CONFIGURATION DU PIC16F1825 :

On utilise l'oscillateur interne et on désactive le watchdog

CONFIGURATION DE L'ADC

Pour configurer le convertisseur, j'ai suivi la procédure décrite dans la datasheet **chapitre 16.1**.

16.1 ADC Configuration

When configuring and using the ADC the following functions must be considered:

- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Result formatting

Le détail de chaque ligne est commenté sur le code.

FONCTION ADC_read :

- Commencer par déterminer un temps d'attente correspondant au temps d'acquisition des données avant conversion. Ce temps d'attente se calcule d'après une formule dans la datasheet **chapitre 16.3**

16.3 A/D Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 16-4. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to Figure 16-4. **The maximum recommended impedance for analog sources is 10 kΩ.** As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an A/D acquisition must be done before the conversion can be started. To calculate the minimum acquisition time, Equation 16-1 may be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

EQUATION 16-1: ACQUISITION TIME EXAMPLE

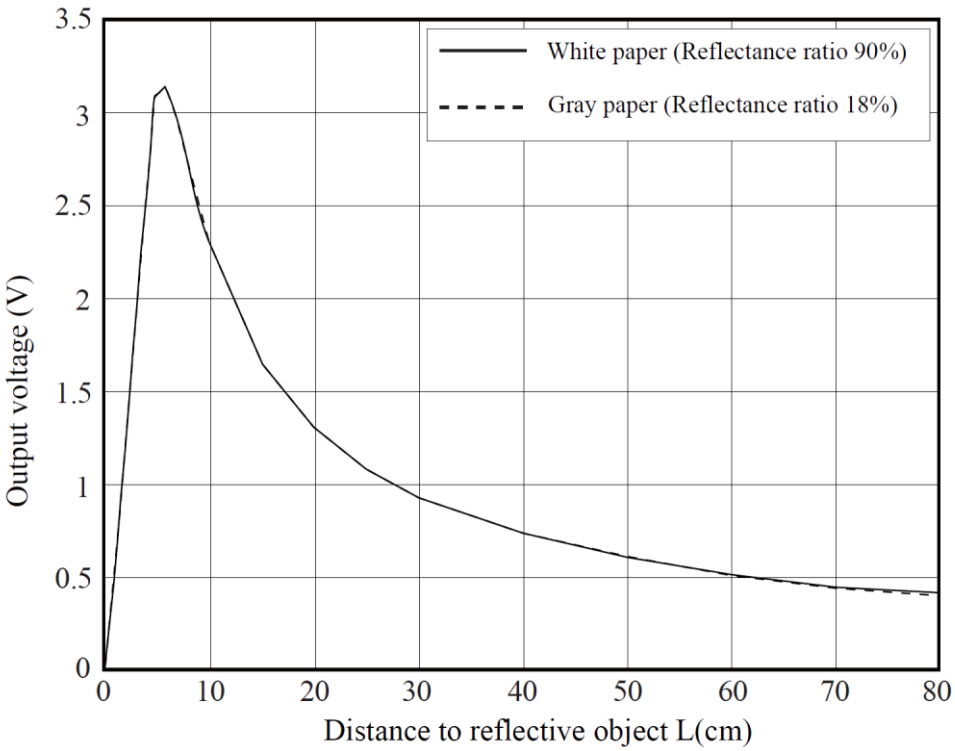
Assumptions: $Temperature = 50°C$ *and external impedance of* $10k\Omega$ $5.0V$ V_{DD}

$$\begin{aligned} T_{ACQ} &= Amplifier\ Settling\ Time + Hold\ Capacitor\ Charging\ Time + Temperature\ Coefficient \\ &= T_{AMP} + T_C + T_{COFF} \\ &= 2\mu s + T_C + [(Temperature - 25°C)(0.05\mu s/°C)] \end{aligned}$$

- On lance la conversion et on attend que celle-ci se finisse.
- On additionne les bits de poids forts et de poids faibles pour avoir la combinaison de 10 bits en sortie du Convertisseur Analogique Numérique.

