Création d'un système de positionnement d'une caméra sur un axe linéaire.

1. Introduction 3
2. Cahier des charges 3
   2.1. Cahier des charges officiel 3
   2.2. Recherches sur le sujet 3
   2.3. Choix de conceptions 4
3. Etude des choix de conceptions 5
   3.1. Méthode de translation 5
      3.1.1. Moyen de glissement 5
      3.1.2. Moyen de traction 6
   3.2. Type de moteur 6
      3.2.1. Contraintes 6
      3.2.2. Choix du type de moteur 7
      3.2.3. Fonctionnement du moteur 7
   3.3. Carte de commande 7
      3.3.1. Méthode « à la main » 8
      3.3.2. Méthode NI 8
      3.3.3. Méthode de la carte autonome 8
   3.4. Carte de puissance 8
   3.5. Programmes de contrôle de la carte 9
   3.6. Codeur incrémental 9
4. Etude théorique et dimensionnement 9
   4.1. Dessin technique et schéma du projet 9
      4.1.1. Synoptique du schéma électrique 9
      4.1.2. Plan globale de la structure mécanique 10
      4.1.3. Vocabulaire de chaque objet 10
   4.2. Méthode de déplacement et de traction et choix du moteur 10
      4.2.1. Dimensionnement 10
      4.2.2. Choix du moteur 10
         4.2.2.1. Branchement 11
         4.2.2.2. Rapport couple/vitesse 11
         4.2.2.3. Schéma 12

4.2.3. Choix de la mécanique 12
4.3. Carte de commande 12
4.3.1. Cahier de contraintes 12
4.3.2. Choix de la carte 13
4.3.3. Etude des possibilités de la carte, et programmation HardWare 14
4.3.3.1. Méthode 14
4.3.3.2. Etude 15
4.3.3.3. Compromis 15
4.4. Carte de puissance 15
4.4.1. Simulation 16
4.4.2. Choix des composants électroniques 17
4.5. Architecture du programme 17
5. Liste du matériel 17
6. Etude et fonctionnement de la carte Flexis DEMOJM 17
6.1. Présentation du paquet 17
6.2. Fonctionnement globale 18
6.3. Préparation 18
6.3.1. Installation de CodeWarrior 18
6.3.1.1. Méthode avec les programmes les plus récents 18
6.3.1.2. Méthode ancienne 19
6.3.1.3. Notes sur les versions 19
6.3.1.4. Liens 19
6.3.2. Préparation de la carte 20
6.3.2.1. Vérifications des jumpers 20
6.3.2.2. Branchement des cables 20
6.3.2.3. Vérification de la cible 21
6.3.2.4. Test d'un programme 21
6.3.2.5. Placement d'une sonde 21
6.3.2.6. Documentations disponibles 21
6.4. Prise en main de l'IDE 22
6.4.1. Premiers pas 22
6.4.2. Création d'un projet 22
6.4.3. L'interface de l'IDE 22
6.4.4. Processor Expert et Device Initialization 23
6.4.5. Votre Code 23

6.4.6. Changer de cible 24
6.4.7. Test du code 24
6.4.7.1. Boutons importants 24
6.4.7.2. Méthode 24
6.5. Prise en main de HiWave 24

1. Introduction

Réalisation d'un chariot commandé numériquement par ordinateur. Le but est de pouvoir déplacer un objet sur un axe, à partir d'un ordinateur (principalement avec LabVIEW). La réalisation de la maquette commence par un choix du support de déplacement, puis du choix d'un moteur. Le moteur est commandé par une carte microcontrôleur, tandis qu'une carte de puissance permet l'alimentation de ce dernier. Un système de correction d'erreur est prévu, et géré par le MCU et par un encodeur incrémental. La liaison du MCU se fera en USB (port COM virtuel), et la carte utilisable à partir d'un programme LabVIEW quelconque.

2. Cahier des charges

2.1. Cahier des charges officiel

- vitesse jusqu'à quelques centimètres par seconde
- top de synchro tous les 1/100ème de millimètre
- course 700 mm
- déplacement linéaire
- charge < 1 kg
- faible coût < 1000 euros
- élément ré-utilisable ? (structure mécanique, codeur, carte ~~compteur~~codeur 6601, bornier)
- pas trop bruyant
- Pilotage avec LabVIEW (info de position, commande)

2.2. Recherches sur le sujet

Après avoir pris connaissance sur le sujet, une phase de recherche et de renseignement s'est imposé, afin de s'imprégner du sujet.

- La CNC, signifie : commande numérique de construction. Beaucoup de sites proposent de petits montages, des retours d'expérience, et des explications sur ce sujet.
- http://www.cncloisirs.com/
- http://cnc25.free.fr/
- Les moteurs pas à pas et les moteurs à courant continue.
- http://fr.wikipedia.org/wiki/Moteur_pas_%C3%A0_pas
- http://etronics.free.fr/dossiers/num/num50/mpap.htm

- http://col2000.free.fr/pasapas/pap_indx.htm
- http://fr.wikipedia.org/wiki/Machine_%C3%A0_courant_continu
- Les cartes de commande et/ou de puissance pour commander un moteur pas à pas.
- http://www.stepgenie.com/
- http://www.conrad.fr/webapps/commande_moteur-37.html
- http://www.conrad.fr/kit_step_easy_p_19247_19257_208569
- Etude des moyens de déplacement mécanique :
- http://cncloisirs.com/Construction/Transmissions
- Catalogue des fournisseurs
- http://www.freescale.com
- http://www.isel.fr/

2.3. Choix de conceptions

Après avoir étudié ce cahier des charges, j'ai été amené à faire des choix sur la manière de réaliser le projet:

- Avec un responsable, nous avons décidé d'utiliser l'USB pour établir la communication entre notre ordinateur et le système, pour des raisons de simplicités d'utilisation. Un port série n'est plus très courant sur la majorité des ordinateurs.
- Nous nous somme également posé la question de la fiabilité de l'appareil à réaliser : faut-il mettre en place un système de correction d'erreur ? Si le moteur avance trop ou pas assez loin, est-ce important de corriger l'erreur ? Avec l'expérience du métier, un responsable m'a expliqué qu'il valait mieux faire quelque chose de fiable, sinon l'intérêt est nul. Donc le système doit gérer au mieux les erreurs. Cela implique le choix d'un moteur adapté à l'utilisation que l'on veut en faire. Entre le moteur à courant continue et son homologue pas à pas, nous choisirons certainement la deuxième option.
- Le système doit également pouvoir être amélioré à l'avenir. Le choix d'une carte avec un microcontrôleur dessus ou le choix de la réalisation d'une petite carte dédié ne pose pas les même possibilité d'évolutions.
- Le système doit être compatible avec LabVIEW. Trois solutions se présentent :
- Nous devons utiliser du matériel proposé par National Instrument, au quel cas nous sommes sûr de la compatibilité entre le matériel et le logiciel
- Nous utilisons les fonctions avancées de LabVIEW, qui permettent notamment de « wrapper » des librairies (dll). (on expliquera ce que c'est plus loin).
- Une autre possibilité consisterait à créer un port COM virtuel associé au port USB. Nous utiliserions une communication de type liaison série dans ce cas.
- Les notions de prix et de temps de réalisation. Certains choix dans la réalisation du projet sont un compromis entre temps de réalisation et coût des produits à acheter. La réalisation d'une carte pour commander les moteurs ne semble pas être très optimisé sur le point de vue temporel, au vue de mes difficultés dans ce domaine. D'un autre coté, acheter une carte National Instrument pour un si petit projet est fort

peu rentable (les premiers prix se situent environ à 900 €, soit tout notre budget). Une solution intermédiaire devra être trouvée.

- Après quelques recherches, il s'est avéré que certains constructeurs fournissaient des carte spécialisées dans la CNC (Contrôle Numérique de Commande). Cependant, le prix élevé de ces produits n'est pas compatible avec notre projet.

3. Etude des choix de conceptions

3.1. Méthode de translation

Plusieurs méthode s'offrent à nous pour gérer mécaniquement le déplacement d'un chariot. On séparera cette étude en deux parties : le moyen de traction et le moyen de glissement. On notera que des solutions toutes prêtes existent.

- Vaut-il mieux créer le système, ou en acheter tout fait ? Question de temps et de prix ? Je pense que ce sera mieux de faire le montage sois même, plutôt que de l'acheter tout fait. Une limite de plus est que souvent ces produits sont en triphasés, je ne sais pas si cela est problématique, si on a un moyen de convertir la source de courant.

