```
//////////////////////////////////////
//////////////////////////////////////
/////Centrale DCC Version 2.1.1/////////
///////////////Mise a jour///////////////
//////////////////////////////////////
/////////Développé par AERONEF//////////
//////////////////////////////////////
///Copyright @ Le Site Ferroviaire//////
//////////////////////////////////////
//////////////Avril 2019/////////////////
//////////////////////////////////////
//////////////////////////////////////
#include <EEPROM.h>
boolean autorisation_interrupteur=0;
//Interdit le fonctionnement des interrupteurs des fonctions
//permet la sélection des locs sans problèmes
boolean confirme_raz;//Variable de confirmation du raz
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration variable pour MEMORISATION
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
boolean autorise_touche_memorisation_loc;
boolean autorise_touche_memorisation_loc_e;//Validation par touche E
boolean traitement_appui_touches_memorisation_loc;
int compteur_touche_memorisation_loc;
long toucheAprogrammation_memorisation_loc;
//Utilisée pour enregistrer la première touche appuyée
byte toucheBprogrammation_memorisation_loc;
//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCprogrammation_memorisation_loc;
//Utilisée pour enregistrer la troisième touche appuyée
byte toucheDprogrammation_memorisation_loc;
//Utilisée pour enregistrer la quatrième touche appuyée
byte toucheEprogrammation_memorisation_loc;
//Utilisée pour enregistrer la cinquième touche appuyée
byte toucheFprogrammation_memorisation_loc;
//Utilisée pour enregistrer la sixième touche appuyée
long nbre_memorisation_loc;
boolean autorise_touche_memorisation_adresse_loc;
boolean autorise_touche_memorisation_adresse_loc_e;//Validation par touche E
```

```
int compteur_touche_memorisation_adresse_loc;
int toucheAprogrammation_memorisation_adresse_loc;
//Utilisée pour enregistrer la première touche appuyée
byte toucheBprogrammation_memorisation_adresse_loc;
//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCprogrammation_memorisation_adresse_loc;
//Utilisée pour enregistrer la troisième touche appuyée
int adresse_memorisation_loc;
boolean prog_memo_loc;
//Autorise ou interdit les accès direct des menus par les touches A, B, C, D
//0 = Autorise, 1 = Interdit
boolean autorise_touche_A;
byte octetmsb;//Byte, obligatoire, pour ne prendre que 8 bits de poids fort
byte octetmidsb;//Byte, obligatoire, pour ne prendre que 8 bits de poids intermédiaire
byte octetlsb;//Byte, obligatoire, pour ne prendre que 8 bits de poids faible
long reconstitueoctet;//Variable pour la reconstitution de l'octet
long reconstitueoctet2;//Variable pour la reconstitution de l'octet
long reconstitueoctet3;//Variable pour la reconstitution de l'octet
long reconstituefin;//Variable pour la reconstitution de l'octet
byte resultatoctetmsb;//Variable pour reconstituer le numéro de locomotive en eeprom
byte resultatoctetmidsb;//Variable pour reconstituer le numéro de locomotive en eeprom
byte resultatoctetlsb;//Variable pour reconstituer le numéro de locomotive en eeprom

int adresse_depart;//Variable pour la boucle de reconstitution du numéro de locomotive en eeprom
int adresse_recherche;//Variable pour la boucle de reconstitution du numéro de locomotive en eeprom
int adresse;//Variable pour la boucle de reconstitution du numéro de locomotive en eeprom
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration variable pour PROGRAMMATION ACCESSOIRE
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int autorise_touche_programmation_accessoire;//Interdit ou Autorise la saisie de chiffre pour programmation_accessoire
boolean autorise_touche_programmation_accessoire_e;//Interdit ou Autorise la saisie de chiffre pour validation par touche E
int compteur_touche_programmation_accessoire;
long toucheAprogrammation_accessoire;//Utilisée pour enregistrer la première touche appuyée
byte toucheBprogrammation_accessoire;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCprogrammation_accessoire;//Utilisée pour enregistrer la troisième touche appuyée
int nbre_programmation_accessoire;
int traitement_appui_touches_programmation_accessoire;
int adresse_programmation_accessoire;
```

```
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
boolean adr7programmation_accessoire;boolean adr6programmation_accessoire;
boolean adr5programmation_accessoire;boolean adr4programmation_accessoire;
boolean adr3programmation_accessoire;boolean adr2programmation_accessoire;
boolean adr1programmation_accessoire;boolean adr0programmation_accessoire;
boolean adr27programmation_accessoire;boolean adr26programmation_accessoire;
boolean adr25programmation_accessoire;boolean adr24programmation_accessoire;
boolean adr23programmation_accessoire;boolean adr22programmation_accessoire;
boolean adr21programmation_accessoire;boolean adr20programmation_accessoire;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
boolean cont7programmation_accessoire;boolean cont6programmation_accessoire;
boolean cont5programmation_accessoire;boolean cont4programmation_accessoire;
boolean cont3programmation_accessoire;boolean cont2programmation_accessoire;
boolean cont1programmation_accessoire;boolean cont0programmation_accessoire;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int autorise_touche_programmation_accessoire2;
//Interdit ou Autorise la saisie de chiffre pour programmation_raz
boolean autorise_touche_programmation_accessoire2_e;
//Interdit ou Autorise la saisie de chiffre pour validation par touche E
int compteur_touche_programmation_accessoire2;
long toucheAprogrammation_accessoire2;//Utilisée pour enregistrer la première touche appuyée
byte toucheBprogrammation_accessoire2;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCprogrammation_accessoire2;//Utilisée pour enregistrer la troisième touche appuyée
int nbre_programmation_accessoire2;
int traitement_appui_touches_programmation_accessoire2;
int donnee_programmation_accessoire2;
boolean donnee7programmation_accessoire2;boolean donnee6programmation_accessoire2;
boolean donnee5programmation_accessoire2;boolean donnee4programmation_accessoire2;
boolean donnee3programmation_accessoire2;boolean donnee2programmation_accessoire2;
boolean donnee1programmation_accessoire2;boolean donnee0programmation_accessoire2;
//Prépositionnement des bytes de la trame accessoire pour accès
//et écriture de toutes les CVs accessoires
boolean comm7programmation_accessoire2 = 1;boolean comm6programmation_accessoire2 = 1;
boolean comm5programmation_accessoire2 = 1;boolean comm4programmation_accessoire2 = 0;
//Bytes fixes pour la trame Cv
boolean comm3programmation_accessoire2 = 1;boolean comm2programmation_accessoire2 = 1;
```

```
//Bytes fixes pour la trame accessoire,
boolean comm1programmation_accessoire2;boolean comm0programmation_accessoire2;
//Bytes variables pour la trame accessoire,
boolean cont7programmation_accessoire2;boolean cont6programmation_accessoire2;
boolean cont5programmation_accessoire2;boolean cont4programmation_accessoire2;
boolean cont3programmation_accessoire2;boolean cont2programmation_accessoire2;
boolean cont1programmation_accessoire2;boolean cont0programmation_accessoire2;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int autorise_touche_programmation_accessoire4;
//Interdit ou Autorise la saisie de chiffre pour programmation_raz
boolean autorise_touche_programmation_accessoire4_e;
//Interdit ou Autorise la saisie de chiffre pour validation par touche E
int compteur_touche_programmation_accessoire4;
long toucheAprogrammation_accessoire4;//Utilisée pour enregistrer la première touche appuyée
byte toucheBprogrammation_accessoire4;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCprogrammation_accessoire4;//Utilisée pour enregistrer la troisième touche appuyée
byte toucheDprogrammation_accessoire4;//Utilisée pour enregistrer la quatrième touche appuyée
int nbre_programmation_accessoire4;
int traitement_appui_touches_programmation_accessoire4;
int commcv_programmation_accessoire4;
boolean donnee7programmation_accessoire4;boolean donnee6programmation_accessoire4;
boolean donnee5programmation_accessoire4;boolean donnee4programmation_accessoire4;
boolean donnee3programmation_accessoire4;boolean donnee2programmation_accessoire4;
boolean donnee1programmation_accessoire4;boolean donnee0programmation_accessoire4;
//Prépositionnement des bytes de la trame accessoire pour accès
//et écriture de toutes les Cvs accessoire
boolean comm15programmation_accessoire4 = 1;boolean comm14programmation_accessoire4 = 1;
boolean comm13programmation_accessoire4 = 1;//Tramefixe
boolean comm12programmation_accessoire4 = 0;boolean comm11programmation_accessoire4 = 1;
boolean comm10programmation_accessoire4 = 1;//Tramefixe

boolean commcv9programmation_accessoire4;boolean commcv8programmation_accessoire4;
boolean commcv7programmation_accessoire4;boolean commcv6programmation_accessoire4;
boolean commcv5programmation_accessoire4;boolean commcv4programmation_accessoire4;
boolean commcv3programmation_accessoire4;boolean commcv2programmation_accessoire4;
boolean commcv1programmation_accessoire4;boolean commcv0programmation_accessoire4;

boolean cont7programmation_accessoire4;boolean cont6programmation_accessoire4;
```

```
boolean cont5programmation_accessoire4;boolean cont4programmation_accessoire4;
boolean cont3programmation_accessoire4;boolean cont2programmation_accessoire4;
boolean cont1programmation_accessoire4;boolean cont0programmation_accessoire4;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration variable commande ACCESSOIRE CHOIX DE L'ACCESSOIRE
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int autorise_touche_accessoire;
//Interdit ou Autorise la saisie de chiffre pour programmation_accessoire
boolean autorise_touche_accessoire_e;
//Interdit ou Autorise la saisie de chiffre pour validation par touche E
int compteur_touche_accessoire;
long toucheAaccessoire;//Utilisée pour enregistrer la première touche appuyée
byte toucheBaccessoire;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCaccessoire;//Utilisée pour enregistrer la troisième touche appuyée

int nbre_accessoire;
int traitement_appui_touches_accessoire;
int adresse_accessoire;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
boolean adr7accessoire;boolean adr6accessoire;boolean adr5accessoire;boolean adr4accessoire;
boolean adr3accessoire;boolean adr2accessoire;boolean adr1accessoire;boolean adr0accessoire;

boolean adr27accessoire;boolean adr26accessoire;boolean adr25accessoire;
boolean adr24accessoire;boolean adr23accessoire;boolean adr22accessoire;
boolean adr21accessoire;boolean adr20accessoire;

boolean cont7accessoire;boolean cont6accessoire;boolean cont5accessoire;boolean cont4accessoire;
boolean cont3accessoire;boolean cont2accessoire;boolean cont1accessoire;boolean cont0accessoire;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

boolean autorise_touche_accessoire4;
//Autorise la prise en compte des chiffres de commande d'accessoire
boolean autorise_touche_accessoire44;
//Autorise la prise en compte de l'appui touche pour la touche 'D' de fonction
```

```
boolean autorise_touche_accessoire4_e;
//Autorise la prise en compte de l'appui touche pour la touche 'D' pour validation par touche E
//ou la prise en compte de l'accessoire sélectionné si à 1
int compteur_touche_accessoire4;
long toucheAaccessoire4;//Utilisée pour enregistrer la première touche appuyée
byte toucheBaccessoire4;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCaccessoire4;//Utilisée pour enregistrer la troisième touche appuyée
byte toucheDaccessoire4;//Utilisée pour enregistrer la quatrième touche appuyée

int nbre_accessoire4;
int traitement_appui_touches_accessoire4;
int commande_accessoire4;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
boolean adr7accessoire4;boolean adr6accessoire4;boolean adr5accessoire4;boolean adr4accessoire4;
boolean adr3accessoire4;boolean adr2accessoire4;boolean adr1accessoire4;boolean adr0accessoire4;

boolean commande7accessoire4;
boolean commande6accessoire4;
