

## TP N° 04    Conversion Analogique-Numérique

### 1. Objectif

Dans ce TP expérience, nous allons apprendre à utiliser le module Conversion Analogique Numérique (ADC pour Analog to Digital Converter) du pic 16f877. Le signal à convertir peut-être une tension d'un capteur analogique qui sera simulé par un potentiomètre.

### 2. Introduction

Un convertisseur analogique-numérique permet de faire l'acquisition de signaux issus du monde extérieur et de les traduire sous forme numérique. Son rôle est donc de convertir une tension  $V$ , généralement comprise entre  $V_{ref-}$  et  $V_{ref+}$ , (tensions de référence). La valeur numérique issue de la conversion est comprise entre  $0$  et  $2^N - 1$  avec  $N$  le nombre de bits de résolution du convertisseur.

Le convertisseur analogique numérique du PIC16F877 est à approximations successives et il possède une résolution de 10 bits. Il est composé de :

- Un multiplexeur analogique à 8 entrées.
- Un échantillonneur bloqueur.
- Un Convertisseur Analogique Numérique de 10 bits.

La valeur Numérique résultante de la conversion est égale à :

$$N \text{ (valeur numérisée)} = ((V_{IN} - V_{ref-}) / (V_{ref+} - V_{ref-})) * 1023$$

Si  $V_{ref+} = V_{DD} = 5V$  et  $V_{ref-} = V_{SS} = 0V$  alors

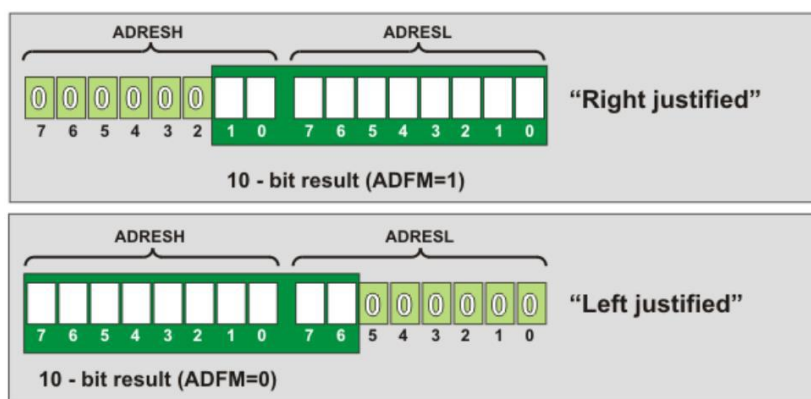
$$N \text{ (valeur numérisée)} = 1023 * (V_{IN} / 5)$$

La résolution du convertisseur est alors donnée par :

$$\text{Résolution} = (V_{ref+} - V_{ref-}) / 1023 = 5 / 1023 = 0.004887V$$

Le module ADC est doté de 4 registres : ADRESH, ADRESL, ADCON0, et ADCON1 :

- Les deux registres ADRESL et ADRESH Contiennent successivement les poids faibles les poids fort du résultat de conversion ;



- Les deux registres ADCON0 et ADCON1 permettent de configurer le module ADC, comme le montre le datasheet du microcontrôleur.

**REGISTER 11-1: ADCON0 REGISTER (ADDRESS: 1Fh)**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits

00 =  $F_{OSC}/2$

01 =  $F_{OSC}/8$

10 =  $F_{OSC}/32$

11 = FRC (clock derived from the internal A/D module RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits

000 = channel 0, (RA0/AN0)

001 = channel 1, (RA1/AN1)

010 = channel 2, (RA2/AN2)

011 = channel 3, (RA3/AN3)

100 = channel 4, (RA5/AN4)

101 = channel 5, (RE0/AN5)<sup>(1)</sup>

110 = channel 6, (RE1/AN6)<sup>(1)</sup>

111 = channel 7, (RE2/AN7)<sup>(1)</sup>

bit 2 **GO/DONE:** A/D Conversion Status bit

If ADON = 1:

1 = A/D conversion in progress (setting this bit starts the A/D conversion)

0 = A/D conversion not in progress (this bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1 **Unimplemented:** Read as '0'

bit 0 **ADON:** A/D On bit

1 = A/D converter module is operating

0 = A/D converter module is shut-off and consumes no operating current

**REGISTER 11-2: ADCON1 REGISTER (ADDRESS 9Fh)**

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit

1 = Right justified. 6 Most Significant bits of ADRESH are read as '0'.

