



Gestion du timer 4 base de temps du robot RAPE

Dans ce cours nous allons gérer les bases de temps du robot (sur lesquelles reposent toutes les tempos utilisées dans le programme), pour vérifier les bases de temps nous utiliserons une sortie de réserve afin de faire des mesures sur oscilloscope.

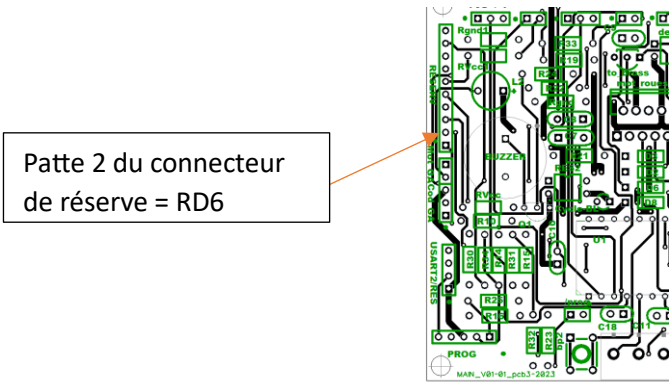
Les 3 bases de temps utilisées sont : 10ms, 100ms, 500ms.

Table des matières

Gestion du timer 4 base de temps du robot RAPE	1
Attribution par le MCC d'une sortie sur RD6	2
Modifications dans le programme principal.....	3
Création dans la temporisation d'un bit qui bascule au rythme de la tempo	4
Mesures des tempo 10ms, 100ms et 500ms.....	5

L'utilisation de `__delay()` n'est pas approprié car le temps est « bloqué » pendant l'utilisation de cette fonction, il est plus judicieux de se faire sa propre base de temps. Nous avons décidé de partir sur une base de 10ms, il nous faut une tempo de 5ms pour qu'entre deux basculements donc de front, on est 10ms. Cette base de temps donnera la base que nous démultiplierons afin de créer 100ms puis 0.5s (500ms) toujours sur le même principe chaque comptage.

Choix d'une sortie libre pour mettre l'oscilloscope et vérifier les tempos : RD6





Attribution par le MCC d'une sortie sur RD6

Nous allons prendre une sortie inoccupée pour contrôler la fréquence.

Il faut lancer le MCC

```

168
169
170
171
172
173
174
175
176 void THM4_SetInterruptHandler(void (* InterruptHandler)(void)) {
177     THM4_InterruptHandler = InterruptHandler;
178 }
179
180 void THM4_DefaultInterruptHandler(void) {
181     // add your THM4 interrupt custom code
182     // or set custom function using THM4_SetInterruptHandler()
183
184     bit_10ms=bit_10ms;
185     cpt_tps_500ms++;
186     if (cpt_tps_500ms==49) {cpt_tps_500ms=0; bit_500ms=bit_500ms; timer_D4 =timer_D4;}
187
188     // 100ms
189     cpt_tps_100ms++;
190     if (cpt_tps_100ms==10) {cpt_tps_100ms=0; bit_100ms=bit_100ms; }
191     // clear the THM4 interrupt flag
192     PIRbits.THM4IF = 0;
193
194
195
196
197 }
198
199 /**
200 End of File
  
```

Puis accéder aux « pin module »

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	OD	IOC
RA0	ADCC	ANA0	channel_AN0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RA1	ADCC	ANA1	channel_AN1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RA2	ADCC	ANA2	channel_AN2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RA3	ADCC	ANA3	channel_AN3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RA4	Pin Module	GPIO	IO_RA4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RA5	Pin Module	GPIO	IO_RA5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB0	EXT_INT	INT0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	posi...
RB1	EXT_INT	INT1		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	posi...

Première chose à faire mettre la sortie (output) en service RD6 par exemple (libre).

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	OD	IOC
RD1	Pin Module	GPIO	IO_RD1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RD2	Pin Module	GPIO	IO_RD2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RD3	Pin Module	GPIO	IO_RD3	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
RD4	Pin Module	GPIO	IO_RD4	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RD5	Pin Module	GPIO	IO_RD5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RD6	Pin Module	GPIO	IO_RD6	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RD7	Pin Module	GPIO	IO_RD7	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
RE0	Pin Module	GPIO	IO_RE0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Ensuite on fait un « generate » pour valider la modification.



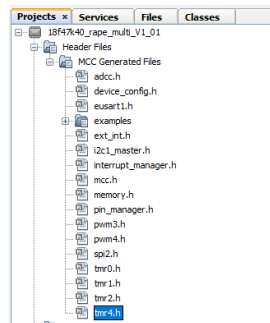
Modifications dans le programme principal

Pour que la sortie soit prise en compte dans la tempo, il faut la supprimer du programme principal et la mettre dans la tempo 4 en mcc dans les fichiers de tête.