boolean commande5accessoire4;
boolean commande4accessoire4;
boolean commande3accessoire4;boolean commande2accessoire4;
boolean commande1accessoire4;boolean commande0accessoire4;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
char autorise_touche_choix;//Interdit ou Autorise la saisie de chiffre pour raz ou programmation Cv 1 ou 4
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration variable pour commande locomotive
boolean autorise_touche_7_et_9;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration variable pour programmation FONCTION
int autorise_touche_fonction;//Interdit ou Autorise la saisie de chiffre pour programmation_raz
boolean autorise_touche_fonction_e;
//Interdit ou Autorise la saisie de chiffre pour validation par touche E
int compteur_touche_fonction;
long toucheAfonction;//Utilisée pour enregistrer la première touche appuyée
byte toucheBfonction;//Utilisée pour enregistrer la deuxième touche appuyée
```

```
byte toucheCfonction;//Utilisée pour enregistrer la troisième touche appuyée
byte toucheDfonction;//Utilisée pour enregistrer la quatrième touche appuyée
byte toucheEfonction;//Utilisée pour enregistrer la cinquième touche appuyée

int nbre_fonction;
int traitement_appui_touches_fonction;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration variable pour programmation CV
int compteur_touche_cv;
byte toucheAcv;//Utilisée pour enregistrer la première touche appuyée
byte toucheBcv;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCcv;//Utilisée pour enregistrer la troisième touche appuyée
byte toucheDcv;//Utilisée pour enregistrer la quatrième touche appuyée

int nbre_cv;
int adresse_cv;
boolean autorise_touche_Cv;//Interdit ou Autorise la rentrée dans la fonction programmationCv
boolean autorise_touche_Cv_e;//Interdit ou Autorise la validation par touche E
boolean cont7cv;boolean cont6cv;boolean cont5cv;boolean cont4cv;
boolean cont3cv;boolean cont2cv;boolean cont1cv;boolean cont0cv;

//Prépositionnement des bytes de la trame cv pour accès et écriture de toutes les Cv
boolean comm7cv = 1;boolean comm6cv = 1;boolean comm5cv = 1;boolean comm4cv = 0;
//Bytes fixes pour la trame Cv
boolean comm3cv = 1;boolean comm2cv = 1;//Bytes fixes pour la trame Cv,
boolean comm1cv;boolean comm0cv;//Bytes variables pour la trame Cv,

boolean adr7cv;boolean adr6cv;boolean adr5cv;boolean adr4cv;
boolean adr3cv;boolean adr2cv;boolean adr1cv;boolean adr0cv;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration variable pour programmation CV2
int compteur_touche_cv2;
byte toucheAcv2;//Utilisée pour enregistrer la première touche appuyée
byte toucheBcv2;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCcv2;//Utilisée pour enregistrer la troisième touche appuyée
```

```
byte toucheDcv2;//Utilisée pour enregistrer la quatrième touche appuyée

int nbre_cv2;
int adresse_cv2;
boolean autorise_touche_Cv2;
//Interdit ou Autorise la rentrée dans la fonction programmationCv2
boolean autorise_touche_Cv2_e;//Interdit ou Autorise la validation par touche E
boolean adr7cv2;boolean adr6cv2;boolean adr5cv2;boolean adr4cv2;
boolean adr3cv2;boolean adr2cv2;boolean adr1cv2;boolean adr0cv2;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration variable pour programmation CV4
int compteur_touche_cv4;
byte toucheAcv4;//Utilisée pour enregistrer la première touche appuyée
byte toucheBcv4;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheCcv4;//Utilisée pour enregistrer la troisième touche appuyée
byte toucheDcv4;//Utilisée pour enregistrer la quatrième touche appuyée

int nbre_cv4;
int adresse_cv4;
boolean autorise_touche_Cv4;
//Interdit ou Autorise la rentrée dans la fonction programmationCv4
boolean autorise_touche_Cv4_e;//Interdit ou Autorise la validation par touche E
boolean adr7cv4;boolean adr6cv4;boolean adr5cv4;boolean adr4cv4;
boolean adr3cv4;boolean adr2cv4;boolean adr1cv4;boolean adr0cv4;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int dureetempo = 250;
int dureetempoaccessoire = 100;
int dureetempofonction = 150;
int dureetempomenuprogrammation = 180;
int dureetempolectureeeprom = 180;
int dureetempomemorisationloc = 200;
int dureetempomemoloceeprom = 250;
int dureetemporazeeprom = 200;
int dureetempoprogaccess = 100;
int dureetempocommaccess = 100;

int compteurbuzzer;
```

```
int dureetempobuzzer = 1;
const byte buzzer = 5;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
boolean locSelectionne = 0;//Prépositionne la variable locSelectionne : Priorité loc 1
boolean loc2Selectionne = 0;//Prépositionne la variable loc2Selectionne

boolean Miseenmemoirelocselectionne;//Mise en memoire de la loc sélectionnée :
//RAZ de locSelectionne lors de la commande fonction pour empêcher l'envoi de la trame locomotive
//Permet de retrouver la loc sélectionnée auparavant
//pour retrouver l'affichage et la commande de la loc sélectionnée après
//l'utilisation de la commande fonction
boolean Miseenmemoireloc2selectionne;//Mise en memoire de la loc sélectionnée :
//RAZ de locSelectionne lors de la commande fonction pour empêcher l'envoi de la trame locomotive
//Permet de retrouver la loc sélectionnée auparavant
//pour retrouver l'affichage et la commande de la loc sélectionnée après
//l'utilisation de la commande fonction

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Loc A
char compteur_touche_locomotive = 0;
boolean traitement_appui_touches_locomotive = 0;//Indique qu'un chiffre ou qu'aucun chiffre
//n'a été saisie et stocké dans la variable nbre et interdit ensuite le traitement du chiffre saisie
byte toucheA_locomotive;//Utilisée pour enregistrer la première touche appuyée
byte toucheB_locomotive;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheC_locomotive;//Utilisée pour enregistrer la troisième touche appuyée
byte toucheD_locomotive;//Utilisée pour enregistrer la quatrième touche appuyée
byte toucheE_locomotive;//Utilisée pour enregistrer la cinquième touche appuyée
byte toucheF_locomotive;//Utilisée pour enregistrer la sixième touche appuyée
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Loc B
char compteur_touche_locomotive2 = 0;
boolean traitement_appui_touches_locomotive2 = 0;//Indique qu'un chiffre ou qu'aucun chiffre n'a été saisie
//et stocké dans la variable nbre et interdit ensuite le traitement du chiffre saisie
byte toucheA_locomotive2;//Utilisée pour enregistrer la première touche appuyée
byte toucheB_locomotive2;//Utilisée pour enregistrer la deuxième touche appuyée
byte toucheC_locomotive2;//Utilisée pour enregistrer la troisième touche appuyée
byte toucheD_locomotive2;//Utilisée pour enregistrer la quatrième touche appuyée
byte toucheE_locomotive2;//Utilisée pour enregistrer la cinquième touche appuyée
```

```
byte toucheF_locomotive2;//Utilisée pour enregistrer la sixième touche appuyée
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration des variables de sélection de l'adresse de la locomotive A octet 1
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 sélection de l'adresse de la locomotive B octet 1
unsigned char adr20loc;unsigned char adr21loc;unsigned char adr22loc;unsigned char adr23loc;
unsigned char adr24loc;unsigned char adr25loc;unsigned char adr26loc;unsigned char adr27loc;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration des variables de commande de la locomotive A
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 de commande de la locomotive B
unsigned char comm20loc;unsigned char comm21loc;unsigned char comm22loc;unsigned char comm23loc;
unsigned char comm24loc;unsigned char comm25loc;unsigned char comm26loc;unsigned char comm27loc;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration des variables pour le calcul du OU exclusif loc A
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 le calcul du OU exclusif loc B
unsigned char cont20loc;unsigned char cont21loc;unsigned char cont22loc;unsigned char cont23loc;
unsigned char cont24loc;unsigned char cont25loc;unsigned char cont26loc;unsigned char cont27loc;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration variable valeur Lue pour potentiomètre de vitesse loc A
unsigned int ValeurLue; //Détermination de la vitesse par le potentiomètre
//Déclaration variable valeur Lue pour potentiomètre de vitesse loc B
unsigned int ValeurLue2; //Détermination de la vitesse par le potentiomètre
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration sortie DCC
const byte sdcc = 2; //Pin 2 en sortie pour signal DCC
const byte sdcc2 = 3; //Pin 3 en sortie pour signal DCC 2
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration des variables des LED
const byte ledbleue=23;//Pin 23
const byte ledverte=25;//Pin 25
```

```
const byte ledjaune=A12;//Pin A12
const byte ledrouge=A13;//Pin A13
const byte ledSensArriere=A14;//Pin A14
const byte ledSensAvant=A15;//Pin A15

const byte ledSensArriere2=A11;//Pin A11
const byte ledSensAvant2=A10;//Pin A10
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Déclaration des boutons en entrée
const byte ArretUrg = 7;//Déclaration bouton Arrêt Urgence Pin 7
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Loc sur bouton A
const byte SensdeMarche = 8;//Déclaration bouton Valid Trame Pin 8
boolean VarComm5;//Variable état sens de marche
boolean sensmarcheloc1;

//Loc sur bouton B
const byte SensdeMarche2 = 6;//Déclaration bouton Valid Trame Pin 6
boolean VarComm52;//Variable état sens de marche
boolean sensmarcheloc2;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//char vitesse;
//Loc A
int adresse_locomotive;//copie du nombre_locomotive
long nbre_locomotive;//Utilisée pour enregistrer l'intégralité du chiffre saisie
//Loc B
int adresse_locomotive2;//copie du nombre_locomotive
long nbre_locomotive2;//Utilisée pour enregistrer l'intégralité du chiffre saisie

boolean mode_manoeuvre = 0;
boolean mode2_manoeuvre = 0;
int ConversionValeurLue;
boolean autorise_touche_1_et_4;
int Conversionaffichage;
int Conversionmodemanoeuvre;
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int i;
```

```
int compteur;//compteur pour la trame fonction

////////////////////////////////////////////////////////////////////////////
boolean retourdetrame;//Equivaut à une pression sur la touche *

////////////////////////////////////////////////////////////////////////////
//Déclaration des variable de fonction
int FonctionFL;
int FonctionF1;
int FonctionF2;
int FonctionF3;
int FonctionF4;
boolean boucleTrameFL0;//Permet la permutation de la fonction FL Haut ou Bas
boolean boucleTrameFL1;//Permet la permutation de la fonction FL Haut ou Bas
boolean boucleTrameF11;//Permet la permutation de la fonction F1 Haut ou Bas
boolean boucleTrameF10;//Permet la permutation de la fonction F1 Haut ou Bas
boolean boucleTrameF21;//Permet la permutation de la fonction F2 Haut ou Bas
boolean boucleTrameF20;//Permet la permutation de la fonction F2 Haut ou Bas
boolean boucleTrameF31;//Permet la permutation de la fonction F3 Haut ou Bas
boolean boucleTrameF30;//Permet la permutation de la fonction F3 Haut ou Bas
boolean boucleTrameF41;//Permet la permutation de la fonction F4 Haut ou Bas
boolean boucleTrameF40;//Permet la permutation de la fonction F4 Haut ou Bas

//Déclaration pour position interrupteur fonction
boolean octetFonctionFL;
boolean octetFonctionF4;
boolean octetFonctionF3;
boolean octetFonctionF2;
boolean octetFonctionF1;

int octetFonction = 0b10000000;//Prépositionne l'octet de commande de fonction
int octetFonction1011 = 0b10110000;//Prépositionne l'octet de commande de fonction
int octetFonction1010 = 0b10100000;//Prépositionne l'octet de commande de fonction
int octetFonction11011110 = 0b00000000;//Prépositionne l'octet de commande de fonction
int octetFonction11011111 = 0b00000000;//Prépositionne l'octet de commande de fonction

////////////////////////////////////////////////////////////
/////////Variables pour l'octet 3 des fonctions 13 à 28//////
////////////////////////////////////////////////////////////
boolean fonc17;boolean fonc16;boolean fonc15;boolean fonc14;
boolean fonc13;boolean fonc12;boolean fonc11;boolean fonc10;
```

