

### 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

Te	est des entrées RAPE sur MPLABX	1
	Charger le programme des entrées	2
	Description du programme :	4
	Evaluation du programme test des entrées :	<del>-</del> л
	Explication sur le programme test des entrees :	4
	Connexions necessaires	5
	Programme :	5
	Test pratique sur le robot.	12

La vidéo associée :



# Charger le programme des entrées

																_		-	
MPLAB X IDE v6.00	- 18f47k40_ri	ape_multi_V	1_01 : default															- 0	×
File Edit View Navigat	e Source R	efactor Pro	duction Debu	g Tean	Tools Winde	ow Help											Q Search	(Ctrl+I)	
·		1			Ka Kay	N 10			: 🍙	:		a : 🔿	:						
: 🗂 📶 📑 🖷 :	76	default		✓ SRC	• 🛯 • 🔛	· • • 3	<u></u>	, * 🖬 💷	> * : 🖤		🧠 i 🖗	7 8	PC: 0x0	n ov z dc c	: W:0x0 : ban	k 0 Ho	w do I? Keywo		
	γ	r	Y				_				_								
Proje × Files	Classes	Services	test ca	-															
□ □ 18f47k40_rape_m	ulti_V1_01																		
🕀 💼 Header Files																			
🗄 👘 Important File	s																		
🗄 💼 Linker Files																			
Source Files																			
	test 12.c																		
	test carre c																		
E 10647140	test suls s		unti Lumbra DC																
1014/K40	_test_cycle_ro	iue_avec_raie	sid+retour_Po	C															
1014/K40	_test_cycle_ro	iue_avec_rale	nussement_P4	c															
18t47k40	_test_cycle_ro	iue_p5.c																	
🔠 18f47k40	_test_rotation	.c																	
···땐 main.c																			
Image: MCC Gene	erated Files																		
🐨 prog1.c																			
🕾 prog_esca	alier_V1.c																		
🖭 test cvde	_aleatoire_P7	l.c																	
test cycle	roue continu	1 P3.c																	
test cycle	roue continu	P4 arret ob	istade.c																
(B) test_cycle	a victuel c		ner ener selfter i fer																
(a) test depla	a_virtuei.c																		
test_entre	ees_base_P1.0																		
test_Entre	ees_P1_option	IS.C																	
test_sortie	es_P2.c																		
🕀 📻 Libraries																			
E Loadables																			
k40 rane multi V1 0	1 - Dachboa	and Nav	igator x																
				_															
Output ×																			
Project Loading Warning	× Configura	ation Loading	Error × PIC	kit 4-18	f47k40_rape_	multi_V1_0	1 × MP	LAB® Code (	Configurat	or ×	18f47k40	_rape_m	ulti_V1_0	01 (Build, Lo	ad,) ×				
The following me	emory area	(s) will h	ce programm	ed:															
program memory:	start add	ress = 0x0	), end addr	ess =	0x27f														
program memory:	start add	ress = 0x1	fe80, end a	ddress	= 0x1347f														
configuration me	mory																		
User Id Memory																			
Programming/Veri	try comple	te																	
Connecting to MP	LAB PICki	t 4																	
Currently loaded	versions	:																	
Application vers	10n		1.07																
Boot version			0.00																
script version			.87																
Script build num	wer	Isfat	Lieesc																1
1001 pack versio	m		.1305																
				_												_			
																	(32)		

« test\_Entrees\_base\_P1 »



	main	1.c									
÷-	MCC	Generated Files									
	prog	1.c									
	prog	escalier_V1.c									
···· 🖭	test	cycle_aleatoire_P7.c									
E	test	_cycle_roue_continu_P3.c									
···· 🖭	test_cycle_roue_continu_P4_arret obstacle.c										
	test	_depla_virtuel.c									
	test test	Open									
_ e	test	Cut	Ctrl+X								
🕀 / 🕞 Libi	raries	Сору	Ctrl+C								
🖽 🜆 Loa	adabi	Paste	Ctrl+V								
k40 mana m		Include file(s) in current	t configuration								
кио_гаре_п	IUIU	Compile File	-								
		Remove From Project									
		Rename									
		Save As Template									
		History	>								
		Tools	>								
		Properties									
		MISRA Check File									
	L										

