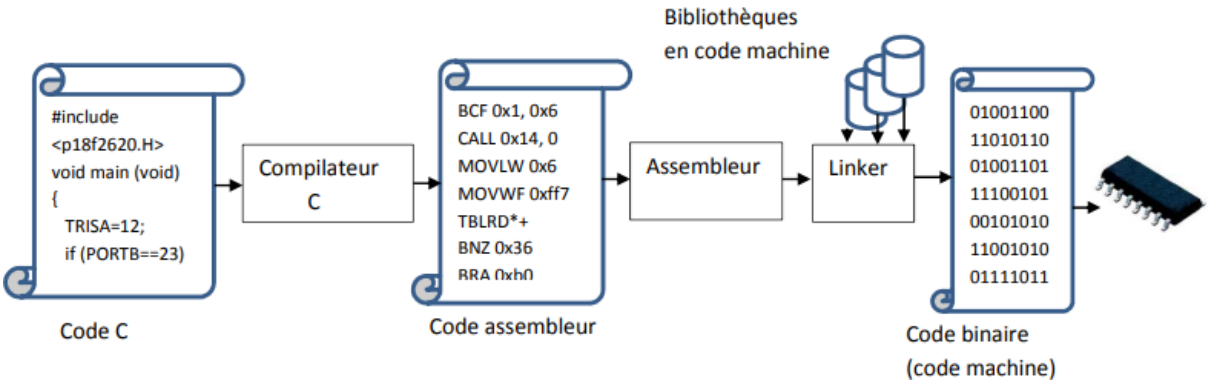
Instruction de base pour robot RAPE

Dans ce cours nous présentons les instructions de base utilisée dans les programmes du robot RAPE

Table des matières

Rappel sur le principe de fonctionnement du microchip Pic 18f



Les compilateurs sont toujours accompagnés de fichiers de déclaration appelés « header » avec l'extension .h Les fichiers (.h) peuvent contenir les déclarations de fonctions précompilés en bibliothèques comme <stdio.h> Ils peuvent également contenir les déclarations de registres et bits d'un microcontrôleur. (nous en reparlerons)

Le langage C utilisé dans le programme RAPE

Le langage C a été créé pour programmer les ordinateurs à des fins de traitement de données. De nombreuses fonctionnalités du langage ne sont pas ou peu utilisées dans le cadre de la programmation d'un microcontrôleur. En revanche le programme d'un microcontrôleur devra accéder aux périphériques et à leurs configurations.

En C toute donnée doit avoir été déclarée avant d'être utilisée. La déclaration indique la taille en octets de la variable ainsi que son étendue (toujours positive ou pas)

Type	Longeur	Domaine de valeurs	utilisation
Char	8 bits	-128 à 127	Nombre entier signé
Unsigned char	8 bits	0 à 255	Nombre entier positif ou code ascii
Int	16bits	-32768 à 32767	Nombre entier signé
Unsigned int	16 bits	0 à 65535	Nombre entier positif
Long	32 bits	-2 147 483 648 à 2 147 483 647	Nombre entier signé
Unsigned long	32 bits	0 à 4 294 967 295	Nombre entier positif
Float	32 bits	3.4 * (10**-38) à 3.4 * (10**+38)	Nombre réel
Double	64 bits	1.7 * (10**-308) à 1.7 * (10**+308)	

Les étoiles indiquent la puissance.

Dans le programme RAPE nous utiliserons essentiellement char, int, long, et float (signé ou non).
Par exemple la valeur lue par l'entrée analogique est de 1024 maxi donc un « int » nous suffira
Pour le comptage des roues nous utiliserons un « long ».
Pour les calculs compliqués sur une position avec gyroscope « float ».

D'une manière générale, nous utiliserons des « integer » afin de réduire la taille mémoire.
C'est le LINKER qui attribuera une espace mémoire à chaque variable.

Les variables :

Elles doivent être déclarées avec le type.
Exemples :

int compt_cod_ga; dans ce cas un integer sur le compteur du codeur gauche
unsigned char copie_etapes_graf_cycle, etapes_graf_cycle; dans ce cas les étapes grafcet sont stockées dans ces mots et ne pourrons jamais excéder quelques dizaines d'étape donc non signée et inférieur à 255.

On peut mettre une valeur dés le départ ou dans le programme.

Exemple :

int mot_16bits_essai : 0 ; on met zéro dans le mot lors de sa déclaration

mot_16bits_essai = 12 ; on met 12 dans le mot dans le programme.

Une instruction doit se terminer par un point-virgule

Exemple :

*consig_pour_un_pas= 750*4-10;* on peut aussi faire des calculs dans une expression

if (etapes_graf_cycle == 1) {cde_avance_un_pas=1;} si l'étape de cycle est égale à 1 alors on commande l'avance d'un pas

les données peuvent prendre plusieurs représentation

a=21 ; valeur décimale de 21

b=0x342 ; valeur hexa

c=b ; une autre variable qui doit être du même type.

Les entrées sorties :

Définie par Microchip on remplacera par une variable plus simple à comprendre :

Exemple :

#define det_cent_ANA PORTAbits.RA0 // ;RA0 P2 ANA **entrée** sur détecteur optique central

#define sortie_mot_enable_B_PW4 LATCbits.LATC0 // patte 15 commande du pwm gauche **sortie**

Les *#define* doivent être placées en dehors de toute fonction et en début de programme.