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
boolean fonc27;boolean fonc26;boolean fonc25;boolean fonc24;
boolean fonc23;boolean fonc22;boolean fonc21;boolean fonc20;
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

//Déclaration des pins de l'arduino pour les boutons de fonction
const byte Fonction0 = 13;//Déclaration bouton FonctionFL Pin 13
const byte Fonction1 = 12;//Déclaration bouton Fonction1 Pin 12
const byte Fonction2 = 11;//Déclaration bouton Fonction2 Pin 11
const byte Fonction3 = 10;//Déclaration bouton Fonction3 Pin 10
const byte Fonction4 = 9;//Déclaration bouton Fonction4 Pin 9

// --- Déclaration des variables du clavier ---
//Loc A
boolean autorise_touche_locomotive;//Variable locomotive,
boolean autorise_touche_locomotive_e;//Variable locomotive, validation par touche E
//empêche ou autorise l'affichage de nombres si dans la boucle locomotive ou pas
//Loc B
boolean autorise_touche_locomotive2;//Variable locomotive2,
boolean autorise_touche_locomotive2_e;//Variable locomotive, validation par touche E
//empêche ou autorise l'affichage de nombres si dans la boucle locomotive ou pas
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

// --- Fonctionnalités utilisées ---
// Utilise un afficheur LCD alphanumérique 4x20 en mode 4 bits
// Utilise un Clavier matriciel 4x4 (16 touches)

// --- Circuit à réaliser ---
// Connecter  sur la broche 45 la Colonne 1 du Clavier
// Connecter  sur la broche 43 la Colonne 2 du Clavier
// Connecter  sur la broche 41 la Colonne 2 du Clavier
// Connecter  sur la broche 39 la Colonne 4 du Clavier

// Connecter  sur la broche 37 la Ligne 1 du Clavier
// Connecter  sur la broche 35 la Ligne 2 du Clavier
// Connecter  sur la broche 33 la Ligne 3 du Clavier
// Connecter  sur la broche 31 la Ligne 4 du Clavier
```

```
// Connecter  sur la broche 32 la broche RS du LCD
// Connecter  sur la broche 30 la broche E du LCD
// Connecter  sur la broche 22 la broche D4 du LCD
// Connecter  sur la broche 24 la broche D5 du LCD
// Connecter  sur la broche 26 la broche D6 du LCD
// Connecter  sur la broche 28 la broche D7 du LCD

//**************** Entête déclarative *******
// A ce niveau sont déclarées les librairies, les constantes, les variables...

// --- Inclusion des librairies utilisées ---

#include <LiquidCrystalFast.h> // Inclusion de la librairie pour afficheur LCD
// Création d'un objet LiquidCrystal,initialisation LCD en mode 4 bits
byte Locomotive[] =
{
  B01110,
  B10001,
  B10001,
  B11111,
  B11111,
  B11111,
  B01010,
  B11111
};

byte sensAv[] =
{
  B00100,
  B01110,
  B10101,
  B00100,
  B00100,
  B00100,
  B00100,
  B01110
  };

byte sensAr[] =
{
```

```
  B01110,
  B00100,
  B00100,
  B00100,
  B00100,
  B10101,
  B01110,
  B00100
  };

#include <Keypad.h>

// --- Déclaration des constantes ---

//--- Constantes utilisées avec le clavier 4x4
const byte LIGNES = 4; // 4 lignes
const byte COLONNES = 4; //4 colonnes


// --- constantes des broches ---
//Clavier
//Colonnes
const byte C1=45; //declaration constante de broche 45
const byte C2=43; //declaration constante de broche 43
const byte C3=41; //declaration constante de broche 41
const byte C4=39; //declaration constante de broche 39
//Lignes
const byte L1=37; //declaration constante de broche 37
const byte L2=35; //declaration constante de broche 35
const byte L3=33; //declaration constante de broche 33
const byte L4=31; //declaration constante de broche 31

//LCD
const byte RS=32; //declaration constante de broche 32
const byte E=30; //declaration constante de broche 30
const byte D4=22; //declaration constante de broche 22
const byte D5=24; //declaration constante de broche 24
const byte D6=26; //declaration constante de broche 26
const byte D7=28; //declaration constante de broche 28
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
```

```
// --- Déclaration des variables de boucle du clavier ---
// --- Déclaration des variables globales ---
//--- Définition des touches tableau 1
char touches[LIGNES][COLONNES] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'E','0','=','D'}
};

// tableaux de lignes et colonnes
byte BrochesLignes[LIGNES] = {L1, L2, L3, L4}; //connexions utilisées pour les broches de lignes du clavier
byte BrochesColonnes[COLONNES] = {C1, C2, C3, C4}; //connexions utilisées pour les broches de colonnes du clavier

char touche; // variable de stockage valeur touche appuyée

// --- Déclaration des objets utiles pour les fonctionnalités utilisées ---

LiquidCrystalFast lcd(RS, E, D4, D5, D6, D7);
// Création d'un objet LiquidCrystal,initialisation LCD en mode 4 bits


// création d'un objet keypad = initialisation clavier
Keypad keypad = Keypad( makeKeymap(touches), BrochesLignes, BrochesColonnes, LIGNES, COLONNES );//tableau 1

// les broches de lignes sont automatiquement configurées en ENTREE avec pullup interne activé
// les broches de colonnes sont automatiquement configurées en SORTIE


//**************** FONCTION SETUP = Code d'initialisation *****
// La fonction setup() est exécutée en premier et 1 seule fois, au démarrage du programme

void setup()   { // debut de la fonction setup()
// cree des nouveaux caracteres
lcd.createChar(0, Locomotive);
lcd.createChar(1, sensAv);
lcd.createChar(2, sensAr);

pinMode(buzzer, OUTPUT);
```

```
pinMode(ledbleue, OUTPUT);
pinMode(ledverte, OUTPUT);
pinMode(ledjaune, OUTPUT);
pinMode(ledrouge, OUTPUT);
pinMode(ledSensArriere, OUTPUT);
pinMode(ledSensAvant, OUTPUT);
pinMode(ledSensArriere2, OUTPUT);
pinMode(ledSensAvant2, OUTPUT);

//Déclaration des boutons en entrées
pinMode(Fonction0, INPUT_PULLUP);//On met le bouton en entrée A1
pinMode(Fonction1, INPUT_PULLUP);//On met le bouton en entrée A2
pinMode(Fonction2, INPUT_PULLUP);//On met le bouton en entrée A3
pinMode(Fonction3, INPUT_PULLUP);//On met le bouton en entrée A4
pinMode(Fonction4, INPUT_PULLUP);//On met le bouton en entrée A5

//Déclaration des boutons en entrées
pinMode(ArretUrg, INPUT_PULLUP);//On met le bouton en entrée 7
pinMode(SensdeMarche, INPUT_PULLUP);//On met le bouton en entrée 8
pinMode(SensdeMarche2, INPUT_PULLUP);//On met le bouton en entrée 6
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
//Déclaration sortie DCC
pinMode(sdcc, OUTPUT); //Pin 2 en sortie pour signal DCC
pinMode(sdcc2,OUTPUT); //Pin 3 en sortie pour signal DCC 2

////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
// --- ici instructions à exécuter au démarrage ---

lcd.begin(20,4); // Initialise le LCD avec 20 colonnes x 4 lignes

delay(10); // Pause rapide pour laisser le temps d'initialisation

// Test du LCD
lcd.setCursor(4,0);//Place le curseur colonne 4, ligne 1
lcd.print("CENTRALE DCC") ; // Affiche la chaîne texte
```

```
lcd.setCursor(3,1);//Place le curseur colonne 5, ligne 2
lcd.print("Version 2.1.1") ; // Affiche la chaîne texte

lcd.setCursor(0,2);//Place le curseur colonne 0, ligne 2
lcd.print("Programme by AERONEF") ; // Affiche la chaîne texte
delay(2000); // pause de 2 secondes

lcd.noDisplay();//Efface le texte, mais le garde en mémoire
delay(1000); // pause de 1 seconde

lcd.display();//Affiche le texte gardé en mémoire
delay(1500); // pause de 1.5 secondes

lcd.clear(); // // Efface l'écran
delay(10); // Pour laisser le temps d'effacer écran

lcd.setCursor(0,1);//Place le curseur colonne 0, ligne 2
lcd.print("Le Site Ferroviaire") ; // Affiche la chaîne texte
delay(1500); // pause de 1.5 secondes

lcd.noDisplay();//Efface le texte, mais le garde en mémoire
delay(1000); // pause de 1 seconde

lcd.display();//Affiche le texte gardé en mémoire
delay(1500); // pause de 1.5 secondes

lcd.clear(); // // Efface l'écran
delay(10); // Pour laisser le temps d'effacer l'écran
lcd.setCursor(0,0);//Place le curseur en haut à gauche
lcd.noBlink();//Empêche le clignotement du curseur

lcd.print("     En ATTENTE") ; // Affiche la chaîne texte
lcd.setCursor(0,2);//Place le curseur colonne 0, ligne 3
lcd.print("    APPUI TOUCHE") ; // Affiche la chaîne texte
lcd.noBlink();//Empêche le clignotement du curseur

// les broches de lignes et d'entrée sont configurées
//automatiquement lors de l'initialisation du clavier
Serial.begin (9600);

} // fin de la fonction setup()
```

```
// ********************************************************************************
// ********************************************************************************
// ********************************************************************************
// ********** FONCTION LOOP = Boucle sans fin = coeur du programme ***********
// la fonction loop() s'exécute sans fin en boucle aussi longtemps que l'Arduino est sous tension

void loop(){ // Début de la fonction loop()
// Détermination de la combinaison des boutons activés,
//ArrêtUrg ou ValidTrame avec priorité au bouton ArrêtUrg
//Trame Arrêt Urgence

while (digitalRead(ArretUrg) == HIGH )// Tant que ArretUrg est High = Trame Urgence
//Envoyée Centrale Inopérente pour le reste
{
PORTK |=  (1<<5);//Equivalent à digitalWrite(Ledrouge,HIGH)
PORTA &=~ (1<<1);//Equivalent à digitalWrite(Ledbleue,LOW)
PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
PORTK &=~ (1<<4);//Equivalent à digitalWrite(Ledjaune,LOW)

TrameArretUrg();//Envoie Trame Arrêt Urgence
}

PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)

////////////////////////////////////////////////////////////
if (retourdetrame == 1)//Si retourdetrame est à envoi la RAZ variables
{envoi_raz_retourdetrame ();}//Test recréé ici suite à implémentation trame Accessoire
////////////////////////////////////////////////////////////

touche = keypad.getKey(); // Lecture de la touche appuyée

if (touche != NO_KEY)// Si une touche a été frappée -- gestion de la touche appuyée
      delay(15); // Pause entre 2 appuis : anti-rebond

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
//////////////////Menu multiple/////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (touche == '=')// || retourdetrame == 1)  //RAZ variables remis en tête de LOOP
 {
```

```
    envoi_raz();//Envoi raz

    autorisation_interrupteur = 0;//Interdit le fonctionnement des interrupteurs des fonctions

    locSelectionne = 0;//Pour éviter un Bug d'affichage
    loc2Selectionne = 0;//Pour éviter un Bug d'affichage

    lcd.clear(); // Efface écran si appui
    lcd.setCursor(8,0);//Place le curseur colonne 1, ligne 1
    lcd.print("MENU") ; // Affiche la chaîne texte

    lcd.setCursor(3,1);//Place le curseur colonne 1, ligne 1
    lcd.print("PROGRAMMATION") ; // Affiche la chaîne texte

    lcd.setCursor(9,2);//Place le curseur colonne 1, ligne 1
    lcd.print("et") ; // Affiche la chaîne texte

    lcd.setCursor(3,3);//Place le curseur colonne 1, ligne 1
    lcd.print("REMISE a ZERO") ; // Affiche la chaîne texte

    for(int compteurtempo = 0; compteurtempo < dureetempomenuprogrammation; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    lcd.clear();

    lcd.setCursor(0,0);//Place le curseur colonne 8, ligne 3
    lcd.print("Prog: CV=5, ACCESS=8") ; // Affiche la chaîne texte

    lcd.setCursor(0,1);//Place le curseur colonne 2, ligne 2
    lcd.print("Lect EEPROM = 3") ; // Affiche la chaîne texte

    lcd.setCursor(0,2);//Place le curseur colonne 2, ligne 2
    lcd.print("ENR EEPROM = 0") ; // Affiche la chaîne texte

    lcd.setCursor(0,3);//Place le curseur colonne 0, ligne 4
    lcd.print("RaZ EEPROM = 7") ; // Affiche la chaîne texte
```

