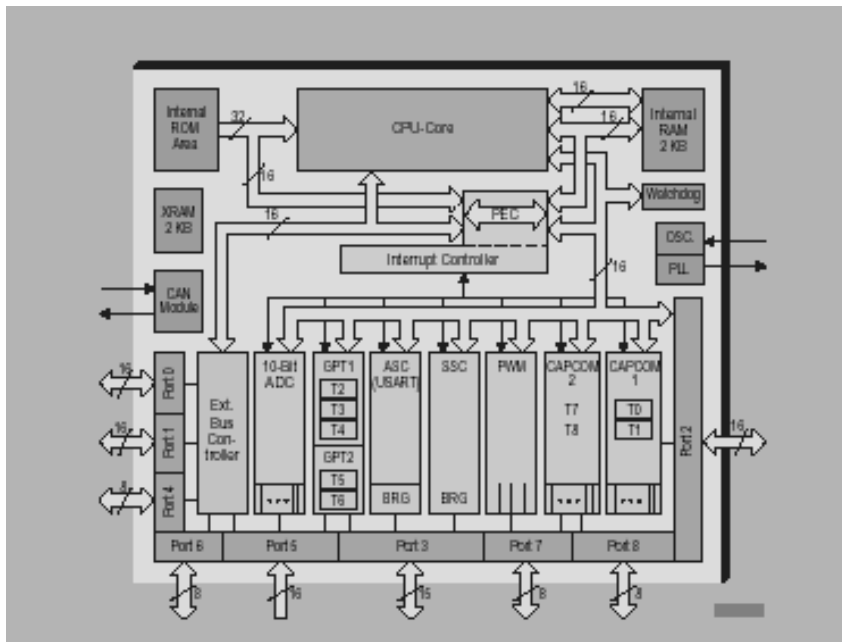


UNIVERSITE PAUL SABATIER

TOULOUSE III

Master 1 STS EEA

TP Micro-contrôleur C167



2M7E12M

2011/2012

Table des matières

Le micro-contrôleur Siemens C167	3
1 La carte micro-contrôleur et son interface	3
1.1 Carte microsys167-Eth	3
1.2 L'interface carte microsys167-Eth ↔ extérieur	3
1.3 Les interruptions	6
2 Environnement logiciel	8
2.1 Compilation et édition de liens	8
2.2 Téléchargement	8
2.3 Mise au point, le débogueur	9
2.4 Accès aux registres et aux bits de l'espace adressable par bits	10
TP1 : Conversion analogique numérique	11
1 Conversion simple	11
2 Conversion échantillonnée	11
2.1	12
2.2	12
3 Emission série	12
3.1 Initialisation de ASC0	12
3.2 Emission de données vers le PC	12
TP2 : Séquenceur programmable	13
1 Séquencement de la commande	14
2 Exécution d'un cycle	14
3 Téléchargement du cycle	14
4 Réglage de la fréquence	15
TP3 : Commande MLI d'un servo-moteur	17
1 Commande simple	18
2 Commande proportionnelle à une tension	18
3 Commande à distance par l'intermédiaire de la ligne série	18
4 Affichage de l'angle	18
5 Commande par boutons poussoirs	18
6 Commande par interrupteurs	19