3.1.1. Moyen de glissement

- Pour un glissement linéaire, les solutions existantes se ressemblent toutes: un support sur lequel se déplace un chariot. Apres, il existes plusieurs technologies différentes, citons les roulements à bille, déplacement sur galet..
- Le constructeur ISEL propose beaucoup de modèles différents, à prix modéré. Apres la lecture de leur catalogue, et un appel au service commercial, je me suis arrêté sur le système du chariot à galet, approprié à notre cahier des charges. Si dessous une illustration en explique le fonctionnement :

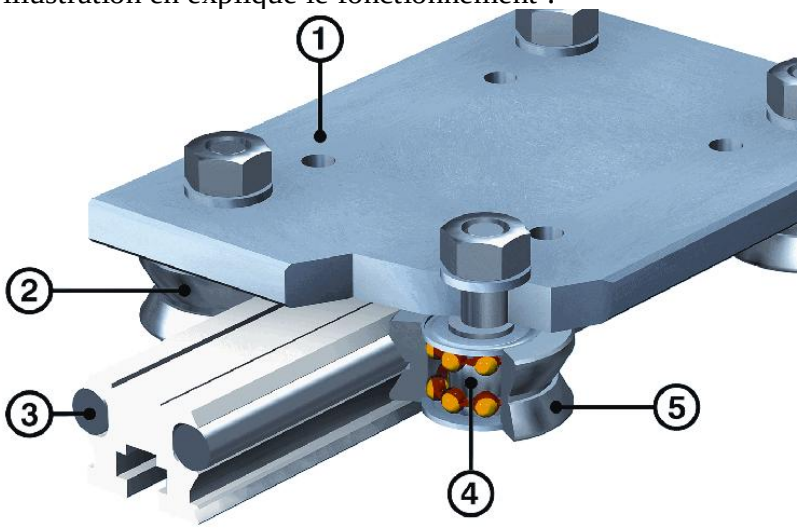


Figure 1: Schéma du chariot à galet (Source ISEL)

- Ce type de chariot nous permet un déplacement fluide et adapté pour des petites charges. Notre cahier des charges nous impose une charge moyenne utile de 1 kilogramme, mais une charge de 5 ou 6 kilogrammes ne devrait pas poser de problème. Ce système est donc adapté.

- Faut-il mener une étude précise, avec les données constructeurs ???

- Le chariot serait-il horizontal ou vertical ? Ca change rien ?
- Le temps de vie du chariot est-il à calculer ?
- La répartition des forces est-elle importante dans notre cas ?
- Quel constructeur faut-il prendre ? ISEL ? L'avantage de tout commander chez le même fournisseur est intéressant pour la livraison des produits.

3.1.2. Moyen de traction

Il existe plusieurs type de traction possibles ; vis sans fin, câble, courroies… Le site cncloisir.com (http://cncloisirs.com/Construction/Transmissions) répertorie les plus courants, avec une petite description des avantages et inconvénients de chaque système.

La solution retenue est celle de la courroie crantée. En effet, plusieurs avantages :

- Pas de dérèglement
- Solide
- On peut étendre la longueur que l'on veut, en remplaçant simplement la courroie par une plus grande, et en étendant le rail.

Figure 2: Image d'un système de traction à roue crantée (Source: CNCLoisir.com)

Ce sera donc le système retenu.

3.2. Type de moteur

Le moteur est un élément important à choisir. En effet, si ce dernier est mal dimensionné, le projet sera un échec total.

3.2.1. Contraintes

Pour dimensionner le moteur, nous avons quelques données dans le cahier des charges :

- Vitesse jusqu'à quelques centimètres par seconde (nous fixons 10 cm / sec)
- Course de 700 mm
- Top de synchro tous les 1/100ème de millimètre
- Charge < 1 kg (nous fixons une valeur maximale de 5 kg)
- Pas trop bruyant

On a ici une contrainte de charge, de vitesse et de précision. C'est ce cahier des charges qui va nous aider à trouver le bon type de moteur

3.2.2. Choix du type de moteur

La lecture de l'article des machines électrique sur WikiPédia[1] nous renseignera

- Le moteur à courant continu est l'un des moteurs les plus couramment utilisés. Cependant, il présente de nombreux défauts, comme le fait de ne pas être très précis, ou comme le fait que l'ensemble balais/collecteur rotatif s'use rapidement. Un autre problème limite les vitesses d'utilisation élevées de ces moteurs lorsque le rotor est bobiné, c'est le phénomène de « défrettage », la force centrifuge finissant par casser les liens assurant la tenue des ensembles de spires (le frettage). Ce type de moteur ne convient donc pas à notre projet.
- Le moteur sans balais, ou « moteur brushless », est un moteur synchrone, dont le rotor est constitué d'un ou de plusieurs aimants permanents et pourvus d'origine d'un capteur de position rotorique (capteur à effet Hall, synchro-résolver, codeur incrémental par exemple). Vu de l'extérieur, il fonctionne en courant continu. Son appellation Brushless vient du fait que ce type de moteur ne contient aucun balai. Sont très souvent utilisé dans le modélisme, ou dans les robots industriels.
- Le moteur linéaire est essentiellement un moteur électrique qui « a été déroulé » de sorte qu'au lieu de produire un couple (rotation), il produise une force linéaire sur sa longueur en installant un champ électromagnétique de déplacement. Actuellement utilisée par les trains à sustentation électromagnétique, cette technologie n'est pas à notre portée.
- Les moteurs pas à pas: un rotor interne contenant des aimants permanents est déplacé par un ensemble d'électroaimants placés dans le stator commutés par une électronique de puissance. L'alimentation ou non de chacun définit une position angulaire différente (l'enchaînement permet le mouvement). Les moteurs pas-à-pas simples ont un nombre limité de positions, mais les moteurs pas-à-pas à commande proportionnelle (alimentation variable des bobines) peuvent être extrêmement précis. On parle alors de « micro pas » puisque le moteur peut s'équilibrer entre deux pas.
  C'est cette qualité que nous recherchons, ce type de moteur semble tout à fait adapté a notre projet.

Notre choix s'arrêtera sur un moteur de type pas à pas.

3.2.3. Fonctionnement du moteur

Le principe de fonctionnement est expliqué dans un article[2] de Wikipédia. Différents modes existent, dans le cas d'un moteur à aimants permanents: pas complet, couple maximal, demi-pas, mode unipolaire et bipolaire. Le choix de ces modes dépendra du moteur que l'on choisira, en fonction du nombre de fils disponibles, des possibilités de la carte de puissance, du MCU, etc.

3.3. Carte de commande

La carte de commande permet de piloter le moteur, de lui donner une vitesse et un temps de fonctionnement.

---

[1] Lien : http://fr.wikipedia.org/wiki/Machine_%C3%A9lectrique#Machines_tournantes

[2] Lien : http://fr.wikipedia.org/wiki/Moteur_pas_%C3%A0_pas#Moteur_.C3.A0_aimants_permanents

3.3.1. Méthode « à la main »

La première idée a été de réaliser une petite carte pour contrôler les commandes et la puissance du moteur. Devant la difficulté technique que cela représente pour moi, l'idée a été vite oubliée, malgré la grosse source d'aide disponible sur Internet.

3.3.2. Méthode NI

La seconde étape a été d'aller chercher du matériel chez National Instrument. Après quelques difficultés à trouver un produit qui correspondait à mes besoins, j'ai vite déchanté. En effet, les prix de ces produits étaient très élevés, absorbant tout mon budget. De plus, l'interface proposée était une carte PCI. C'est-à-dire que pour utiliser la table, il faudra rajouter une carte PCI sur le PC à partir duquel on voudra commander. Pas forcément très pratique. L'avantage d'aller chez NI, c'est qu'on a une intégration excellente entre le logiciel et le matériel. Pas besoin de se prendre la tête avec des drivers, etc. Donc, c'était intéressant en terme de temps de développement. Un autre désavantage était celui qu'il fallait tout de même réaliser la carte de puissance pour le moteur.