### Le programme passe en gras





### Cliquez deux fois dessus :

X MPLAB X IDE v6.00 - 18f47k40_rape_multi_V1_01 : default		-	
File Edit View Navigate Source Refactor Production Debug Tear	m Tools	Window Help	ol+I)
👚 🎦 🔡 🍓 🍤 🍊 🛛 default 🤍 🏠	• 7	🦉 * 🕨 * 🏪 * 🖫 * 🕼 🚯 * 🔯 🤓 🛒 🎯 PC: 0x0 n ov z dcc : W:0x0 : bank 0 How do 12 (Cryword(s)	
Proje × Files Classes Services test ca 🖃	test_E	trees_base_P1.c ×	
□- □ 18f47k40_rape_multi_V1_01	Source	History 🗐 🔯 🔁 📲 📲 🖓 🖉 😓 🔛 💜 🕬 👝 📰 🌌	æ
🖶 🛅 Header Files			
🛞 🕼 Important Files	1 1		1 +
🐵 🕋 Linker Files	2	ARE A	
🖨 🚰 Source Files	3		
@ 18f47k40_test_12.c	4	* sur carte MAIN_V01-01_pcb3-2023	-
18f47k40_test_carre.c	5	* robot RAPE a programmer avec MPLAB et picist47/840	
🙀 18f47k40_test_cycle_roue_avec_ralenti+retour_P6.c	6		
😰 18f47k40_test_cycle_roue_avec_ralentissement_P4.c	7	* programme N°1	
18f47k40_test_cycle_roue_p5.c	8	* premier essai	
R 18f47k40 test rotation.c	9	* <u>programme</u> de test <u>des Entrées</u> sur <u>le schéma</u> de base <u>uniquement</u>	
mein.c	10	* la led <u>verte</u> doit <u>clignoter</u> une <u>fois le programme injecté</u>	
MCC Generated Files	11	* L'affichage sur pc fonctionne et permet de voir les compteurs roue et les tensions ana	
- @ progl.c	12	*	
- M prog escaler V1.c	13	*	
test cycle aleatoire P7.c	14	* La led <u>verte</u> doit <u>clignoter</u> en permanence <u>tous</u> les 100ms <u>donc</u> 0.1s	
test cycle raue continu P3 c	15	* <u>l'affichage</u> doit <u>être présent</u> sur <u>l'écran</u> du <u>p</u> c avec la liaison RS232 <u>établie</u>	
test cycle roue continu P4 arret obstacle.c	16	*	
test denia virtuel c	17	* ************************************	
art Entreer bare P1 c	18	* les trois BP un appui led bleue	
The Entrance Pl online c	19	* le bouclier avant	
test_bid ces_pi_opuoris.c	20	* lorsque l'on fait tourner les roues à la main les compteurs s'incrémentent à voir sur l'affichage.	
time test_portes_pizze	21	*	
in the Londollar	22	<i>x</i>	
Coabables	23	* ************************************	
	24	* cellule centrale tension variable de 300 à 650.	
k40_rape_multi_V1_01 - Dashboard Navigator ×	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 remet le	fil on a
ANA_BATT	26	*	
@ FM_tempo_100ms()	27	*la <u>consommation</u> à vide <u>est</u> de 60mA	
@ FM_tempo_500ms()	28	x	
# 12c_SCL	29	- */	
#= I2c_SDA	30		
# PGC_prog	31 0	finclude "mcc generated files/mcc.h"	
PGD_prog	32	<pre>#include <stdio.h></stdio.h></pre>	
# SI_mem_SDO2	33	finclude <xc.h></xc.h>	
# SO_mem_SDI	34	finclude <stdint.h></stdint.h>	
#= bp1	35	finclude "mcc generated files/examples/i2cl master example.h" // sans ca pas de fonctionnement du I2C;	
#= bp2	36	tinclude (math.b)	
	37		
	38		
	39		
#- com_pc_mess_RX1	35		-
com_pc_mess_TX1	Find:	tempo attente fin mouvement 🗸 🖳 Previous 🚓 Next 📣 🧢 🗭 🕅	15 matches ×
(a) commande sorties()			

### 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 :

ATTENTION ne pas mettre du 5V en direct sur la diode

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.

····· [28:7	18f47k40_test_cycle_roue_avec_ralentissement_P4.0
···· 🔁	18f47k40_test_cycle_roue_p5.c
···· 🖭	18f47k40_test_rotation.c
···· 🖭	main.c
÷ 👍	MCC Generated Files
	🖭 adcc.c
	🖭 device_config.c
	🖭 eusart1.c
	examples
	🖭 ext_int.c
	🖭 i2c1_master.c
	🖭 interrupt_manager.c
	🖭 mcc.c
	🖭 memory.c
	🖭 pin_manager.c
	🖭 pwm3.c

### Puis cliquez sur « ext\_int.c »

Proje ×	Files	Classes	Services	test ca	_								
···· 🔁	18f47k40_	test_carre.c											
	18f47k40_test_cycle_roue_avec_ralenti+retour_P6.c												
	18f47k40_	18f47k40_test_cycle_roue_avec_ralentissement_P4.c											
<b>e</b>	18f47k40_test_cycle_roue_p5.c												
<b>P</b>	18f47k40_test_rotation.c												
<b>P</b>	main.c												
- MCC Generated Files													
	🔮 adcc.c												
	evice_config.c												
	🖭 eusart:	1.c											
	⊕ 庙 examples												
	🖭 ext_int.c												
	mcc.c												
	🖭 memor	y.c											
	🖭 pin_ma	nager.c											
	🖭 pwm3.c												
	🖭 pwm4.												

Voici le programme qui est dans ce fichier :



**<u>Remarque</u>** : 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 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\_INTO\_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

\*\*\*\*\*\* <u>explications</u> : 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

<u>explications</u> : 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 INT0\_SetInterruptHandler(void (\* InterruptHandler)(void)){
 INT0\_InterruptHandler = InterruptHandler;

```
}
```

void INT0\_DefaultInterruptHandler(void){

// add your INTO interrupt custom code

// or set custom function using INT0\_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

\*\*\*\*\*\*\*\*\*\*\*\*

<u>explications</u> : 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;

<u>explications</u> : 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\_INTO\_InterruptFlagClear();

EXT\_INTO\_risingEdgeSet();

// Set Default Interrupt Handler

INTO\_SetInterruptHandler(INTO\_DefaultInterruptHandler);

EXT\_INTO\_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 bitl: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 cde_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
- Faites tourner les roues à la main
   Sur chaque compteur affichée sur l'écran du pc on doit voir s'incrémenter les points.