



## Test des entrées RAPE sur MPLABX

Nous nous proposons dans ce cours de charger le programme du test des entrées moteurs puis de les tester sur le robot RAPE enfin de contrôler que toutes les codeurs fonctionnent correctement.

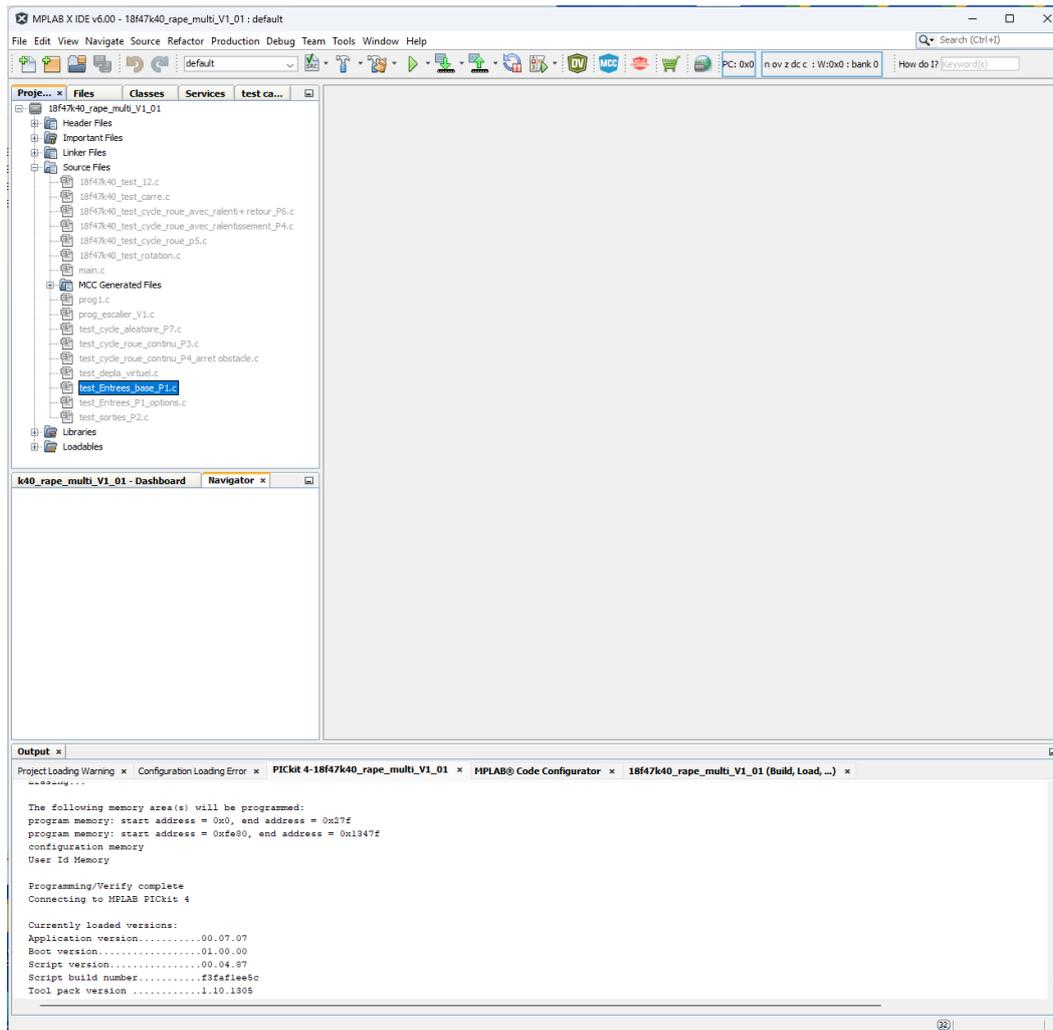
### Table des matières

Test des entrées RAPE sur MPLABX .....	1
Charger le programme des entrées .....	2
Description du programme : .....	4
Explication sur le programme test des entrées : .....	4
Connexions nécessaires.....	5
Programme : .....	5
Test pratique sur le robot. ....	12

