

```

//Déclaration des variables d'Interruption
byte interrupt = 0; //A mettre à 1 pour appel du clignotement
int ComptageInterrupt = 120; //Fréquence clignotement LedCS
int trame = 215; //Délai pour verrouillage oscilloscope

//Déclaration des variables LedCS
byte varCompteur = 0; // La variable compteur interruption
#define LedCS digitalWrite (LedAR, !digitalRead(LedAR)) // Changement d'état de la LED changement Signal

//Déclaration des variables des LED
unsigned char LedAvant = A0; //Sortie led sortie A0
unsigned char LedArriere = A1; //Sortie led sortie A1
unsigned char LedAR = A3; //Sortie led accusé réception sortie A3

//Déclaration des variables du signal DCC
unsigned char sdcc = 2; //Sortie signal DCC sur sortie 2
unsigned char sdcc2 = 9; //Sortie signal DCC sur sortie 9

//Déclaration des variables des Locomotives
unsigned char NbreLoc2; //Choix des locomotives
unsigned char PremiereLoc = A4; //Un bouton sur la broche 1
unsigned char DeuxiemeLoc = A5; //Un bouton sur la broche 2
unsigned char TroisiemeLoc = 3; //un bouton sur la broche 3
unsigned char QuatriemeLoc = 4; //un bouton sur la broche 4
unsigned char CinquiemeLoc = 5; //un bouton sur la broche 5
unsigned char SixiemeLoc = 6; //un bouton sur la broche 6
unsigned char SeptiemeLoc = 7; //un bouton sur la broche 7

//Déclaration des boutons en entrée
unsigned char ArretUrg = 10; //Déclaration bouton Arrêt Urgence Pin 10
unsigned char NbreLoc = 11; //Déclaration bouton Nombre de Loc Pin 11
unsigned char VarNbreLoc; //Variable état nombre de loc
unsigned char ValidTrame = 12; //Déclaration bouton Valid Trame Pin 12

//Déclaration du bouton en entrée
unsigned char SensdeMarche = 13; //Déclaration bouton Sens de Marche Pin 13
unsigned char VarComm5; //Variable état sens de marche

//Déclaration variable valeur Lue pour potentiomètre de vitesse
unsigned int ValeurLue; //Détermination de la vitesse par le potentiomètre

//Déclaration variable de boucle trame de synchronisation
unsigned char i; //Variable pour la boucle

//Déclaration des variables de sélection du numéro de locomotive
int etat1; int etat2; int etat3; int etat4; int etat5; int etat6; int etat7; int etat8;

```

```

//Déclaration des variables de sélection de l'adresse de la locomotive (octet d'adresse)
unsigned char adr0;unsigned char adr1;unsigned char adr2;unsigned char adr3;
unsigned char adr4;unsigned char adr5;unsigned char adr6;unsigned char adr7;

//Déclaration des variables de commande de la locomotive (octet de commande)
unsigned char comm0;unsigned char comm1;unsigned char comm2;unsigned char comm3;
unsigned char comm4;unsigned char comm5;unsigned char comm6;unsigned char comm7;

//Déclaration des variables pour le calcul du OU exclusif (octet de contrôle)
unsigned char cont0;unsigned char cont1;unsigned char cont2;unsigned char cont3;
unsigned char cont4;unsigned char cont5;unsigned char cont6;unsigned char cont7;

//Déclaration des variables pour la FonctionFL
unsigned char Fonction = 8;//Déclaration bouton FonctionFL Pin 8
int FonctionFL;
int boucleTrameFL1 = 0;
int boucleTrameFL0 = 0;
int compteur;//Compteur boucle fonction
////////////////////////////////////
void setup()
{
//Déclaration Interruption
cli();// Désactive l'interruption globale
bitClear (TCCR2A, WGM20); // WGM20 = 0 //Mise à 0 bit WG20 du registre TCCR2A
bitClear (TCCR2A, WGM21); // WGM21 = 0 //Mise à 0 bit WG21 du registre TCCR2A
TCCR2B = 0b00000110; // Mise à 0 bit WG22 du registre TCCR2B et Réglage CS22 à 1, CS21 à 1 et CS20 à 0 pour une division par 256 (soit 16 micro-s) du signal d'horloge
TIMSK2 = 0b00000001; // Bit TOIE2 à 1 pour autoriser les interruptions locales
sei();// Active l'interruption globale
////////////////////////////////////
//Déclaration des PINS analogiques en numériques sauf celui utilisé pour la détermination de la vitesse (A2)et sens de marche (A0 et A1)
pinMode(A0, OUTPUT);
pinMode(A1, OUTPUT);
pinMode(A3, OUTPUT);
pinMode(A4, OUTPUT);
pinMode(A5, OUTPUT);
////////////////////////////////////
//Déclaration sortie DCC
pinMode(sdcc, OUTPUT); //Pin 2 en sortie pour signal DCC
pinMode(sdcc2,OUTPUT); //Pin 9 en sortie pour signal DCC 2
////////////////////////////////////
//Données constantes de la trame DCC : A déplacer dans le void Loop car sinon la trame transmis n'est plus correcte
//du fait du changement de l'octet de commande
//adr7 = 0; //Le bit7 d'adresse à 0 signale un octet d'adresse courte
//comm7 = 0; //Le bit7 de commande à 0 signale un octet de commande, indique un codage de vitesse de 14 ou 28 pas
//comm6 = 1; //Le bit6 de commande à 1 signale un octet de commande, indique un codage de vitesse de 14 ou 28 pas
////////////////////////////////////