Le micro-contrôleur Siemens C167

1 La carte micro-contrôleur et son interface

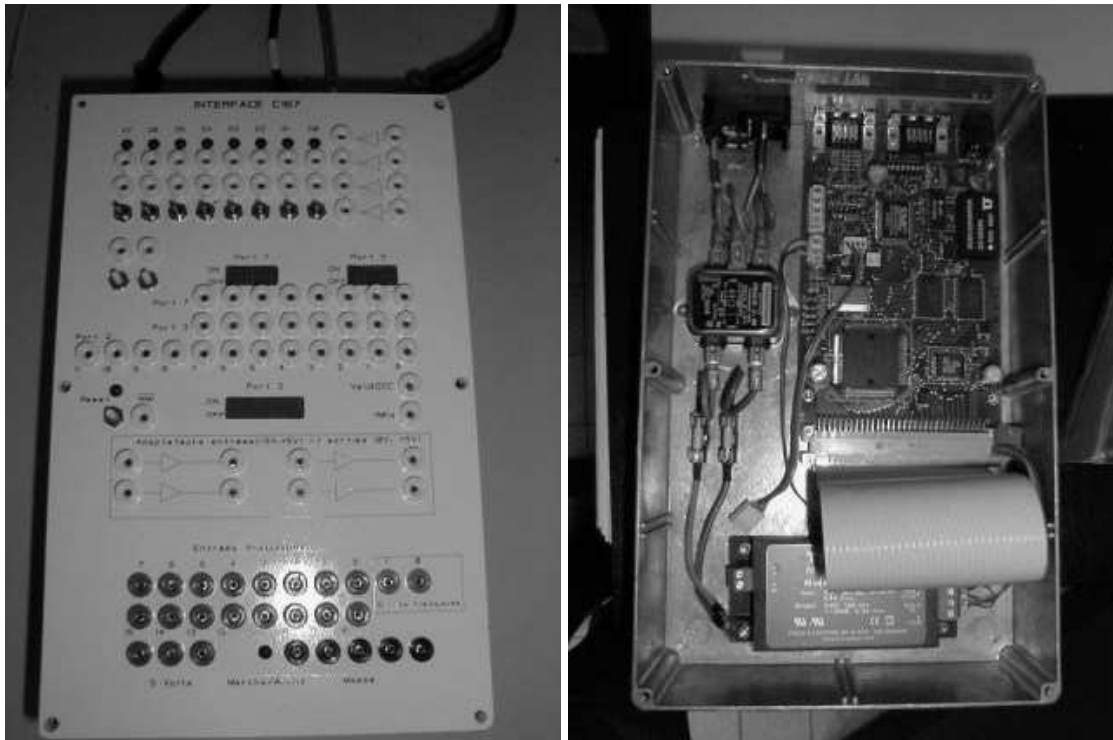
1.1 Carte microsys167-Eth

Cette carte a été conçue par la société Hightec (Allemagne) autour d'un micro-contrôleur Siemens C167 CR cadencé à 20MHz. Sur la carte sont disponibles 1Mo de RAM et 512 Ko de Flash-EPROM et les interfaces de communication suivantes :

- une interface série asynchrone et synchrone (RS232),
- une interface série synchrone (RS485),
- une interface bus CAN (Controller Area Network),
- une liaison ethernet qui sera utilisée pour le téléchargement et la mise au point (débugueur),
- un connecteur externe de 96 points. Un certain nombre de ces points sont ramenés sur la face avant d'un boîtier d'interface (Ports P7, P3, P2 et P5, reset, \overline{NMI}).

La face avant est représentée en fin de présentation de la partie matériel. Les connecteurs correspondant aux interfaces série, CAN, ethernet se retrouvent en face arrière du boîtier d'interface.

1.2 L'interface carte microsys167-Eth ↔ extérieur



Sur la face avant sont disponibles :

- huit interrupteurs, deux boutons poussoirs,
- huit LEDs,
- quatre inverseurs TTL,
- un bouton **reset**, une entrée \overline{NMI}
- une horloge de 1 MHz et son entrée de validation (**ValidOSC**)
- quatre adaptateurs analogiques permettant de ramener une tension $\in [-5V, +5V]$ en une tension $\in [0, 5V]$,
- les ports d'entrée-sortie logiques Port P7 (huit bits), Port P3 (huit bits), Port P2 (douze bits). Ces ports sont programmables en entrée/sortie mais il faut aussi sur l'interface positionner les lignes en entrée/sortie par des interrupteurs deux positions ON/OFF (de type "DIL"). Cette contrainte est liée à l'interface, les interrupteurs pilotent des buffers de bus placés sur les lignes des ports. Les interrupteurs ON(entrée)/OFF(sortie) sont positionnés pour les applications qui seront traitées en TP, il ne sera pas nécessaire de les manipuler.
- seize entrées analogiques correspondant au Port 5,
- deux sorties analogiques 0 et 1, sélectionnées par Chip Select et correspondant respectivement aux adresses 0x2xxxxx et 0x3xxxxx (voir photocopié de cours, chapitre 3) .

Les fonctions alternatives des ports, disponibles sur la face avant, sont rappelées ici.

Port P7

Les huit bits du port sont programmables en entrées/sorties logiques, les fonctions alternatives sont les sorties de l'unité PWM et les entrées capture/sorties comparaison de l'unité CAPCOM2.

Fonction	Sens	Fonction	Sens
P7.0	↔	POUT0	→
P7.1	↔	POUT1	→
P7.2	↔	POUT2	→
P7.3	↔	POUT3	→
P7.4	↔	CC28IO	↔
P7.5	↔	CC29IO	↔
P7.6	↔	CC30IO	↔
P7.7	↔	CC31IO	↔

Port P3

Les broches du Port P3 sont des entrées/sorties logiques ou bien des fonctions d'entrée ou de sortie de différents timers, de l'unité ASC0 (série asynchrone) et de l'unité SSC (série synchrone).

Fonction	Sens	Fonction	Sens
P3.0	↔	T0IN	←
P3.1	↔	T6OUT	→
P3.2	↔	CAPIN	←
P3.3	↔	T3OUT	→
P3.4	↔	T3EUD	←
P3.5	↔	T4IN	←
P3.6	↔	T3IN	←
P3.7	↔	T2IN	←

Pour la signification des fonctions on pourra se rapporter au polycopié de cours. De plus le port P3 est un port 16 bits et les broches P3.10 et P3.11 correspondent aux lignes TxD0 et RxD0 de l'interface série ASC0.