3.3.3. Méthode de la carte autonome

La solution retenue consiste à utiliser une carte de laboratoire avec un petit microcontrôleur dessus. Beaucoup d'avantages en découlent:

- Programmation en langage C : Rapidité de développement, en tout cas dans mon cas.
- Pas besoin d'utiliser systématiquement un PC pour utiliser la table, grâce aux boutons présents sur la carte.
- Permets de gérer à la fois les signaux de commande moteur, ainsi que la liaison avec le PC, et permets de programmer différents modes de fonctionnement, dans le cas d'un moteur pas à pas. (cf. modes de fonctionnement d'un moteur pas à pas)
- Souplesse pour des évolutions futures: cette solution peut-être reprogrammée pour entre améliorée, sans changer le matériel existant.
- Peut gérer de manière simple un système de correction d'erreur.
- Le prix de ce genre de carte s'étale entre une cinquantaine d'euros jusqu'à 200 € pour les modèles les plus chers.
- Le fabricant fournit des pilotes et garantissent une utilisation haut niveau simple, ce qui fait économiser du temps sur la réalisation du projet.

En revanche:

- Il faudra tout de même créer une carte de puissance, en sortie de notre carte microcontrôleur.
- Il faudra créer et développer une librairie sous Windows, et l'utiliser sous LabVIEW, grâce à des « wrappers ». Le système ne fonctionnera (malheureusement) que sous Windows.

3.4. Carte de puissance

La carte de puissance nous permettra à partir des sorties de la carte de commande de fournir l'énergie nécessaire au bon fonctionnement de notre moteur.

Pour le pilotage de la bobine, nous utiliserons le principe de point en H, montage très courant dans ce genre d'application.

- La carte de puissance me pose beaucoup de problèmes, quant à sa conception. Plusieurs schémas existent sur Internet, mais j'avoue galérer pas mal.

3.5. Programmes de contrôle de la carte

Le programme de contrôle de la carte permettra à l'ordinateur de piloter le chariot. Le programme qui sera exécuté sur le MCU devra fonctionner en parfaite symbiose avec l'ordinateur, ou sans l'ordinateur. Dans un premier temps, nous pouvons fixer un mini cahier des charges pour la réalisation de ce logiciel:

- Dois toujours afficher la valeur exacte du positionnement, par une vérification des indicateurs d'erreurs.
- Le chariot doit pouvoir être piloté à partir de la carte, ou à partir de l'ordinateur.
- La vitesse et le sens de déplacement doivent être variables.
- Il faut un bouton d'arrêt d'urgence.
- Un bouton de mise à zéro (pour fixer l'origine de l'axe, sachant que notre système de positionnement est relatif)

3.6. Comptdeur incrémental

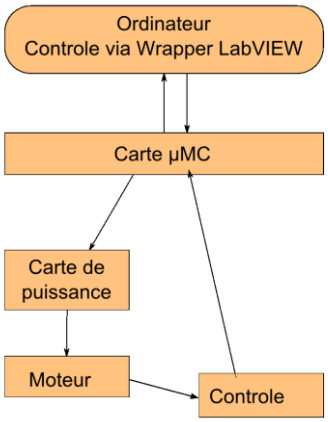
Le ~~compteur~~codeur incrémental permettra de vérifier que le moteur réagit correctement aux commandes qu'on lui fournit. Une courroie reliera le rotor du moteur et du ~~compteur~~codeur. Le ~~compteur~~codeur renverra des salves, en fonction de l'état de rotation du rotor du moteur. Un ~~compteur~~codeur peut envoyer, par exemple, 1000 salves par tour, et avoir une fréquence de fonctionnement de 100 kHz maximum.

4. Etude théorique et dimensionnement

Cette étude va nous permettre de choisir les différents composants ainsi que leur modèle afin de réaliser l'assemblage final.

4.1. Dessin technique et schéma du projet

4.1.1. Synoptique du schéma électrique



4.1.2. Plan global de la structure mécanique
4.1.3. Vocabulaire de chaque objet

4.2. Méthode de déplacement et de traction et choix du moteur

Le choix du moteur se fait en fonction des différents éléments mécaniques. En effet, le dimensionnement du moteur va dépendre de la charge à porter, de la vitesse que l'on souhaite avoir, du temps d'accélération et de la taille de roue de l'engrenage… Expliquons un peu plus en détail.

4.2.1. Dimensionnement

Pour dimensionner le moteur à acheter, il faut considérer plusieurs choses. En effet, le rotor du moteur entraînera une roue crantée qui permettra de déplacer le chariot. Le diamètre D de cette roue est donc important. On suppose prendre une roue d'un diamètre de 4 cm.. Notons aussi que le système de glissement n'est pas idéal, et sur lequel la notion de frottement est non négligeable.

Pour choisir un moteur, il faut connaître la puissance et le couple du moteur à acheter. Une feuille Excel rassemble l'ensemble des calculs effectués, et une valeur de couple et de puissance pour chaque cas.

	Valeur des paramètres					Calculs intermédiaires					Résultats	
	Masse	Vitesse	Temps	Diametre rou		Force	Accélération	Tours par sec	Energie cinétique		Puissance	Couple
	Kg	m/s	s	m		N	m/s/s	Tour/s	J		W	Nm
Cas minimal	1	0,1	1	0,04		0,1	0,1	1,25663706	0,005		0,005	0,0098696
Cas large	10	0,1	1	0,04		1	0,1	1,25663706	0,05		0,05	0,09869604
Cas moyen	5	0,1	1	0,04		0,5	0,1	1,25663706	0,025		0,025	0,04934802
	0	0,1	1	0,05		0	#DIV/0!	1,57079633	0		0	0
	0	0,1	1	0,05		0	#DIV/0!	1,57079633	0		0	0

Tableau 1: Présentation des valeurs de couple et de puissance en fonction de paramètres variables.

- Ce tableau est-il bon? Les valeurs de couples et de puissance me semblent très faibles, par rapport à ce que proposent les modèles dédiés à la CNC.

4.2.2. Choix du moteur

Sur le site de l'ISEL, nous trouvons le modèle MS 058 HT, un moteur unipolaire, qui corresponds à nos besoins. Cependant, les dates de livraisons sont très longues. L'intérêt serait de trouver un modèle similaire, chez un autre fournisseur.

4.2.2.1. Branchement

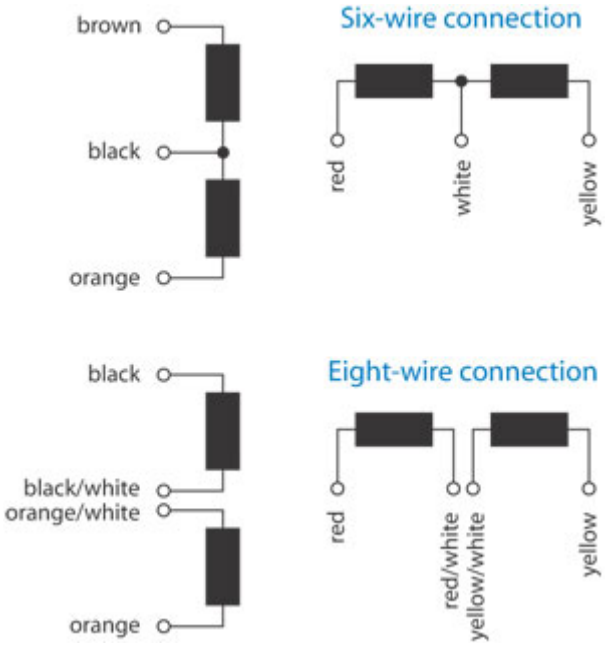


Figure 3: Mode de connexion (Source ISEL)

L'intérêt de pouvoir choisir entre 6 et 8 fil, c'est que cela nous laisse une certaine liberté pour la création de la partie commande. On pourra soit travailler sur 6 ou 8 bits, en fonction de nos besoins/possibilités.

4.2.2.2. Rapport couple/vitesse

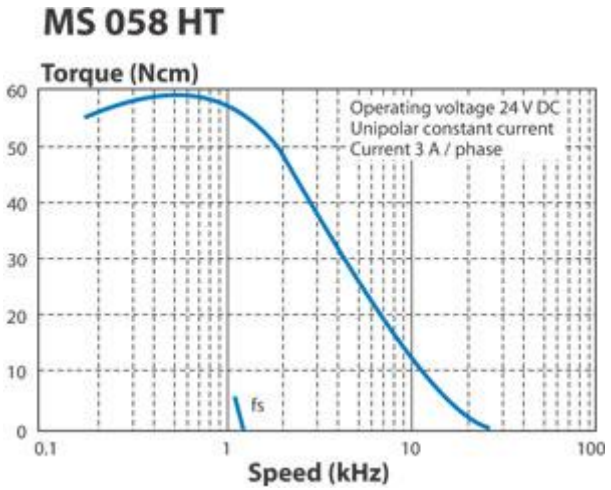


Figure 4: Couple en fonction de la vitesse (Source ISEL)

Cette caractéristique nous montre que le couple sera le plus important à faible vitesse.