```

```

//Déclaration des boutons en entrées
pinMode(ArretUrg, INPUT_PULLUP);//On met le bouton en entrée 10
pinMode(NbreLoc, INPUT_PULLUP);//On met le bouton en entrée 11
pinMode(ValidTrame, INPUT_PULLUP);//On met le bouton en entrée 12
pinMode(SensdeMarche, INPUT_PULLUP);//On met le bouton en entrée 13
pinMode(Fonction, INPUT_PULLUP);//On met le bouton en entrée 8

////////////////////////////////////

//Déclaration des Locs en entrées
digitalWrite(PremiereLoc, INPUT_PULLUP);//On active la résistance de pull-up en mettant la broche à l'état haut mais cela reste toujours une entrée)
digitalWrite(DeuxiemeLoc, INPUT_PULLUP);//On active la résistance de pull-up en mettant la broche à l'état haut mais cela reste toujours une entrée)
digitalWrite(TroisiemeLoc, INPUT_PULLUP);//on active la résistance de pull-up en mettant la broche à l'état haut mais cela reste toujours une entrée)
digitalWrite(QuatriemeLoc, INPUT_PULLUP);//on active la résistance de pull-up en mettant la broche à l'état haut mais cela reste toujours une entrée)
digitalWrite(CinquiemeLoc, INPUT_PULLUP);//on active la résistance de pull-up en mettant la broche à l'état haut mais cela reste toujours une entrée)
digitalWrite(SixiemeLoc, INPUT_PULLUP);//on active la résistance de pull-up en mettant la broche à l'état haut mais cela reste toujours une entrée)
digitalWrite(SeptiemeLoc, INPUT_PULLUP);//on active la résistance de pull-up en mettant la broche à l'état haut mais cela reste toujours une entrée)
}

////////////////////////////////////

// Routine d'interruption
ISR(TIMER2_OVF_vect) //Interruption par débordement du TIMER 2
{
TCNT2 = 256 - 30; // Préchargement d'une valeur dans TCNT2 pour déterminer temps de débordement du TIMER 2 (30 x 16 µS = 0.48 ms)

if (varCompteur++ > ComptageInterrupt && interrupt==1) // Détermine fréquence de clignotement par variable varCompteur et interrupt à 1 (125=60ms 120=57.6ms)
{
varCompteur = 0;
LedCS; //Changement d'état de la LED changement Signal
}
}

////////////////////////////////////
////////////////////////////////////

void loop()
{
interrupt = 0;//Remise à zéro de la variable interrupt pour éteindre la LEDCS

////Données constantes de la trame DCC : Déplacées dans le void Loop car sinon la trame transmise n'est plus correcte

//du fait du changement de l'octet de commande
adr7 = 0; //Le bit7 d'adresse à 0 signale un octet d'adresse courte
comm7 = 0; //Le bit7 de commande à 0 signale un octet de commande, indique un codage de vitesse de 14 ou 28 pas
comm6 = 1; //Le bit6 de commande à 1 signale un octet de commande, indique un codage de vitesse de 14 ou 28 pas

////////////////////////////////////

// Détermination de la combinaison des boutons activés, ArrêtUrg ou ValidTrame avec priorité au bouton ArrêtUrg

//Trame Arrêt Urgence
while (digitalRead(ArretUrg) == HIGH && digitalRead(ValidTrame) == LOW)// Tant que ArretUrg est High et ValidTrame est LOW = Trame Urgence Envoyée Centrale
Inopérente pour le reste
{
digitalWrite(LedAR,HIGH);//Lampe CS Allumée, centrale bloquée
delayMicroseconds(trame);//Verrouillage oscilloscope
TrameArretUrg();//Envoie Trame Arrêt Urgence
}
}