0 = Left justified. 6 Least Significant bits of ADRESL are read as '0'.

bit 6-4 **Unimplemented:** Read as '0'

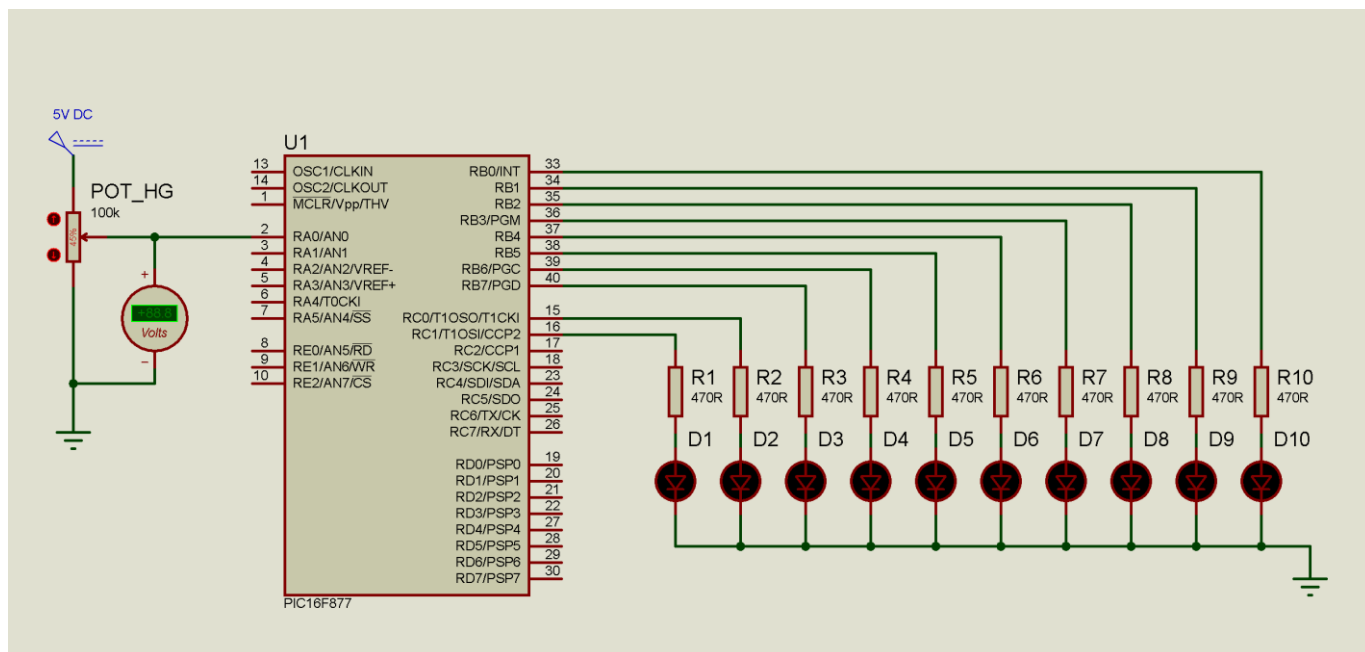
bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN7 <sup>(1)</sup> RE2	AN6 <sup>(1)</sup> RE1	AN5 <sup>(1)</sup> RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs <sup>(2)</sup>
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Analog input    D = Digital I/O