4.2.2.3. Schéma

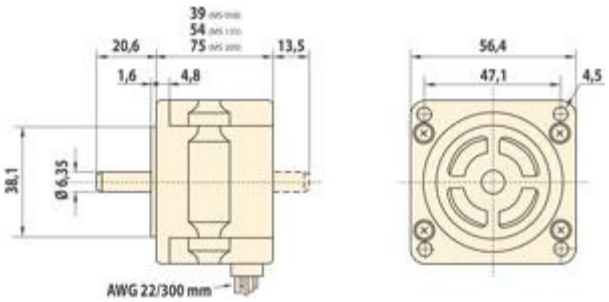


Figure 5: Schéma et dimensions du moteur (Source ISEL)

Les dimensions du moteur sont raisonnables par rapport à la taille du système global. Notons tout de même le diamètre du rotor, détail important pour le choix des engrenages: 6.35 mm

4.2.3. Choix de la mécanique

Le choix des pièces mécanique ne relève pas tellement de mes compétences professionnelles. Ce choix se fera de manière empirique, la seule contrainte à respecter étant le diamètre de la roue crantée fixée sur le rotor du moteur.

Nous avons besoin de :

- Une roue crantée de diamètre D.
- Deux petites roues crantées (voir figure 1).
- Une courroie crantée.
- Un kit pour entraîner le ~~compteur~~codeur (permettant l'asservissement)
- Deux supports pour fixer la courroie à ses extrémités.
- Rotors divers et fixations mécaniques.

4.3. Carte de commande

Il existe déjà des circuits intégrés pour s'occuper de ce genre de chose. On peut citer les circuits StepGenies, MC3419, SLA7026M… Cependant, dans notre projet est prévu un système de correction d'erreur (c'est donc une sorte de système asservi). A la vue de mes compétences électroniques, j'ai pensé qu'il valait mieux utiliser une carte Microcontrôleur (µC ou MCU, pour Micro Controler Unit), un domaine connu et maîtrisé. L'avantage de passer par un MCU est que l'on utilise une seule cible pour programmer le programme. De plus, si le projet devait à être mis à jour, il est plus simple de modifier un programme bien commenté en C, que de récréer une nouvelle carte de commande.

4.3.1. Cahier de contraintes

Pour choisir mon MCU, j'avais un cahier de contraintes:

- Un port USB: tant pour la programmation de la carte, que pour l'utilisation quotidienne.
- Un port CAN: me permettra de lire l'état du ~~compteur~~codeur (qui permettra d'ajuster la commande moteur et de détecter les erreurs). Au moins sur 4 bits, pour avoir un petit peu de marge de manœuvre.

- Un port CNA: qui permettra de commander le moteur. Le port devra au minimum être sur 6 bits.
- Un port série: cela peut toujours servir, mais ce ne sera pas un critère déterminant dans le choix de la carte.
- Une interface homme-machine: Au moins 4 boutons, et un écran LCD. L'ordinateur deviendra alors facultatif.
- Faible prix: Une carte se situant sous la barre des 200 €.
- Pilotes PlugAndPlay: Sous cette appellation abusive, j'entends pouvoir utiliser la communication USB sans perdre du temps à écrire un pilote.
- Programmation en C : Certaines cartes se programment en VHDL ou en VeriLog.

4.3.2. Choix de la carte

Mon choix s'est plus ou moins naturellement orienté vers Freescale, étant déjà habitué à leur environnement de travail grâce à l'IUT. La plus grande difficulté a été de trouver la carte qui correspondait à mes besoins, parmi leur catalogue. Après quelques coups de téléphone infructueux à destination de leur service commercial délocalisé en Inde, je me résigne à contacter par mail les « experts américains », comme me l'a conseillé le Pakistanais. En attendant la réponse du support, je poste sur quelques forums spécialistes, qui me proposent globalement un MCU. La réponse du support arrive le lendemain, et ils me conseillent le même MCU. La carte est choisie.

Il s'agit du MCU « Flexis™ JM Demonstration Board », qui propose au choix deux Microcontrôleurs:

- S08JM : en 8-bits, sortie en 2001
- MCF51JM: en 32-bits, ColdFire V1 sortit en 2008

Ces deux Microcontrôleurs sont livrés dans le même pack, sans surplus de prix. A ce moment, j'imagine que le programme sera écrit pour le MCF51JM, bien que d'après Freescale, le changement de microcontrôleur ne nécessite pas de modification de code du programme.

Je n'ai malheureusement pas trouvé de datasheet sur la carte de test, en revanche les datasheets des deux microcontrôleurs sont disponibles sur le site de Freescale. Cependant, une note pour les utilisateurs explique de manière assez globale le fonctionnement de la carte.

- Dispose d'un accéléromètre : fonction qui ne sert à rien à priori, mais il pourrait être marrant de piloter la plate-forme juste en inclinant la carte d'un coté ou d'un autre.
- Un port USB : Mini-AB, en mode hôte ou périphérique.
- Port CAN : 12-bits
- 8 Leds
- Un buzzer piézo
- 5 boutons
- Prix de 100€

Cette carte conviendra parfaitement à nos besoins. Cependant, d'après le retour de certains utilisateurs, ils ont eu quelques difficultés quand à l'utilisation des DLL sous Windows pour

piloter la carte via un programme. Globalement, il est plus simple d'utiliser la fonction de port COM virtuel (un « stack » est disponible, en version limitée, sur le site du constructeur).

- Est-ce un bon choix ? Je suis pas forcement convaincu, sachant que je n'ai pas trouvé de datasheet sur cette carte. Peut-être qu'elle sera fournie avec la carte, mais je trouve étonnant de ne pas la trouver.

4.3.3. Etude des possibilités de la carte, et programmation Hardware

La carte de démonstration possède comme nous avons pu le voir une grande quantité de services matériels (LEDs, ADC, Timers, Accéléromètre …). Seulement, notre microcontrôleur dispose d'un certain nombre d'entrés et sorties. Notre carte fournissant 2 microcontrôleurs différents, nous ferons notre étude sur le MCF51 (64 broches) bien plus récente et performant que le MC9S08. Cependant, leur brochage d'après Freescale est sensé être le même.

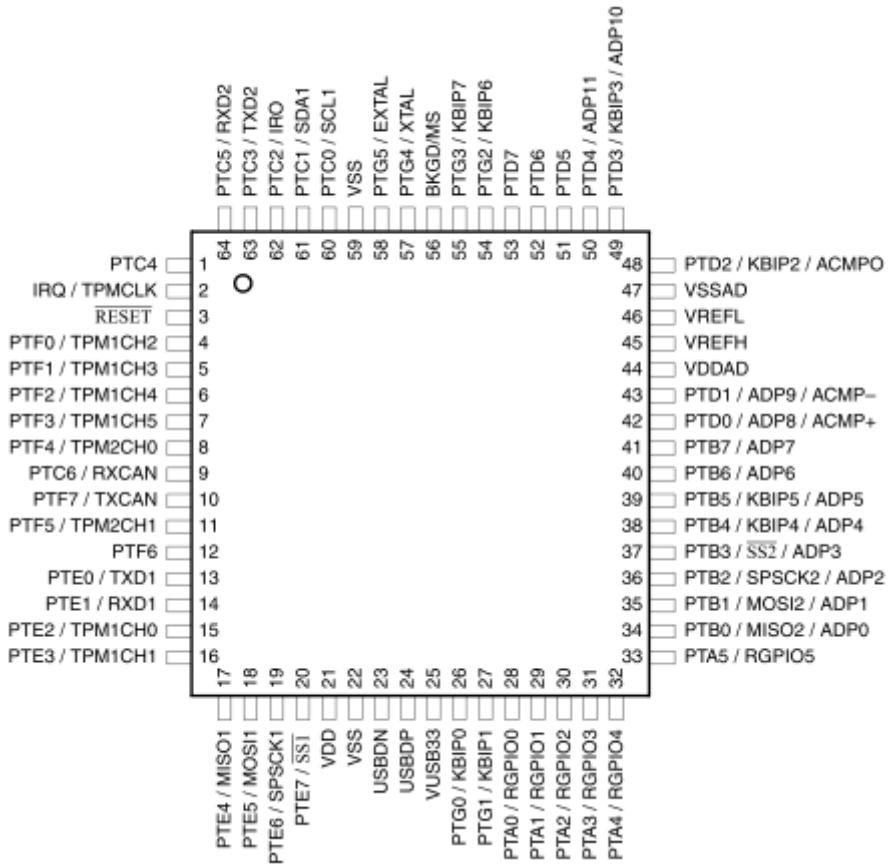


Figure 6: Détail des pattes du MC ColdFire (modèle 64 pins)

4.3.3.1. Méthode

Les ports sont numérotés de A à J, et chaque bit est lié à une unité de calcul du MC (ADC, USB, TIMER, RGPIO …). Il faut savoir si notre carte peut supporter ce qu'on lui demandera. Pour cela, il faut étudier le PCB de la carte, voir ce qui est obligatoire et ce qui ne rentre pas dans notre cahier des charges. Toutes ces documentations sont fournies par le constructeur.

