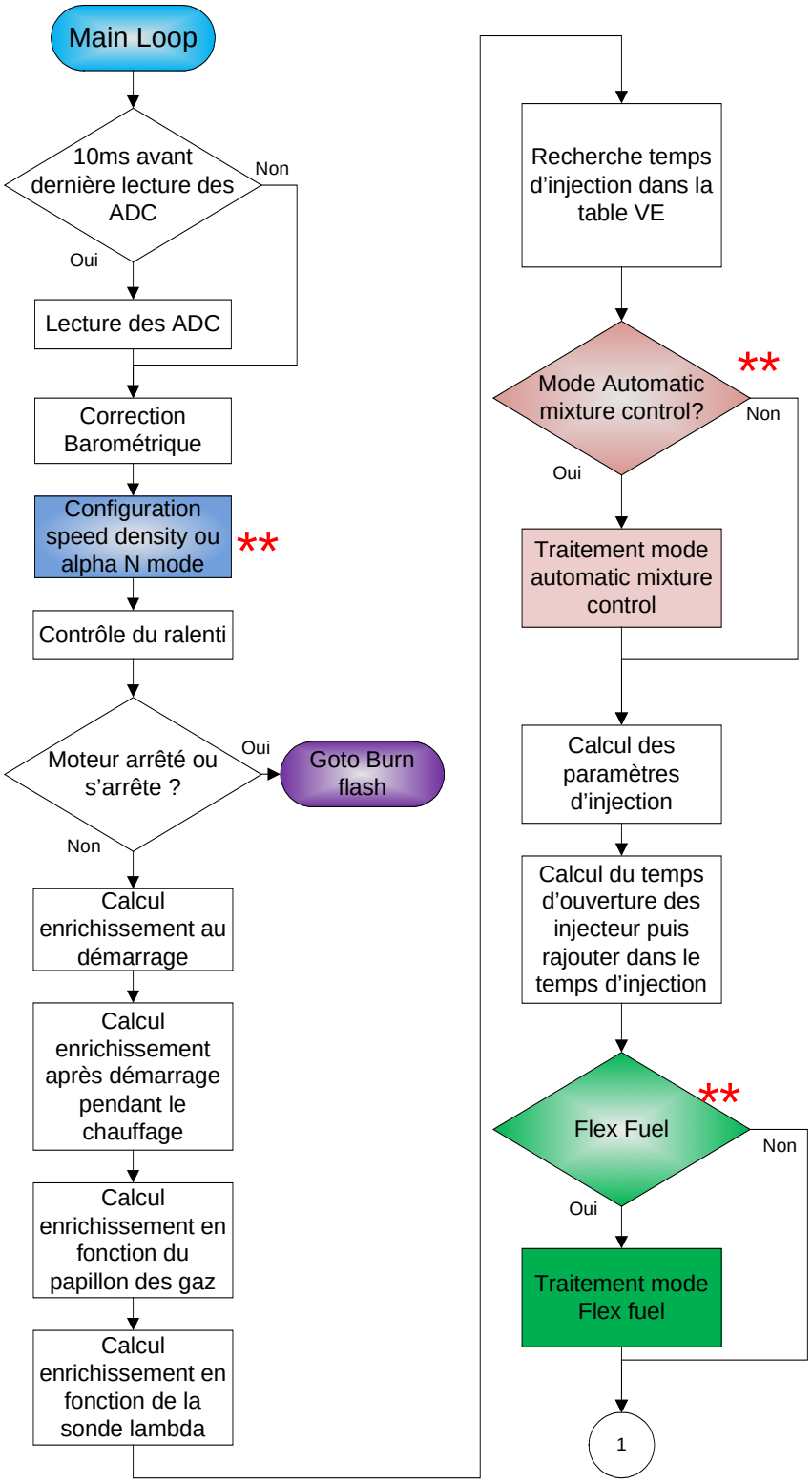
18.4.2 Startup

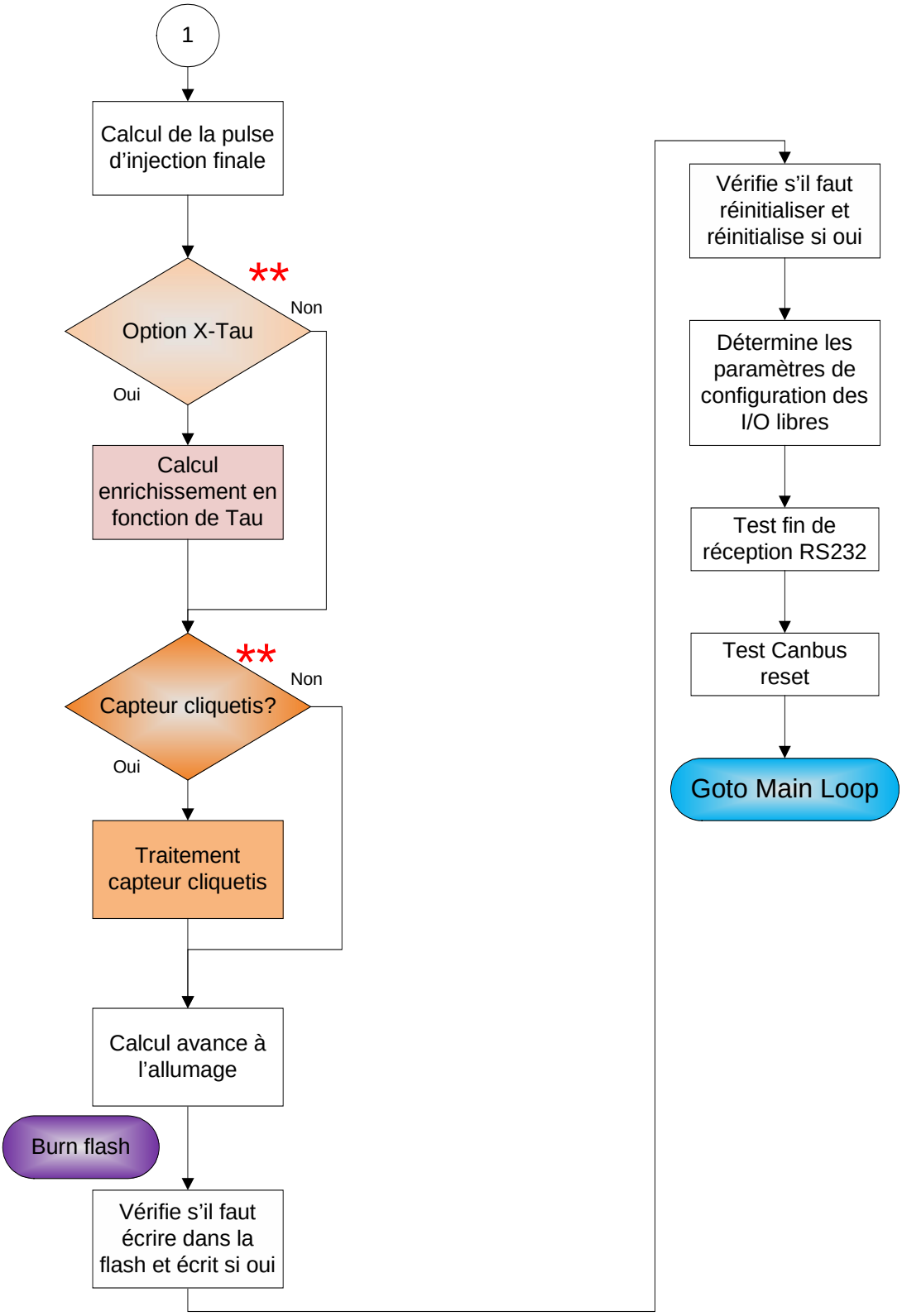


18.4.3 Main

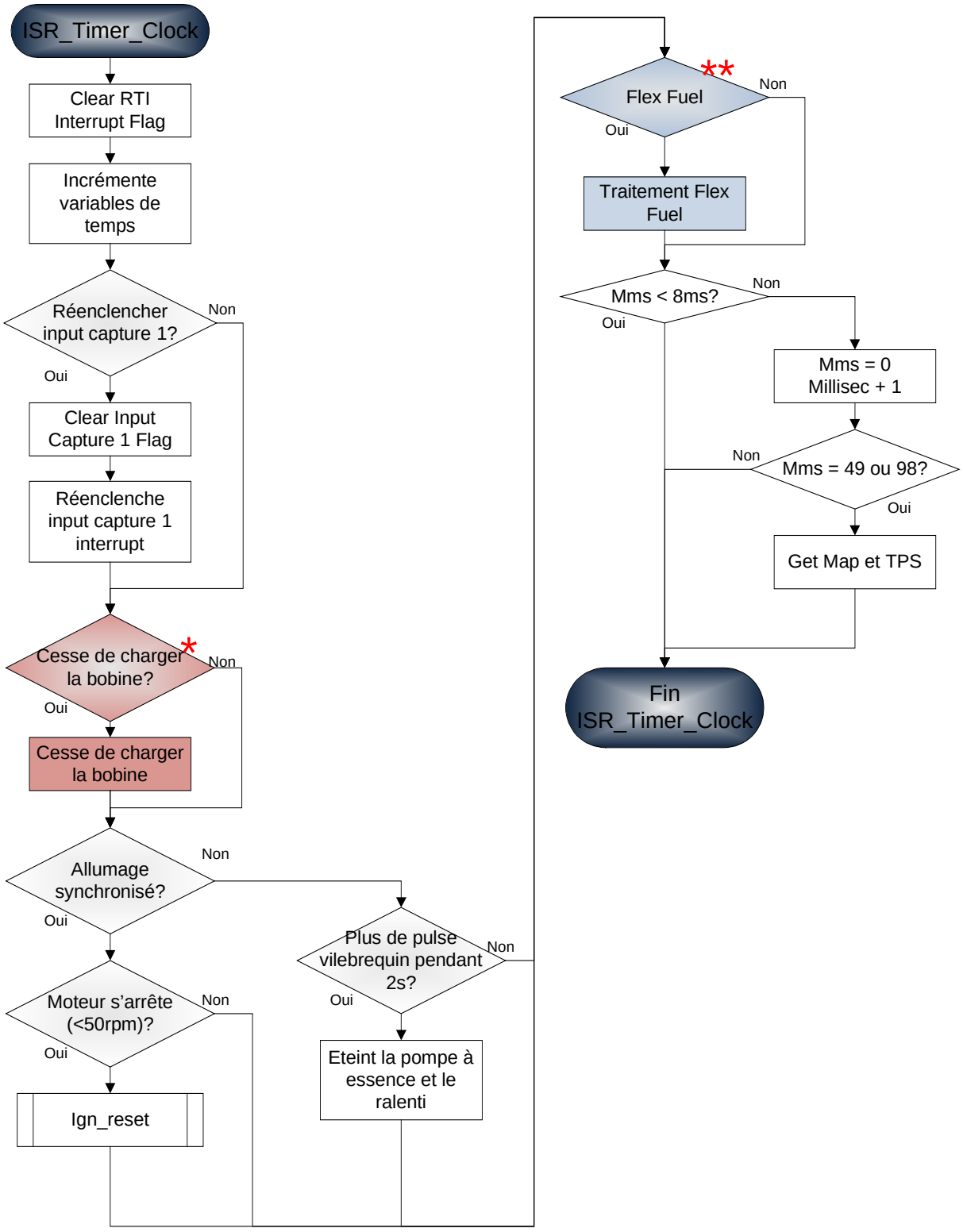

Main
Configuration PLL
Chargement des User inputs de la Flash vers la Ram
Configuration des ports I/O
Configuration des variables d'injection
Mise à zéro des sorties utilisées ainsi que configuration des ports non utilisés
Configuration de l'interruption ISR_Timer_Clock chaque 0.128ms
Configuration du module SCI (RS232)
Configuration des Timers
Ign_reset
Configuration du module ADC
Get_ADC
*
Initialisation des variables
ENABLE INTERRUPTS
Configuration Prime Pulse
Prime Pulse?
Non
Oui
Première Injection
Fin Main