```
  lcd.blink();//Autorise le clignotement du curseur

  autorise_touche_choix = 1;//Permet l'entrée des chiffres 5, 8, 3, 7, 0 uniquement

}//Fin du if (touche == '=')// || retourdetrame == 1)  //RAZ variables
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////Contrôle l'appui sur les touches 1 ou 4/////////////////
///////////Si autorise touche choix est à 1/////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if  (( touche == '5' || touche == '8' || touche == '0' || touche =='3'|| touche =='7') && autorise_touche_choix == 1)
//Traitement touche autorise choix ici. Pas au niveau des touches numériques
//if  (( touche == '2'|| touche == '5' || touche == '8' || touche == '0' || touche =='3'|| touche =='7') && autorise_touche_choix == 1)
//Traitement touche autorise choix ici. Pas au niveau des touches numériques
{

      if (touche == '5')
      {
       autorise_touche_choix = 0;//Interdit l'entrée des chiffres, 5, 8, 0, 3, 7 uniquement
       touche = ' ';//Empêche la saisie du nombre comme premier chiffre

       locSelectionne = 0;
       //Interdit la rentrée dans la boucle locomotive
       loc2Selectionne = 0;
       //Interdit la rentrée dans la boucle locomotive

       Cv();//Envoi menu programmation cv
       }//fermeture else if (touche == '5')

       else if (touche == '8')
      {
       autorise_touche_choix = 0;//Interdit l'entrée des chiffres, 5, 8, 0, 3, 7 uniquement
       touche = ' ';//Empêche la saisie du nombre comme premier chiffre

       locSelectionne = Miseenmemoirelocselectionne;
       //Permet de retrouver la locomotive sélectionnée après appui sur la touche "D" de fonction
       loc2Selectionne = Miseenmemoireloc2selectionne;
       //Car lors de l'appui de cette touche les deux variables loc (2)Selectionne sont remises à zéros
```

```
 programmation_accessoire_depart ();//Envoi trame accessoire
 }//fermeture else if (touche == '8')

 else if (touche == '0')
{
 autorise_touche_choix = 0;//Interdit l'entrée des chiffres, 5, 8, 0, 3, 7 uniquement
 touche = ' ';//Empêche la saisie du nombre comme premier chiffre

 locSelectionne = 0;
 //Interdit la rentrée dans la boucle locomotive
 loc2Selectionne = 0;
 //Interdit la rentrée dans la boucle locomotive

 memorisation_numero_loc ();//Envoi memorisation_numero_loc
 }//fermeture else if (touche == '0')

 else if (touche == '3')
{
 autorise_touche_choix = 0;//Interdit l'entrée des chiffres, 5, 8, 0, 3, 7 uniquement
 touche = ' ';//Empêche la saisie du nombre comme premier chiffre

 locSelectionne = Miseenmemoirelocselectionne;
 //Permet de retrouver la locomotive sélectionnée après appui sur la touche "D" de fonction
 loc2Selectionne = Miseenmemoireloc2selectionne;
 //Car lors de l'appui de cette touche les deux variables loc (2)Selectionne sont remises à zéros

 lecture_entiere_memoire_eeprom ();//Envoi memorisation_numero_loc

 }//fermeture else if (touche == '3')


 else if (touche == '7')
{
 autorise_touche_choix = 0;//Interdit l'entrée des chiffres, 5, 8, 0, 3, 7 uniquement
 touche = ' ';//Empêche la saisie du nombre comme premier chiffre

 locSelectionne = 0;
 //Interdit la rentrée dans la boucle locomotive
 loc2Selectionne = 0;
 //Interdit la rentrée dans la boucle locomotive
```

```
        lcd.clear();
        lcd.setCursor (2,0);
        lcd.print ("EFFACEMENT EEPROM");
        lcd.setCursor (5,1);
        lcd.print ("CONFIRMER");
        lcd.setCursor (6,2);
        lcd.print ("Touche 5");
        lcd.setCursor (9,3);

        confirme_raz = 1;
       }//fermeture else if (touche == '7')
      }//Fermeture(( touche == '2'|| touche == '5' || touche == '8' || touche == '0' || touche =='3'|| touche =='7')

    if  (( touche == '5') && confirme_raz == 1)
       //Traitement touche autrise choix ici. Pas au niveau des touches numériques

        {
         confirme_raz = 0;

         raz_eeprom();
       }//(( touche == '5') && confirme_raz == 1))


////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
if ((touche == 'A' && prog_memo_loc == 0)) //Locomotive et si programmation adresse loc pas en cours
{
   envoi_raz();

   autorise_touche_locomotive = 1;//Autorise l'entrée et l'affichage de nombres dans la boucle locomotive

   autorisation_interrupteur = 0;//Interdit le fonctionnement des interrupteurs des fonctions

   //Autorise le traitement appui touche locomotive
   traitement_appui_touches_locomotive = 1;

   lcd.clear(); // Efface écran si appui = sinon  affiche touche
   lcd.setCursor(0,0);//Place le curseur colonne 0, ligne 1

   lcd.print("N Locomotive : ?"); // Affiche la chaîne texte
```

```
    lcd.setCursor(0,2);//Place le curseur colonne 0, ligne 3
    lcd.print ("Choix MPJ");

    lcd.setCursor(0,3);//Place le curseur colonne 0, ligne 4
    lcd.print ("Choix VITESSE");

    lcd.setCursor(8,1);//Place le curseur colonne 8, ligne 2

    lcd.noBlink();//Empêche le clignotement du curseur


} // fin du if (touche == 'A')

if (touche == 'B' && prog_memo_loc == 0) //Locomotive2 et si programmation adresse loc pas en cours
{
    envoi_raz();

    autorise_touche_locomotive2 = 1;//Autorise l'entrée et l'affichage de nombres dans la boucle locomotive

    autorisation_interrupteur = 0;//Interdit le fonctionnement des interrupteurs des fonctions

    //Autorise le traitement appui touche locomotive
    traitement_appui_touches_locomotive2 = 1;

    lcd.clear(); // Efface écran si appui = sinon  affiche touche
    lcd.setCursor(0,0);//Place le curseur colonne 1, ligne 1

    lcd.print("N Locomotive2 : ?"); // Affiche la chaîne texte

    lcd.setCursor(0,2);//Place le curseur colonne 0, ligne 3
    lcd.print ("Choix MPJ");

    lcd.setCursor(0,3);//Place le curseur colonne 0, ligne 4
    lcd.print ("Choix VITESSE");

    lcd.setCursor(8,1);//Place le curseur colonne 8, ligne 2

    lcd.noBlink();//Empêche le clignotement du curseur
```

```
} // fin du if (touche == 'B')



if (touche == 'C' && prog_memo_loc == 0)  //FONCTION et si programmation adresse loc pas en cours

{


  if ((locSelectionne == 0) && (loc2Selectionne == 0))//Si aucune locomotives ne sont sélectionnées

{

   lcd.clear(); // Efface écran si appui


   lcd.setCursor(0,0);//Place le curseur colonne 0, ligne 2

   lcd.print("Aucunes locomotives") ; // Affiche la chaîne texte


   lcd.setCursor(4,1);//Place le curseur colonne 0, ligne 2

   lcd.print("selectionnees") ; // Affiche la chaîne texte


   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran


   lcd.noBlink();//Empêche le clignotement du curseur


   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)

   //Permet une temporisation


   {

    trame_idle();//Envoi trame idle

   }//fermeture du for


   retourdetrame = 1;//En attente appui touche


}//Fin de (nbre_locomotive2 == 0 && loc2Selectionne == 1


else

{

   envoi_raz();


   autorise_touche_fonction = 1;//Autorise l'entrée et l'affichage de nombres dans la boucle fonction


   autorisation_interrupteur = 0;//Interdit le fonctionnement des interrupteurs des fonctions


   traitement_appui_touches_fonction = 1;

   //Autorise la rentrée dans la boucle traitement_appui_touches_fonction quand appui sur 'E'
```

```
   lcd.clear(); // Efface écran si appui = sinon  affiche touche
   lcd.setCursor(7,0);//Place le curseur colonne 7, ligne 1
   lcd.print ("ENTRER");

   lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
   lcd.print ("Le NUMERO de");

   lcd.setCursor(6,2);//Place le curseur colonne 6, ligne 3
   lcd.print ("FONCTION");

   lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4

   lcd.blink();//Autorise le clignotement du curseur

}//Fin du else

}// fin du if (touche == 'C')



if ((touche == 'D') && (autorise_touche_accessoire44 == 0)  && (prog_memo_loc == 0))
//FONCTION et si programmation adresse loc pas en cours//ACCESSOIRE
{
   envoi_raz();

   autorise_touche_accessoire = 1;//Autorise la saisie de chiffre pour accessoire

   autorisation_interrupteur = 0;//Interdit le fonctionnement des interrupteurs des fonctions

   traitement_appui_touches_accessoire = 1;
   //Autorise la rentrée dans la boucle traitement_appui_touches_accessoire quand appui sur 'E'

   lcd.clear(); // Efface écran si appui = sinon  affiche touche
   lcd.setCursor(7,0);//Place le curseur colonne 7, ligne 1
   lcd.print ("ENTRER     C");

   lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
   lcd.print ("L'ADRESSE de");

   lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
```

```
   lcd.print ("L'ACCESSOIRE  E");

   lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4

   lcd.blink();//Autorise le clignotement du curseur

}//fin du if (touche == 'D')

//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
////////////Clavier universel//////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
if (touche == 'A' && autorise_touche_memorisation_loc == 1)
{
   lcd.print(touche);
{

if (autorise_touche_memorisation_loc == 1)
   {touche_appuyee_numerique_memorisation_loc ();}//Commande memorisation loc
}
}

if (touche == '1' && autorise_touche_A == 1)
{
   lcd.print(touche);
   traitement_appui_touche_memorisation_loc_eeprom ();//Commande memorisation loc
}



if (touche == '0' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
   lcd.print(touche);
```

```
{
   if (autorise_touche_locomotive == 1)
   {touche_appuyee_numerique_locomotive();}

   if (autorise_touche_locomotive2 == 1)
   {touche_appuyee_numerique_locomotive2();}

   if (autorise_touche_Cv == 1)
   {touche_appuyee_numerique_cv();}

   if (autorise_touche_Cv2 == 1)
   {touche_appuyee_numerique_cv2();}

   if (autorise_touche_Cv4 == 1)
   {touche_appuyee_numerique_cv4();}

   if (autorise_touche_fonction == 1)
   {touche_appuyee_numerique_fonction();}

   if (autorise_touche_programmation_accessoire == 1)
   {touche_appuyee_numerique_programmation_accessoire ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire2 == 1)
   {touche_appuyee_numerique_programmation_accessoire2 ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire4 == 1)
   {touche_appuyee_numerique_programmation_accessoire4 ();}
   //Commande programmation accessoire

   if (autorise_touche_memorisation_loc == 1)
   {touche_appuyee_numerique_memorisation_loc ();}//Commande memorisation loc

   if (autorise_touche_memorisation_adresse_loc == 1)
   {touche_appuyee_numerique_memorisation_adresse_loc();}//Commande memorisation loc

}  // fin des if 0
} // fin gestion de la touche appuyée

if (touche == '0' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
```

```
{
 lcd.print(touche);
 {
 if (autorise_touche_accessoire == 1)
  {touche_appuyee_numerique_accessoire();}

 if (autorise_touche_accessoire4 == 1)
  {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
}  // fin des if 0
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (touche == '1' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
  lcd.print(touche);
{
  if (autorise_touche_locomotive == 1)
  {touche_appuyee_numerique_locomotive();}

  if (autorise_touche_locomotive2 == 1)
  {touche_appuyee_numerique_locomotive2();}

  if (autorise_touche_Cv == 1)
  {touche_appuyee_numerique_cv();}

  if (autorise_touche_Cv2 == 1)
  {touche_appuyee_numerique_cv2();}

  if (autorise_touche_Cv4 == 1)
  {touche_appuyee_numerique_cv4();}

  if (autorise_touche_fonction == 1)
  {touche_appuyee_numerique_fonction();}
```

```
   if (autorise_touche_programmation_accessoire == 1)
   {touche_appuyee_numerique_programmation_accessoire ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire2 == 1)
   {touche_appuyee_numerique_programmation_accessoire2 ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire4 == 1)
   {touche_appuyee_numerique_programmation_accessoire4 ();}
   //Commande programmation accessoire

   if (autorise_touche_memorisation_loc == 1)
   {touche_appuyee_numerique_memorisation_loc ();}//Commande memorisation loc

   if (autorise_touche_memorisation_adresse_loc == 1)
   {touche_appuyee_numerique_memorisation_adresse_loc();}//Commande memorisation loc