4.3.3.2. Etude

En étudiant le PCB de la carte de démonstration, nous allons dans un premier temps faire le tri, entre les fonctionnalités (blocs) dont l'on a besoin, du µC et de la carte de démonstration.

Les nécessaires sont :

- RGPIO : Nous permettra d'envoyer et de recevoir des signaux logiques à une vitesse proche de la fréquence de fonctionnement du µC. On en aura besoin de 4, 6 ou 8 pour contrôler le moteur, et 2 ou 4 pour contrôler le codeur incrémental (on verra si l'on peut se passer du ADC (CAN en Français) pour le codeur).
- PORTA: 0-7
- PORTH : 2,3,4
- PORTJ : 0-4
- USB : Pour la communication avec le PC, et pour l'alimentation aussi.
- Les boutons poussoirs : Pour contrôler la machine sans ordinateur.
- PORTG : 0-3
- ADC : Eventuellement pour lire le codeur incrémental, si on ne peut pas le faire avec le RGPIO. On pourra également utiliser le potentiomètre pour régler la vitesse.
- PORTB :0-7
- PORTD :0,1,3,4

On est donc sensé pouvoir ne pas utiliser les autres blocs, en tout cas dans une première version du programme.

4.3.3.3. Compromis

Nous allons voir ce que l'on devra ne pas utiliser/débrancher sur la carte pour répondre à notre cahier des charges.

Pour les ports associés au module RGPIO, seuls PTA4 et PTA5 sont utilisés par l'USB (parce qu'on met la carte en mode périphérique, et non pas hôte). Nous disposons alors de 6 ports RGPIO disponibles, moins les deux utilisés par l'USB.

Nous pouvons utiliser d'autres ports (pas les RGPIO) pour la lecture et l'écriture des signaux entrants et sortant. Il faut néanmoins qu'ils soient suffisamment rapides en écriture et/ou en lecture pour lires des signaux à quelques dizaines de kilohertz…

4.4. Carte de puissance

La carte de puissance est la carte qui permettra de faire le lien entre le moteur et la carte de commande « Flexis™ JM Demonstration Board ». En effet, la carte ne peut pas délivrer la puissance demandée par le moteur, sous risque d'endommager de manière définitive la carte. Pour cela, nous devons créer une petite carte intermédiaire, qui s'occupera de la partie puissance.

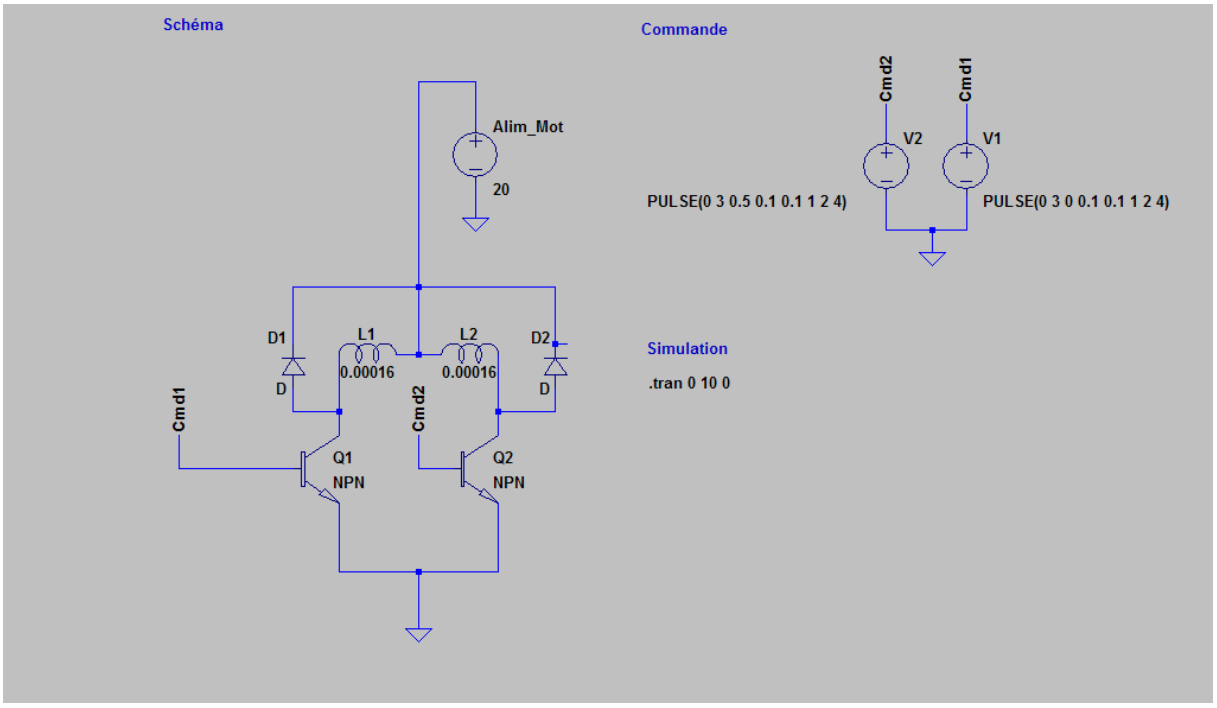
Après moult recherches sur les forums et les sites, la structure en H semble la plus simple à mettre en oeuvre:

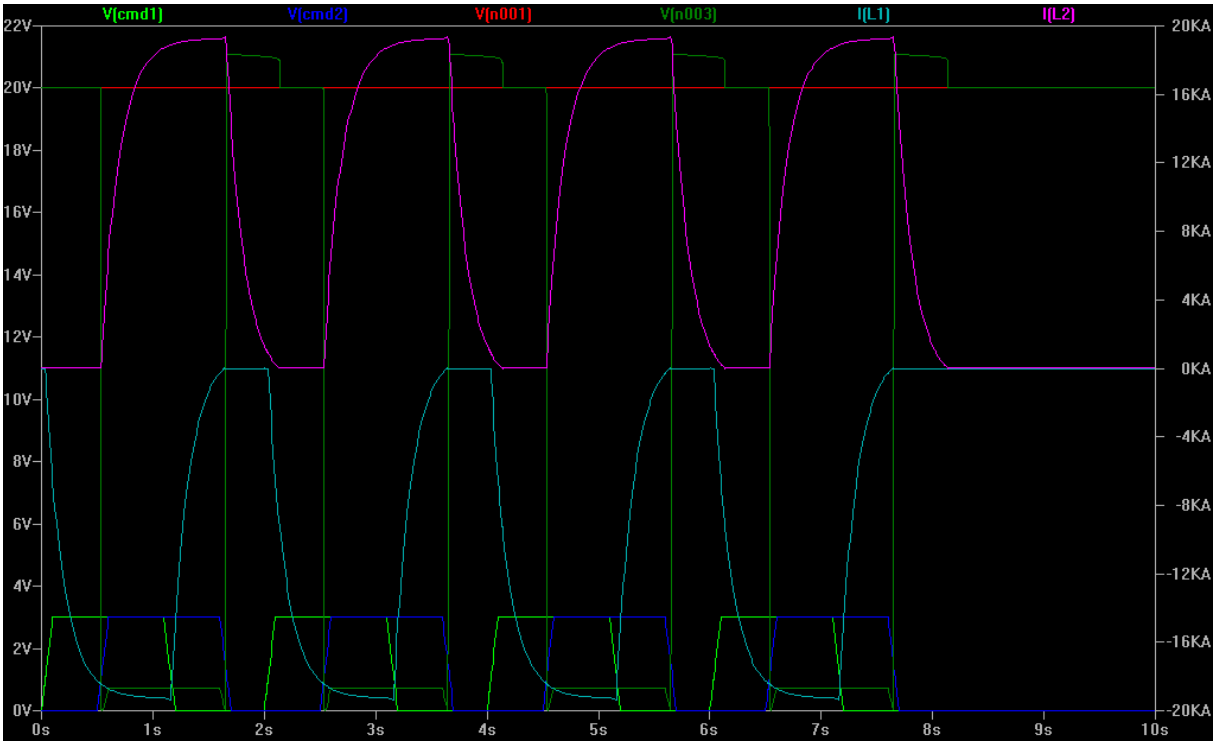
- Alors, là grosse galère, je n'arrive à rien, même avec mes simulations sous SwitcherCAD. Sinon, il faut que j'achète une carte de puissance déjà construite.