Suppression dans prog principal :

```
#define sortie_led_bleue LATDbits.LATD5 // patte 28
// #define reserve_D6 PORTDbits.RD6 // patte 29
#define reserve_D7 LATDbits.LATD7 // patte 30
#define sortie_led_CS mem1 INTFbits.INTF0 // patte 30
```

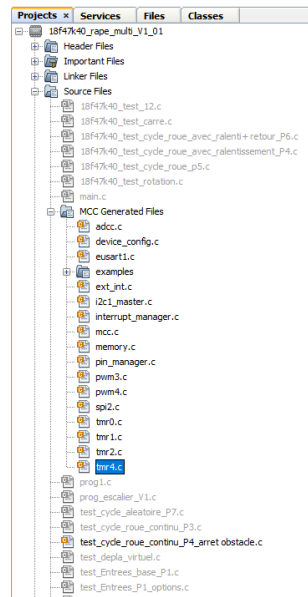
Ajout dans le fichier de tête (Header Files\ tm4.h) de la déclaration du bit:



```
49 /**
50 Section: Included Files
51 */
52
53
54 #include <as4int.h>
55 #include <as4bool.h>
56
57 unsigned int tps_500ms;
58 unsigned int tps_100ms;
59
60 struct {
61     unsigned bit0 : 1; //
62     unsigned bit1 : 1;
63     unsigned bit2 : 1;
64     unsigned bit3 : 1;
65     unsigned bit4 : 1;
66     unsigned bit5 : 1;
67     unsigned bit6 : 1;
68     unsigned bit7 : 1;
69 } tps; //
70
71 #define timer_D6 PORTDbits.RD6 // patte 29
72 #define bit_10ms tps.bit0
73 #define bit_500ms tps.bit1
74 #define bit_500ms tps.bit2
75 #define timer_100ms tps.bit3
76 #define bit_100ms tps.bit4
77 #define bit_100ms tps.bit5
78 #define timer_100ms tps.bit6
79 // #define sortie_led led_bit7
80
81
82
83
```



Création dans la temporisation d'un bit qui bascule au rythme de la tempo Dans le timer 4 : tmr4.c



Nous ajoutons la dernière ligne « timer_D6 =!timer_D6; »

Ensuite nous déplacerons cette écriture sur chaque tempo.

```
void TMR4_DefaultInterruptHandler(void) {
    // add your TMR4 interrupt custom code
    // or set custom function using TMR4_SetInterruptHandler()

    bit_10ms=!bit_10ms;
    cpt_tps_500ms++;
    if (cpt_tps_500ms>=49) {cpt_tps_500ms=0, bit_500ms=!bit_500ms;}
    //100ms
    cpt_tps_100ms++;
    if (cpt_tps_100ms>=9) {cpt_tps_100ms=0, bit_100ms=!bit_100ms; }
    // clear the TMR4 interrupt flag
    PIR4bits.TMR4IF = 0;

    timer_D6 =!timer_D6;
}
```

Nous pourrions aussi faire cela :

```
void TMR4_DefaultInterruptHandler(void) {
    // add your TMR4 interrupt custom code
    // or set custom function using TMR4_SetInterruptHandler()

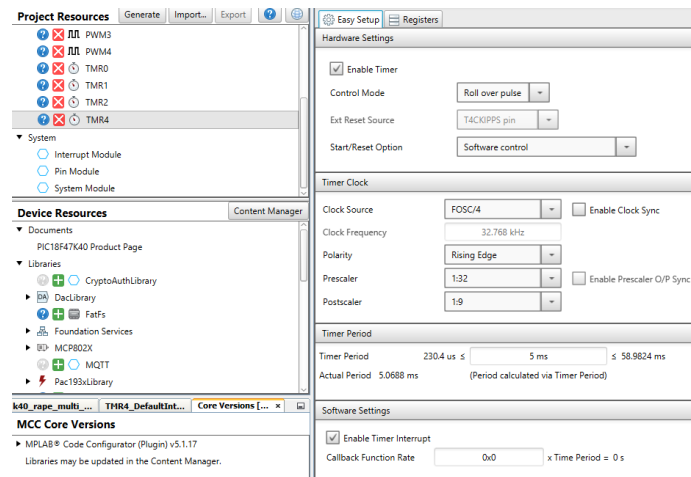
    bit_10ms=!bit_10ms;
    cpt_tps_500ms++;
    if (cpt_tps_500ms>=49) {cpt_tps_500ms=0, bit_500ms=!bit_500ms; }
    //100ms
    cpt_tps_100ms++;
    if (cpt_tps_100ms>=10) {cpt_tps_100ms=0, bit_100ms=!bit_100ms; }
    // clear the TMR4 interrupt flag

    timer_D6 = bit_10ms;

    PIR4bits.TMR4IF = 0;
}
```



