

```

//=====
//=====
// ***** SOUS-PROGRAMME SUIVI DE LIGNE ELEVE *****
//=====
//=====
//  Mode_Run() : LE ROBOT EFFECTUE SON PARCOURS
//-----
// !!!!!!!! Attention : on sort du menu Mode_Run par une coupure d'alimentation !!!!!!!!
//=====
void Mode_Run()
{
//-----
// Variables locale à Mode_Run()
//-----
// type      nom      Commentaires
//-----
unsigned char i,j,k;          // Compteurs des boucles utilisées pour le chargement de l'eprom en ram
unsigned char VisuPosition;   //
unsigned char PWM[11][2];     // Table chargée avec les coefficients
                                // PWM de réglage des moteurs
                                // issus de la table PWMEEPROM
unsigned char Cde_Del_IR;     // Permet de multiplexer la commande des del IR
unsigned int Phase_Arret = 50; // Compteur de boucle : Autorise la mesure de la position du robot
                                // pendant 200ms après l'arrêt des moteurs devant la cible
//=====

//=====
// INITIALISATION
//-----

// Initialisation du mode Run: Chargement de la table PWM[][] avec les paramètres en EEPROM
// i, k: compteur ligne, j compteur colonne
//-----
for (i=0;i<11;i++)
{
    for (j=0;j<2;j++)
    {
        for (k=0;k<19;k++)
        {

```

```

    if (PWM_EEPROM[i][j] == Table_Calcul_OCR1x[k][0])
    {
        PWM[i][j] = Table_Calcul_OCR1x[k][1];
        break;
    }
}
}
}

//-----
// Global enable interrupts
// #asm("sei")
//-----
// Fin initialisation
//=====

//=====
// BOUCLE INFINIE
//-----

while (1)
    { // début Mode_Run()
//=====
// GESTION DE LA BALLE          A retirer dans programme élève
//-----
// Phase d'arrêt devant la cible: Mesure de la position et évolution du graphe pendant
// [4.3*10E-3*Phase_Arret] puis lancement du sous-programme Prise_Balle()
//-----
/*      if (Cible == 1) // Cible passe à 1 lorsque le robot est devant la cible
    {
        if (Phase_Arret == 0)
            Prise_Balle(); // Cible <- dans INT0
        else
            Phase_Arret--;
    } */
//=====

/*=====
// SUIVI DE LIGNE
//=====
//-----

```

```
// Début FS11: "Calculer la position du robot par rapport à la ligne"          // A COMPLETER PAR ELEVE
```

```
//-----
```

Rôle: Calculer une valeur numérique permettant de sélectionner la LED IR (émetteur IR) devant émettre un faisceau infrarouge et calculer la valeur numérique représentative de la position du robot par rapport à la ligne.

Entrée:      AC0                      bit              Associé à la sortie du comparateur analogique (interne au µC).  
Représentatif de la position d'un capteur IR par rapport à la ligne.

Sorties:      Cde\_Del\_IR              octet              Permet de multiplexer la commande des LED IR.  
Comme Cde\_Del\_IR est transféré dans le port A, on donne ci-dessous le détail des bits de commande des LED IR.

Emetteur IR connecté au bit			Remarque
-----			
E_D_IR	"	PA0	Emetteur IR droit
E_CD_IR	"	PA1	Emetteur IR centre droit
E_C_IR	"	PA2	Emetteur IR centre
E_CG_IR	"	PA3	Emetteur IR centre Gauche
E_G_IR	"	PA4	Emetteur IR gauche