Port P2

Les fonctions alternatives du port P2 sont les entrées capture/sorties comparaison de l'unité CAP-COM1 et les interruptions externes rapides.

Fonction	Sens	Fonction	Sens	Fonction	Sens
P2.0	↔	CC0IO	↔		
P2.1	↔	CC1IO	↔		
P2.2	↔	CC2IO	↔		
P2.3	↔	CC3IO	↔		
P2.4	↔	CC4IO	↔		
P2.5	↔	CC5IO	↔		
P2.6	↔	CC6IO	↔		
P2.7	↔	CC7IO	↔		
P2.8	↔	CC8IO	↔	EX0IN	←
P2.9	↔	CC9IO	↔	EX1IN	←
P2.10	↔	CC10IO	↔	EX2IN	←
P2.11	↔	CC11IO	↔	EX3IN	←

Port P5

Le port P5 fonctionne uniquement en entrée, il est destiné aux entrées analogiques mais les lignes de ce port peuvent être aussi utilisées en entrées logiques ou encore en entrées externes de certains timers.

Fonction	Sens	Fonction	Sens	Fonction	Sens
P5.0	←	AN0	←		
P5.1	←	AN1	←		
P5.2	←	AN2	←		
P5.3	←	AN3	←		
P5.4	←	AN4	←		
P5.5	←	AN5	←		
P5.6	←	AN6	←		
P5.7	←	AN7	←		
P5.8	←	AN8	←		
P5.9	←	AN9	←		
P5.10	←	AN10	←	T6EUD	←
P5.11	←	AN11	←	T5EUD	←
P5.12	←	AN12	←	T6IN	←
P5.13	←	AN13	←	T5IN	←
P5.14	←	AN14	←	T4EUD	←
P5.15	←	AN15	←	T2EUD	←

1.3 Les interruptions

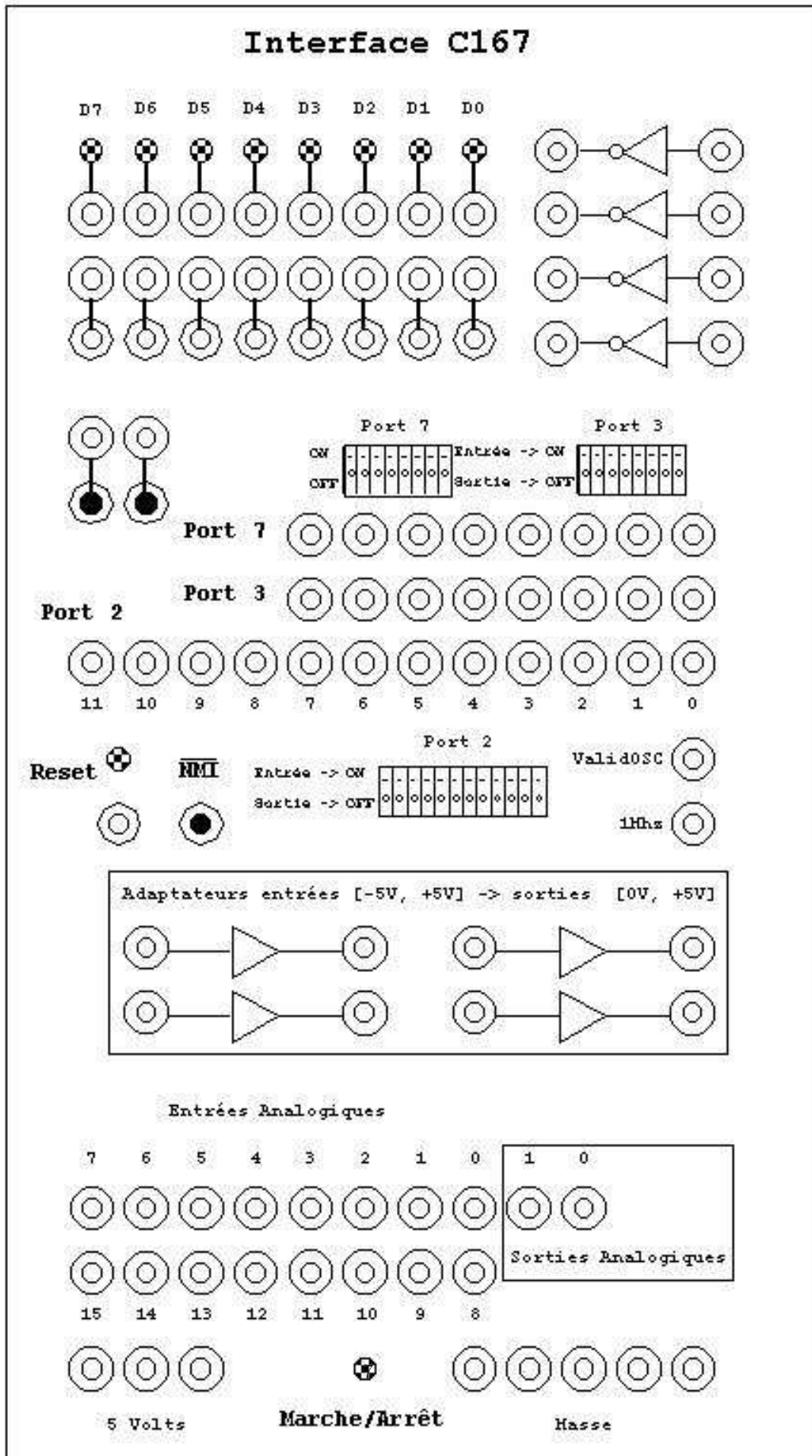
Le C167 offre 56 sources d'interruptions associées à des périphériques. Pour une source d'interruption donnée, notée par exemple source, le registre de contrôle de l'interruption est sourceIC, le bit d'autorisation est sourceIE, et le drapeau d'interruption est sourceIR, à cette interruption est associée un numéro de trappe auquel correspond une adresse dans la table des vecteurs d'interruption (adresse = $4 \times$ numéro de trappe); pour T0 nous aurons respectivement T0IC, T0IE, T0IR, le numéro de trappe est 0x20, l'adresse du vecteur d'interruption est 0x000080.