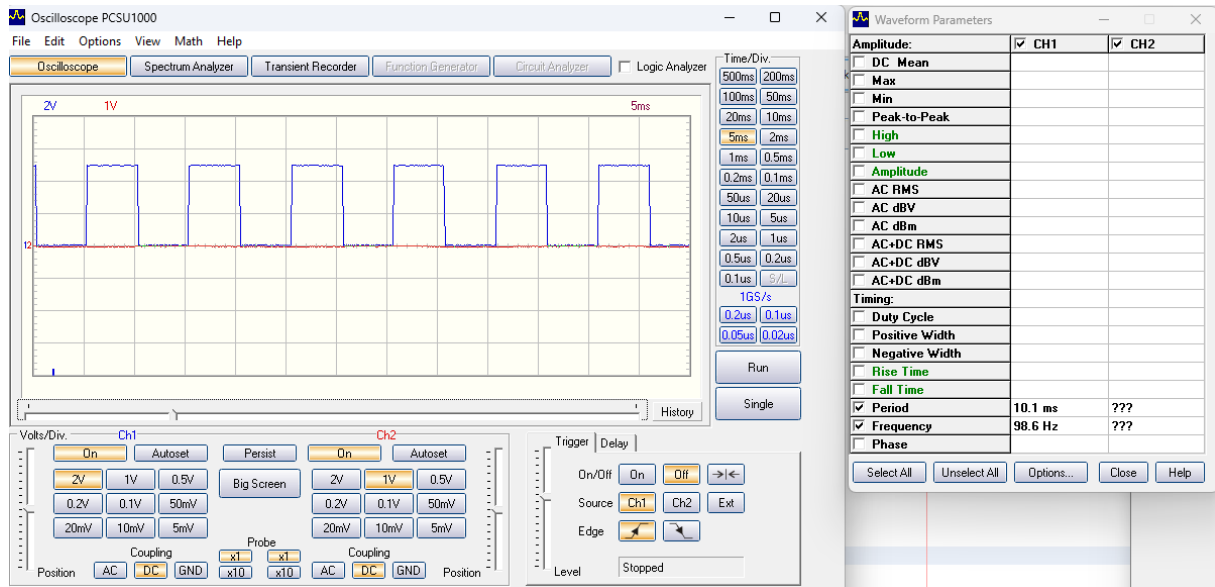
Rappel du timer 4 :



On remarque que l'on a pas tout à fait 5ms (donc 10ms entre deux fronts) on 5.0688ms nous ne cherchons pas un temps précis donc nous pouvons accepter cette erreur.

Mesures des tempo 10ms, 100ms et 500ms.

Mesure sur la tempo de 10ms

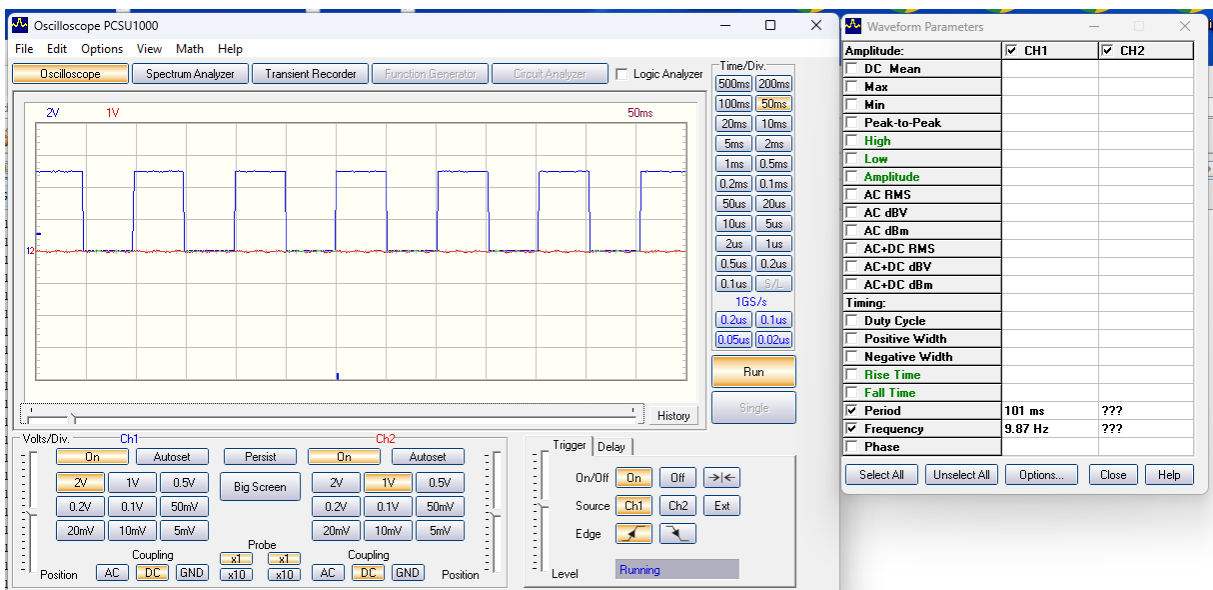


La temporisation est bien de ~10ms (avec une erreur de 1%) entre deux fronts montants.