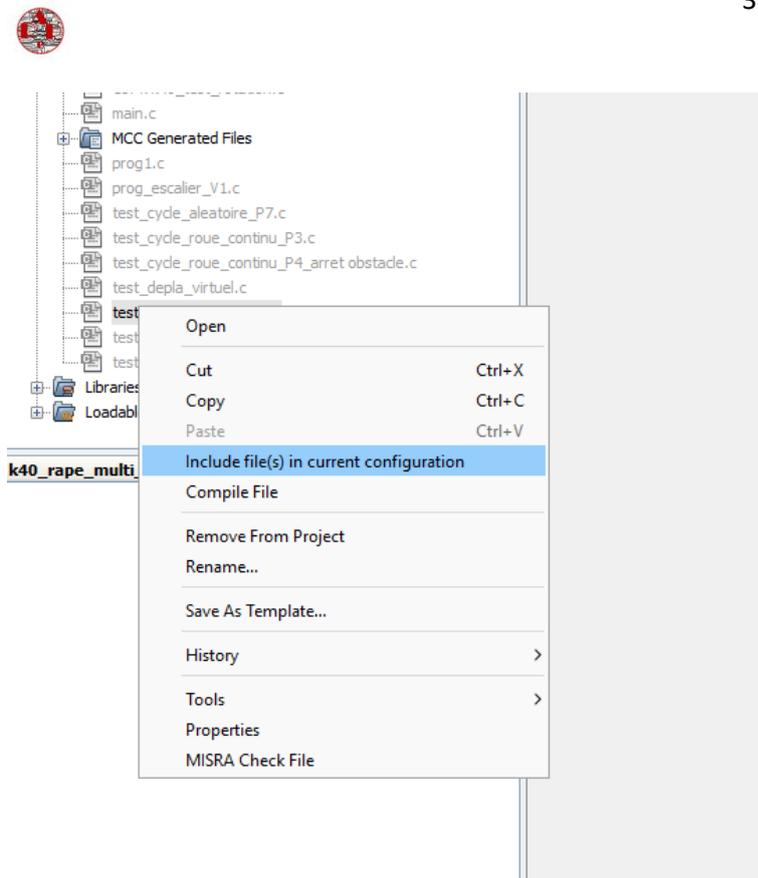
**La vidéo associée :**

11/2023

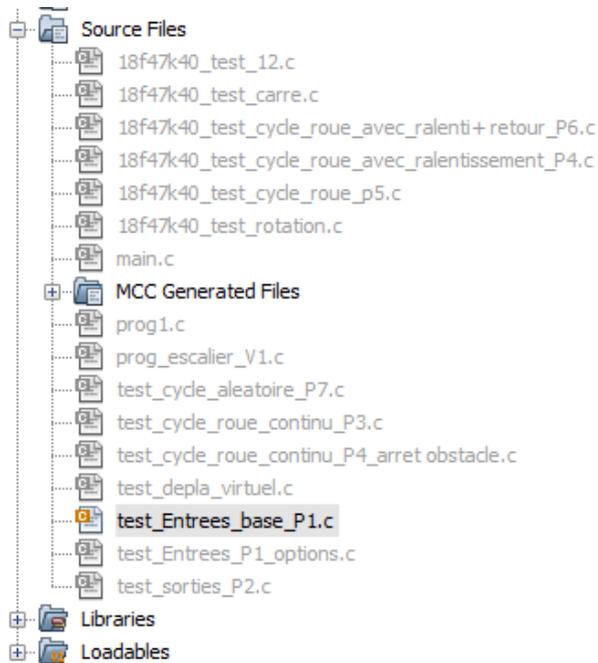
## Charger le programme des entrées



« test\_Entrees\_base\_P1 »