Nous rappelons ici pour certaines interruptions le numéro de trappe qu'il est nécessaire de connaître pour mettre en œuvre des fonctions d'interruptions.

Source de l'interruption (ou du service PEC)	nom	numéro de trappe
CAPCOM registre 0	CC0	0x10
...
CAPCOM reg. 8 ou interr. externe EX0	CC8	0x18
CAPCOM reg 9 ou interr. externe EX1	CC9	0x19
CAPCOM reg 10 ou interr. externe EX2	CC10	0x1A
CAPCOM reg 11 ou interr. externe EX3	CC11	0x1B
...
CAPCOM registre 31	CC31	0x46
CAPCOM Timer 0	T0	0x20
CAPCOM Timer 1	T0	0x21
CAPCOM Timer 7	T0	0x3D
CAPCOM Timer 8	T0	0x3E
GPT1 Timer 2	T2	0x22
GPT1 Timer 3	T3	0x23
GPT1 Timer 4	T4	0x24
GPT2 Timer 5	T5	0x25
GPT2 Timer 6	T6	0x26
GPT2 CAPREL	CR	0x27
AD Conversion terminée	ADC	0x28
AD Erreur d'écrasement	ADE	0x29
ASC0 Transmission	S0T	0x2A
ASC0 Buffer de transm. transféré	S0TB	0x47
ASC0 Réception	S0R	0x2B
ASC0 Erreur Réception	S0E	0x2C
PWM canal 0..3	PWM	0x3F

Les interruptions externes rapides EX0 à EX7 ont pour vecteur d'interruption le vecteur de l'interruption normale, c'est à dire celui de la voie CAPCOM correspondante (de CC8 pour EX0 à CC15 pour EX7).

Interface C167 – Sérigraphie Face avant -



2 Environnement logiciel

Un chapitre du polycopié de cours a été consacré au système de développement qui sera utilisé en Travaux Pratiques. Ces outils sont des outils classiques GNU adaptés pour le C167 par la société Hightec.

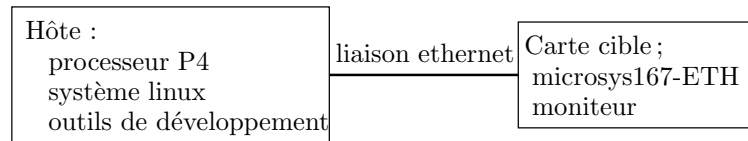


FIG. .1 – Ensemble station de développement et carte cible

Dans le cas de l'installation en salle de TP, une machine linux de nom XXX communique par liaison ethernet avec une carte nommée **mXXX**, par exemple à la machine *tamia* est associé la carte micro-contrôleur *mtamia* (m pour micro-contrôleur).

2.1 Compilation et édition de liens

Le compilateur `gcc166` est invoqué par la commande :

```
gcc166 -Wall -m7 -g -o prog prog.c
(m7 précise que la compilation doit se faire pour une cible C167).
```

2.2 Téléchargement

Le chargement de l'exécutable sur la cible et la mise au point à distance se font par l'intermédiaire du débogueur (`tgdb166`, adaptation du débogueur `gdb`).

Lancement : `tgdb166`

Une fenêtre débogueur s'ouvre (cf. figure page suivante).

