Rapport de Laboratoire d'électronique digitale de 3BE : Programmation d'un PIC18 en Langage C

Selleslagh Tom, Wéry Benoît

18 décembre 2016

Thème : La notion de convertisseur analogique-numérique

1 Introduction

Le but de ce laboratoire, est de comprendre le concept de conversion analogique/numérique au moyen d'une carte de développement dont le microprocesseur est de la famille des PIC18F.

Dans la quasi-totalité des cas, le monde physique est perçu par des signaux analogiques (Le son, la lumière...). Le traitement de ces signaux peuvent se faire de manière analogique ou numérique. Cependant, le recours au traitement numérique offre plusieurs avantages :

- facilité de stockage de l'information
- reproductibilité des traitements du signal d'entrée.
- réalisation plus aisée de fonctionnalités de traitement complexe
- ...

L'interface couramment utilisée pour faire le lien entre ces deux monde est le convertisseur analogique numérique (CAN) ou Analogic Digital Converter (ADC) en anglais.

2 Principe Générale

Le principe de ce laboratoire est d'utiliser l'ADC que contient le PIC18F et de convertir la tension d'entrée en une valeur numérique codée sur 10 bits stockée dans une variable de 16 bits. Il sera donc important de spécifier s'il l'on désire aligner nos bits à gauche ou à droite.(cf FIG 1)

2.1 paramètres de l'ADC

Tensions de références

Borne de tension entre lesquelles le PIC18F convertira le signal d'entrée.(Dans ce laboratoire nous choisirons V_{SS} et V_{DD} comme bornes de tensions)

FIGURE 1 – Alignement des 10 bits encodée dans la variable

Temps de conversion analogique → digitale (T_{AD})

le T_{AD} est le temps minimum nécessaire à l'ADC pour mesure un bit. Le T_{AD} se configure comme un multiple en puissance de 2 de la période d'oscillation de la clock du PIC18F.

Temps d'acquisition (T_{ACQ})

le T_{ACQ} est le temps minimum nécessaire pour que le condensateur CHOLD soit charger. le T_{ACQ} se configure comme un multiple de 2 du T_{AD}.

3 Programmation

Ne seront présentés dans cette section que le paramétrage de l'ADC, la codage de son utilisation sera explicité dans la section 4

Il existe deux méthodes pour configurer ces registres :

3.1 Sans Libraire

Il s'agit de modifier directement les bits dans les registres. Le paramétrage de l'ADC se fera sur le registres ADCON0, ADCON1 et ADCON2.

Tension de référence : on configure pour que les tensions de références sont V_{SS} et V_{DD}

```
ADCON1bits.VCFG0 = 0;
ADCON1bits.VCFG1 = 0;
```

Alignement des bits encodé : alignement à droite.

```
ADCON2bits.ADFM = 1;
```

T_{AD} : paramétrage pour un $T_{AD} = \frac{F_{OSC}}{16}$

```
ADCON2bits.ADCS2 = 1;
ADCON2bits.ADCS1 = 0;
ADCON2bits.ADCS0 = 1;
```

T_{ACQ} : paramétrage pour un $T_{ACQ} = 6\ T_{AD}$

```
ADCON2bits.ACQT2 = 0;
ADCON2bits.ACQT1 = 1;
ADCON2bits.ACQT0 = 1;
```

Choix de l'entrée analogique : dans cette configuration on choisi l'entrée AN0 que l'on place en "mode" analogique.

```
ADCON0bits.CHS0 = 0;
ADCON0bits.CHS1 = 0;
ADCON0bits.CHS2 = 0;
ADCON0bits.CHS3 = 0;
ADCON1bits.PCFG0 = 0;
```

3.2 Avec la libraire

en utilisant la libraire XC8 nous pouvont utiliser la fonction **OpenADC()** et **SetChanADC()** pour réaliser les opérations ci-dessus.

```
OpenADC(ADC_FOSC_16 & ADC_RIGHT_JUST & ADC_6_TAD,
           ADC_REF_VDD_VSS,
           0);

SetChanADC(ADC_CH0);
ADCON1bits.PCFG0 = 0;
```

4 Notion avancées

L'utilisation de l' ADC peut trouver son utilitée dans la modulation d'une PWM (Pulse Width Modulation) ou MLI (Modulation de largeur d'impulsions) en français.

Le principe général est qu'en appliquant une succession d'états discrets pendant des durées bien choisies, on peut obtenir en moyenne sur une certaine durée n'importe quelle valeur intermédiaire. un exemple pour une tension d'entrée de 3V vous est présenté à la FIGURE 2.

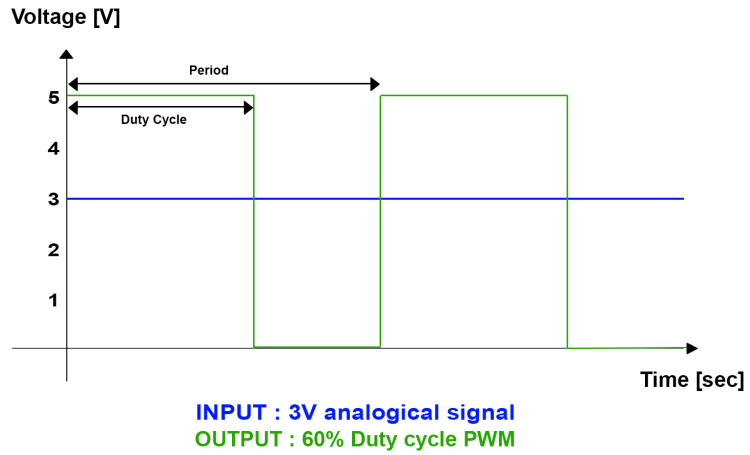

FIGURE 2 – Transformation d'un signal analogique en signal PWM de tension équivalente

En se servant de la librairie XC8. Il est très facile d'utiliser le signal numérique de sortie de l'ADC pour créer une PWM. Voici un exemple pour la création d'une PWM :

Création de la PWM paramétrage de la période [1] et du mode.

```
OpenPWM1(0xFF);
SetOutputPWM1(SINGLE_OUT, PWM_MODE_1);
```

Conversion On suppose l'ADC correctement paramétré (voir Section 3).

```
ConvertADC();
while(BusyADC());
value = ReadADC();
```

1. le calcul de la durée de la période de la PWM ce fait grâce à la formule $period = [(period)+1]x4xTOSCxTMR2prescaler$, cette valeur sera renseignée comme paramètre de la fonction OpenPMMx()

on *redirige* notre valeur vers la PWM .

```
SetDCPWM1(value);
```

5 Conclusion

Annexes