18.4.4 Main Loop

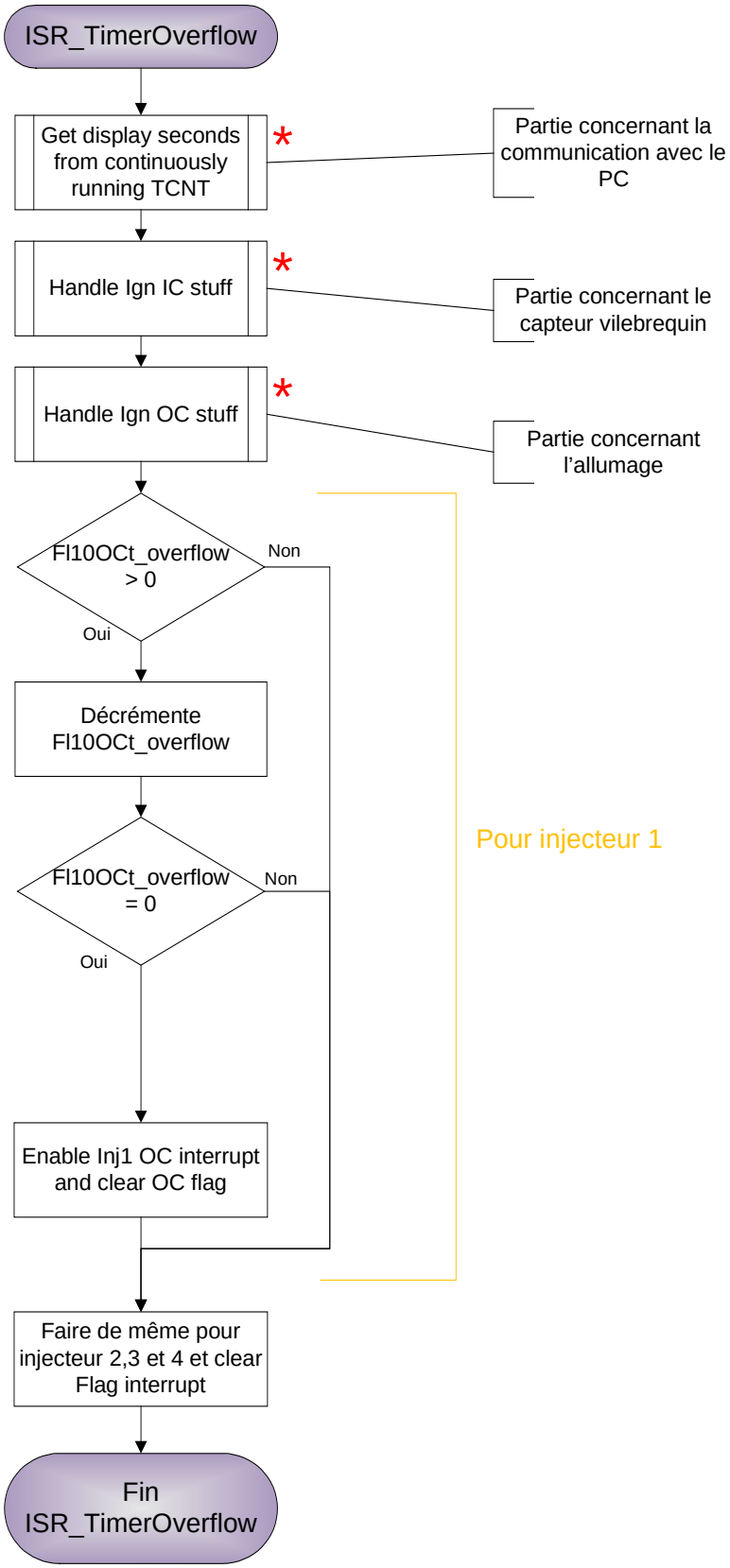



1
Calcul de la pulse d'injection finale
**
Option X-Tau
Non
Oui
Calcul enrichissement en fonction de Tau
**
Capteur cliquetis?
Non
Oui
Traitement capteur cliquetis
Calcul avance à l'allumage
Burn flash
Vérifie s'il faut écrire dans la flash et écrit si oui
Vérifie s'il faut réinitialiser et réinitialise si oui
Détermine les paramètres de configuration des I/O libres
Test fin de réception RS232
Test Canbus reset
Goto Main Loop