} // fin des if
} // fin gestion de la touche appuyée

if (touche == '1' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
{
 lcd.print(touche);
 {
 if (autorise_touche_accessoire == 1)
  {touche_appuyee_numerique_accessoire();}

 if (autorise_touche_accessoire4 == 1)
  {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
} // fin des if 0
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (touche == '2' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
```

```
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
  lcd.print(touche);
{
  if (autorise_touche_locomotive == 1)
  {touche_appuyee_numerique_locomotive();}

  if (autorise_touche_locomotive2 == 1)
  {touche_appuyee_numerique_locomotive2();}

  if (autorise_touche_Cv == 1)
  {touche_appuyee_numerique_cv();}

  if (autorise_touche_Cv2 == 1)
  {touche_appuyee_numerique_cv2();}

  if (autorise_touche_Cv4 == 1)
  {touche_appuyee_numerique_cv4();}

  if (autorise_touche_fonction == 1)
  {touche_appuyee_numerique_fonction();}

  if (autorise_touche_programmation_accessoire == 1)
  {touche_appuyee_numerique_programmation_accessoire ();}
  //Commande programmation accessoire

  if (autorise_touche_programmation_accessoire2 == 1)
  {touche_appuyee_numerique_programmation_accessoire2 ();}
  //Commande programmation accessoire

  if (autorise_touche_programmation_accessoire4 == 1)
  {touche_appuyee_numerique_programmation_accessoire4 ();}
  //Commande programmation accessoire

  if (autorise_touche_memorisation_loc == 1)
  {touche_appuyee_numerique_memorisation_loc ();}//Commande memorisation loc

  if (autorise_touche_memorisation_adresse_loc == 1)
  {touche_appuyee_numerique_memorisation_adresse_loc();}//Commande memorisation loc
```

```
}  // fin du if 2
} // fin gestion de la touche appuyée

if (touche == '2' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
{
  lcd.print(touche);
  {
  if (autorise_touche_accessoire == 1)
    {touche_appuyee_numerique_accessoire();}

  if (autorise_touche_accessoire4 == 1)
    {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
}  // fin des if 0
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (touche == '3' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
   lcd.print(touche);
{
   if (autorise_touche_locomotive == 1)
   {touche_appuyee_numerique_locomotive();}

   if (autorise_touche_locomotive2 == 1)
   {touche_appuyee_numerique_locomotive2();}

   if (autorise_touche_Cv == 1)
   {touche_appuyee_numerique_cv();}

   if (autorise_touche_Cv2 == 1)
   {touche_appuyee_numerique_cv2();}
```

```
    if (autorise_touche_Cv4 == 1)
    {touche_appuyee_numerique_cv4();}

    if (autorise_touche_fonction == 1)
    {touche_appuyee_numerique_fonction();}

    if (autorise_touche_programmation_accessoire == 1)
    {touche_appuyee_numerique_programmation_accessoire ();}
    //Commande programmation accessoire

    if (autorise_touche_programmation_accessoire2 == 1)
    {touche_appuyee_numerique_programmation_accessoire2 ();}
    //Commande programmation accessoire

    if (autorise_touche_programmation_accessoire4 == 1)
    {touche_appuyee_numerique_programmation_accessoire4 ();}
    //Commande programmation accessoire

    if (autorise_touche_memorisation_loc == 1)
    {touche_appuyee_numerique_memorisation_loc ();}
    //Commande memorisation loc

    if (autorise_touche_memorisation_adresse_loc == 1)
    {touche_appuyee_numerique_memorisation_adresse_loc();}
    //Commande memorisation loc

}  // fin du if 3
} // fin gestion de la touche appuyée

if (touche == '3' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
{
  lcd.print(touche);
  {
  if (autorise_touche_accessoire == 1)
    {touche_appuyee_numerique_accessoire();}

  if (autorise_touche_accessoire4 == 1)
    {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
}  // fin des if 0
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////
if (touche == '4' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
   lcd.print(touche);
{
   if (autorise_touche_locomotive == 1)
   {touche_appuyee_numerique_locomotive();}

   if (autorise_touche_locomotive2 == 1)
   {touche_appuyee_numerique_locomotive2();}

   if (autorise_touche_Cv == 1)
   {touche_appuyee_numerique_cv();}

   if (autorise_touche_Cv2 == 1)
   {touche_appuyee_numerique_cv2();}

   if (autorise_touche_Cv4 == 1)
   {touche_appuyee_numerique_cv4();}

   if (autorise_touche_fonction == 1)
   {touche_appuyee_numerique_fonction();}

   if (autorise_touche_programmation_accessoire == 1)
   {touche_appuyee_numerique_programmation_accessoire ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire2 == 1)
   {touche_appuyee_numerique_programmation_accessoire2 ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire4 == 1)
   {touche_appuyee_numerique_programmation_accessoire4 ();}
```

```
   //Commande programmation accessoire

   if (autorise_touche_memorisation_loc == 1)
   {touche_appuyee_numerique_memorisation_loc ();}
   //Commande memorisation loc

   if (autorise_touche_memorisation_adresse_loc == 1)
   {touche_appuyee_numerique_memorisation_adresse_loc();}
   //Commande memorisation loc

} // fin du if 4
} // fin gestion de la touche appuyée

if (touche == '4' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
{
 lcd.print(touche);
 {
 if (autorise_touche_accessoire == 1)
  {touche_appuyee_numerique_accessoire();}

 if (autorise_touche_accessoire4 == 1)
  {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
} // fin des if 0
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (touche == '5' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
   lcd.print(touche);
{
   if (autorise_touche_locomotive == 1)
   {touche_appuyee_numerique_locomotive();}
```

```
    if (autorise_touche_locomotive2 == 1)
    {touche_appuyee_numerique_locomotive2();}

    if (autorise_touche_Cv == 1)
    {touche_appuyee_numerique_cv();}

    if (autorise_touche_Cv2 == 1)
    {touche_appuyee_numerique_cv2();}

    if (autorise_touche_Cv4 == 1)
    {touche_appuyee_numerique_cv4();}

    if (autorise_touche_fonction == 1)
    {touche_appuyee_numerique_fonction();}

    if (autorise_touche_programmation_accessoire == 1)
    {touche_appuyee_numerique_programmation_accessoire ();}
    //Commande programmation accessoire

    if (autorise_touche_programmation_accessoire2 == 1)
    {touche_appuyee_numerique_programmation_accessoire2 ();}
    //Commande programmation accessoire

    if (autorise_touche_programmation_accessoire4 == 1)
    {touche_appuyee_numerique_programmation_accessoire4 ();}
    //Commande programmation accessoire

    if (autorise_touche_memorisation_loc == 1)
    {touche_appuyee_numerique_memorisation_loc ();}
    //Commande memorisation loc

    if (autorise_touche_memorisation_adresse_loc == 1)
    {touche_appuyee_numerique_memorisation_adresse_loc();}
    //Commande memorisation loc

}  // fin du if 5
} // fin gestion de la touche appuyée

if (touche == '5' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
{
  lcd.print(touche);
```

```
 {
 if (autorise_touche_accessoire == 1)
  {touche_appuyee_numerique_accessoire();}

 if (autorise_touche_accessoire4 == 1)
  {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
} // fin des if 0
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (touche == '6' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
  lcd.print(touche);
{
  if (autorise_touche_locomotive == 1)
  {touche_appuyee_numerique_locomotive();}

  if (autorise_touche_locomotive2 == 1)
  {touche_appuyee_numerique_locomotive2();}

  if (autorise_touche_Cv == 1)
  {touche_appuyee_numerique_cv();}

  if (autorise_touche_Cv2 == 1)
  {touche_appuyee_numerique_cv2();}

  if (autorise_touche_Cv4 == 1)
  {touche_appuyee_numerique_cv4();}

  if (autorise_touche_fonction == 1)
  {touche_appuyee_numerique_fonction();}

  if (autorise_touche_programmation_accessoire == 1)
```

```
   {touche_appuyee_numerique_programmation_accessoire ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire2 == 1)
   {touche_appuyee_numerique_programmation_accessoire2 ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire4 == 1)
   {touche_appuyee_numerique_programmation_accessoire4 ();}
   //Commande programmation accessoire

   if (autorise_touche_memorisation_loc == 1)
   {touche_appuyee_numerique_memorisation_loc ();}
   //Commande memorisation loc

   if (autorise_touche_memorisation_adresse_loc == 1)
   {touche_appuyee_numerique_memorisation_adresse_loc();}
   //Commande memorisation loc

} // fin du if 6
} // fin gestion de la touche appuyée

if (touche == '6' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
{
 lcd.print(touche);
 {
 if (autorise_touche_accessoire == 1)
  {touche_appuyee_numerique_accessoire();}

 if (autorise_touche_accessoire4 == 1)
  {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
} // fin des if 0
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (touche == '7' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
```

```
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
  lcd.print(touche);
{
  if (autorise_touche_locomotive == 1)
  {touche_appuyee_numerique_locomotive();}

  if (autorise_touche_locomotive2 == 1)
  {touche_appuyee_numerique_locomotive2();}

  if (autorise_touche_Cv == 1)
  {touche_appuyee_numerique_cv();}

  if (autorise_touche_Cv2 == 1)
  {touche_appuyee_numerique_cv2();}

  if (autorise_touche_Cv4 == 1)
  {touche_appuyee_numerique_cv4();}

  if (autorise_touche_fonction == 1)
  {touche_appuyee_numerique_fonction();}

  if (autorise_touche_programmation_accessoire == 1)
  {touche_appuyee_numerique_programmation_accessoire ();}
  //Commande programmation accessoire

  if (autorise_touche_programmation_accessoire2 == 1)
  {touche_appuyee_numerique_programmation_accessoire2 ();}
  //Commande programmation accessoire

  if (autorise_touche_programmation_accessoire4 == 1)
  {touche_appuyee_numerique_programmation_accessoire4 ();}
  //Commande programmation accessoire

  if (autorise_touche_memorisation_loc == 1)
  {touche_appuyee_numerique_memorisation_loc ();}
  //Commande memorisation loc

  if (autorise_touche_memorisation_adresse_loc == 1)
```

```
   {touche_appuyee_numerique_memorisation_adresse_loc();}
   //Commande memorisation loc

}  // fin du if 7
} // fin gestion de la touche appuyée

if (touche == '7' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
{
  lcd.print(touche);
  {
  if (autorise_touche_accessoire == 1)
    {touche_appuyee_numerique_accessoire();}

  if (autorise_touche_accessoire4 == 1)
    {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
}  // fin des if 0
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (touche == '8' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
   lcd.print(touche);
{
   if (autorise_touche_locomotive == 1)
   {touche_appuyee_numerique_locomotive();}

   if (autorise_touche_locomotive2 == 1)
   {touche_appuyee_numerique_locomotive2();}

   if (autorise_touche_Cv == 1)
   {touche_appuyee_numerique_cv();}

   if (autorise_touche_Cv2 == 1)
```

```
    {touche_appuyee_numerique_cv2();}

    if (autorise_touche_Cv4 == 1)
    {touche_appuyee_numerique_cv4();}

    if (autorise_touche_fonction == 1)
    {touche_appuyee_numerique_fonction();}

    if (autorise_touche_programmation_accessoire == 1)
    {touche_appuyee_numerique_programmation_accessoire ();}
    //Commande programmation accessoire

    if (autorise_touche_programmation_accessoire2 == 1)
    {touche_appuyee_numerique_programmation_accessoire2 ();}
    //Commande programmation accessoire

    if (autorise_touche_programmation_accessoire4 == 1)
    {touche_appuyee_numerique_programmation_accessoire4 ();}
    //Commande programmation accessoire

    if (autorise_touche_memorisation_loc == 1)
    {touche_appuyee_numerique_memorisation_loc ();}
    //Commande memorisation loc

    if (autorise_touche_memorisation_adresse_loc == 1)
    {touche_appuyee_numerique_memorisation_adresse_loc();}
    //Commande memorisation loc