- L'idéal serait le fonctionnement à couple maximal, en bipolaire. http://stielec.ac-aix-marseille.fr/cours/abati/elecpas.htm
- Ou alors, l'alimenter en mode unipolaire directement.

4.4.1. Simulation

Simulation d'un pont H pour le contrôle d'une des deux bobines du moteur choisi.





4.4.2. Choix des composants électroniques

Aucune idée pour le moment…

4.5. Architecture du programme

L'étude du

5. Liste du matériel

- Moteur: MS 058 HT, de chez ISEL. Lien: http://www.isel-cnc.fr/isel_fr/Electronique/Index.HTML, puis moteurs/ pas à pas/Série MS/MS 058 HT. Référence: 470 520. Coût: 85.62 €
- Guidage linéaire: http://www.isel-cnc.fr/isel_fr/Meca/Index.HTML, puis Guidages linéaires/Série MLF. Coût: 102.26 €
- Rail pour guidage linéaire: 100 cm, Coût: 93.63 €
- Petite mécanique:
- Courroie crantée linéaire (longueur un peu plus de 70 cm)
- 3 roues crantées (Diamètre environ 4 cm et un de 3 cm)
- deux mini roues crantées
- Carte MCU: 100 €

6. Etude et fonctionnement de la carte Flexis DEMOJM

Dans ce chapitre, nous verrons tout ce qui est lié à la manipulation de la carte Flexis DEMOJM.

6.1. Présentation du paquet

Une fois le paquet (commandé chez Farnell le jeudi 28 mai 2009, reçu le lendemain matin) reçu, on trouve différentes choses dedans. Petite présentation:

- La carte DEMOJM, avec sa carte fille verte (daughter dans les datasheets). Par défaut, les switchs sont configurées de manière optimale. La configuration par défaut est indiquée dans les documentations DEMOJM Lab Supplément, un pour chaque µC.
- Une deuxième carte fille, contenant le µC MCF51JM128.
- Plein de documents papier (garantie, pub, guides, trucs divers qui ne servent à rien). Une version en PDF est disponible pour tout ce qui est documentation liée à la carte.
- Une clef USB contenant des exemples et des programmes supplémentaires.
- Un DVD-Rom contenant toutes les applications nécessaires au développement sur la carte.
- Un pack d'adaptateur USB et un câble USB gris.

6.2. Fonctionnement global

Avant d'installer quoi que ce soit, il s'agit de comprendre les bases pour ne pas être trop perdu. Les personnes qui ont déjà développé des cartes µC de chez Freescale (ou Motorola) peuvent à priori passer cette partie.

Pour programmer la carte, il faut avoir un minimum de connaissances en langage C (ou éventuellement en assembleur), ainsi que savoir comment fonctionne de manière générale un µC. Les versions des logiciels que l'on utilise sont des versions limitées, mais tout à fait appropriées à ce que l'on cherche à faire. Par exemple, nous utiliserons Metrowerks CodeWarrior dédiée au µC.

La méthode de développement suit ces étapes:

1. Lecture de la documentation technique du µC, ou du moins des parties intéressantes (elle fait environ 600 pages)
2. Création du code d'initialisation des registres, puis des fonctions qui permettent d'utiliser différentes capacités du µC (timers, PWM, Port Série, USB, ADC…)
3. Paramétrage de la carte: vérifier la position des jumpers, que le µC est le bon, que tout est bien connecté au PC.
4. Ecrire le programme dans la mémoire du µC et débugger/tester via HiWave.

Nous allons détailler toutes ces étapes pour une prise en main facile de la carte et de l'utilisation des outils dédiés.

6.3. Préparation

Il faut installer différents programmes et pilotes pour pouvoir utiliser la carte.

6.3.1. Installation de CodeWarrior

CodeWarrior sera notre environnement de développement principal. Il est fourni par Freescale. Nous allons installer tout ce qu'il faut pour développer sous les 2 µC. Les liens vers les différents logiciels à télécharger sont disponibles ci-dessous.

Vous avez 2 possibilités à ce stade.

6.3.1.1. Méthode avec les programmes les plus récents

C'est la méthode conseillée, sachant que la version 6.1 n'est plus supportée par le constructeur (ou en tout cas plus en téléchargement sur leur site).

Utiliser cette méthode est sensé fonctionner du premier coup. Vous êtes sûr d'avoir les derniers trucs tip-top; enfin, il n'y a pas de changements très importants, mais ça peut être intéressant de l'utiliser pour tester des exemples qui ont été développement sous la version 6.2.

1. Installer CW 6.2 for µC
2. Il n'y a priori pas besoin d'installer les Services Pack JM60 et JM128, car ils sont déjà inclus dans cette version.

A partir de ce stade, vous pouvez programmer votre carte. Le reste n'est que programmes supplémentaires, et facultatifs. Ils proposent néanmoins quelques exemples. Nous en reparlerons plus tard. Si vous débutez, installez-les.

1. Installer le Stack USB-Lite by CMX pour MCF51JM128 et pour MC9S08JM60.

3. Installer PEMICRO Embedded Multilink Toolkit.

A ce stade tous les logiciels sont installés pour développer votre premier programme.

6.3.1.2. Méthode ancienne

Utiliser la version 6.1, fournie sur le DVD, qui est dépassée à ce jour. Mais elle à l'avantage de fonctionner à tous les coups, et de ne pas poser de problèmes de connections avec la carte. Il faudra néanmoins mettre à jour certains de ses composants. Si vous choisissez ce cas, vous pouvez suivre à la lettre le Quick Start Guide, en installant les différents programmes du DVD. Dans l'ordre:

2. Installer CW 6.1 for µC
3. Installer les JM128 Service Pack et JM60 Service Pack (pour pouvoir utiliser les deux cartes filles).

A partir de ce stade, vous pouvez programmer votre carte. Le reste n'est que programmes supplémentaires, et facultatifs. Ils proposent néanmoins quelques exemples. Nous en reparlerons plus tard. Si vous débutez, installez-les.

4. Installer le Stack USB-Lite by CMX pour MCF51JM128 et pour MC9S08JM60.
5. Installer PEMICRO Embedded Multilink Toolkit.

Pour faire bien, il faut mettre à jour Processor-Expert, surtout si vous utilisez le MCF51JM128.

6. Mettre à jour Processor Expert vers une version 3.03 pour CW 6.1, disponible sur le site de Freescale.

A ce stade tous les logiciels sont installés pour développer votre premier programme.

6.3.1.3. Notes sur les versions

Freescale propose plusieurs versions de CodeWarrior, notamment des versions 7 ou 8. mais notez que vous ne pourrez pas les utiliser avec la carte DEMOJM, car ce ne sont pas des versions adaptées pour le développement sous des µC. donc, restez dans les versions 6.x.

Freescale propose une version « spéciale » et une version « évaluation ». C'est la première qu'il faut choisir. Elle est limitée en quantité de fichiers par projet (32) et en taille de code (64 ko), mais c'est amplement très suffisant pour notre application. La version évaluation permet d'avoir la version complète du programme pendant 30 jours, mais que pendant 30 jours.