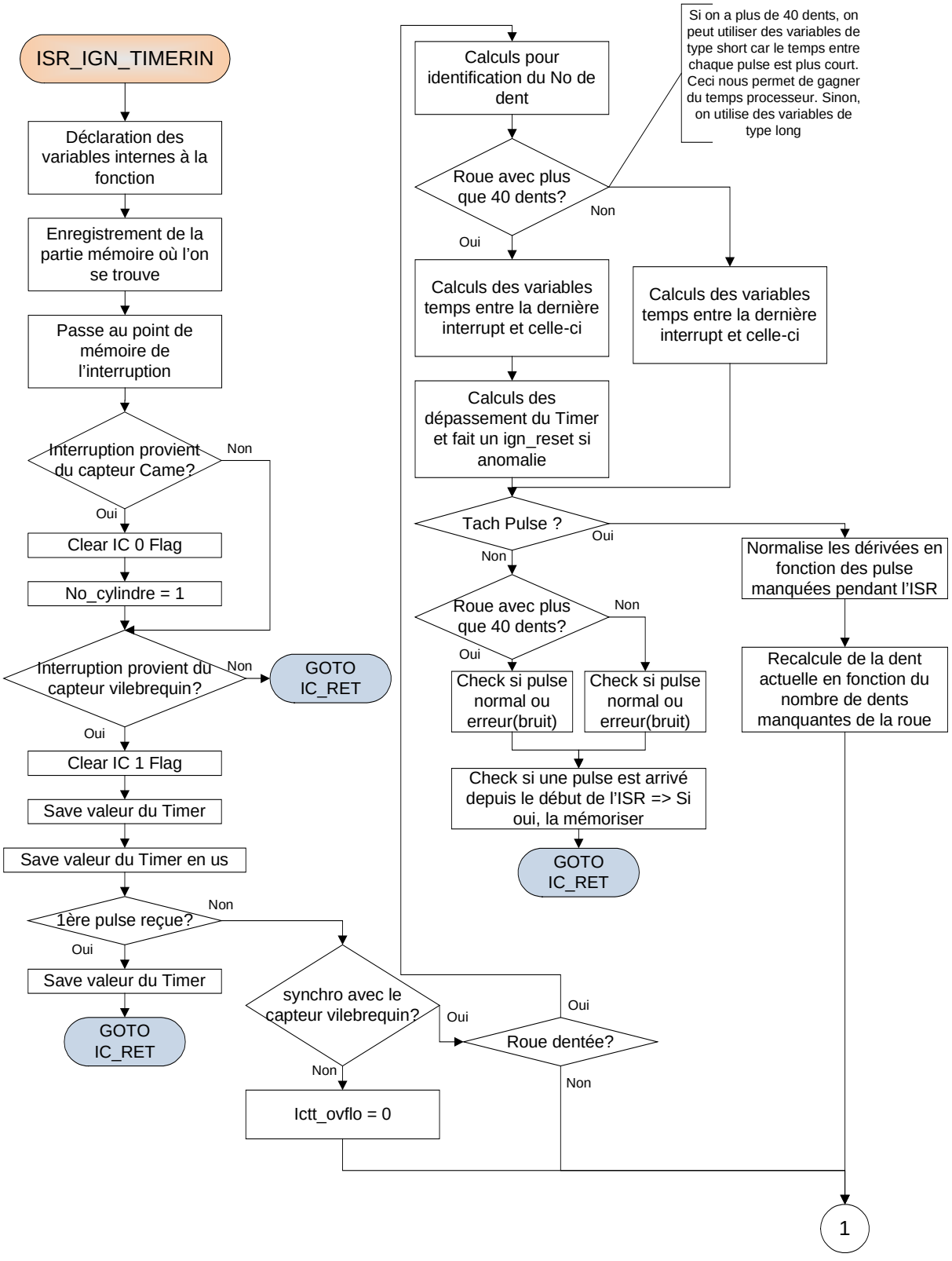
18.4.5 ISR_Timer_Clock

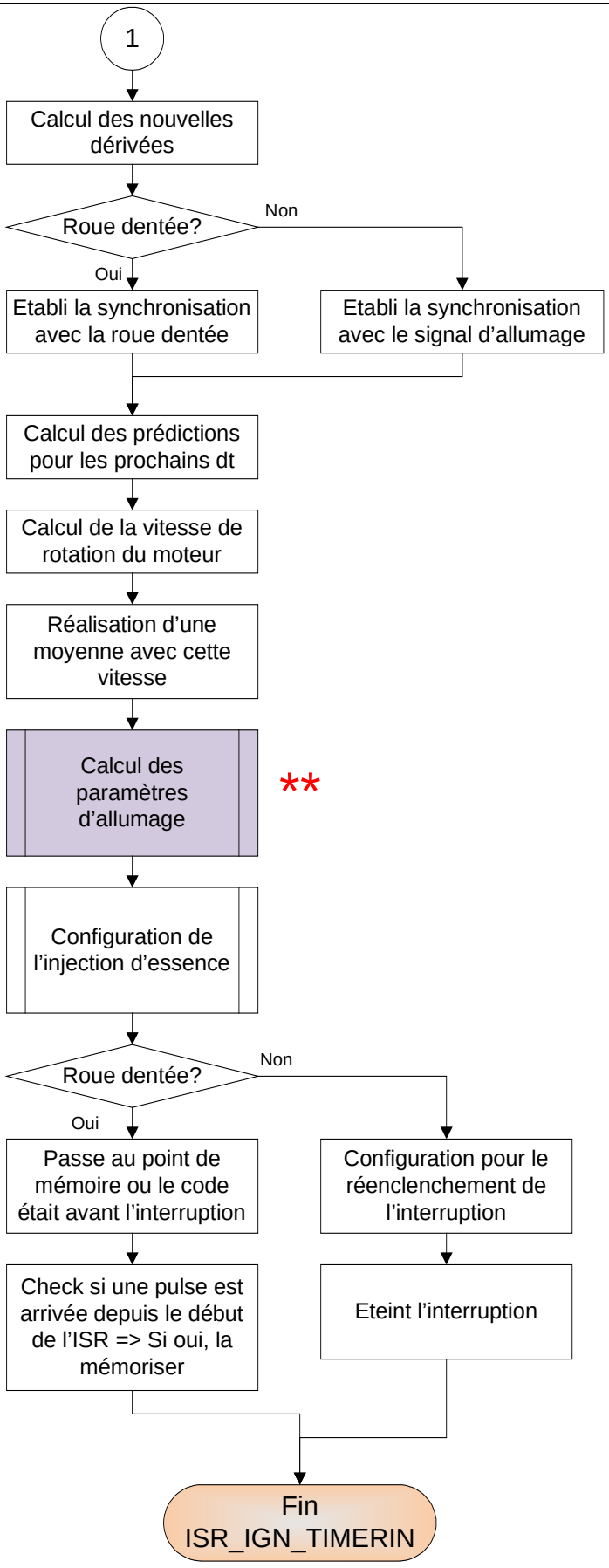


18.4.6 ISR_TimerOverflow



18.4.7 ISR_Ign_TimerIn




1
Calcul des nouvelles dérivées
Roue dentée?
Non
Oui
Etabli la synchronisation avec la roue dentée
Etabli la synchronisation avec le signal d'allumage
Calcul des prédictions pour les prochains dt
Calcul de la vitesse de rotation du moteur
Réalisation d'une moyenne avec cette vitesse
Calcul des paramètres d'allumage
**
Configuration de l'injection d'essence
Roue dentée?
Non
Oui
Passe au point de mémoire ou le code était avant l'interruption
Configuration pour le réenclenchement de l'interruption
Check si une pulse est arrivée depuis le début de l'ISR => Si oui, la mémoriser
Eteint l'interruption
Fin ISR_IGN_TIMERIN

18.4.8 Configuration de l'injection d'essence

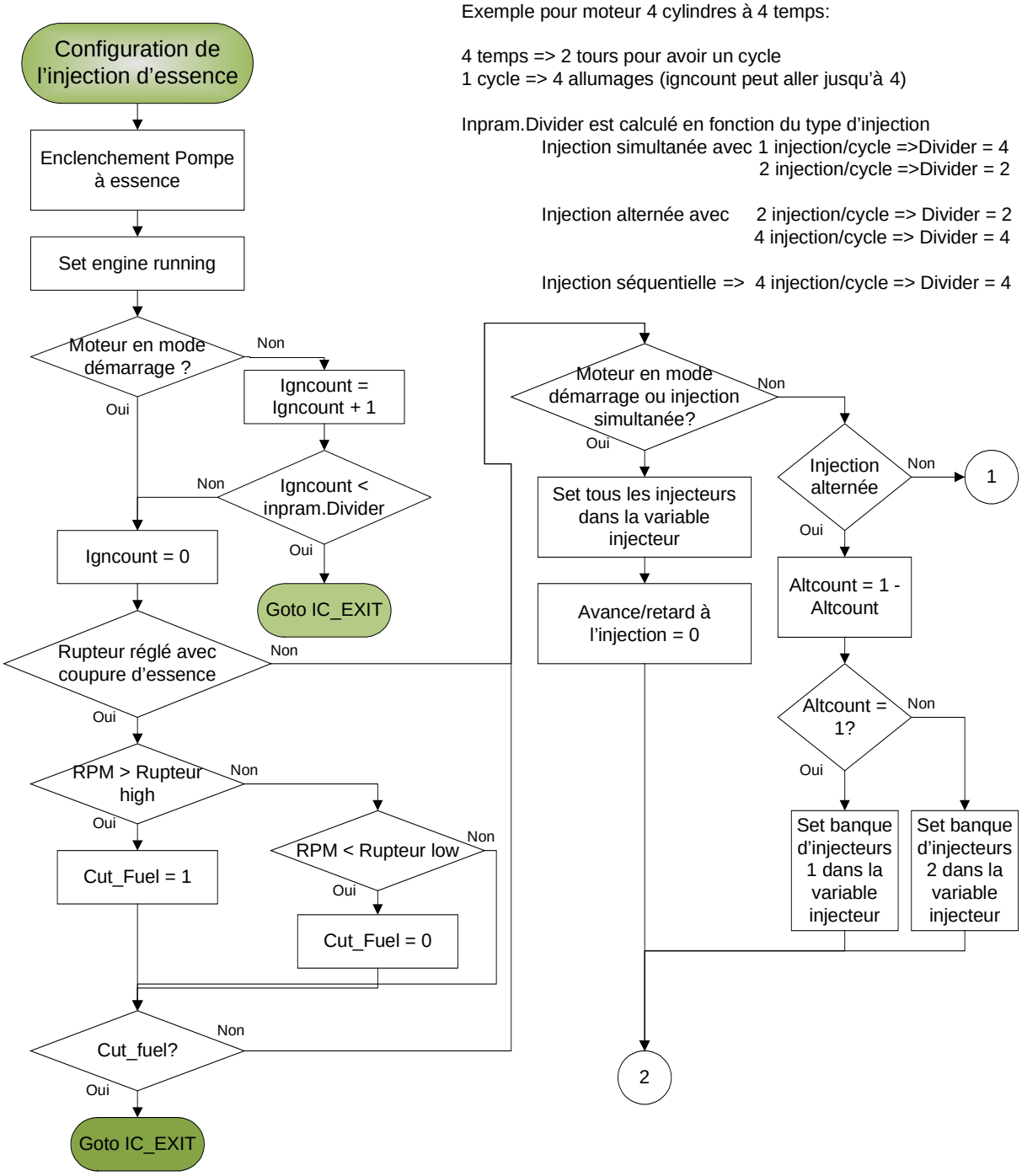
Exemple pour moteur 4 cylindres à 4 temps:

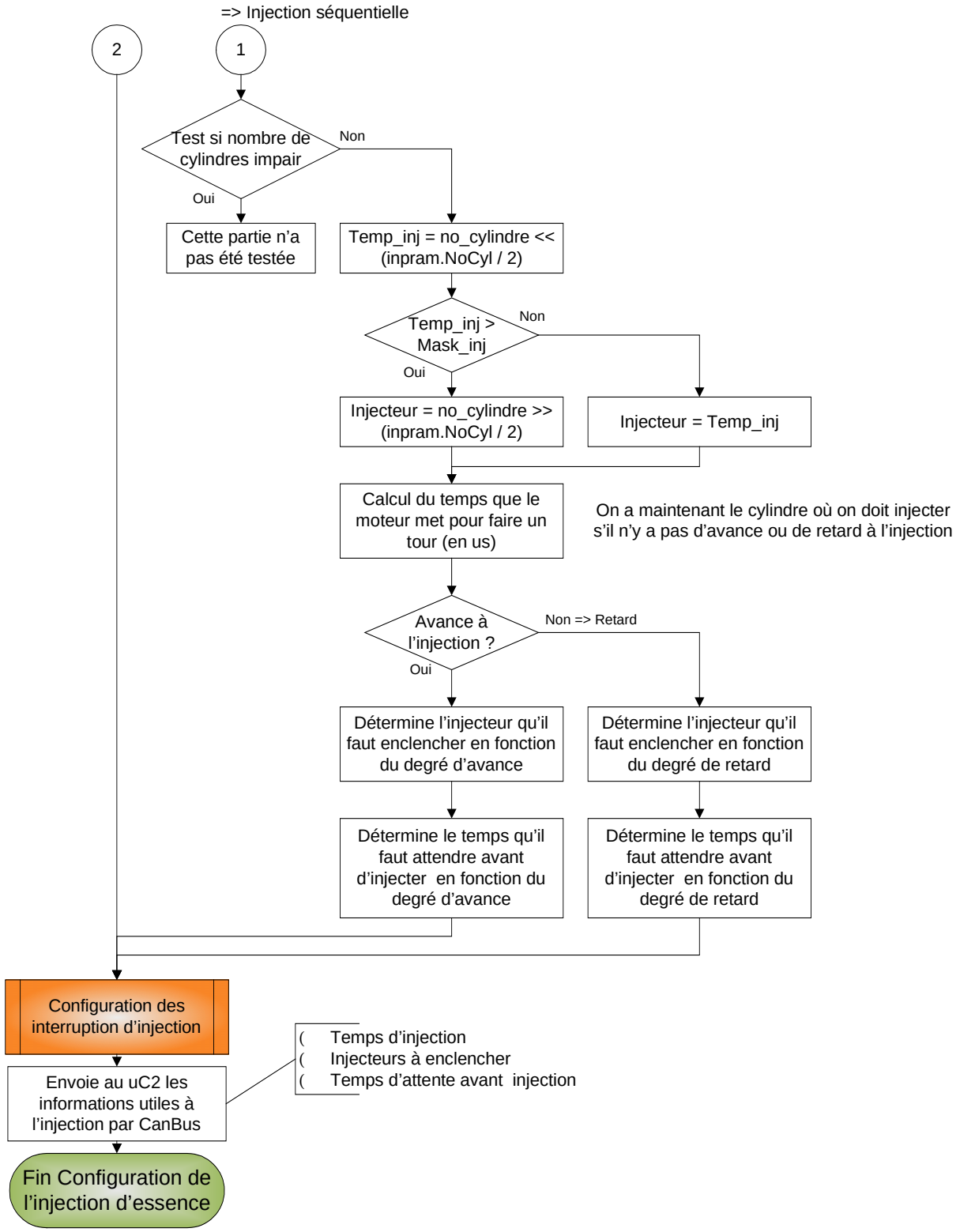
4 temps => 2 tours pour avoir un cycle
1 cycle => 4 allumages (igncount peut aller jusqu'à 4)

Inpram.Divider est calculé en fonction du type d'injection
Injection simultanée avec 1 injection/cycle =>Divider = 4
2 injection/cycle =>Divider = 2