} // fin du if 8
} // fin gestion de la touche appuyée

if (touche == '8' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
{
  lcd.print(touche);
  {
  if (autorise_touche_accessoire == 1)
    {touche_appuyee_numerique_accessoire();}

  if (autorise_touche_accessoire4 == 1)
    {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
} // fin des if 0
```

```
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (touche == '9' &&
(autorise_touche_locomotive == 1 || autorise_touche_locomotive2 == 1
||autorise_touche_Cv == 1 ||autorise_touche_Cv2 == 1
||autorise_touche_Cv4 == 1 ||autorise_touche_fonction == 1
|| autorise_touche_programmation_accessoire == 1
|| autorise_touche_programmation_accessoire2 == 1
|| autorise_touche_programmation_accessoire4 == 1
|| autorise_touche_memorisation_loc == 1 || autorise_touche_memorisation_adresse_loc == 1))

{
  lcd.print(touche);
{
  if (autorise_touche_locomotive == 1)
  {touche_appuyee_numerique_locomotive();}

  if (autorise_touche_locomotive2 == 1)
  {touche_appuyee_numerique_locomotive2();}

  if (autorise_touche_Cv == 1)
  {touche_appuyee_numerique_cv();}

  if (autorise_touche_Cv2 == 1)
  {touche_appuyee_numerique_cv2();}

  if (autorise_touche_Cv4 == 1)
  {touche_appuyee_numerique_cv4();}

  if (autorise_touche_fonction == 1)
  {touche_appuyee_numerique_fonction();}

  if (autorise_touche_programmation_accessoire == 1)
  {touche_appuyee_numerique_programmation_accessoire ();}
  //Commande programmation accessoire

  if (autorise_touche_programmation_accessoire2 == 1)
  {touche_appuyee_numerique_programmation_accessoire2 ();}
  //Commande programmation accessoire
```

```
   if (autorise_touche_programmation_accessoire4 == 1)
   {touche_appuyee_numerique_programmation_accessoire4 ();}
   //Commande programmation accessoire

   if (autorise_touche_memorisation_loc == 1)
   {touche_appuyee_numerique_memorisation_loc ();}
   //Commande memorisation loc

   if (autorise_touche_memorisation_adresse_loc == 1)
   {touche_appuyee_numerique_memorisation_adresse_loc();}
   //Commande memorisation loc

}  // fin du if 9
} // fin gestion de la touche appuyée

if (touche == '9' && (autorise_touche_accessoire == 1 || autorise_touche_accessoire4 == 1))
{
  lcd.print(touche);
  {
  if (autorise_touche_accessoire == 1)
    {touche_appuyee_numerique_accessoire();}

  if (autorise_touche_accessoire4 == 1)
    {touche_appuyee_numerique_accessoire4 ();}//Commande accessoire
}  // fin des if 0
} // fin gestion de la touche appuyée
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
///////Prise en compte pour validation du chiffre saisi///////
////////// du chiffre saisi de la fonction voulue//////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
 if (touche == 'E' && (autorise_touche_locomotive_e == 1 || autorise_touche_locomotive2_e == 1
 || autorise_touche_Cv_e == 1 || autorise_touche_Cv2_e == 1 || autorise_touche_Cv4_e == 1
 || autorise_touche_fonction_e == 1
 || autorise_touche_programmation_accessoire_e == 1
 || autorise_touche_programmation_accessoire2_e == 1
 || autorise_touche_programmation_accessoire4_e == 1
 || autorise_touche_memorisation_loc_e == 1
 || autorise_touche_memorisation_adresse_loc_e == 1))
```

```
{
   if (autorise_touche_locomotive_e == 1)
   {traitement_appui_touche_locomotive();}

   if (autorise_touche_locomotive2_e == 1)
   {traitement_appui_touche_locomotive2();}

   if (autorise_touche_Cv_e == 1)
   {traitement_appui_touche_cv();}

   if (autorise_touche_Cv2_e == 1)
   {traitement_appui_touche_cv2();}

   if (autorise_touche_Cv4_e == 1)
   {traitement_appui_touche_cv4();}

   if (autorise_touche_fonction_e == 1)
   {traitement_appui_touche_fonction();}

   if (autorise_touche_programmation_accessoire_e == 1)
   {traitement_appui_touche_programmation_accessoire ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire2_e == 1)
   {traitement_appui_touche_programmation_accessoire2 ();}
   //Commande programmation accessoire

   if (autorise_touche_programmation_accessoire4_e == 1)
   {traitement_appui_touche_programmation_accessoire4 ();}
   //Commande programmation accessoire

   if (autorise_touche_memorisation_loc_e == 1)
   {traitement_appui_touche_memorisation_loc ();}
   //Commande memorisation loc

   if (autorise_touche_memorisation_adresse_loc_e == 1)
   {traitement_appui_touche_memorisation_memorisation_loc ();}
   //Commande memorisation loc


} //Fin du If
```

```
 if (touche == 'E' && autorise_touche_accessoire_e == 1)
    {traitement_appui_touche_accessoire();}//Commande accessoire

 if (touche == 'D' && autorise_touche_accessoire4_e == 1)
   {traitement_appui_touche_accessoire4();}//Commande accessoire



////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
//Appel fonctions
FonctionFL = digitalRead(Fonction0);
FonctionF1 = digitalRead(Fonction1);
FonctionF2 = digitalRead(Fonction2);
FonctionF3 = digitalRead(Fonction3);
FonctionF4 = digitalRead(Fonction4);

//Appel fonctionFL

if (FonctionFL == HIGH && boucleTrameFL1 == 0 && autorisation_interrupteur == 1)
{delay (15);
boucleTrameFL1 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles
boucleTrameFL0 = 0;//Permet la rentrée dans la boucle. Croisement des boucles

octetFonctionFL = 1;//Active la fonction

envoi_trame_fonction();// Appel trame_fonction
}//Fin FonctionFL

if (FonctionFL == LOW && boucleTrameFL0 == 0 && autorisation_interrupteur == 1)
{delay (15);
boucleTrameFL1 = 0;//Permet la rentrée dans la boucle. Croisement des boucles
boucleTrameFL0 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles

octetFonctionFL = 0;//Désactive la fonction

envoi_trame_fonction();// Appel trame_fonction
```

```
}//Fin FonctionFL
//////////////////////////////////////////
//////////////////////////////////////////
//Appel fonctionF1

if (FonctionF1 == HIGH && boucleTrameF11 == 0 && autorisation_interrupteur == 1)
{delay (15);
boucleTrameF11 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles
boucleTrameF10 = 0;//Permet la rentrée dans la boucle. Croisement des boucles

octetFonctionF1 = 1;//Active la fonction

envoi_trame_fonction();// Appel trame_fonction
}//Fin FonctionF1

if (FonctionF1 == LOW && boucleTrameF10 == 0 && autorisation_interrupteur == 1)
{delay (15);
boucleTrameF11 = 0;//Permet la rentrée dans la boucle. Croisement des boucles
boucleTrameF10 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles

octetFonctionF1 = 0;//Désactive la fonction

envoi_trame_fonction();// Appel trame_fonction

}//Fin FonctionF1
//////////////////////////////////////////
//////////////////////////////////////////
//Appel fonctionF2

if (FonctionF2 == HIGH && boucleTrameF21 == 0 && autorisation_interrupteur == 1)
{delay (15);
boucleTrameF21 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles
boucleTrameF20 = 0;//Permet la rentrée dans la boucle. Croisement des boucles

octetFonctionF2 = 1;//Active la fonction

envoi_trame_fonction();// Appel trame_fonction
}//Fin FonctionF1

if (FonctionF2 == LOW && boucleTrameF20 == 0 && autorisation_interrupteur == 1)
{delay (15);
```

```
boucleTrameF21 = 0;//Permet la rentrée dans la boucle. Croisement des boucles
boucleTrameF20 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles

octetFonctionF2 = 0;//Désactive la fonction

envoi_trame_fonction();// Appel trame_fonction

}//Fin FonctionF2
//////////////////////////////////////////
//////////////////////////////////////////
//Appel fonctionF3

if (FonctionF3 == HIGH && boucleTrameF31 == 0 && autorisation_interrupteur == 1)
{delay (15);
boucleTrameF31 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles
boucleTrameF30 = 0;//Permet la rentrée dans la boucle. Croisement des boucles

octetFonctionF3 = 1;//Active la fonction

envoi_trame_fonction();// Appel trame_fonction
}//Fin FonctionF1

if (FonctionF3 == LOW && boucleTrameF30 == 0 && autorisation_interrupteur == 1)
{delay (15);
boucleTrameF31 = 0;//Permet la rentrée dans la boucle. Croisement des boucles
boucleTrameF30 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles

octetFonctionF3 = 0;//Désactive la fonction

envoi_trame_fonction();// Appel trame_fonction

}//Fin FonctionF3
//////////////////////////////////////////
//////////////////////////////////////////
//Appel fonctionF4

if (FonctionF4 == HIGH && boucleTrameF41 == 0 && autorisation_interrupteur == 1)
{delay (15);
boucleTrameF41 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles
boucleTrameF40 = 0;//Permet la rentrée dans la boucle. Croisement des boucles
```

```
octetFonctionF4 = 1;//Active la fonction

envoi_trame_fonction();// Appel trame_fonction
}//Fin FonctionF1

if (FonctionF4 == LOW && boucleTrameF40 == 0 && autorisation_interrupteur == 1)
{delay (15);
boucleTrameF41 = 0;//Permet la rentrée dans la boucle. Croisement des boucles
boucleTrameF40 = 1;//Empêche la rentrée dans la boucle en permanence. Croisement des boucles

octetFonctionF4 = 0;//Désactive la fonction

envoi_trame_fonction();// Appel trame_fonction
}//Fin FonctionF4

////////////////////////////////////////////////
///////////////Sens de Marche//////////////////
////////////Commande Locomotive 1/////////////////
////////////////////////////////////////////////
//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 && locSelectionne == 1 )
{
comm7=1;//Sens marche avant loc 1
PORTK |= (1<<7);//Equivalent à digitalWrite(LedAvant,HIGH)
PORTK &=~ (1<<6);//Equivalent à digitalWrite(LedArrière,LOW)

PORTK &=~ (1<<2);//Equivalent à digitalWrite(LedAvant2,LOW)
PORTK &=~ (1<<3);//Equivalent à digitalWrite(LedArrière2,LOW)
}

if (VarComm5 == LOW && locSelectionne == 1 )
{
comm7=0;//Sens marche arrière loc 1
PORTK &=~ (1<<7);//Equivalent à digitalWrite(LedAvant,LOW)
PORTK |= (1<<6);//Equivalent à digitalWrite(LedArrière,HIGH)
```

```
PORTK &=~ (1<<2);//Equivalent à digitalWrite(LedAvant2,LOW)
PORTK &=~ (1<<3);//Equivalent à digitalWrite(LedArrière2,LOW)
}

////////////////////////////////////////////////
///////////////Sens de Marche//////////////////
////////////Commande Locomotive 2/////////////////
////////////////////////////////////////////////
//Détermine le sens de marche
//valeur bit commande 5 sens de marche
VarComm52 = digitalRead(SensdeMarche2);
//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 (VarComm52 == HIGH && loc2Selectionne == 1)
{
comm27loc=1;//Sens marche avant loc 2
PORTK |= (1<<2);//Equivalent à digitalWrite(LedAvant2,HIGH)
PORTK &=~ (1<<3);//Equivalent à digitalWrite(LedArrière2,LOW)

PORTK &=~ (1<<7);//Equivalent à digitalWrite(LedAvant,LOW)
PORTK &=~ (1<<6);//Equivalent à digitalWrite(LedArrière,LOW)
}

if (VarComm52 == LOW && loc2Selectionne == 1)
{
comm27loc=0;//Sens marche arrière loc 2