Elle présente une barre de menu (**File, Options, Running, ...**) et trois fenêtres :

- une fenêtre source,
- un bandeau de commande : il présente un certain nombre de commandes sous forme d'icônes,
- une fenêtre de travail : entrée des commandes et affichage des résultats des commandes.

La connexion du débogueur sur la machine XXX au moniteur de la carte cible nommée **mXXX** est obtenue par la commande `target` :

```
target c16x mXXX$10200
```

soit sur la machine *tamia* :

```
target c16x mtamia$10200
```






`c16x` désigne le type de carte(C167), `mXXX` le nom de la carte, `$10200` est un service réseau pour le téléchargement.


2.3 Mise au point, le débogueur

Une session de travail type (par exemple sur *tamia* et la carte *mtamia*) se présente ainsi :

- édition du fichier source (`emacs prog.c &` par exemple).
- compilation : `gcc166 -m7 -g -o prog prog.c`
- lancement du débogueur : `tgdb166`

A partir de l'interface graphique de *tgdb* :

- choix du fichier à télécharger
 - Menu **File**, puis **Load Program**, choix de `prog` dans la liste des programmes présentés,
- connexion : `target c16x mtamia$10200` dans la fenêtre inférieure.
 - Après demande de confirmation du chargement, et reset éventuel de la carte cible, le programme sera chargé.
- ce programme doit apparaître dans la fenêtre source, sinon repasser dans **File**, puis **Visit** et choisir le programme `prog.c` ou utiliser l'icône : .
- le programme peut être lancé (icône  du bandeau de commande)
- ou exécuté pas à pas (sans entrer dans les fonctions  ou bien en pas à pas dans les fonctions .
- des points d'arrêt sont placés et supprimés sur un clic du bouton droit de la souris,
- des variables peuvent être visualisées en les sélectionnant et cliquant sur l'icône .

- des variables peuvent être suivies lors de la mise au point dans une fenêtre d'affichage (menu **Windows** puis **Watches**). La sélection des variables se fait par sélection avec la souris, puis clic sur l'icône .

2.4 Accès aux registres et aux bits de l'espace adressable par bits

Un fichier à inclure `c167.h` contient pour tous les registres du micro-contrôleur une déclaration du type suivant :

```
#define registre (* (unsigned int *) adresse)
```

soit un forçage de type de la constante adresse vers un pointeur et une indirection de ce pointeur, le résultat est associé au registre. Ceci permet un accès au registre en écriture et en lecture par une affectation :

- lecture `valeur=registre;`
- écriture `registre=donnee;`

Des macros incluses automatiquement par référence à `c167.h` permettent l'accès aux bits des registres adressables par bits, les principales macros (voir polycopié de cours et travaux dirigés) sont :

- `SET_SFRBIT(BIT)` : met à 1 le bit `BIT`
- `CLR_SFRBIT(BIT)` : met à 0 le bit `BIT`
- `SFR_BIT(BIT)` : renvoie la valeur du bit `BIT`
- `WAIT_UNTIL_BIT_SET(BIT)` attend le passage à 1 du bit `BIT`
- `WAIT_UNTIL_BIT_CLR(BIT)` attend le passage à 0 du bit `BIT`

La mise en place des interruptions s'effectue grâce à des macros contenues dans le fichier `gnutrap.h` :

```
TRAP(num_trap,nom_fonction)
```

```
TRAP_noMD(num_trap,nom_fonction)
```

TP1 : Conversion analogique numérique

1 Conversion simple

On souhaite lire une tension en mode conversion unique sur l'entrée analogique 5 de l'unité ADC. L'algorithme de la fonction de lecture d'une voie peut s'écrire ainsi :

```
début
  mise à zéro du drapeau de fin de conversion (* bit ADCIR *)
  lancer conversion                          (* bit ADST *)
  attendre fin de conversion                 (* bit ADCIR *)
  lire donnée convertie en masquant le numéro de canal (* registre ADDAT *)
fin
```

Dans le programme principal il faut avoir programmé ADCON (mode de fonctionnement et numéro du canal).

Ecrire votre programme et vérifier la pertinence du résultat avec le débogueur : prendre connaissance de la valeur lue sur un point d'arrêt ou bien faire une exécution pas à pas.

La commande `OutputCNA` exécutée sur le PC permet d'obtenir une tension réglable sur la sortie `Vout1` de l'interface. Vérifier pour différentes valeurs de tension le bon fonctionnement de votre programme.