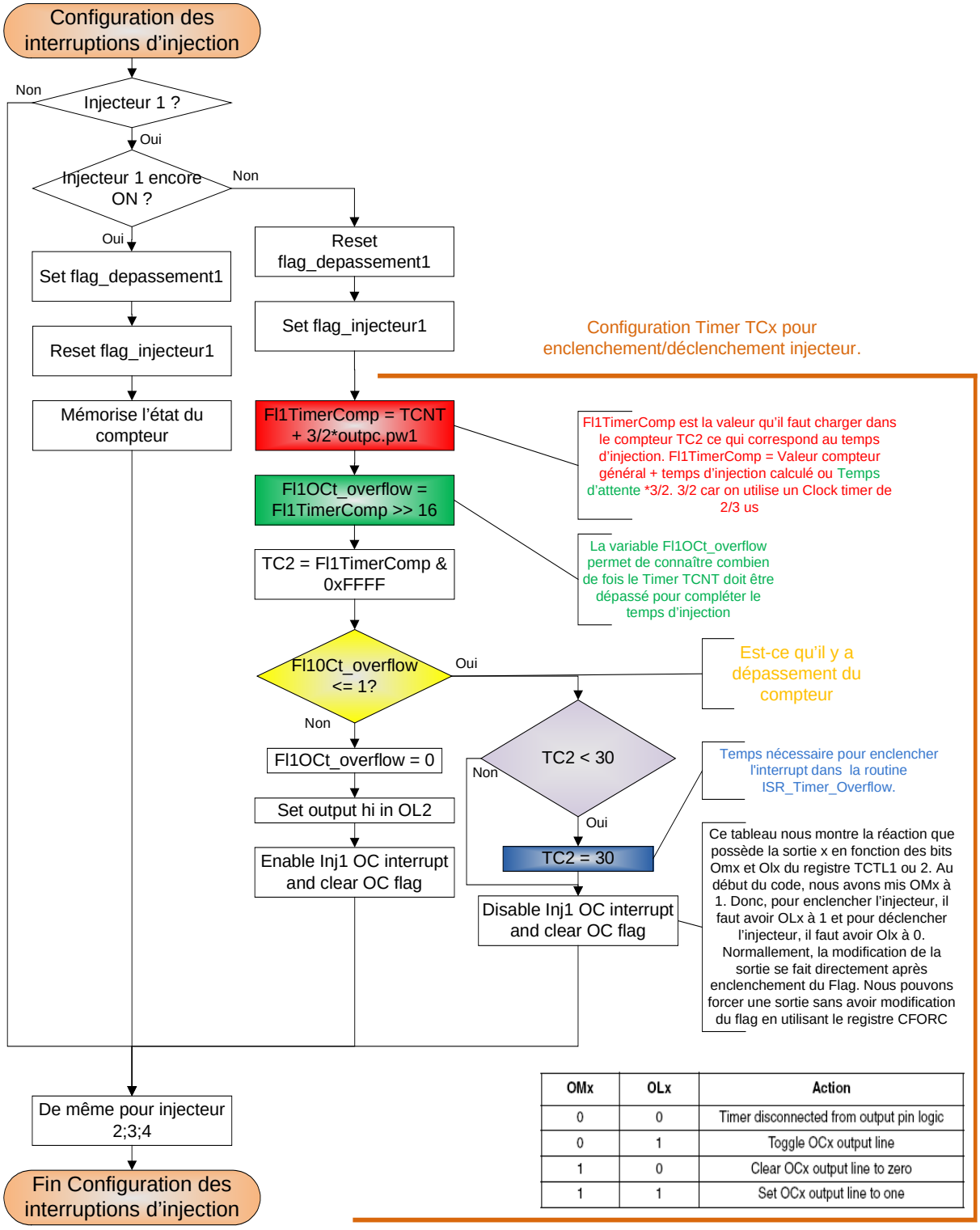
Injection alternée avec 2 injection/cycle => Divider = 2
4 injection/cycle => Divider = 4