```

```

}

//Si Arrêt Urgence est HIGH et ValidTrame est HIGH

//Trame Arrêt Urgence

while (digitalRead(ArretUrg) == HIGH && digitalRead(ValidTrame) == HIGH)// Tant que ArretUrg est High et ValidTrame est HIGH = Trame Urgence Envoyée Centrale Inopérente pour le reste

{

digitalWrite(LedAR,HIGH);//Lampe CS Allumée, centrale bloquée

delayMicroseconds(trame);//Verrouillage oscilloscope

TrameArretUrg();//Envoie Trame Arrêt Urgence

}

// Trame Validation Trame

while (digitalRead(ValidTrame) == HIGH && digitalRead(ArretUrg) == LOW)// Tant que ValidTrame est High et ArretUrg est LOW = ValidTrame Envoyée Centrale Inopérente pour le reste

{digitalWrite(LedAR,HIGH);//Lampe CS Allumée, centrale bloquée

delayMicroseconds(trame);//Verrouillage oscilloscope

ArretValidTrame();//Envoie Trame ValidTrame

}

////////////////////

//Appel fonctionFL1 ou FonctionFLO

FonctionFL = digitalRead(Fonction);

if (FonctionFL == HIGH && (boucleTrameFL1) == 0)

{

boucleTrameFL1 = 1;//Empêche la rentrée dans la boucle en permanence

for(int compteur = 0; compteur < 5; compteur++) //Envoi 5 fois la FonctionFL1 pour sécuriser la réalisation de l'ordre

{FonctionFL1();} // Appel FonctionFL1

}

if (FonctionFL == LOW && (boucleTrameFLO) == 0)

{

boucleTrameFLO = 1;//Empêche la rentrée dans la boucle en permanence

for(int compteur = 0; compteur < 5; compteur++)//Envoi 5 fois la FonctionFLO pour sécuriser la réalisation de l'ordre

{FonctionFLO();} // Appel FonctionFLO

}

////////////////////

//Détermine si le bouton nombre de loc est actif

//valeur bit commande nombre de machine

VarNbreLoc = digitalRead(NbreLoc);

if (VarNbreLoc == HIGH)

{NbreLoc2=1;} // Huit deuxième loc

if (VarNbreLoc == LOW)

{NbreLoc2=0;} // Huit première loc

////////////////////

//Détermine le sens de marche

//valeur bit commande 5 sens de marche

VarComm5 = digitalRead(SensdeMarche); //valeur bit 5 octet commande : bit sens de marche, un seul bouton pour toutes les locs.

//Choisir le sens en fonction du mouvement de la loc avant d'envoyer la salve

if (VarComm5 == HIGH)

{

```

```

comm5=1;
PORTC |= (1<<0); //Equivalent à digitalWrite(LedAvant,HIGH)
PORTC &~ (1<<1); //Equivalent à digitalWrite(LedArriere,LOW)
}
if (VarComm5 == LOW)
{
comm5=0; //Sens marche arrière
PORTC &~ (1<<0); //Equivalent à digitalWrite(LedAvant,LOW)
PORTC |= (1<<1); //Equivalent à digitalWrite(LedArriere,HIGH)
}
////////////////////
//Détermine le numéro de la locomotive sélectionnée avec position du bouton "nombre de loc"
//Première Loc
etat1 = digitalRead(PremiereLoc); //Choix Loc 1 ou 8
if (etat1 == LOW && (NbreLoc2) == 0)
{
AdresseLoc1(); //Appel Adresse loc 1
}
if (etat1 == LOW && (NbreLoc2) == 1) //Choix Loc 9
{
AdresseLoc8(); //Appel Adresse loc 8 (65)
}
//Deuxième Loc
etat2 = digitalRead(DeuxiemeLoc); //Choix Loc 2 ou 9
if (etat2 == LOW && (NbreLoc2) == 0)
{
AdresseLoc2(); //Appel Adresse loc 2
}
if (etat2 == LOW && (NbreLoc2) == 1) //Choix Loc 9
{
AdresseLoc9(); //Appel Adresse loc 9 (66)
}
//Troisième Loc
etat3 = digitalRead(TroisiemeLoc); //Choix Loc 3 ou 10
if (etat3 == LOW && (NbreLoc2) == 0)
{
AdresseLoc3(); //Appel Adresse loc 3
}
if (etat3 == LOW && (NbreLoc2) == 1)
{
AdresseLoc10(); //Appel Adresse loc 10 (67)
}
//Quatrième Loc
etat4 = digitalRead(QuatriemeLoc); //Choix Loc 4 ou 11
if (etat4 == LOW && (NbreLoc2) == 0)
{