2 Conversion échantillonnée

Le programme suivant permet, en utilisant le timer T3, de générer une interruption toutes les 26ms et d'exécuter la fonction associée.

```
#include <c167.h>
#include <gnutrap.h>

TRAP(0x23,echantillonnage);
void echantillonnage()
{
  /* exécutée toute les 26 ms */
  /* a compléter...          */ }

void main()
{
  T3CON=0x00; /* programmation de T3 - période 26ms*/
  T3IC=0x08; /* IT lié à T3 - Niveau 2 groupe 0 */
  SET_SFRBIT(T3IE); /* autorisation de l'IT sur T3 */
  SET_SFRBIT(T3R); /* activation du timer */
```

```
SET_SFRBIT(IEN); /* autorisation générale des IT */  
  
while(1);  
}
```

2.1

Modifier ce programme pour qu'il fasse 50 acquisitions du signal analogique avec un pas d'échantillonnage de 26ms et les stocke dans un tableau. Chaque acquisition sera faite dans la fonction d'interruption `echantillonnage`.

Tester et vérifier le bon fonctionnement avec le débogueur.

Un signal lentement variable est disponible sur la sortie `Vout1` du boitier d'interface a condition de lancer le programme `signalBF` dans une fenêtre terminal du PC. En reliant la sortie `Vout1` à l'entrée `Vin1` du boitier d'interface et en lançant l'exécutable `oscillo`, on peut par ailleurs visualiser l'allure du signal sur l'écran du PC.

2.2

Modifier le programme de manière à : régler à 125 ms la période de l'interruption (ce qui nécessite un fonctionnement en rechargement de T3)

Pour modifier la période d'échantillonnage :

- programmer T4CON (ou T2CON) en rechargement de T3 sur chaque changement d'état de la bascule T3OTL.
- programmer T3 et T4 (ou T2) avec une valeur permettant d'obtenir la période souhaitée. Rappel : en mode comptage T3 s'incrémente de T4 (ou T2) à 0xFFFF et en mode décomptage, il se décrémente de T4 (ou T2) à 0.

Tester et vérifier le bon fonctionnement avec le débogueur.

3 Emission série

La ligne série du C167 est reliée à la ligne série du PC. L'exécutable `reception` sur le PC permet d'afficher à l'écran les données reçues sur la ligne série.

La transmission s'effectue a 19200 bauds avec des mots de 8 bits sans bit de parité et avec un seul bit de stop.

3.1 Initialisation de ASC0

- Donner l'initialisation de l'unité série ASC0 compte tenu des paramètres de transmission.
- Vérifier le bon fonctionnement en écrivant un petit programme qui envoie un ou plusieurs caractères sur la ligne série du C167 à destination du PC.

3.2 Emission de données vers le PC

Modifier et compléter le programme sur l'acquisition d'un signal analogique par échantillonnage de manière à :

- rajouter la fonction d'initialisation de ASC0.
- convertir en dixièmes de volts les valeurs acquises (stockées comme des entiers).
- calculer la moyenne de 50 acquisitions en dixièmes de volts (valeur entière).
- la transmettre sur la ligne série une fois qu'elle a été calculée.
- Faire reboucler le programme pour réaliser des séquences répétitives de 50 acquisitions.
- calculer les valeurs minimales et maximales (toutes les 50 acquisitions aussi)
- transmettre sur la ligne série les valeurs moyenne, minimum et maximum une fois qu'elles ont été calculées.

TP2 : Séquenceur programmable

Introduction

On souhaite réaliser un séquenceur à l'aide du micro-contrôleur C167. Ce séquenceur devra être programmable à partir d'un PC relié au micro-contrôleur par une ligne série. Sa fréquence de fonctionnement sera par ailleurs réglable à l'aide de deux boutons-poussoirs.

Ce séquenceur sera appliqué à la commande d'une série de leds (appelé couramment chenillard).

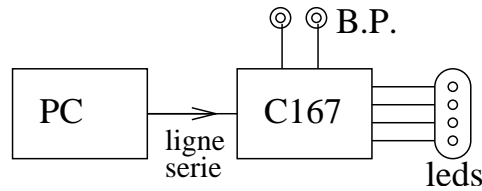


FIG. .1 – Synoptique du dispositif

Le principe du chenillard consiste simplement à allumer et éteindre une série de leds suivant un cycle particulier comme illustré sur le chronogramme suivant :