Injection séquentielle => 4 injection/cycle => Divider = 4



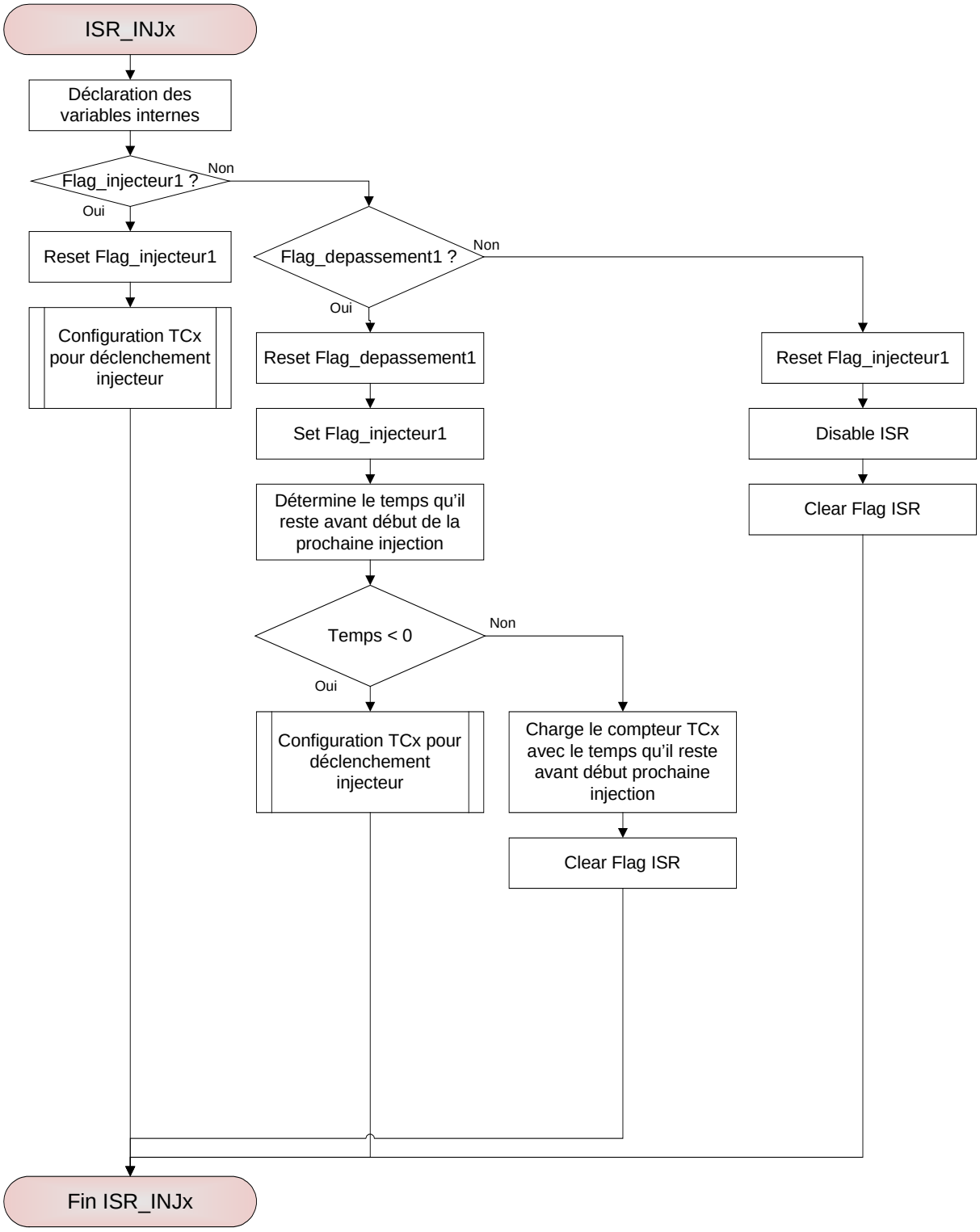

=> Injection séquentielle
2
1
Test si nombre de cylindres impair
Non
Oui
Cette partie n'a pas été testée
Temp_inj = no_cylindre << (inpram.NoCyl / 2)
Temp_inj > Mask_inj
Non
Oui
Injecteur = no_cylindre >> (inpram.NoCyl / 2)
Injecteur = Temp_inj
Calcul du temps que le moteur met pour faire un tour (en us)
On a maintenant le cylindre où on doit injecter s'il n'y a pas d'avance ou de retard à l'injection
Avance à l'injection ?
Non => Retard
Oui
Détermine l'injecteur qu'il faut enclencher en fonction du degré d'avance
Détermine l'injecteur qu'il faut enclencher en fonction du degré de retard
Détermine le temps qu'il faut attendre avant d'injecter en fonction du degré d'avance
Détermine le temps qu'il faut attendre avant d'injecter en fonction du degré de retard
Configuration des interruption d'injection
Envoie au uC2 les informations utiles à l'injection par CanBus
(Temps d'injection
(Injecteurs à enclencher
(Temps d'attente avant injection
Fin Configuration de l'injection d'essence

18.4.9 Configuration des interruptions d'injection

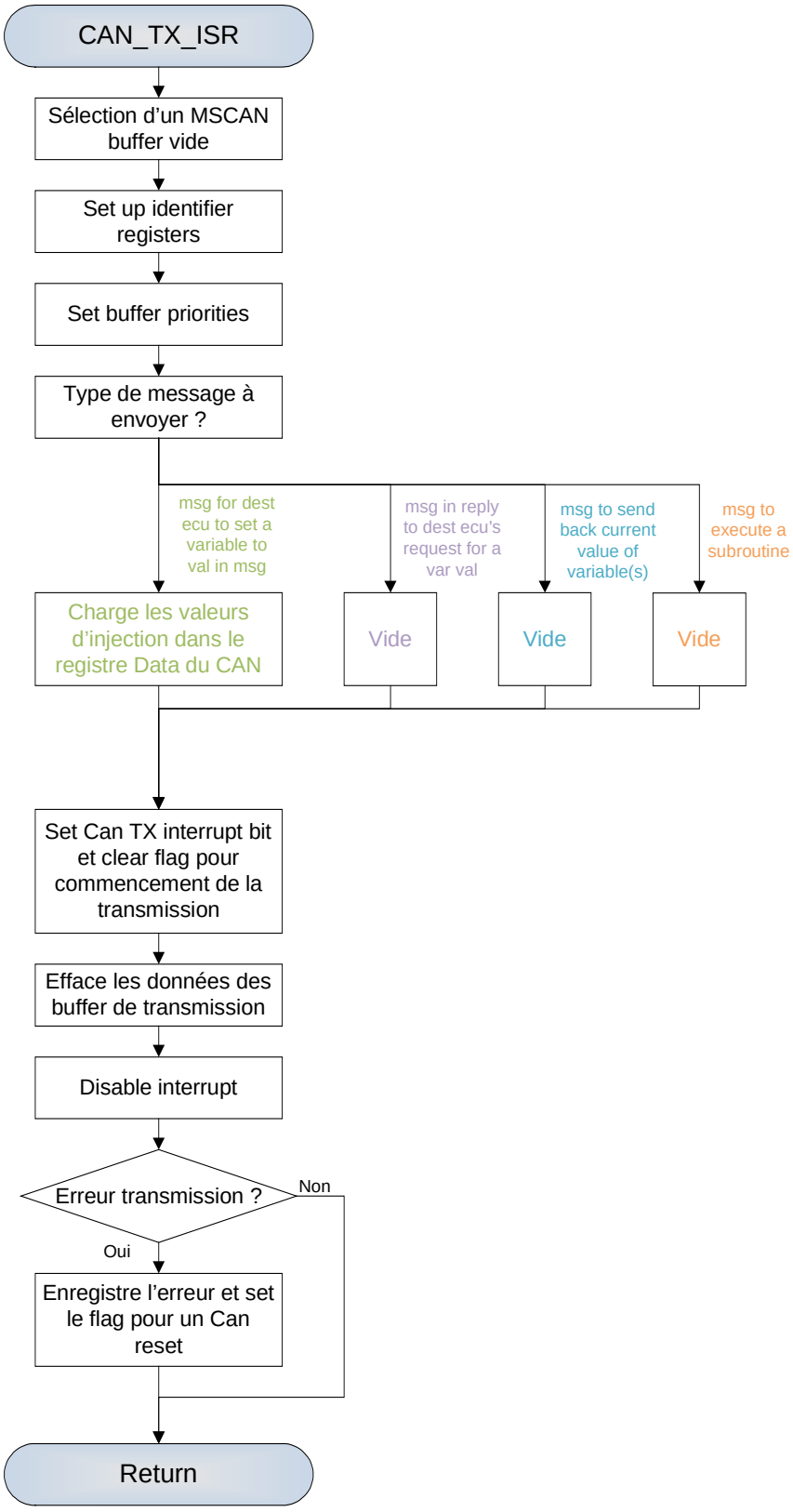


OMx	OLx	Action
0	0	Timer disconnected from output pin logic
0	1	Toggle OCx output line
1	0	Clear OCx output line to zero
1	1	Set OCx output line to one