Le programme passe en gras





11/2023

Cliquez deux fois dessus :

```
1  /**
2  * RAPE
3  *
4  * sur carte MAIN_V01-01_pcb3-2023
5  * robot RAPE à programmer avec MPLAB et pic18f47k40
6  *
7  * PROGRAMME N°1
8  * premier essaï
9  * PROGRAMME de test des Entrées sur le schéma de base uniquement
10 * la led verte doit clignoter une fois le programme injecté
11 * l'affichage sur pc fonctionne et permet de voir les compteurs roues et les tensions ana
12 *
13 *
14 * La led verte doit clignoter en permanence sous les 100ms dans 0.1s
15 * l'affichage doit être présent sur l'écran du pc avec la liaison RS232 stable
16 *
17 * *****entrées TOR :
18 * les trois BP un appui led bleue
19 * le bouton avant
20 * lorsque l'on fait tourner les roues à la main les compteurs s'incrémentent à voir sur l'affichage.
21 *
22 *
23 * *****entrées analogiques :
24 * cellule centrale tension variable de 300 à 650.
25 * tension batterie (on retire le fil du bornier) on met en série une résistance de 10ohm on lit 500 sur l'entrée ana on rampe le fil on a
26 *
27 * la consommation à vide est de 60mA
28 *
29 */
30
31 #include "mcc_generated_files/mcc.h"
32 #include <stdio.h>
33 #include <xc.h>
34 #include <stdint.h>
35 #include "mcc_generated_files/examples/i2c1_master_example.h" // sans ça pas de fonctionnement du I2C.....;
36 #include <math.h>
37
38
39
```

Description du programme :

Dans ce petit programme de « test des entrées » nous allons tester les entrées liées aux moteurs.

Eventuellement allez voir sur le cours « **1-4 structure d'un programme RAPE** »

Explication sur le programme test des entrées :

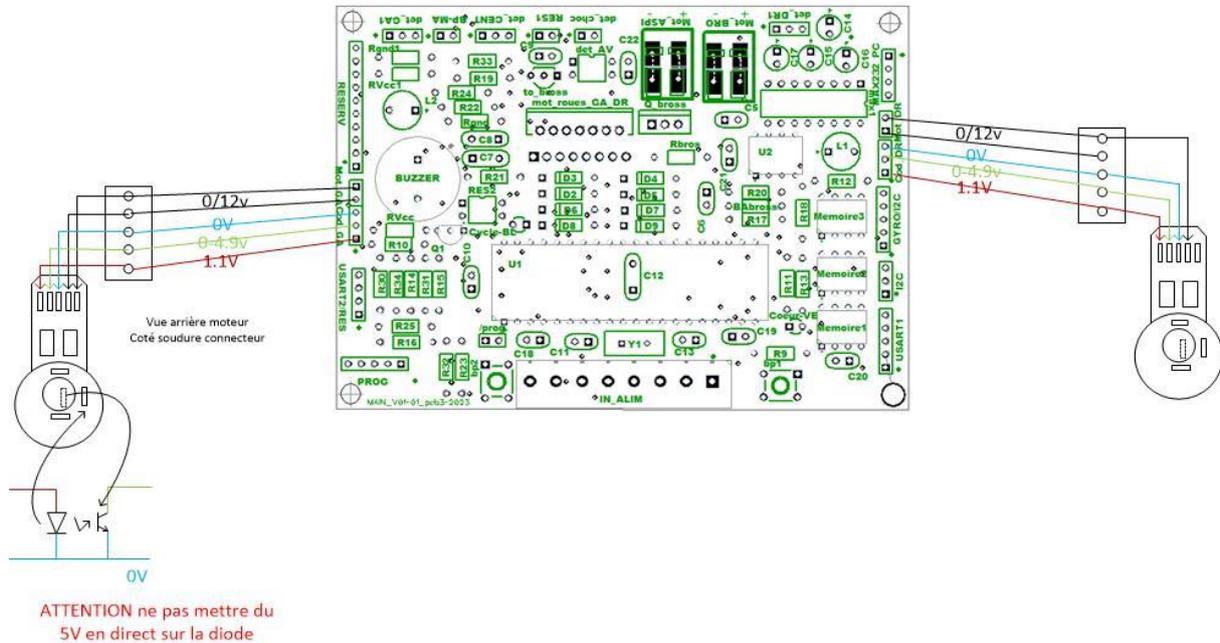
Le programme est déjà décortiqué dans le 2-1