CALCUL DISTANCE SHARP :

Le **SHARP GP2Y0A21YK0F** renvoie une tension selon la distance qui suit cette courbe caractéristique.

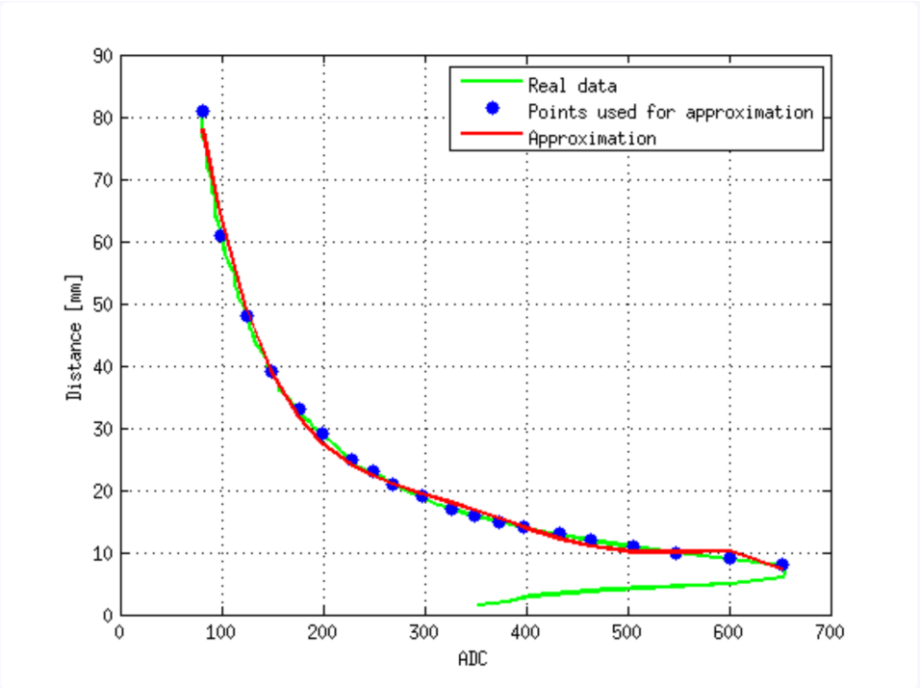


Pour obtenir la distance à partir de la tension, il va falloir tout d'abord numériser le signal de tension à l'aide du convertisseur analogique numérique.

Le convertisseur donne en sortie un mot de 10 bits.

On choisit une plage de tension d'entrée allant de 0 V et 5 V, soit la masse et la tension d'alimentation du PIC16F1825.

On a un pas q = (5 – 0) / 1024 = 4.88 mV.



On retrouve donc la même allure, mais ici nous avons la distance en fonction du signal numérisé en mots de 10 bits.

Une fois le signal numérisé il est exploitable pour pouvoir calculer la distance associée.

On déduit une formule à partir du graphique.

On peut effectuer une linéarisation ou même faire une approximation polynomiale

J'ai choisi la formule : dist = (3027,4/N) ^ 1.2134 avec : dist : la distance mesurée,
N : la valeur du mot en sortie de CAN

Cette formule approxime bien la courbe dans l'intervalle 10cm - 80cm, qui correspond à la plage de mesure du **SHARP GP2Y0A21YK0F.**

EXECUTION DU PROGRAMME :

Dans le programme, on active le Convertisseur Analogique Numérique.
Et au sein d'une boucle while, on va récupérer la valeur N = ADC_read() qui correspond à la sortie du CAN puis calculer la distance avec la formule associée.
A chaque itération de la boucle while, la valeur du CAN s'actualise et la distance s'actualise donc aussi.

PROBLEMES :

La formule de type float utilise des nombres flottants qui demandent bien plus de mémoire que le PIC16F1825 n'en a. Le programme ne compile pas à cause de ce manque de mémoire.