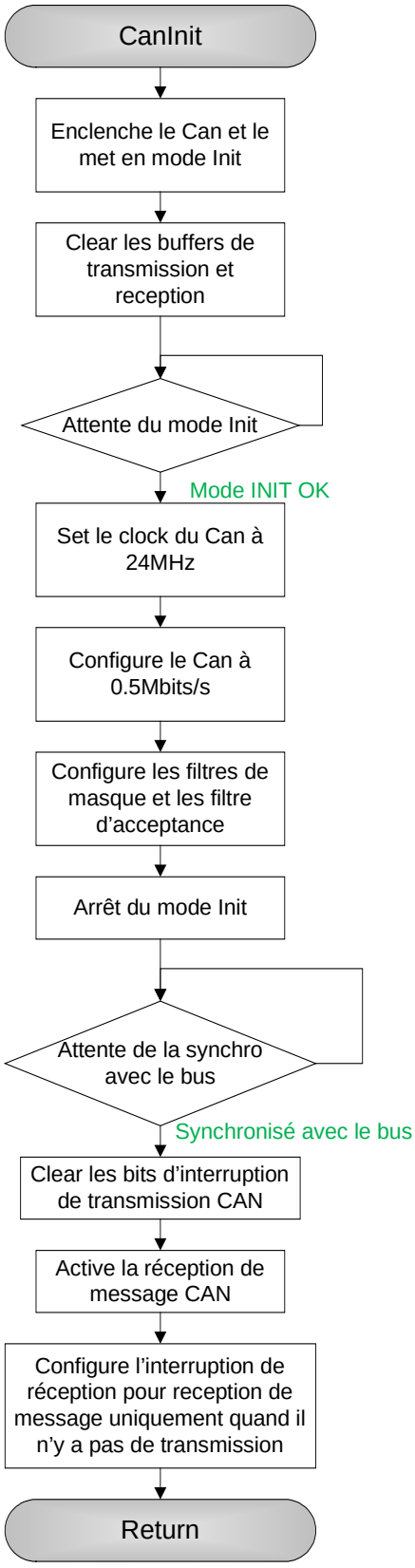
18.4.10 ISR_InjX_TimerOut



18.4.11 Can_Tx_ISR



18.4.12 CanInit



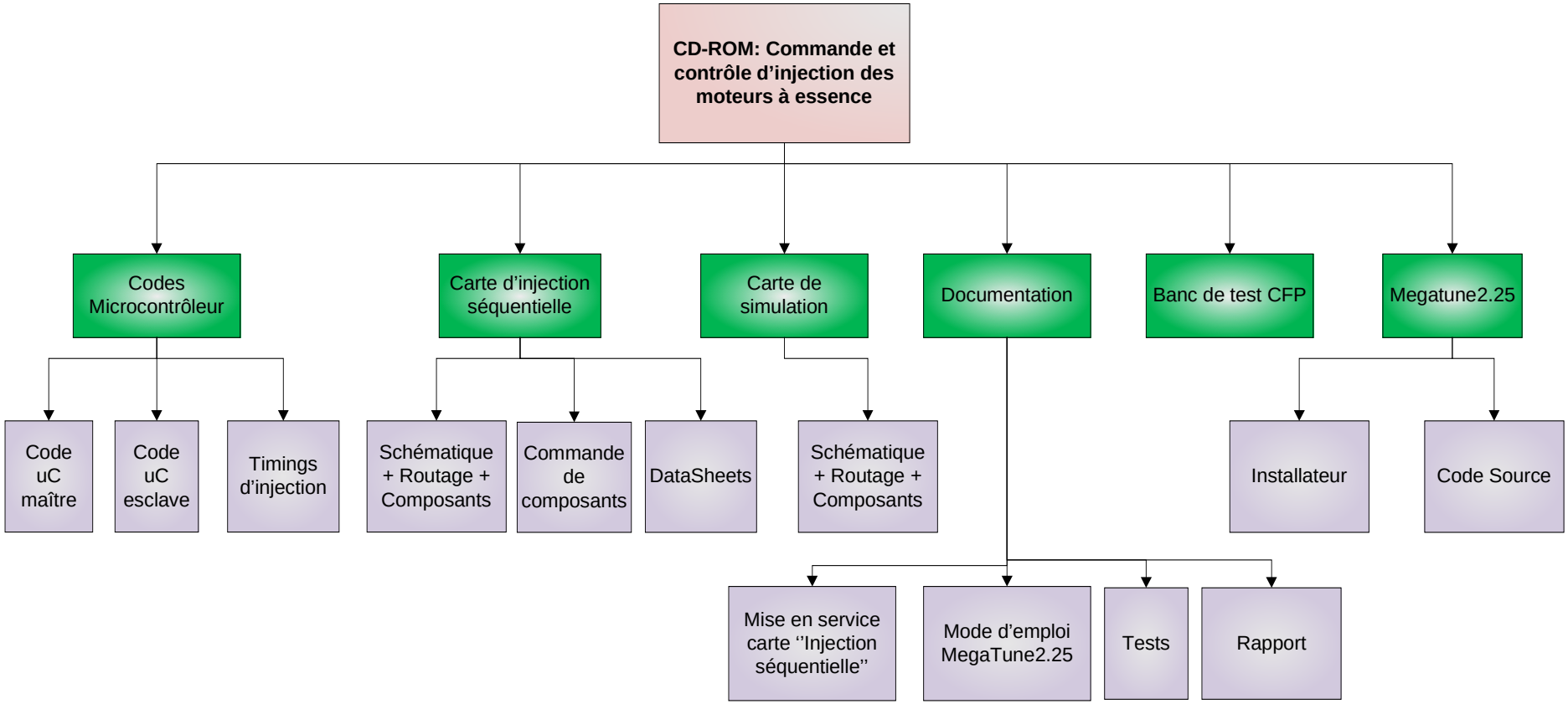
18.5 *Tests*

18.5.1 Test général

Voici les tests effectuées pour un moteur 8 cylindres. **Les tests pour les autres types de moteurs se trouvent dans le CD-ROM.**

Test Injection moteur 8 cylindres					
Pour tous les tests => RPM de 0 à 8000tr/min avec rupteur(coupure injection) à 6000tr/min					
Simultanée		Alternée		Séquentielle(Nb inj/cy:4)	
Nb injections/cycle	OK/NOK	Nb injections/cycle	OK/NOK	Angle retard/avance	OK/NOK
1	OK	2	OK	-360°	OK
2	OK	4	OK	-340°	OK
4	OK	8	OK	-320°	OK
8	OK			-300°	OK
				-280°	OK
				-260°	OK
				-240°	OK
				-220°	OK
				-200°	OK
				-180°	OK
				-160°	OK
				-140°	OK
				-120°	OK
				-100°	OK
				-80°	OK
				-60°	OK
				-40°	OK
				-20°	OK
				0°	OK
				20°	OK
				40°	OK
				60°	OK
				80°	OK
				100°	OK
				120°	OK
				140°	OK
				160°	OK
				180°	OK
				190°	OK
				200°	OK
				220°	OK
				240°	OK
				260°	OK
				280°	OK
				300°	OK
				320°	OK

18.6 *CD-ROM*


CD-ROM: Commande et contrôle d'injection des moteurs à essence
Codes Microcontrôleur
Carte d'injection séquentielle
Carte de simulation
Documentation
Banc de test CFP
Megatune2.25
Code uC maître
Code uC esclave
Timings d'injection
Schématique + Routage + Composants
Commande de composants
DataSheets
Schématique + Routage + Composants
Installateur
Code Source
Mise en service carte ''Injection séquentielle''
Mode d'emploi MegaTune2.25
Tests
Rapport