Connexions nécessaires

## Connexions moteurs

Il y a un moteur droit et gauche mais les connecteurs sont les mêmes



Programme :

Pour le comptage des impulsions sur le codeur moteur nous une interruption.

Principe d'une interruption :

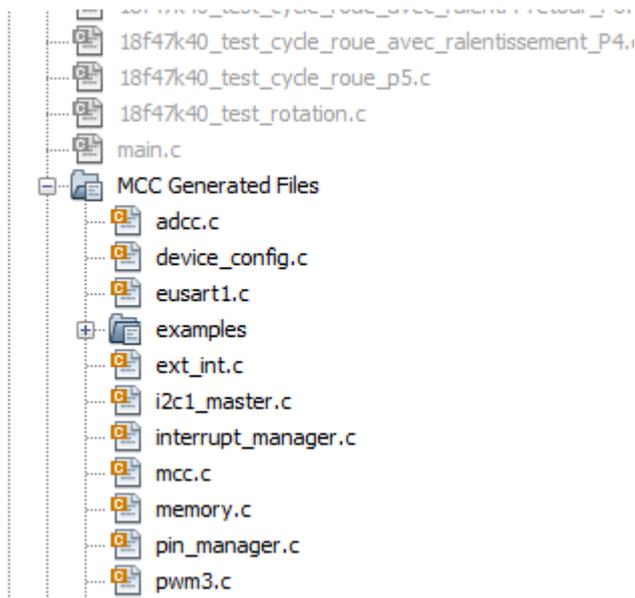
L'interruption est un mécanisme fondamental de tout processeur. Il permet de prendre en compte des événements extérieurs au processeur et de leur associer un traitement spécifique..

Il faut noter que l'exécution d'une instruction n'est jamais interrompue ; c'est à la fin de l'instruction en cours lors de l'arrivée de l'événement que le sous-programme d'interruption est exécuté. A la fin de cette procédure, le microcontrôleur reprend le programme principal à l'endroit où il l'a laissé.

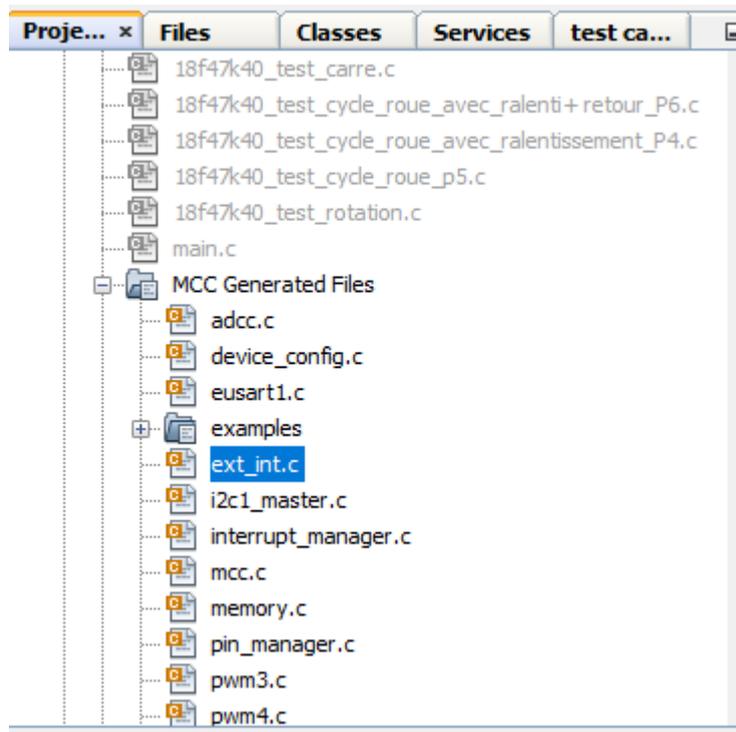
Nous utilisons des interruptions pour les temporisations et le comptage des impulsions sur roues.