```

```

AdresseLoc4());//Appel Adresse loc 4
}
if (etat4 == LOW && (NbreLoc2) == 1)
{
AdresseLoc11());//Appel Adresse loc 11 (68)
}
//Cinquième Loc
etat5 = digitalRead(CinquiemeLoc);//Choix Loc 5 ou 12
if (etat5 == LOW && (NbreLoc2) == 0)
{
AdresseLoc5());//Appel Adresse loc 5
}
if (etat5 == LOW && (NbreLoc2) == 1)
{AdresseLoc12());//Appel Adresse loc 12 (69)
}
//Sixième Loc
etat6 = digitalRead(SixiemeLoc);//Choix Loc 6 ou 13
if (etat6 == LOW && (NbreLoc2) == 0)
{
AdresseLoc6());//Appel Adresse loc 6
}
if (etat6 == LOW && (NbreLoc2) == 1)
{
AdresseLoc13());//Appel Adresse loc 13 (70)
}
//Septième Loc
etat7 = digitalRead(SeptiemeLoc);//Choix Loc 7 ou 14
if (etat7 == LOW && (NbreLoc2) == 0)
{
AdresseLoc7());//Appel Adresse loc 7
}
if (etat7 == LOW && (NbreLoc2) == 1)
{
AdresseLoc14());//Appel Adresse loc 14 (71)
}
////////////////////
//Valeur Lue par le convertisseur analogique/numérique en fonction de la position du potentiomètre de vitesse
ValeurLue = analogRead(A2); //la valeur lue sera comprise entre 0 et 1023 (potentiomètre) sur la broche A2
//La tension est égale à : ((ValeurLue*5)/1024)
//En fonction du résultat du convertisseur analogique/numérique, positionnement des bits pour régler le pas de vitesse dans la trame.
if (ValeurLue < 10)(comm4=0,comm3=0,comm2=0,comm1=0,comm0=0); //0
else if (ValeurLue > 10 && ValeurLue <= 30)(comm4=0,comm3=0,comm2=0,comm1=1,comm0=0); //1
else if (ValeurLue > 30 && ValeurLue <= 65)(comm4=1,comm3=0,comm2=0,comm1=1,comm0=0); //2
else if (ValeurLue > 65 && ValeurLue <= 100)(comm4=0,comm3=0,comm2=0,comm1=1,comm0=1); //3
else if (ValeurLue > 100 && ValeurLue <= 138)(comm4=1,comm3=0,comm2=0,comm1=1,comm0=1); //4
else if (ValeurLue > 138 && ValeurLue <= 175)(comm4=0,comm3=0,comm2=1,comm1=0,comm0=0); //5

