Rapport de laboratoire d'électronique digitale de 3BE : Programmation d'un PIC18 en langage C

Selleslagh Tom, Wéry Benoît

21 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 peut 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 des fonctionnalités de traitements complexes
- ...

L'interface couramment utilisée pour faire le lien entre ces deux mondes est le Convertisseur Analogique Numérique (CAN) ou Analogic Digital Converter (ADC) en anglais.

2 Principe général

L'objectif 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 et stockée dans une variable de 16 bits. Il sera donc important de spécifier si l'on désire aligner nos bits à gauche ou à droite.(cf FIG 1)

2.1 Paramètres de l'ADC

Tensions de références

Bornes de tension entre lesquelles le PIC18F convertira le signal d'entrée. Si la tension mesurée se situe entre les deux bornes, alors la valeur digitale retournée sera calculée proportionnellement aux valeurs binaires extrêmes.(Dans ce laboratoire nous choisirons V_{SS} et V_{DD} comme tensions de références basse et haute)

FIGURE 1 – Alignement des 10 bits encodés dans la variable sur 16 bits

Temps de conversion Analogique → Digital (T_{AD})

Le T_{AD} est le temps minimum nécessaire à l'ADC pour mesurer un bit. Le T_{AD} se configure comme un multiple en puissance de 2 de la période d'oscillation de la clock du PIC18F ($T_{osc} = 25ns$).

Temps d'acquisition (T_{ACQ})

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

3 Programmation

Nous ne présenterons ici que le **paramétrage** de l'ADC, le codage de son utilisation sera explicité au moyen d'un exemple dans la section 4

Il existe deux méthodes pour configurer les registres de l'ADC :

3.1 Sans libraire

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

Tensions de références : on configure pour que les tensions de références soient V_{SS} et V_{DD}

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

Alignement des bits encodés : 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 pouvons utiliser les fonctions **OpenADC()** et **SetChanADC()** pour réaliser ces mêmes opérations.

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

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

4 Notions avancées

Un exemple d'utilisation de l' ADC est la gestion des signaux PWM (Pulse Width Modulation) ou MLI (Modulation de Largeur d'Impulsions) en français.

Le principe général de la modulation PWM est qu'en appliquant une succession d'états discrets pendant des durées bien définies, on peut générer des signaux continus. La valeur moyenne du signal obtenu est alors égale au rapport cyclique du module PWM : rapport cyclique $= \frac{\text{Temps haut}}{\text{Période}}$. Un exemple pour une tension d'entrée de 3V vous est présenté à la FIGURE 2.

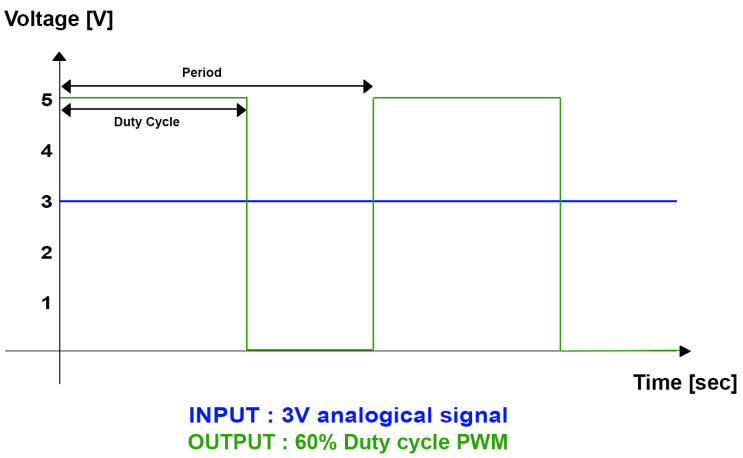

FIGURE 2 – Transformation d'un signal analogique en signal PWM de tension équivalente

Une application concrète de ces notions est le contrôle d'un moteur DC au moyen d'un signal PWM. Le moteur est placé dans un montage en pont H dont l'ouverture et la fermeture des interrupeurs est commandée par le signal PWM (voir FIGURE 3). Afin de gérer le sens de rotation du moteur ainsi que sa vitesse, on utilise un potentiomètre et un convertisseur Analogique-Numérique. Celui-ci aura alors pour fonction de traduire la valeur analogique de la tension lue sur le potentiomètre en un rapport cyclique pour le module PWM. Ainsi, par exemple, si le potentiomètre est mis sur la résistance minimale, l'ADC convertira cette valeur en un rapport cyclique minimum et le moteur sera à basse vitesse [1].

Utilisation du signal numérique de sortie de l'ADC pour créer une PWM, via les librairies du compilateur XC8 :

1. Nb : nous ne développeront cependant pas le code pour ce genre d'application, car il s'agit d'une utilisation directe d'une PWM, mais indirecte d'un ADC.

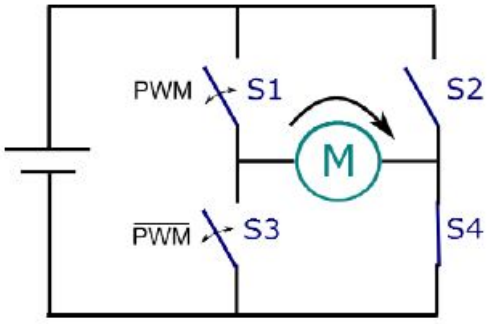

FIGURE 3 – Contrôle d'un moteur DC via un montage en pont H

Création de la PWM paramétrage de la période [2]

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

Conversion On suppose l'ADC correctement paramétré pour lire la tension sur le potentiomètre (voir Section 3). Les fonctions **ConvertADC()**, **BusyADC()** et **ReadADC()** permettent respectivement :

- de demander à l'ADC de lancer une mesure
- de vérifier si il a fini de convertir la tension analogique
- de lire le résultat de la conversion

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

Redirection de la sortie numérique de l'ADC vers la PWM .

```
SetDCPWM1(value);
```

Fermeture de l'ADC à la fin du programme .

```
CloseADC();
```

5 Conclusion

Comme nous le contastons via l'application du moteur DC et comme nous l'avons exprimé précédemment, le CAN permet d'exploiter très facilement une grandeur analogique, en la traitant de manière numérique via des microcontrôleurs par exemple.

Ses utilisations son nombreuses et variées, et les grandeurs analogiques peuvent être mesurées directement dans un circuit ou captées par une antenne par exemple. Bien souvent, le CAN s'accompagne d'un Convertisseur Numérique Analogique qui permet alors de générer une nouvelle grandeur continue après voir effectué un travail numériquement. C'est le cas du traitement des signaux en télécommunication comme nous le voyons actuellement.

2. le calcul de la durée de la période de la PWM se fait grâce à la formule $period = [(period) + 1]x4xTOSCxTMR2prescaler$. Cette valeur sera renseignée comme paramètre de la fonction OpenPMMx() et du mode.

Annexes

Sources :

— `https://fr.wikipedia.org/wiki/Convertisseur_analogique-num.`
— Note de laboratoires 2 et 3 d'électroniques Numériques, ECAM , 2016.