Faisons la même chose pour 100ms

```
void TMR4_DefaultInterruptHandler(void) {  
  // add your TMR4 interrupt custom code  
  // or set custom function using TMR4_SetInterruptHandler()  
  
  bit_10ms=!bit_10ms;  
  cpt_tps_500ms++;  
  if (cpt_tps_500ms>=49) (cpt_tps_500ms=0, bit_500ms=!bit_500ms; )  
  //100ms  
  cpt_tps_100ms++;  
  if (cpt_tps_100ms>=10) (cpt_tps_100ms=0, bit_100ms=!bit_100ms; timer_D6 !=timer_D6; )  
  // clear the TMR4 interrupt flag  
  PIR4bits.TMR4IF = 0;  
}
```



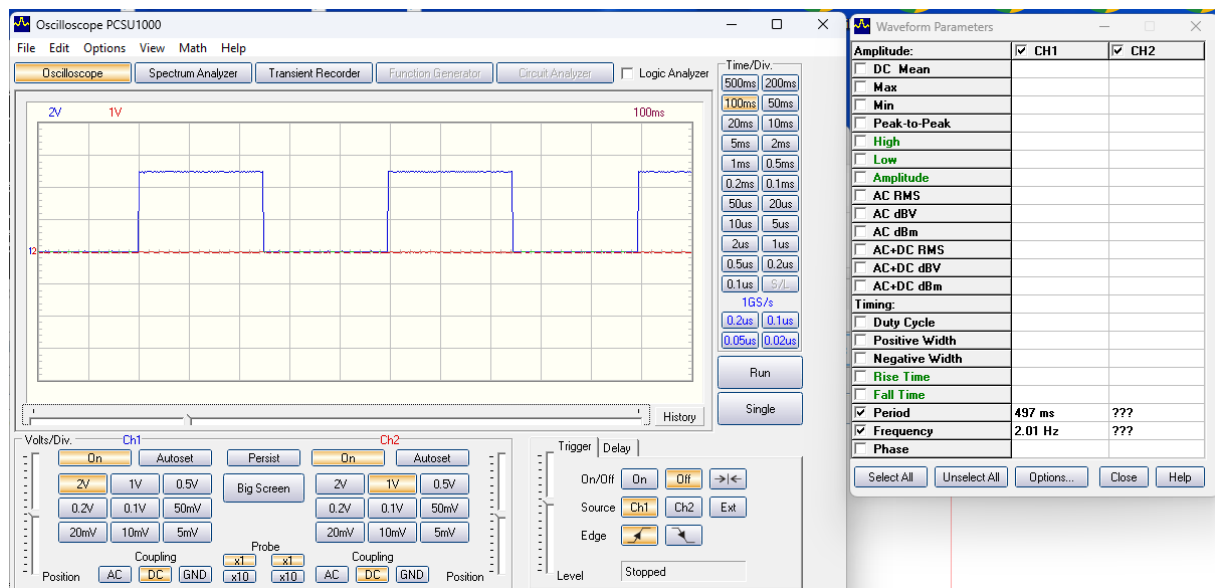
La aussi nous avons une tempo légèrement supérieure. (si on met 9 à la place de 10 on a 91ms.) Nous garderons 10

On voit que l'on a 91ms avec une valeur de 9 et 101ms avec 10 nous laisserons donc 10.



Tempo 500ms

Le même raisonnement s'applique avec les 500ms. On a 497ms ou 507. Nous garderons 49 pour réduire l'erreur.



Pour utiliser ces bases de temps nous utiliserons des compteurs sur front montant.

Exemple d'utilisation de tempo :

//temporisation attente fin mouvement

consigne de la tempo 3x100ms

```
consigne_tempo=3;
```

compteur de la tempo

```
if (start_tempo_attente_fin_mouvement & fm_bit_100ms) {cpt_tpo_attente_fin_mvt++;}
```

```
if (cpt_tpo_attente_fin_mvt >= consigne_tempo )
```

```
{fin_tempo_attente_fin_mouvement=1 ; cpt_tpo_attente_fin_mvt=0;start_tempo_attente_fin_mouvement=0;}//
```

```
if ( !start_tempo_attente_fin_mouvement) {cpt_tpo_attente_fin_mvt=0;}
```

Le compteur tempo est supérieur à la consigne donc on remet à zéro tous les bis utilisé dans la tempo ainsi que le compteur.