```

```

else if (ValeurLue > 175 && ValeurLue <= 212)(comm4=1,comm3=0,comm2=1,comm1=0,comm0=0); //6
else if (ValeurLue > 212 && ValeurLue <= 249)(comm4=0,comm3=0,comm2=1,comm1=0,comm0=1); //7
else if (ValeurLue > 249 && ValeurLue <= 285)(comm4=1,comm3=0,comm2=1,comm1=0,comm0=1); //8
else if (ValeurLue > 285 && ValeurLue <= 320)(comm4=0,comm3=0,comm2=1,comm1=1,comm0=0); //9
else if (ValeurLue > 320 && ValeurLue <= 360)(comm4=1,comm3=0,comm2=1,comm1=1,comm0=0); //10
else if (ValeurLue > 360 && ValeurLue <= 395)(comm4=0,comm3=0,comm2=1,comm1=1,comm0=1); //11
else if (ValeurLue > 395 && ValeurLue <= 432)(comm4=1,comm3=0,comm2=1,comm1=1,comm0=1); //12
else if (ValeurLue > 432 && ValeurLue <= 467)(comm4=0,comm3=1,comm2=0,comm1=0,comm0=0); //13
else if (ValeurLue > 467 && ValeurLue <= 504)(comm4=1,comm3=1,comm2=0,comm1=0,comm0=0); //14
else if (ValeurLue > 504 && ValeurLue <= 540)(comm4=0,comm3=1,comm2=0,comm1=0,comm0=1); //15
else if (ValeurLue > 540 && ValeurLue <= 577)(comm4=1,comm3=1,comm2=0,comm1=0,comm0=1); //16
else if (ValeurLue > 577 && ValeurLue <= 603)(comm4=0,comm3=1,comm2=0,comm1=1,comm0=0); //17
else if (ValeurLue > 603 && ValeurLue <= 649)(comm4=1,comm3=1,comm2=0,comm1=1,comm0=0); //18
else if (ValeurLue > 649 && ValeurLue <= 687)(comm4=0,comm3=1,comm2=0,comm1=1,comm0=1); //19
else if (ValeurLue > 687 && ValeurLue <= 723)(comm4=1,comm3=1,comm2=0,comm1=1,comm0=1); //20
else if (ValeurLue > 723 && ValeurLue <= 760)(comm4=0,comm3=1,comm2=1,comm1=0,comm0=0); //21
else if (ValeurLue > 760 && ValeurLue <= 796)(comm4=1,comm3=1,comm2=1,comm1=0,comm0=0); //22
else if (ValeurLue > 796 && ValeurLue <= 833)(comm4=0,comm3=1,comm2=1,comm1=0,comm0=1); //23
else if (ValeurLue > 833 && ValeurLue <= 869)(comm4=1,comm3=1,comm2=1,comm1=0,comm0=1); //24
else if (ValeurLue > 869 && ValeurLue <= 906)(comm4=0,comm3=1,comm2=1,comm1=1,comm0=0); //25
else if (ValeurLue > 906 && ValeurLue <= 941)(comm4=1,comm3=1,comm2=1,comm1=1,comm0=0); //26
else if (ValeurLue > 941 && ValeurLue <= 977)(comm4=0,comm3=1,comm2=1,comm1=1,comm0=1); //27
else if (ValeurLue > 977 && ValeurLue < 1024)(comm4=1,comm3=1,comm2=1,comm1=1,comm0=1); //28

//Serial.println (ValeurLue);

////////////////////////////////////

//Calcul du OU Exclusif

cont0 = adr0 ^ comm0; // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont1 = adr1 ^ comm1; // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont2 = adr2 ^ comm2; // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont3 = adr3 ^ comm3; // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont4 = adr4 ^ comm4; // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont5 = adr5 ^ comm5; // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont6 = adr6 ^ comm6; // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont7 = adr7 ^ comm7; // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle

////////////////////////////////////

// Génération des paquets DCC

// Octet de synchronisation

for ( i=0; i <= 16; i++)
{
  bitun(); // La centrale transmet 16 bits à 1
}

////////////////////////////////////

bitzero(); // Bit à 0 de séparation

////////////////////////////////////

// Octet d'adresse

//La centrale transmet l'octet d'adresse