Voici comment le comptage s'effectue sur les roues :

Cliquez sur « MCC Generated Files » dans la fenêtre projet.



Puis cliquez sur « ext\_int.c »



Voici le programme qui est dans ce fichier :



**Remarque :** toutes les écritures en bleue sont créées par microchip lors de la selection de la fonction interruption par des entrées.

Les écritures ajoutées manuellement par nos soins sont en vertes.

Les deux fonctions que nous utilisons sont :

`void INTO_CallBack(void)`

fonction utilisée pour le codeur de droite

`void INT1_CallBack(void)`

fonction utilisée pour le codeur de gauche

```
/**
```

```
EXT_INT Generated Driver File
```

```
.....
```

```
Bla bla bla de microchip....en commentaire
```

```
*/
```

```
/**
```

```
Section: Includes
```

```
*/
```

```
#include <xc.h>
```

```
#include "ext_int.h"
```

```
#include "tmr1.h"
```

```
void (*INT0_InterruptHandler)(void);
```

```
void (*INT1_InterruptHandler)(void);
```

```
void INTO_ISR(void)
```

```
{
```

```
    EXT_INT0_InterruptFlagClear();
```



```
// Callback function gets called everytime this ISR executes
INTO_CallBack();
}

void INTO_CallBack(void)
{
    // Add your custom callback code here

    explications : on arrive dans cette fonction uniquement lors d'un front montant sur le codeur roue droite

    {cmpt_roue_DR++;} //
    ajout*****
    ***** explications : cmpt_roue_DR++ signifie qu'à chaque fois qu'il y a une interruption due à un front montant
    (géré par le pic voir les config d'entrée sortie) on compte un de plus que l'ancien comptage.

    Bien sûr la remise à zéro doit se faire dans le programme principal à chaque fois que l'on démarre un mouvement.

    cpt_tempo_manque_puissance=0;//à chaque fois qu'une entrée bascule on
    remet à zéro la tempo de surveillance de la puissance

    explications : la temporisation est mise à zéro à chaque fois qu'un mouvement est vu par le codeur en effet cette
    tempo à pour but de ne pas laisser le robot bloqué par manque de puissance.

    if(INTO_InterruptHandler)
    {
        INTO_InterruptHandler();
    }
}

void INTO_SetInterruptHandler(void (* InterruptHandler)(void)){
    INTO_InterruptHandler = InterruptHandler;
}

void INTO_DefaultInterruptHandler(void){
    // add your INTO interrupt custom code
    // or set custom function using INTO_SetInterruptHandler()
```



```
}  
void INT1_ISR(void)  
{  
    EXT_INT1_InterruptFlagClear();  
  
    // Callback function gets called everytime this ISR executes  
    INT1_CallBack();  
}  
  
void INT1_CallBack(void)  
{  
    // Add your custom callback code here  
  
    explications : on arrive dans cette fonction uniquement lors d'un front montant sur le codeur roue gauche  
  
    cmpt_roue_GA++; // ajout  
    *****  
  
    explications : cmpt_roue_GA++ signifie qu'à chaque fois qu'il y a une interruption due à un front montant (géré par le pic voir les config d'entrée sortie) on compte un de plus que l'ancien comptage.  
  
    Bien sûr la remise à zéro doit se faire dans le programme principal à chaque fois que l'on démarre un mouvement  
  
    cpt_tempo_manque_puissance=0;  
  
    explications : la temporisation est mise à zéro à chaque fois qu'un mouvement est vu par le codeur en effet cette tempo à pour but de ne pas laisser le robot bloqué par manque de puissance.  
  
    if(INT1_InterruptHandler)  
    {  
        INT1_InterruptHandler();  
    }  
}  
  
void INT1_SetInterruptHandler(void (* InterruptHandler)(void)){  
    INT1_InterruptHandler = InterruptHandler;  
}
```



```
void INT1_DefaultInterruptHandler(void){

    // add your INT1 interrupt custom code
    // or set custom function using INT1_SetInterruptHandler()
}

void EXT_INT_Initialize(void)
{

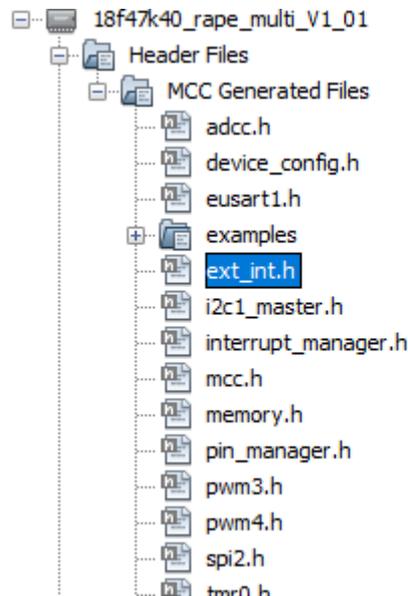
    // Clear the interrupt flag
    // Set the external interrupt edge detect
    EXT_INT0_InterruptFlagClear();
    EXT_INT0_risingEdgeSet();
    // Set Default Interrupt Handler
    INT0_SetInterruptHandler(INT0_DefaultInterruptHandler);
    EXT_INT0_InterruptEnable();

    // Clear the interrupt flag
    // Set the external interrupt edge detect
    EXT_INT1_InterruptFlagClear();
    EXT_INT1_risingEdgeSet();
    // Set Default Interrupt Handler
    INT1_SetInterruptHandler(INT1_DefaultInterruptHandler);
    EXT_INT1_InterruptEnable();

}
```