6.3.1.4. Liens

- Page de présentation de la carte DEMOJM, avec tout plein d'infos:
  https://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=DEMOJM
- Version 6.1 de CW pour µC :
  (n'est plus disponible est ligne)
- Version 6.2 de CW pour µC :
  http://www.freescale.com/lgfiles/devsuites/HC08/CW_MCU_V6_2_SE_11262008.exe
- Mise à jour de Processor Expert 3.03 pour CW 6.1:
  http://www.freescale.com/webapp/sps/download/license.jsp?colCode=MCU_V61_PE

_3_03&location=null&fpsp=1&WT_TYPE=Updates%20&%20Patches&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=exe&WT_ASSET=Downloads&Parent_nodeId=11951510718747185307 36&Parent_pageType=product

- JM60 Service Pack: http://www.freescale.com/webapp/sps/download/license.jsp?colCode=MCU_V6-1_JM60_SP&location=null&fpsp=1&WT_TYPE=Updates%20&%20Patches&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=exe&WT_ASSET=Downloads&Parent_nodeId=11951510718747185307 36&Parent_pageType=product
- JM128 Service Pack: https://www.freescale.com/webapp/Download?colCode=MCU_V6-1_JM128_SP&nodeId=0127260061788213CF333D&location=overview&WT_TYPE=Updates%20&%20Patches&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=exe&WT_ASSET=Downloads&Parent_nodeId=11955961071387147443 33&Parent_pageType=product&Parent_nodeId=11955961071387147443 33&Parent_pageType=product

Notez que les liens ne seront sûrement plus valide, et que les logiciels existeront dans des versions plus récentes. Je vous conseille quand même d'utiliser le moteur de recherche le moteur de recherche de téléchargement avec les noms ci-dessus.

6.3.2. Préparation de la carte

Préparer la carte est un bien grand mot. Il suffit juste de vérifier les jumpers, le branchement des câbles et la cible.

6.3.2.1. Vérifications des jumpers

La vérification du placement des jumpers est importante pour utiliser toutes les fonctions de la carte. De base, on peut quasiment utiliser toutes les fonctions de la carte (l'accéléromètre, le buzzer, les leds, les boutons poussoirs, le potard…)

Pour obtenir la configuration en sortie d'usine, vous pouvez vous référer aux guides DEMOJM Lab Supplement, dans la partie « Default Jumpers Settings ».

Pour une configuration plus fine, vous pouvez vous référer au

pour obtenir la configuration en sortie d'usine, vous pouvez vous référer aux guides DEMOJM Lab Supplement, dans la partie « Default Jumpers Settings ».

Pour une configuration plus fine, vous pouvez vous référer au User Guide de la Carte, qui montre la disposition des différents jumpers, en fonction des éléments que l'on veut utiliser.

6.3.2.2. Branchement des câbles

Rien de bien compliqué. Le câble USB gris est le seul nécessaire dans l'absolu. Il fournit l'alimentation électrique de la carte, ainsi que l'interface de débogage de la carte (appelé P&E Embedded MultiLink Toolkit. C'est cette interface qui va permettre de programmer la carte.

Si vous souhaitez également utiliser les fonctions d'USB, vous devez utiliser le second port mini-USB. N'oubliez pas d'activer le switch on/off, qui établira ou non la connexion USB avec le PC.

6.3.2.3. Vérification de la cible

Il s'agit tout simplement de vérifier que la carte daughter est la bonne est qu'elle est bien placée.

- Vérifiez que vous n'avez pas interverti le MCF51JM120 et le MC09S08JM60
- Vérifiez que le détrompeur est bien placé du bon côté (en haut à droite)
- Vérifiez que les pattes sont bien toutes en contact avec les broches (enfin, juste faire attention à ça)

Notes:

- Pas la peine d'enfoncer à fond la carte fille, âpre, c'est pénible à retirer, surtout si vous la changez souvent.
- Prenez un petit tournevis pour soulever délicatement la carte. Attention au coup de tournevis qui dérape et qui détruit la carte (= fin du monde, PL vous tue !)

6.3.2.4. Test d'un programme

De base, on vous fournit des programmes et des exemples. Vous pouvez tester celui fournit par le Stack USB de CMX. Celui qui fonctionne est celui du HID, dans le dossier: CMXUSB_LITE\usb-peripheral\projects\CodeWarrior-6.x\mcf51xx\hid-demo

Cette démo fonctionne, et aura pour conséquence de faire votre souris de gauche à droite, et de droite à gauche, et de gauche à droite, et de droite à gauche, et de gauche à droite, et de droite à gauche, et…

Notez que c'est la seule démonstration que j'ai réussi à faire fonctionner. Les autres, on ne sait pas trop à quoi elles servent, sachant que leur fonctionnement et leur but ne sont même pas documentés.

Sachez aussi que les applications fournies par le P&E Micro n'ont jamais fonctionné avec moi (à part Terminal Utility, et encore, il fonctionnait mal). Après, rien ne vous empêche de les tester.

6.3.2.5. Placement d'une sonde

Si durant votre développement, vous êtes amené à étudier un signal (du style PWM) que vous mettez en sorties, vous pouvez l'analyser grâce à une sonde et un oscilloscope. La petite difficulté consiste à placer la masse. D'après le schéma de la carte, différents pins de jumpers sont reliés à la masse.

Vous branchez alors la masse de votre sonde à une pin de la masse. (ou autre bricolage exotique avec des jumpers)

6.3.2.6. Documentations disponibles

Plusieurs documentations sont disponibles. Face à leur très grande importance, ce chapitre montre celle qu'il est indispensable de connaître ou du moins de consulter :

- Documentation des µC MCF51JM128 et MC09S08JM60: l'une ou l'autre, permets de savoir comment paramétrer tout les registres, avec une explication détaillée de chaque paramètre. Attention, elles sont… imposantes!

- Documentations de la carte DEMOJM. Est disponible le User Guide (avec la configuration des jumpers, l'installation des éventuels drivers…), ainsi que le schéma de la carte (important, pour connaître les ports associés aux boutons, au buzzer… etc.)

6.4. Prise en main de l'IDE

6.4.1. Premiers pas

Pour développer un programme sur la carte, on aura besoin de Freescale Metrowerks Code Warrior. Vous pouvez le lancer via le menu démarrer.

Un menu d'accueil apparaît. Soit, vous choisissez d'ouvrir un exemple, d'ouvrir un projet (ceux du stack USB de CMX, par exemple), ou de créer un nouveau projet.