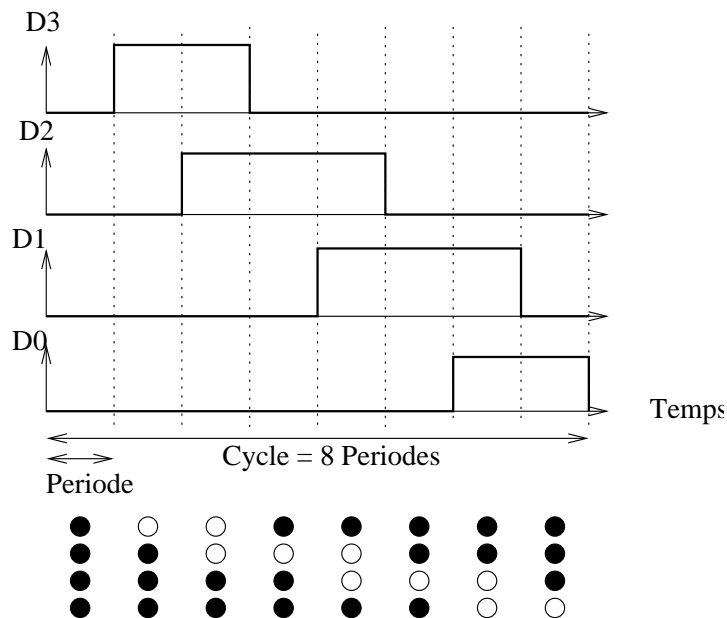


FIG. .2 – Chronogramme de commande des leds

- Les leds commandées seront au nombre de 4 et reliées aux bits 0 à 3 de P3.
- La longueur du cycle sera variable (8 périodes dans l'exemple), chaque période durant 240 ms (cette valeur pourra être modifiée par la suite).
- Un cycle sera stocké dans un tableau d'entiers dont la taille sera fixée par la longueur de la séquence (8 dans l'exemple).
- Les 4 bits de poids faible de chaque élément du tableau coderont l'état des leds correspondantes (0 pour éteint, 1 pour allumé). Par exemple, pour le cycle représenté sur la figure, pour allumer les leds D1 et D2 on utilisera la valeur 0110b soit 6.
- L'ensemble du cycle sera stocké dans un tableau, par exemple pour le chronogramme de la figure 2 on pourra initialiser le tableau ainsi :

```
int cycle[8]={0, 8, 12, 4, 6, 2, 3, 1}
```

1 Séquencement de la commande

On choisit d'allumer ou d'éteindre les leds à la période fixée (240 ms) en utilisant l'interruption périodique provoquée par un timer de l'unité GPT1.

Ecrire la fonction d'initialisation du timer et la fonction d'interruption périodique correspondante. Dans cette fonction on se contentera, dans un premier temps, d'allumer puis d'éteindre une led.

Ne pas oublier l'initialisation du port P3.

2 Exécution d'un cycle

Le cycle de fonctionnement du chenillard étant fixé et stocké dans un tableau, modifier le programme précédent pour que celui-ci exécute ce cycle en commandant les 4 leds.

3 Téléchargement du cycle

Pour modifier le cycle à partir du PC on se propose d'écrire une fonction de réception sur la ligne série qui fonctionne par interruption (exécutée à chaque réception d'une donnée). Cette fonction permet de lire les x octets envoyés du PC vers le micro-contrôleur.

Le protocole de communication suivant est choisi :

- La transmission s'effectue à 19200 bauds avec des mots de 8 bits sans bit de parité et avec un seul bit de stop.
- Le premier octet reçu représente la longueur x du cycle (nombre de périodes).
- Le début de la transmission (réception du premier octet) doit entraîner l'arrêt du chenillard. (On pourra utiliser l'instruction `CLR_SFRBIT(T3R)` pour arrêter le timer et `SET_SFRBIT(T3R)` pour le relancer).
- Les octets suivants reçus représentent la valeur des leds pour une période donnée en commençant par la période 0. Les valeurs des 4 leds (0 : éteinte, 1 : allumée) sont codées dans les 4 bits de poids faibles de l'octet transmis.
- Après réception des x octets, le chenillard repart à partir de la première période du nouveau cycle.

A titre d'exemple, le cycle illustré par le chronogramme correspond à la réception consécutive des valeurs suivantes : **8, 0, 8, 12, 4, 6, 2, 3, 1**

Modifier le programme précédent afin de pouvoir télécharger le cycle envoyé sur la ligne série à partir du PC, et ensuite de l'exécuter.

La longueur du cycle pouvant varier, on utilisera un tableau statique de dimension suffisante (20 par exemple).

Remarque : L'exécutable `emission` lancé dans une fenêtre du PC permet d'envoyer des octets sur la ligne série.

4 Réglage de la fréquence

On souhaite d'autre part modifier la période de base du chenillard en utilisant deux boutons-poussoirs :

- Un bouton permettra de doubler la période à chaque appui (avec une période maximale de 1,92 s),
- l'autre bouton divisera par deux la période avec une période minimale de 60 ms.

On utilisera pour cela deux entrées d'interruption externes de l'unité CAPCOM.

Modifier le programme précédent pour réaliser ce fonctionnement.

TP3 : Commande MLI d'un servo-moteur

Introduction

Un servo-moteur désigne un dispositif utilisé en modélisme constitué d'un petit moteur à courant continu et d'un asservissement de position électronique. Le signal de commande qui pilote le moteur est une consigne de position, le moteur tourne d'un angle proportionnel à la consigne et s'arrête lorsque la position cible est atteinte.