Les fonctions :

En C on utilise des sous-programme que l'on appelle fonction, elles sont appélées le programme principal. Une fonction est un ensemble d'instructions groupées et encadrées par des accolades {},.

le nom de la fonction représente l'adresse ou celle-ci est rangée en mémoire, pour exécuter une fonction il suffit d'écrire son nom. Elle peut posséder un ou plusieurs paramètres d'entrée mais un seul paramètre de sortie, les types des paramètres sont déclarés comme les variables. S'il n'y a pas de paramètre on écrit void (vide).

Exemple :

void FM_tempo_100ms(void); déclarée en tête de programme.

……

while (1) appelé dans le prog principal pour être active

{ FM_tempo_100ms(); }

Fin de programme principal

……

void FM_tempo_100ms(void) la fonction est écrite ici

{

……

}

Vous pouvez aller dans le programme et vous retrouverez cette fonction

Structure d'un programme :

/**

Indications claires sur le programme en commentaire

*/

```
//appel des fichiers nécessaires à la compilation
#include "mcc_generated_files/mcc.h"
……
//définition de notre pic avec ses entrées et sorties

#define det_cent_ANA            PORTAbits.RA0        //        ;RA0 P2  ANA entrée sur détecteur optique central

//**********définition des mots et bits utilisés dans notre programme

        int compt_cod_ga;    déclaration de la valeur codeur en integer
……
         int signed ecart_roue =0;   déclartion en integer d'une valeur qui sera l'écart entre le codeur droit et gauche

//*****************************************************************************************
Déclaration de bits associés à un mot de 8 bits (byte)
 union {
 unsigned char byte ; déclaration du  mot de 8 bits byte
 struct {
   unsigned b0:1;
   unsigned b1:1;
   unsigned b2:1;
   unsigned b3:1;
   unsigned b4:1;
   unsigned b5:1;
```

```
  unsigned b6:1;
  unsigned b7:1;
 };
} mot_etape_gra;   // union de mot_etape_gra et byte(qui peut être appelé autrement)


#define E0    mot_etape_gra.b0 // mot envoyé au wago sur l'état de la carte
……
#define E7    mot_etape_gra.b7 //
#define E0    mot_etape_gra.b0 //
……
#define E7    mot_etape_gra.b7 //
```

Déclaration de bits

```
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;
} drap1,…… ;
#define cde_avance                drap1.bit0//bit_10_ms
……
#define inter_cde_mvt_antihoraire       drap1.bit7
```

Déclaration des mots utilisés dans le prog

```
unsigned int consig_pour_un_pas; //
……
```

Déclaration des fonctions

///*******************VOID*****************************

//****** fronts montants

void FM_tempo_100ms(void);

……..

Programme principal on ne passe qu'une seul fois dans cette partie lors du démarrage

main(void) il ne peut y avoir qu'un seul main

{

Initialisation demandée par les MCC dans ce cas le programme va dans les sous programmes créés par le générateur de configuration.

SYSTEM_Initialize();

……

On met les consignes à zéro (pourrait être mis lors de la création de la variable

copie_etapes_graf_cycle = etapes_graf_cycle = 0;

on peut mettre aussi les sorties pour fixer leur état

sortie_buzzer=0;

programme principal en boucle

while (1)

{

// appel des fonctions pour qu'elles soient actives

FM_tempo_100ms();

//-------------------------APPEL DES SOUS PROGRAMMES---------------------------------------

//**********************cde moteur**********************************

sortie_leds();

cycle (); //cycle grafcet

sortie_buzzer = bp1;

}//fin while(1))

}//fin main

Fin du programme

Appel des sous programmes ou fonction

```
void cycle ()
 {
……..
}
void gestion_moteur()
   {…….
}
……..
```

Fin du programme

Les tests

If …..then …..else

Si (conditions) alors on fait ça {…… ;}

Exemple

`if (etapes_graf_cycle == 1) {cde_avance_un_pas=1;}` //si l'étape du cycle égale à 1 on fait la commande d'avance d'un pas

on peut ajouter l'autre « else »

```
 if (cde_avance_un_pas) ) {cde_avance_un_pas_GA=1;} else {cde_avance_un_pas_GA=0;}
```

dans ce cas on aurait pu écrire aussi

```
cde_avance_un_pas_GA=  cde_avance_un_pas ;
```

les boucles while, do-while et for

while :

Très peu utilisé source de boucle perpétuel menant à un défaut de watchdog

```
while (a>5)     si a est plus petit que 5 on fait ce qui est entre les accolades
        {
                ……..
        }
```

Pas utilisé dans le programme RAPE

```
do { a= PORTB ; PORTA=PARTA++ ; }

while (a>5) // si a est plus petit que 5 on sort de la boucle
```

for

attention au boucle qui peuvent déclencher le watchdog

```
    for (y = 0 ; y<=valeur_maxi_affichage_ram + 5 ;y++ )

     { //debut for

          SPI_Write(y,0x00);    écriture de zéro dans y qui passe de 0 à valeur maxi d'affichage

     }
```

Consulter les très nombreux cours en ligne sur le C (http://genelaix.free.fr)