### 3. Manipulation

**3.1.** Dans cette partie, nous allons lire tension variable à la sortie d'un potentiomètre par l'ADC du microcontrôleur puis on affiche le résultat de conversion sous forme numérique par des LEDs placées sur les ports B et C.



```

void main( void )
{
    unsigned int Donnee;           //Déclaration des variables.

    TRISB = 0; TRISC = 0;         //Initialisation des ports.
    TRISA = 0x01 ;                // RA0 en entrée
    ADCON0 = 0 ; ADCON1 = 0x0E ;

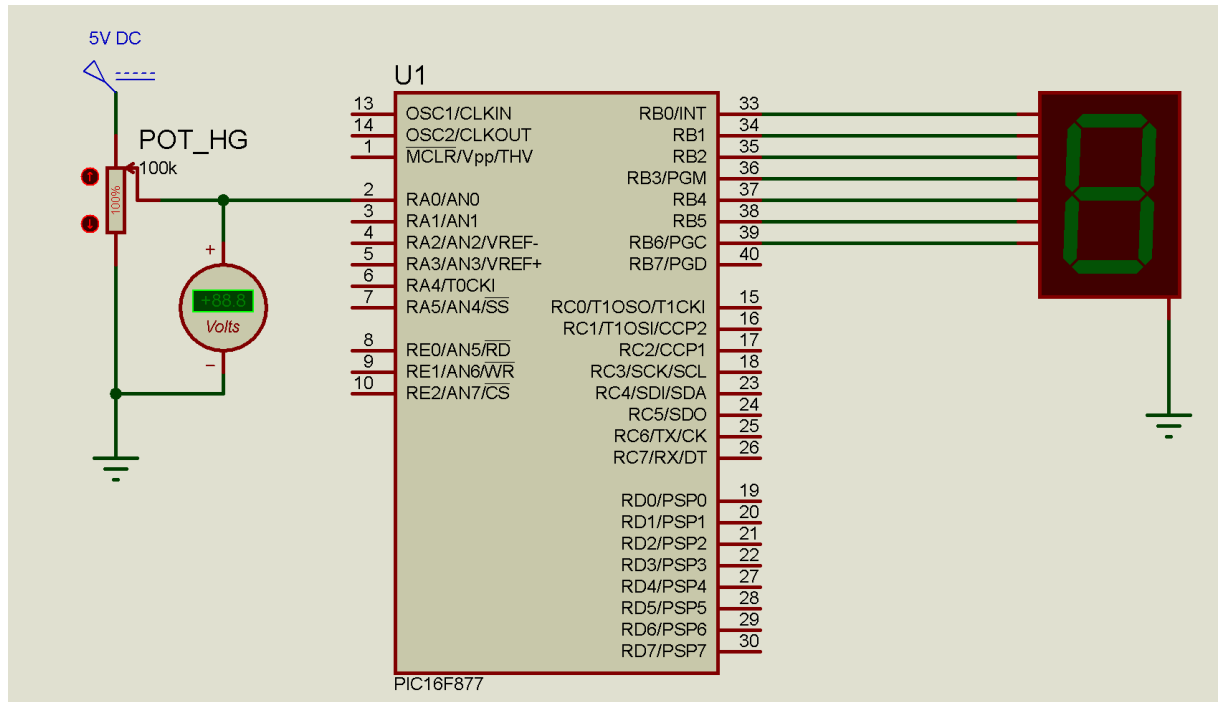
    while(1)                      //Boucle infinie.
    {
        Donnee = ADC_Read(0);     //Conversion sur le canal 0.
        PORTB = Donnee;           //Sortir les 8 bits de poids le plus faible sur le port B.
        PORTC = (Donnee>>8);     // Sortir les 2 bits de poids le plus fort sur le port C.
    }
}

```

- Sous ISIS, réaliser le circuit ci-dessus, Ecrire le programme sur mikroC PRO, générer le fichier **.hex** et lancer la simulation.
- Faite varier la position de potentiomètre puis relever le la tension d'entrée et la valeur numérique correspondante, et remplissez le tableau et notez vos remarques.
- Tracer la valeur numérique en fonction de la tension d'entrée.

Tension d'entrée	Valeur Numérique Binaire	Valeur Numérique Décimale	Tension Equivalente à la valeur Numérique
0 V	0000000000	0	0 V
0.5 V	0001100110	66	0.32 V
⋮	⋮	⋮	⋮
5 V			

**3.2.** Dans cette partie, nous allons lire tension variable à la sortie d'un potentiomètre par l'ADC du microcontrôleur puis on affiche le résultat de conversion sous forme décimale par un afficheur 7 segments.



```

char unit ;
unsigned long int temp ;
char tab[9] = { 0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x07, 0x7f, 0x6f } ; //code 7seg

void main() {
    PORTC = 0;      PORTB = 0;
    TRISC = 0;      TRISB = 0 ;
    TRISA = 0x01 ; // entrée CAN
    ADCON0 = 0 ; ADCON1 = 0x8E ;

    while(1) {
        Delay_ms(10);
        temp = Adc_Read(0);
        unit = floor(temp*5/1000) ; // extrait les unité
        PORTB = tab[unit] ; // afficher les unité
    }
}

```

- Simuler le circuit avec le code donné.
- En se basant sur ce code, ajouter deux autres afficheur 7 segments sur le PortB, afin d'afficher en claire la valeur de la tension qui est sur le curseur du potentiomètre.