Dans ce TP on s'intéresse à la génération du signal de commande d'un servo-moteur.

Ce signal de commande est un signal $[0, 5]V$ modulé en largeur de période 20ms. Le servo-moteur prend en compte la largeur temporelle de l'impulsion qu'il convertit proportionnellement en un angle. La position 0° correspond à une largeur d'impulsion de 1,5ms ; par rapport à cette largeur une variation de +0.9ms (largeur 2,4 ms) correspond à un angle de $+90^\circ$, une variation de -0.9ms (largeur 0,6 ms) correspond à un angle de -90° . Les figures .1 et .2 illustrent la correspondance entre largeur d'impulsion et angle.

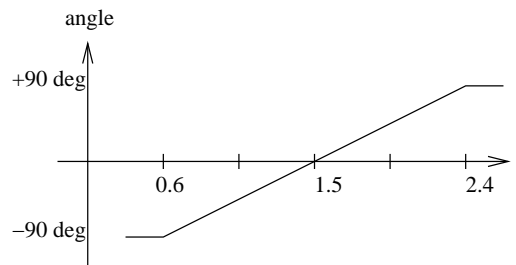


FIG. .1 – Correspondance largeur d'impulsion-angle

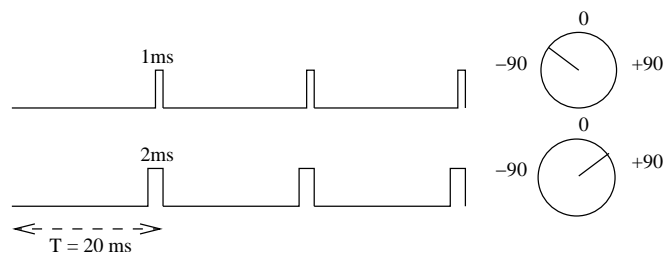


FIG. .2 – Signaux et angles associés

Le signal est dit modulé en largeur d'impulsion (MLI ou PWM)¹ et le signal sera créé à partir de l'unité PWM du C167.

Différents types de commande sont envisagés.

1 Commande simple

Par envoi d'impulsions de largeur fixe à des périodes de 20ms placer l'arbre de sortie du moteur dans les positions correspondant à 0° , $+45^\circ$, $+90^\circ$, -45° , -90° . La consigne d'angle sera changée entre différentes exécutions.

2 Commande proportionnelle à une tension

La maquette comporte un potentiomètre 10 tours, engendrant une tension variable $[0, 5]V$. On veut commander le servo-moteur à partir de cette tension variable lue sur une entrée du convertisseur analogique-numérique (unité ADC) qui sera échantillonnée toutes les 20ms. On considèrera que 0V correspond à la position -90° et 5V à 90° .

Réalisez et testez cette commande.

3 Commande à distance par l'intermédiaire de la ligne série

Un exécutable `com_servo` est fourni côté PC, il permet la saisie d'un angle et son émission sur la ligne série. Cet angle est récupéré sur la ligne série côté C167, sa réception provoquant une interruption. La commande correspondant à l'angle est appliquée sur le servo-moteur après lecture de la consigne. La ligne série asynchrone est configurée à 19200 bits/s, 8 bits de donnée sans bit de parité, un bit de stop

4 Commande par boutons poussoirs

Deux boutons-poussoirs reliés à deux entrées d'interruptions externes du C167 doivent permettre de commander le servo-moteur.

Chaque appui sur le BP de gauche doit faire tourner le servo-moteur de 10 degrés vers la gauche, et chaque appui sur le BP de droite doit le faire tourner de 10 degrés vers la droite.

Quand le servo-moteur arrive en butée (gauche à -90 degrés ou droite à $+90$ degrés) une led doit s'allumer. Elle doit s'éteindre dès que le servo-moteur n'est plus en butée.

Ecrire le programme permettant de réaliser ce mode de commande.

5 Commande par interrupteurs

On remplace les deux boutons-poussoirs de la question précédente par deux interrupteurs reliés à deux entrées logiques du C167.

Le fonctionnement souhaité est le suivant : Pendant l'activation de l'interrupteur de gauche, le servo-moteur doit tourner vers la gauche à la vitesse de 32 degrés par seconde. Il doit tourner à la même vitesse vers la droite pendant l'activation de l'interrupteur de droite. (Si les deux interrupteurs sont activés, le servo-moteur doit rester à l'arrêt).

L'état des deux interrupteurs sera testé périodiquement toutes les 20 ms.

Ecrire le programme permettant de réaliser ce mode de commande.

¹Un signal PWM est un signal dont le rapport cyclique varie de 0 à 1. Pour un servo il faudrait parler de Signal Modulé en Code d'Impulsion (Impulsion Code Modulated Signal).