Position\_Robot    octet              Image de la position du robot par rapport à la ligne  
(détails dans RBELEVE.h)

```
-----*/
```

```
Position_Robot = HORS_LIGNE; // Initialisation à 0
```

```
Cde_Del_IR = 1;
```

```
// Début Acquisition de la position du robot par rapport à la ligne
```

```
for (i=?;?;i++) // <-- A COMPLETER
```

```
{
```

```
  CAPTEUR_IR = ~Cde_Del_IR;
```

```
  delay_us(30); // Activation d'un capteur à ajuster (19µs par défaut)
```

```
  if (ACO == 1) Position_Robot = Position_Robot | Cde_Del_IR;
```

```
  CAPTEUR_IR = 0xFF;
```

```
  Cde_Del_IR = Cde_Del_IR << 1 ;
```

```
  delay_us(540); // A ajuster (540µs par défaut)
```

```
}
```

```
// Fin FS11
```

```
//-----
```

```

/*-----
// Début FS13 "Choisir la valeur à afficher sur les LEDs"           // A COMPLETER PAR ELEVE
//-----
Rôle: Choisir la valeur numérique nécessaire à la commande des LEDs rouge pour que leur éclairage soit
représentatif de la position du robot par rapport à la ligne noire.

Entrée
Position_Robot: octet      Représentatif de la position du robot. Voir les valeurs qu'il prend dans RBELEVE.h
                           et les modifier si nécessaire

Sortie
VisuPosition:  octet      Représentatif des Leds à éclairer. Le bit 0 de VisuPosition commande la Led D0 de
                           l'interface IHM etc. Les valeurs à placer dans VisuPosition sont définies
                           dans le fichier RBELEVE.h
-----*/

switch (Position_Robot)
{
//      Position_Robot      Modification de la valeur de commande des Leds
//-----
    case HORS_LIGNE:        VisuPosition = Visu_Hors_ligne; break;           // Hors ligne
    case GAUCHE_3PLUS:      VisuPosition = Visu_GAUCHE_3PLUS; break;        // Gauche3+
    case GAUCHE_2PLUS:      VisuPosition = Visu_GAUCHE_2PLUS; break;        // Gauche2+
    case CENTRE_1:          VisuPosition = Visu_CENTRE_1; break;            // Centre1 non visualisé
    case GAUCHE_PLUS:       VisuPosition = Visu_GAUCHE_PLUS; break;         // Gauche+
    case CENTRE_3:          VisuPosition = Visu_CENTRE_3; break;            // Centre3
    case GAUCHE:            VisuPosition = Visu_GAUCHE; break;              // Gauche
    case DROITE_3PLUS:      VisuPosition = Visu_DROITE_3PLUS; break;        // Droite3+
    case DROITE_2PLUS:      VisuPosition = Visu_DROITE_2PLUS; break;        // Droite2+
    case DROITE_PLUS:       VisuPosition = Visu_DROITE_PLUS; break;         // Droite+
    case DROITE:            VisuPosition = Visu_DROITE; break;              // Droite
    case CENTRE_5:          VisuPosition = Visu_CENTRE_5; break;            // Centre5
}

    Ecrire_Port_I2C_Soft(Add_LED_I2C,VisuPosition);
// Fin FS13
//-----

```

```

/*-----
// Début FS12: "Choisir les valeurs des rapports cycliques"           // A COMPLETER PAR ELEVE
//-----
Rôle: Choisir la valeur du rapport cyclique du signal de commande des moteurs du robot en fonction de
sa position par rapport à la ligne noire.

Entrée:      Position_Robot      octet      (voir ci-dessus)

Sorties:      CDE_MOTD      octet      Correspond au registre (OCR1A). Règle le rapport cyclique
                                         du signal de commande du moteur droit.
              CDE_MOTG      octet      Correspond au registre (OCR1B). Règle le rapport cyclique
                                         du signal de commande du moteur gauche.
-----*/
// Graphe d'état du suivi de ligne
//-----
switch (EtatCdeMot)
{ // Début switch (EtatCdeMot)
  case DEPART: Arret_Moteurs();
    if ((Position_Robot == CENTRE_1) || (Position_Robot == CENTRE_3) || (Position_Robot == CENTRE_5))
    {
      Active_Moteurs();
      EtatCdeMot = CDE_CENTRE;
    }
    else EtatCdeMot = DEPART;
  break;

  case CDE_CENTRE : CDE_MOTG = PWM[0][0]; CDE_MOTD = PWM[0][1];
    if (Position_Robot == GAUCHE) EtatCdeMot = CDE_DROITE;
    else if (Position_Robot == DROITE) EtatCdeMot = CDE_GAUCHE;
    else EtatCdeMot = CDE_CENTRE;
  break;

  // A compléter

  default : EtatCdeMot = CDE_CENTRE;
  break;
}

// Fin FS12

```

```
//-----  
    }// Fin switch (EtatCdeMot)  
//-----  
  
// Fin Traiter  
//=====
```

delay\_us(1000); // à ajuster (704µs par défaut)

```
//=====
```

}

```
} // fin Mode_Run();  
//=====
```