```

```
bitzero();//adr 7

if (adr6==1) bitun(); // Si bit à 1

if (adr6==0) bitzero(); // Si bit à 0

if (adr5==1) bitun(); // Si bit à 1

if (adr5==0) bitzero(); // Si bit à 0

if (adr4==1) bitun(); // Si bit à 1

if (adr4==0) bitzero(); // Si bit à 0

if (adr3==1) bitun(); // Si bit à 1

if (adr3==0) bitzero(); // Si bit à 0

if (adr2==1) bitun(); // Si bit à 1

if (adr2==0) bitzero(); // Si bit à 0

if (adr1==1) bitun(); // Si bit à 1

if (adr1==0) bitzero(); // Si bit à 0

if (adr0==1) bitun(); // Si bit à 1

if (adr0==0) bitzero(); // Si bit à 0

////////////////////////////////////

// Bit à zéro de séparation

bitzero();

////////////////////////////////////

// Octet de commande

//La centrale transmet l'octet de commande

bitzero();//comm 7

bitun();//comm 6

if (comm5==1) bitun(); // Si bit à 1

if (comm5==0) bitzero(); // Si bit à 0

if (comm4==1) bitun(); // Si bit à 1

if (comm4==0) bitzero(); // Si bit à 0

if (comm3==1) bitun(); // Si bit à 1

if (comm3==0) bitzero(); // Si bit à 0

if (comm2==1) bitun(); // Si bit à 1

if (comm2==0) bitzero(); // Si bit à 0

if (comm1==1) bitun(); // Si bit à 1

if (comm1==0) bitzero(); // Si bit à 0

if (comm0==1) bitun(); // Si bit à 1

if (comm0==0) bitzero(); // Si bit à 0

////////////////////////////////////

// Bit à zéro de séparation

bitzero();

////////////////////////////////////

// Envoi de l'octet de contrôle

//La centrale transmet l'octet de contrôle

if (cont7==1) bitun(); // Si bit à 1

if (cont7==0) bitzero(); // Si bit à 0

if (cont6==1) bitun(); // Si bit à 1

if (cont6==0) bitzero(); // Si bit à 0

if (cont5==1) bitun(); // Si bit à 1
```

```

if (cont5==0) bitzero(); // Si bit à 0

if (cont4==1) bitun(); // Si bit à 1

if (cont4==0) bitzero(); // Si bit à 0

if (cont3==1) bitun(); // Si bit à 1

if (cont3==0) bitzero(); // Si bit à 0

if (cont2==1) bitun(); // Si bit à 1

if (cont2==0) bitzero(); // Si bit à 0

if (cont1==1) bitun(); // Si bit à 1

if (cont1==0) bitzero(); // Si bit à 0

if (cont0==1) bitun(); // Si bit à 1

if (cont0==0) bitzero(); // Si bit à 0

////////////////////

// Bit à un de fin de transmission

bitun();

////////////////////

// Pause émission de 26 zéros (5ms) avant nouvelle transmission

for ( i=0; i<=25; i++)

bitzero();

////////////////////

} //Fermeture void loop

////////////////////

// Génère un bit ZERO

void bitzero()

{

PORTD &=~ (1<<2); //Equivalent à digitalWrite(sdcc,LOW), Sortie sdcc, état bas

PORTB |= (1<<1); //Equivalent à digitalWrite(sdcc2,HIGH), Sortie sdcc2, état haut

delayMicroseconds(105); // Pause de 116 microsecondes

PORTD |= (1<<2); //Equivalent à digitalWrite(sdcc,HIGH), Sortie sdcc, état haut

PORTB &=~ (1<<1); //Equivalent à digitalWrite(sdcc2,LOW), Sortie sdcc2, état bas

delayMicroseconds(105); // Pause de 116 microsecondes

}

////////////////////

// Génère un bit à UN

void bitun() //Génère un bit à 1

{

PORTD &=~ (1<<2); //Equivalent à digitalWrite(sdcc,LOW), Sortie sdcc, état bas

PORTB |= (1<<1); //Equivalent à digitalWrite(sdcc2,HIGH), Sortie sdcc2, état haut

delayMicroseconds(58); // Pause de 58 microsecondes

PORTD |= (1<<2); //Equivalent à digitalWrite(sdcc,HIGH), Sortie sdcc, état haut

PORTB &=~ (1<<1); //Equivalent à digitalWrite(sdcc2,LOW), Sortie sdcc2, état bas

delayMicroseconds(58); // Pause de 58 microsecondes

}

////////////////////

// Positionne l'adresse de la loc 1

void AdresseLoc1()

{

```

```
adr6 = 0;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 0;adr1 = 0;adr0 = 1;
```

```
interrupt=1;
```

```
}
```

```
////////////////////////////////////
```

```
// Positionne l'adresse de la loc 2
```

```
void AdresseLoc2()
```

```
{
```

```
adr6 = 0;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 0;adr1 = 1;adr0 = 0;
```

```
interrupt=1;
```

```
}
```

```
////////////////////////////////////
```

```
// Positionne l'adresse de la loc 3
```

```
void AdresseLoc3()
```

```
{
```

```
adr6 = 0;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 0;adr1 = 1;adr0 = 1;
```

```
interrupt=1;
```

```
}
```

```
////////////////////////////////////
```

```
// Positionne l'adresse de la loc 4
```

```
void AdresseLoc4()
```

```
{
```

```
adr6 = 0;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 1;adr1 = 0;adr0 = 0;
```

```
interrupt=1;
```

```
}
```

```
////////////////////////////////////
```

```
// Positionne l'adresse de la loc 5
```

```
void AdresseLoc5()
```

```
{
```

```
adr6 = 0;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 1;adr1 = 0;adr0 = 1;
```

```
interrupt=1;
```

```
}
```

```
////////////////////////////////////
```

```
// Positionne l'adresse de la loc 6
```

```
void AdresseLoc6()
```

```
{
```

```
adr6 = 0;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 1;adr1 = 1;adr0 = 0;
```

```
interrupt=1;
```

```
}
```

```
////////////////////////////////////
```

```
// Positionne l'adresse de la loc 7
```

```
void AdresseLoc7()
```

```
{
```

```
adr6 = 0;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 1;adr1 = 1;adr0 = 1;
```

```
interrupt=1;
```

```
}
```

```
////////////////////////////////////
```

```
// Positionne l'adresse de la loc 8 Adresse 65
void AdresseLoc8()
{
adr6 = 1;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 0;adr1 = 0;adr0 = 1;
interrupt=1;
}
```