Pour pouvoir ajouter une variable (cmpt\_roue\_Ga par exemple) il faut se rendre dans le fichier portant le même nom mais avec une extension différente « .h » donc les fichiers de tête que l'on trouve dans :



Qui contient :

```
struct {
unsigned bit0:1; //
unsigned bit1:1;
unsigned bit2:1;
unsigned bit3:1;
unsigned bit4:1;
unsigned bit5:1;
unsigned bit6:1;
unsigned bit7:1;
} cmpt_rapide;

#define comptage_gauche_droit      cmpt_rapide.bit0//bit_10_ms
#define ode_recul_debut_pas      cmpt_rapide.bit1
//#define reserve                cmpt_rapide.bit2
//#define réserve                cmpt_rapide.bit3
//#define réserve                cmpt_rapide.bit4
//#define réserve                cmpt_rapide.bit5//
//#define réserve                cmpt_rapide.bit6
//#define réserve                cmpt_rapide.bit7

#define sortie_led_rouge          LATDbits.LATD5 // patte 28

signed long   cmpt_roue_DR;
signed long   cmpt_roue_GA;
unsigned int  cpt_tempo_manque_puissance;
```



### Test pratique sur le robot.

- 1- Faire le transfert du programme dans le pic.
- 2- La led verte doit clignoter
- 3- Brancher la liaison rs232, lancer TERA TERM les informations venant du pic doivent apparaitre
- 4- Faites tourner les roues à la main  
Sur chaque compteur affichée sur l'écran du pc on doit voir s'incrémenter les points.