PORTK &=~ (1<<2);//Equivalent à digitalWrite(LedAvant2,LOW)
PORTK |= (1<<3);//Equivalent à digitalWrite(LedArrière2,HIGH)

PORTK &=~ (1<<7);//Equivalent à digitalWrite(LedAvant,LOW)
PORTK &=~ (1<<6);//Equivalent à digitalWrite(LedArrière,LOW)
}

////////////////////////////////////////////////
///////Prend la main sur la locomotive 1/////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (touche == '7' && autorise_touche_7_et_9 == 1)
{
```

```
 lcd.clear(); // Efface écran si appui

if (nbre_locomotive == 0)//Interdit l'échange de loc si loc non sélectionnée
{
   lcd.setCursor(1,0);//Place le curseur colonne 0, ligne 2
   lcd.print("Aucune locomotive") ; // Affiche la chaîne texte

   lcd.setCursor(4,1);//Place le curseur colonne 0, ligne 2
   lcd.print("selectionnee") ; // Affiche la chaîne texte

   locSelectionne = 0;//Interdit l'envoi de la trame locomotive
   loc2Selectionne = 1;//Permet l'envoi de la trame locomotive2
   //Pour retrouver la loc sélectionnée au cas ou la loc2
   //n'aurai pas été sélectionnée

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Permet une temporisation

   {
    trame_idle();//Envoi trame idle
   }//fermeture du for

   lcd.clear();//Pour éviter un bug d'affichage

}//Fin de (nbre_locomotive > 0

else
{
locSelectionne = 1;//Permet l'envoi de la trame locomotive
loc2Selectionne = 0;//Empêche l'envoi de la trame locomotive2

Miseenmemoirelocselectionne = locSelectionne;//Mise en memoire de la loc sélectionnée :
 //Permet de retrouver la loc sélectionnée auparavant
 //pour retrouver l'affichage et la commande de la loc sélectionnée après
 //l'utilisation de la commande fonction

Miseenmemoireloc2selectionne = loc2Selectionne;//Mise en memoire de la loc sélectionnée :
```

```
 //Permet de retrouver la loc sélectionnée auparavant
 //pour retrouver l'affichage et la commande de la loc sélectionnée après
 //l'utilisation de la commande fonction

}//Fin du else

}//Fin du if (nbre_locomotive == 0)

////////////////////////////////////////////////
///////Prend la main sur la locomotive 2/////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (touche == '9' && autorise_touche_7_et_9 == 1)
{
lcd.clear();
if (nbre_locomotive2 == 0)//Interdit l'échange de loc si loc2 non sélectionnée
{

   lcd.setCursor(1,0);//Place le curseur colonne 0, ligne 2
   lcd.print("Aucune locomotive") ; // Affiche la chaîne texte

   lcd.setCursor(4,1);//Place le curseur colonne 0, ligne 2
   lcd.print("selectionnee") ; // Affiche la chaîne texte

   locSelectionne = 1;//Permet l'envoi de la trame locomotive2
   //Pour retrouver la loc sélectionnée au cas ou la loc2
   //n'aurai pas été sélectionnée
   loc2Selectionne = 0;//Interdit l'envoi de la trame locomotive2

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Permet une temporisation

   {
    trame_idle();//Envoi trame idle
   }//fermeture du for

   lcd.clear();//Pour éviter un bug d'affichage
```

```
}//Fin du if (nbre_locomotive2 == 0)

else
{
locSelectionne = 0;//Empêche l'envoi de la trame locomotive
loc2Selectionne = 1;//Permet l'envoi de la trame locomotive2

Miseenmemoirelocselectionne = locSelectionne;//Mise en memoire de la loc sélectionnée :
 //Permet de retrouver la loc sélectionnée auparavant
 //pour retrouver l'affichage et la commande de la loc sélectionnée après
 //l'utilisation de la commande fonction

Miseenmemoireloc2selectionne = loc2Selectionne;//Mise en memoire de la loc sélectionnée :
 //Permet de retrouver la loc sélectionnée auparavant
 //pour retrouver l'affichage et la commande de la loc sélectionnée après
 //l'utilisation de la commande fonction

}//Fin du else
}//Fin du if (touche == '9' && autorise_touche_7_et_9 == 1)

//////////////////////////////////////////////////
//////////////////////Fin//////////////////////////
///////Prend la main sur la locomotive 2/////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
////////////Mode Normal ou Manoeuvre/////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
if (touche == '1' && autorise_touche_1_et_4 == 1)
{
if (locSelectionne == 1)
{
mode_manoeuvre = 0;
}

if (loc2Selectionne == 1)
mode2_manoeuvre = 0;
```

```
}

if (touche == '4' && autorise_touche_1_et_4 == 1)
{
if (locSelectionne == 1)
{
mode_manoeuvre = 1;
}

if (loc2Selectionne == 1)
mode2_manoeuvre = 1;

}

////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
//////////////////Envoi trame//////////////////////
//////////////////En fonction ///////////////////
//////////////de la loc sélectionnée///////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (locSelectionne == 1 && loc2Selectionne == 0)
//Test double pour empêcher la rentrée dans la boucle lors
//d'une mauvaise sélecton du numéro de locomotive
{
//locSelectionne = 1;//Permet l'envoi de la trame locomotive
//loc2Selectionne = 0;//Empêche l'envoi de la trame locomotive2

Miseenmemoirelocselectionne = locSelectionne;//Mise en memoire de la loc sélectionnée :
 //Permet de retrouver la loc sélectionnée auparavant
 //pour retrouver l'affichage et la commande de la loc sélectionnée après
 //l'utilisation de la commande fonction

Miseenmemoireloc2selectionne = loc2Selectionne;//Mise en memoire de la loc sélectionnée :
 //Permet de retrouver la loc sélectionnée auparavant
 //pour retrouver l'affichage et la commande de la loc sélectionnée après
 //l'utilisation de la commande fonction

PORTA &=~ (1<<1);//Equivalent à digitalWrite(Ledbleue,LOW)
PORTA |= (1<<3);//Equivalent à digitalWrite(Ledverte,HIGH)
```

```
PORTK &=~ (1<<4);//Equivalent à digitalWrite(Ledjaune,LOW)

trame_locomotive();//Envoi trame_locomotive
}//Fin du if (locSelectionne == 1 && loc2Selectionne == 0)
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (locSelectionne == 0 && loc2Selectionne == 1)
//Test double pour empêcher la rentrée dans la boucle lors
//d'une mauvaise sélecton du numéro de locomotive
{
//locSelectionne = 0;//Empêche l'envoi de la trame locomotive
//loc2Selectionne = 1;//Permet l'envoi de la trame locomotive2

Miseenmemoirelocselectionne = locSelectionne;//Mise en memoire de la loc sélectionnée :
 //Permet de retrouver la loc sélectionnée auparavant
 //pour retrouver l'affichage et la commande de la loc sélectionnée après
 //l'utilisation de la commande fonction qui met locSelectionne et
 //loc2Selectionne à 0 pour eviter une erreur d'afffichage.

Miseenmemoireloc2selectionne = loc2Selectionne;//Mise en memoire de la loc sélectionnée :
 //Permet de retrouver la loc sélectionnée auparavant
 //pour retrouver l'affichage et la commande de la loc sélectionnée après
 //l'utilisation de la commande fonction qui met locSelectionne et
 //loc2Selectionne à 0 pour eviter une erreur d'afffichage.

//Leds trames
PORTA &=~ (1<<1);//Equivalent à digitalWrite(Ledbleue,LOW)
PORTA |= (1<<3);//Equivalent à digitalWrite(Ledverte,HIGH)
PORTK &=~ (1<<4);//Equivalent à digitalWrite(Ledjaune,LOW)

trame_locomotive2();//Envoi trame_locomotive2
}//Fin du if (locSelectionne == 0 && loc2Selectionne == 1)

////////////////////////////////////////////////
////////////////////Fin/////////////////////////
//////////////////Envoi trame///////////////////
//////////////////En fonction ///////////////////
/////////////de la loc sélectionnée//////////////
////////////////////////////////////////////////
```

```
////////////////////////////////////////////////
////////////////////////////////////////////////
//////////////////////Si//////////////////////////
///////////////Aucune locomotive//////////////////
////////////////Sélectionnée ////////////////////
/////////////Envoi trame idle////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (locSelectionne == 0 && loc2Selectionne == 0)
{
//Leds trames
PORTA &=~ (1<<1);//Equivalent à digitalWrite(Ledbleue,LOW)
PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
PORTK |=  (1<<4);//Equivalent à digitalWrite(Ledjaune,HIGH)

trame_idle();//Envoi trame_idle

}//Fin du if (locSelectionne == 0 && loc2Selectionne == 0)

////////////////////////////////////////////////
////////////////////////////////////////////////
//////////////////////Si//////////////////////////
///////////////Aucune locomotive//////////////////
////////////////Sélectionnée ////////////////////
/////////////Envoi trame idle////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////

}//fin du loop


//////////////////////////// ////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////AUTRES FONCTIONS DU PROGRAMME ////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
/////////////// MISE EN MEMOIRE DES TOUCHES APPUYEES LOCOMOTIVE//////////////////
//////////////////////TOUCHE APPUYEE NUMERIQUE LOCOMOTIVE//////////////////////////
////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_locomotive() // // lecture de la touche appuyée
{
```

```
 compteur_touche_locomotive = compteur_touche_locomotive + 1;
 //Incrémente la variable compteur_touche

if (compteur_touche_locomotive == 1) //Teste si compteur_touche = 1
{ toucheA_locomotive = touche;//Garde trace de la touche 1ere touche appuyée
lcd.blink();

autorise_touche_locomotive_e = 1;//Autorise la prise en compte par la touche E

}//Autorise le clignotement du curseur

if (compteur_touche_locomotive == 2) //Teste si compteur_touche = 2
{toucheB_locomotive = touche;//Garde trace de la touche 2ième touche appuyée
//lcd.blink();
}//Autorise le clignotement du curseur

if (compteur_touche_locomotive == 3) //Teste si compteur_touche = 3
{toucheC_locomotive = touche;
//lcd.blink();
}//Garde trace de la touche 3ième touche appuyée

if (compteur_touche_locomotive == 4) //Teste si compteur_touche = 4
{toucheD_locomotive = touche;
//lcd.blink();
}//Garde trace de la touche 4ième touche appuyée

if (compteur_touche_locomotive == 5) //Teste si compteur_touche = 5
{toucheE_locomotive = touche;
//lcd.blink();
}//Garde trace de la touche 5ième touche appuyée

if (compteur_touche_locomotive == 6) //Teste si compteur_touche = 6
{toucheF_locomotive = touche;
lcd.noBlink();

autorise_touche_locomotive = 0;//Empêche la saisie de chiffre supplémentaire

}//Garde trace de la touche 6ième touche appuyée

}//Fin du VOID MISE EN MEMOIRE DES TOUCHES APPUYEES LOCOMOTIVE
```

```
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
///////////////// TRAITEMENT DES TOUCHES APPUYEES LOCOMOTIVE//////////
//////////////////////TRAITEMENT APPUI TOUCHE LOCOMOTIVE///////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
void traitement_appui_touche_locomotive()
//Correspond à Appui touche 'E' et traitement_appui_touche_locomotive = 1
{

if (compteur_touche_locomotive == 1)//Prise en compte du chiffre saisi pour une entrée
{
  nbre_locomotive = toucheA_locomotive-48;//toucheA contient l'intégralité de la saisie

  if (nbre_locomotive == 0)//Refus du chiffre 0
{
    lcd.clear(); // Efface écran si appui
    lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

    lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

    lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

    lcd.print("doit etre compris") ; // Affiche la chaîne texte

    lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

    lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    locSelectionne = 0;//DéSélectionne la loc 1 et permet l'envoi de la trame_locomotive 1
    loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2


    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
```

```
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0

else
 {
 envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

 locSelectionne = 1;//Sélectionne la loc 1 et permet l'envoi de la trame_locomotive 1
 loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2

 compteur_touche_locomotive = 0;//Réinitialise compteur_touche


 autorise_touche_7_et_9 = 1;
 //Autorise la prise en compte de la touche 7 ou 9 pour changer la locomotive sous contrôle

 autorise_touche_1_et_4 = 1;

  lcd.clear(); // Efface écran

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

  adresse_locomotive = nbre_locomotive;//recopie dans adresse la valeur de nbre en binaire