////////////////////////////////////

```
// Positionne l'adresse de la loc 9 Adresse 66
void AdresseLoc9()
{
adr6 = 1;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 0;adr1 = 1;adr0 = 0;
interrupt=1;
}
```

////////////////////////////////////

```
// Positionne l'adresse de la loc 10 Adresse 67
void AdresseLoc10()
{
adr6 = 1;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 0;adr1 = 1;adr0 = 1;
interrupt=1;
}
```

////////////////////////////////////

```
// Positionne l'adresse de la loc 11 Adresse 68
void AdresseLoc11()
{
adr6 = 1;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 1;adr1 = 0;adr0 = 0;
interrupt=1;
}
```

////////////////////////////////////

```
// Positionne l'adresse de la loc 12 Adresse 69
void AdresseLoc12()
{
adr6 = 1;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 1;adr1 = 0;adr0 = 1;
interrupt=1;
}
```

////////////////////////////////////

```
// Positionne l'adresse de la loc 13 Adresse 70
void AdresseLoc13()
{
adr6 = 1;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 1;adr1 = 1;adr0 = 0;
interrupt=1;
}
```

////////////////////////////////////

```
// Positionne l'adresse de la loc 14 Adresse 71
void AdresseLoc14()
{
adr6 = 1;adr5 = 0;adr4 = 0;adr3 = 0;adr2 = 1;adr1 = 1;adr0 = 1;
}
```



```

//bit zero
bitzero();

////////////////////////////////////

//Octet de données 1
bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();

////////////////////////////////////

//Bit de séparation
//bit zero
bitzero();

////////////////////////////////////

//Octet de données 2
bitzero();bitzero();bitzero();bitzero();bitzero();bitzero();bitzero();bitzero();

////////////////////////////////////

//Bit de séparation
//bit zero
bitzero();

////////////////////////////////////

//Octet de données 3
bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();

////////////////////////////////////

//Fin de transmission
//bit un
bitun();

////////////////////////////////////

// Pause émission de 44 un (5 ms) avant nouvelle transmission
for ( i=0; i<=44; i++)
bitun();

} // Fin Void ArretValidTrame()

// Génère la trame FonctionFL1
void FonctionFL1()//Clignote si aucune action?
{
boucleTrameFLO = 0;//Permet de rentrer dans la boucle FonctionFLO quand manœuvre du bouton en HIGH

//Bit de Synchronisation 16 bits à un
bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();

////////////////////////////////////

//Bit de séparation
//bit zero
bitzero();

////////////////////////////////////

//Octet de données 1
if (adr7==1) bitun(); // Si bit à 1
if (adr7==0) bitzero(); // Si bit à 0
if (adr6==1) bitun(); // Si bit à 1
if (adr6==0) bitzero(); // Si bit à 0
if (adr5==1) bitun(); // Si bit à 1

```

```

if (adr5==0) bitzero(); // Si bit à 0
if (adr4==1) bitun(); // Si bit à 1
if (adr4==0) bitzero(); // Si bit à 0
if (adr3==1) bitun(); // Si bit à 1
if (adr3==0) bitzero(); // Si bit à 0
if (adr2==1) bitun(); // Si bit à 1
if (adr2==0) bitzero(); // Si bit à 0
if (adr1==1) bitun(); // Si bit à 1
if (adr1==0) bitzero(); // Si bit à 0
if (adr0==1) bitun(); // Si bit à 1
if (adr0==0) bitzero(); // Si bit à 0*/
////////////////////
//Bit de séparation
//bit zero
////////////////////
bitzero();
////////////////////
//Octet de données 2
comm7 = 1;comm6 = 0;comm5 = 0;comm4 = 1;comm3 = 0;comm2 = 0;comm1 = 0;comm0 = 0;
bitun();//comm 7
bitzero();//comm 6
if (comm5==1) bitun(); // Si bit à 1
if (comm5==0) bitzero(); // Si bit à 0
if (comm4==1) bitun(); // Si bit à 1
if (comm4==0) bitzero(); // Si bit à 0
if (comm3==1) bitun(); // Si bit à 1
if (comm3==0) bitzero(); // Si bit à 0
if (comm2==1) bitun(); // Si bit à 1
if (comm2==0) bitzero(); // Si bit à 0
if (comm1==1) bitun(); // Si bit à 1
if (comm1==0) bitzero(); // Si bit à 0
if (comm0==1) bitun(); // Si bit à 1
if (comm0==0) bitzero(); // Si bit à 0
////////////////////
//Bit de séparation
//bit zero
////////////////////
bitzero();
////////////////////
//Octet de données 3
//Calcul du OU Exclusif
cont0 = adr0 ^ comm0; // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont1 = adr1 ^ comm1; // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont2 = adr2 ^ comm2; // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont3 = adr3 ^ comm3; // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont4 = adr4 ^ comm4; // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle

```

```

cont5 = adr5 ^ comm5; // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont6 = adr6 ^ comm6; // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont7 = adr7 ^ comm7; // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle

// Envoi de l'octet de contrôle

//La centrale transmet l'octet de contrôle

if (cont7==1) bitun(); // Si bit à 1
if (cont7==0) bitzero(); // Si bit à 0
if (cont6==1) bitun(); // Si bit à 1
if (cont6==0) bitzero(); // Si bit à 0
if (cont5==1) bitun(); // Si bit à 1
if (cont5==0) bitzero(); // Si bit à 0
if (cont4==1) bitun(); // Si bit à 1
if (cont4==0) bitzero(); // Si bit à 0
if (cont3==1) bitun(); // Si bit à 1
if (cont3==0) bitzero(); // Si bit à 0
if (cont2==1) bitun(); // Si bit à 1
if (cont2==0) bitzero(); // Si bit à 0
if (cont1==1) bitun(); // Si bit à 1
if (cont1==0) bitzero(); // Si bit à 0
if (cont0==1) bitun(); // Si bit à 1
if (cont0==0) bitzero(); // Si bit à 0

////////////////////////////////////

//Fin de transmission

//bit un

////////////////////////////////////

bitun();

////////////////////////////////////

// Pause émission de 26 zéros avant nouvelle transmission

////////////////////////////////////

for ( i=0; i<=25; i++)
bitzero();

} //Fin FonctionFL1()

////////////////////////////////////

// Génère la trame FonctionFLO

void FonctionFLO()

{

boucleTrameFL1 = 0; //Permet de rentrer dans la boucle FonctionFL1 quand manœuvre du bouton en HIGH

//Bit de Synchronisation 16 bits à un
bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();bitun();
bitun();bitun();bitun();bitun();bitun();bitun();bitun();

////////////////////////////////////

//Bit de séparation

//bit zero

////////////////////////////////////

bitzero();

////////////////////////////////////

```



```

//Calcul du OU Exclusif
cont0 = adr0 ^ comm0; // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont1 = adr1 ^ comm1; // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont2 = adr2 ^ comm2; // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont3 = adr3 ^ comm3; // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont4 = adr4 ^ comm4; // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont5 = adr5 ^ comm5; // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont6 = adr6 ^ comm6; // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont7 = adr7 ^ comm7; // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle

// Envoi de l'octet de contrôle

//La centrale transmet l'octet de contrôle

if (cont7==1) bitun(); // Si bit à 1
if (cont7==0) bitzero(); // Si bit à 0
if (cont6==1) bitun(); // Si bit à 1
if (cont6==0) bitzero(); // Si bit à 0
if (cont5==1) bitun(); // Si bit à 1
if (cont5==0) bitzero(); // Si bit à 0
if (cont4==1) bitun(); // Si bit à 1
if (cont4==0) bitzero(); // Si bit à 0
if (cont3==1) bitun(); // Si bit à 1
if (cont3==0) bitzero(); // Si bit à 0
if (cont2==1) bitun(); // Si bit à 1
if (cont2==0) bitzero(); // Si bit à 0
if (cont1==1) bitun(); // Si bit à 1
if (cont1==0) bitzero(); // Si bit à 0
if (cont0==1) bitun(); // Si bit à 1
if (cont0==0) bitzero(); // Si bit à 0

////////////////////////////////////
//Fin de transmission

//bit un

////////////////////////////////////

bitun();

////////////////////////////////////

// Pause émission de 26 zéros avant nouvelle transmission

////////////////////////////////////

for ( i=0; i<=25; i++)
bitzero();

} //Fin FonctionFLO

```