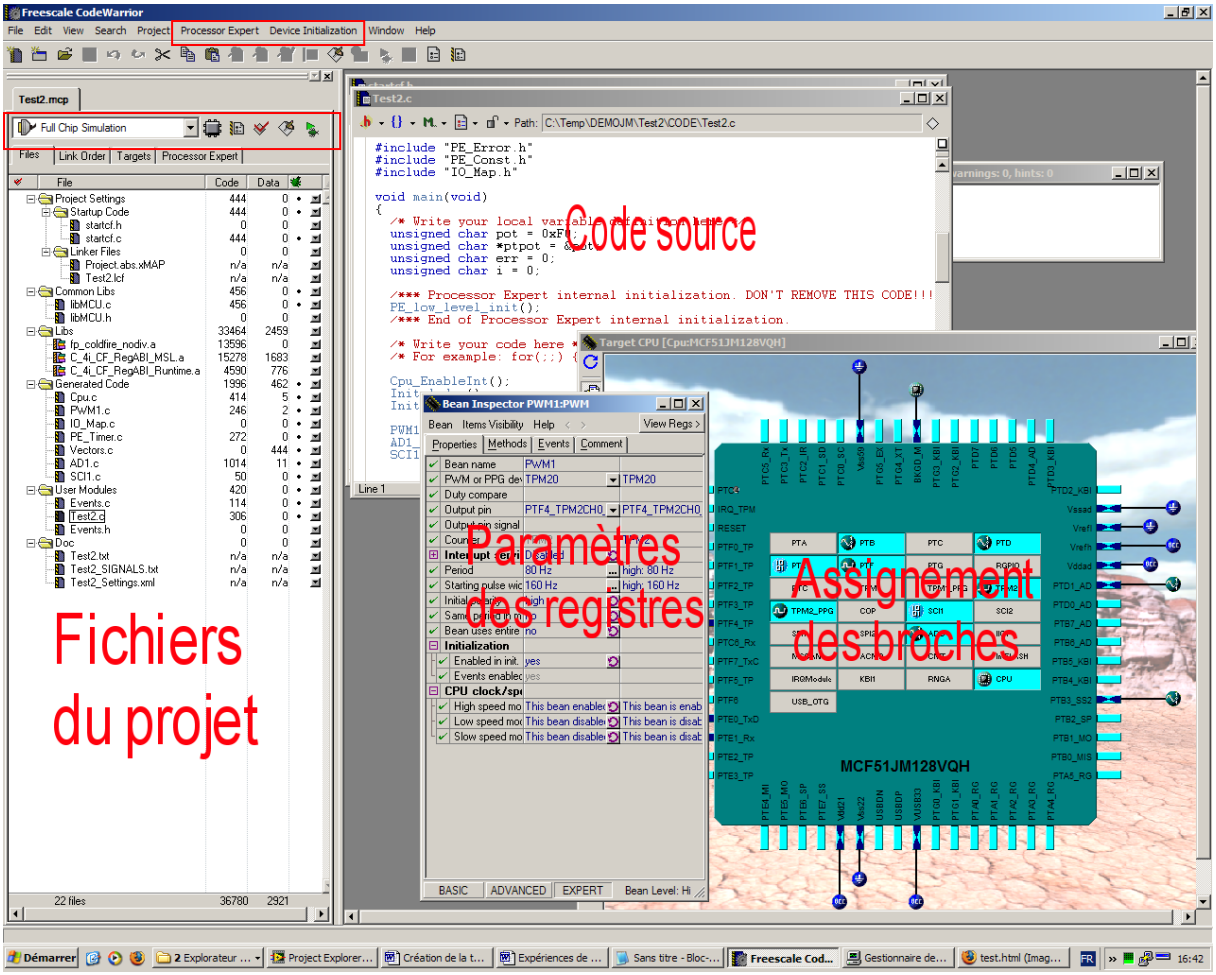
6.4.2. Création d'un projet

- Choisissez la cible. Vous devez aller dans Flexis/JM Family/MCF51JM128 ou dans HCS08/ HCS08JM family/ MC9S08JM60. Choisir comme connexion P&E Multilink/Cyclone Pro. Il vaut mieux choisir le ColdFire par rapport au S08JM, car il est plus performant et plus récent. Mais si vous voulez développer sur les deux plates-formes, choisissez la cible MC9S08, une passerelle est possible dans ce sens vers le MCF51.
- Choisissez un répertoire de travail, ainsi qu'un nom de projet. Essayez d'être le plus claire possible, apres, ça devient vite le fouillis. Comme langage, vous choisissez C, sachant que le C++ n'est pas disponible dans cette version, et que choisir l'assembleur est une tentative de suicide. Suivant.
- Si vous avez des fichiers à ajouter dans votre projet, c'est le moment. A priori, non. De toute façon, vous pouvez toujours le faire plus tard.
- Ensuite, vous avez le choix, pour vous aider à démarrer. Notez que ce choix n'est pas définitif, vous pouvez passer de DI à PE plus tard:
    - None : aucun code ne sera généré, vous pouvez essayer, seulement si vous êtes un puriste, et que vous êtes extrêmement à l'aise avec le µC (à moins que vous ayez déjà écrit vos propres librairies)
    - Device Initialization: La méthode la plus simple pour initialiser votre µC est bien pour une première prise en main.
    - Processor Expert: Est le nec plus ultra de la configuration des registres et de la génération de code. N'est pas forcement facile à prendre en main, mais une fois maitrisé, extrêmement pratique et rapide!
- Choisissez vos paramètres de compilation, et cliquez sur terminer.

Vous avez créé un nouveau projet. Sachez que tout ce que vous avez défini peut être changé après, avec plus ou moins de casse.

6.4.3. L'interface de l'IDE

Nous allons présenter de manière extrêmement rapide l'interface.



C’est rapide, je vous avais prévenue.

6.4.4. Processor Expert et Device Initialization

Note de version : Pour utiliser PE ou DI avec le ColdFire, vous devez au minimum avoir la version 3.03 de ces logiciels. Autrement, vous ne pourrez pas utiliser ce logiciel fort pratique.

Processor Expert, et Device Initialisation sont deux logiciels d’Unis permettant de configurer tous les registres du µC, via une interface graphique simplifiée. Ils ont la même base, seule leur interface graphique diverge.

Ces deux logiciels fonctionnent sur le principe de beans, c'est-à-dire de petits fichiers modulables de description des registres. Il existe donc autant de beans que de µC existant chez Freescale. Il existe un bean aussi pour chaque fonction du µC. L’idéal est d’avoir tous les beans de son µC, mais ils ne sont pas tous fournis et/ou gratuits.

Vous pourrez facilement paramétrer chaque registre en chargeant ces beans. L’avantage est que PE ou DI vous affiche les erreurs, ou les incohérences, vous génère du code tout seul, gère les évènements, les interruptions, les fonctions… Et c’est très simple, une fois compris le principe.

Le seul désavantage est que du code en assembleur est généré

6.4.5. Votre Code

Une fois après s'être amusé avec PE ou DI, il est temps d'écrire votre code. Sachez que tant que PE ou DI sera utilisé, vous ne pourrez pas écrire ou modifier sur le code généré (tout ce que vous écrirez sera perdu). Si vous souhaitez retrouver votre liberté, désactiver PE/DI.

PE/DI génère des fichiers d'information au format texte, qui donne notamment une explication de chaque fonction générée, ainsi que les en-têtes des fonctions. Bien sûr, rien ne vous interdit de créer vous même vos fonctions, et de paramétrer à la main les registres… mais c'est bien plus long.

6.4.6. Changer de cible

Vous pouvez changer de cible, c'est-à-dire passer du MC9S08JM60 au MCF51JM128 sans trop de casse (du moins, c'est prévu pour). Une page expliquant la méthode est disponible dans l'aide de CodeWarrior. Je vous invite à y jeter un coup d'œil.

6.4.7. Test du code

6.4.7.1. Boutons importants

1. Choix de la cible
2. Paramètres du projet
3. Mettre à jour les fichiers modifiés à l'extérieur de l'IDE
4. Compile les fichiers sources, et affiche les warnings et les erreurs.
5. Compile les fichiers sources, affiche les warnings et les erreurs. S'il n'y a pas d'erreurs, il lance HiWave, pour placer le programme sur la cible.

6.4.7.2. Méthode

Une fois que votre portion de code a été écrite, vous devez mettre à jour les dernières modifications des fichiers (très utile dans le cas ou vous utilisez un éditeur de texte externe).

Ensuite, vous lancez la compilation, et vérifiez qu'il n'y a pas d'erreur de syntaxe. Tant que vous avez des warnings, vous pourrez débugger. Si vous avez des erreurs, il faudra absolument les résoudre avant. Vous pouvez régler « la sensibilité » du compilateur dans les options du projet.

Si tout s'est bien déroulé, vous pouvez placer votre programme sur la cible, et utiliser le débugger HiWave.

6.5. Prise en main de HiWave

HiWave s'occupe de toute la partie mise en place du programme sur la cible, ainsi que la partie débogage.

6.5.1. Démarrage

Lorsque vous cliquez sur le bouton « debug » de CodeWarrior, HiWave va se lancer automatiquement. Il va vous demander des paramètres de communication avec la carte. Choisissez les paramètres, s'ils ne sont pas valides. Normalement, vous avez juste besoin de faire « connect », et ça marche. Il va remplacer le programme précédemment mis sur le µC. Si

vous avez des problèmes, et que HiWave n'arrive pas à se connecter, vérifiez les points suivants:

- Vérifier que la daughter card est bien en contact avec toutes les pins
- Vérifier que le câble USB est bien branché (c'est le plus important)
- Vérifiez que vous avez bien choisi le bon µC
- Vérifiez que vous avez bien choisi P&E Multilink/Cyclone Pro. Sinon, ça ne marchera pas.
- Vérifiez que le switch est bien sûr ON
- Le câble noir n'est pas obligatoire pour mettre le programme sur le µC.

Cliquez ensuite sur la flèche verte pour lancer le programme.

6.5.2. Utilisation

HiWave est similaire à tout debugger, vous pouvez avancer en mode pas à pas, en suivant le parcours dans le code source. Pour plus d'informations, vous pouvez vous reporter à l'aide de HiWave.

La carte est indépendante de HiWave, c'est à dire que si vous avez lancé le programme sur le µC, et que vous fermez HiWave, le programme continuera de tourner de manière autonome.