   //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   adr7 = 0;//Adresse courte
   adr6 = bitRead (adresse_locomotive, 6);//Met dans adr6, le bit 6 de la variable adresse
   adr5 = bitRead (adresse_locomotive, 5);
   adr4 = bitRead (adresse_locomotive, 4);
   adr3 = bitRead (adresse_locomotive, 3);
   adr2 = bitRead (adresse_locomotive, 2);
   adr1 = bitRead (adresse_locomotive, 1);
   adr0 = bitRead (adresse_locomotive, 0);
```

```
  }//Fin du else
}// fin du if (compteur_touche_locomotive == 1)
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
if (compteur_touche_locomotive == 2)//Prise en compte du chiffre saisi pour deux entrées
{

   nbre_locomotive = ((toucheA_locomotive-48) * 10) + (toucheB_locomotive-48);
   //nbre contient l'intégralité de la saisie

 if (nbre_locomotive == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2


   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for
```

```
    retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0
 else
 {
     envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    locSelectionne = 1;//Sélectionne la loc 1 et permet l'envoi de la trame_locomotive 1
    loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2

    compteur_touche_locomotive = 0;//Réinitialise compteur_touche

    autorise_touche_7_et_9 = 1;
    //Autorise la prise en compte de la touche 7 ou 9 pour changer la locomotive sous contrôle

    autorise_touche_1_et_4 = 1;

   lcd.clear(); // Efface écran

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

    adresse_locomotive = nbre_locomotive;
    //recopie dans adresse la valeur de nbre en binaire

    //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7 = 0;//Adresse courte
    adr6 = bitRead (adresse_locomotive, 6);//Met dans adr6, le bit 6 de la variable adresse
    adr5 = bitRead (adresse_locomotive, 5);
    adr4 = bitRead (adresse_locomotive, 4);
    adr3 = bitRead (adresse_locomotive, 3);
    adr2 = bitRead (adresse_locomotive, 2);
    adr1 = bitRead (adresse_locomotive, 1);
    adr0 = bitRead (adresse_locomotive, 0);

 }//Fin du else
}// fin du if (compteur_touche_locomotive == 2)
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

```
if (compteur_touche_locomotive == 3)//Prise en compte du chiffre saisi pour trois entrées
{
   nbre_locomotive = ((toucheA_locomotive-48) * 100) + ((toucheB_locomotive-48)*10)
   +(toucheC_locomotive-48);
   //nbre contient l'intégralité de la saisie

   if (nbre_locomotive > 0 && nbre_locomotive < 128)
   {
   compteur_touche_locomotive = 0;//Réinitialise compteur_touche

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   locSelectionne = 1;//Sélectionne la loc 1 et permet l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2

  autorise_touche_7_et_9 = 1;
   //Autorise la prise en compte de la touche 7 ou 9 pour changer la locomotive sous contrôle

  autorise_touche_1_et_4 = 1;

   lcd.clear(); // Efface écran

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

   adresse_locomotive = nbre_locomotive;
   //recopie dans adresse la valeur de nbre en binaire

   //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   adr7 = 0;//Adresse courte
   adr6 = bitRead (adresse_locomotive, 6);//Met dans adr6, le bit 6 de la variable adresse
   adr5 = bitRead (adresse_locomotive, 5);
   adr4 = bitRead (adresse_locomotive, 4);
   adr3 = bitRead (adresse_locomotive, 3);
   adr2 = bitRead (adresse_locomotive, 2);
   adr1 = bitRead (adresse_locomotive, 1);
   adr0 = bitRead (adresse_locomotive, 0);

   }//fin if nbre_locomotive <128
```

```
//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
  if (nbre_locomotive > 127)
  {
   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_locomotive = 0;//Réinitialise compteur_touche

   autorise_touche_7_et_9 = 1;
   //Autorise la prise en compte de la touche 7 ou 9 pour changer la locomotive sous contrôle

  autorise_touche_1_et_4 = 1;

 lecture_memoire_eeprom ();//Appel fonction lecture_memoire_eeprom ())

   }//Fin du If > 127


//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
 if (nbre_locomotive == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2
```

```
   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0

}// fin du if (compteur_touche_locomotive == 3)

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
if (compteur_touche_locomotive == 4)//Prise en compte du chiffre pour quatre entrées
{
   nbre_locomotive = ((toucheA_locomotive-48) * 1000) + ((toucheB_locomotive-48)*100)
   +((toucheC_locomotive-48)*10)+(toucheD_locomotive-48);
   //nbre contient l'intégralité de la saisie

   if (nbre_locomotive == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte
```

```
    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
    loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2


    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    retourdetrame = 1;//En attente appui touche

  }//Fin de if nbre_locomotive == 0

    else
    {
    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    lecture_memoire_eeprom ();//Appel fonction lecture_memoire_eeprom ())

    }//Fin du else
  }//fin if compteur_touche_locomotive == 4;
  ////////////////////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////////////////
  if (compteur_touche_locomotive == 5)//Prise en compte du chiffre pour quatre entrées
  {
    nbre_locomotive = ((toucheA_locomotive-48) * 100) + ((toucheB_locomotive-48)*10)+
    (toucheC_locomotive-48);

    nbre_locomotive = (nbre_locomotive*100) + ((toucheD_locomotive-48)*10)
    +(toucheE_locomotive-48);
    //nbre contient l'intégralité de la saisie
    //Reconstitué en 2 fois sinon erreur dans la reconstitution
```

```
   if (nbre_locomotive == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2


   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0

   else
   {
   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   lecture_memoire_eeprom ();//Appel fonction lecture_memoire_eeprom ())

   }//Fin du else
```

```
}//fin if compteur_touche_locomotive == 5

if (compteur_touche_locomotive == 6)//Prise en compte du chiffre pour quatre entrées
{
   nbre_locomotive = ((toucheA_locomotive-48)* 1000) + ((toucheB_locomotive-48)*100)
   +((toucheC_locomotive-48)*10)+(toucheD_locomotive-48);

   nbre_locomotive = (nbre_locomotive*100) + ((toucheE_locomotive-48)*10)
   + (toucheF_locomotive-48);
   //nbre contient l'intégralité de la saisie
   //Reconstitué en 2 fois sinon erreur dans la reconstitution

   if (nbre_locomotive == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2


   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
```

```
  }//fermeture du for

  retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0

  else
  {
  envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

  lecture_memoire_eeprom ();//Appel fonction lecture_memoire_eeprom ())

  }//Fin du else

}//fin if compteur_touche_locomotive == 6

}//Fin du VOID TRAITEMENT DES APPUI TOUCHE LOCOMOTIVE

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////AUTRES FONCTIONS DU PROGRAMME ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
/////////////// MISE EN MEMOIRE DES TOUCHES APPUYEES LOCOMOTIVE2//////////////////////
//////////////////TOUCHE APPUYEE NUMERIQUE LOCOMOTIVE2////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_locomotive2() // // lecture de la touche appuyée
{
  compteur_touche_locomotive2 = compteur_touche_locomotive2 + 1;
  //Incrémente la variable compteur_touche

if (compteur_touche_locomotive2 == 1) //Teste si compteur_touche = 1
{ toucheA_locomotive2 = touche;//Garde trace de la touche 1ere touche appuyée
lcd.blink();

autorise_touche_locomotive2_e = 1;//Autorise la prise en compte par la touche E

}//Autorise le clignotement du curseur

if (compteur_touche_locomotive2 == 2) //Teste si compteur_touche = 2
{toucheB_locomotive2 = touche;//Garde trace de la touche 2ième touche appuyée
```

```
}//Autorise le clignotement du curseur

if (compteur_touche_locomotive2 == 3) //Teste si compteur_touche = 3
{toucheC_locomotive2 = touche;//Garde trace de la touche 3ième touche appuyée

}//Empêche le clignotement du curseur

if (compteur_touche_locomotive2 == 4) //Teste si compteur_touche = 4
{toucheD_locomotive2 = touche;//Garde trace de la touche 4ième touche appuyée

}//Autorise le clignotement du curseur

if (compteur_touche_locomotive2 == 5) //Teste si compteur_touche = 5
{toucheE_locomotive2 = touche;//Garde trace de la touche 5ième touche appuyée

}//Autorise le clignotement du curseur

if (compteur_touche_locomotive2 == 6) //Teste si compteur_touche = 6
{toucheF_locomotive2 = touche;//Garde trace de la touche 6ième touche appuyée
lcd.noBlink();

autorise_touche_locomotive2 = 0;//Empêche la saisie de chiffre supplémentaire

}//Interdit le clignotement du curseur


}//Fin du VOID MISE EN MEMOIRE DES TOUCHES APPUYEES LOCOMOTIVE2

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////// TRAITEMENT DES TOUCHES APPUYEES LOCOMOTIVE2//////////////////
/////////////////////TRAITEMENT APPUI TOUCHE LOCOMOTIVE2/////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_locomotive2()
//Correspond à Appui touche 'E' et traitement_appui_touche_locomotive2 = 1
{

if (compteur_touche_locomotive2 == 1)//Prise en compte du chiffre saisi pour une entrée
{
```

```
  nbre_locomotive2 = toucheA_locomotive2-48;//toucheA contient l'intégralité de la saisie

if (nbre_locomotive2 == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2


   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0

else
 {

  envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"
```

```
  locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
  loc2Selectionne = 1;//Sélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

  compteur_touche_locomotive2 = 0;//Réinitialise compteur_touche

  autorise_touche_7_et_9 = 1;
  //Autorise la prise en compte de la touche 7 ou 9 pour changer la locomotive sous contrôle

   autorise_touche_1_et_4 = 1;

  lcd.clear(); // Efface écran

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

   adresse_locomotive2 = nbre_locomotive2;//recopie dans adresse la valeur de nbre en binaire

    //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr27loc = 0;//Adresse courte
    adr26loc = bitRead (adresse_locomotive2, 6);//Met dans adr6, le bit 6 de la variable adresse
    adr25loc = bitRead (adresse_locomotive2, 5);
    adr24loc = bitRead (adresse_locomotive2, 4);
    adr23loc = bitRead (adresse_locomotive2, 3);
    adr22loc = bitRead (adresse_locomotive2, 2);
    adr21loc = bitRead (adresse_locomotive2, 1);
    adr20loc = bitRead (adresse_locomotive2, 0);

 }//Fin du else
}// fin du if (compteur_touche_locomotive == 1)
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (compteur_touche_locomotive2 == 2)//Prise en compte du chiffre saisi pour deux entrées
{
    nbre_locomotive2 = ((toucheA_locomotive2-48) * 10) + (toucheB_locomotive2-48);
    //nbre contient l'intégralité de la saisie

    if (nbre_locomotive2 == 0)//Refus du chiffre 0
{
    lcd.clear(); // Efface écran si appui
```

```
    lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

    lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

    lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

    lcd.print("doit etre compris") ; // Affiche la chaîne texte

    lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

    lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
    loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2



    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0

else
  {
    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_locomotive2 = 0;//Réinitialise compteur_touche

    locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
    loc2Selectionne = 1;//Sélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

    autorise_touche_7_et_9 = 1;
```

```
   //Autorise la prise en compte de la touche 7 ou 9 pour changer la locomotive sous contrôle

   autorise_touche_1_et_4 = 1;

   lcd.clear(); // Efface écran

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

   adresse_locomotive2 = nbre_locomotive2;//recopie dans adresse la valeur de nbre en binaire

   //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   adr27loc = 0;//Adresse courte
   adr26loc = bitRead (adresse_locomotive2, 6);//Met dans adr6, le bit 6 de la variable adresse
   adr25loc = bitRead (adresse_locomotive2, 5);
   adr24loc = bitRead (adresse_locomotive2, 4);
   adr23loc = bitRead (adresse_locomotive2, 3);
   adr22loc = bitRead (adresse_locomotive2, 2);
   adr21loc = bitRead (adresse_locomotive2, 1);
   adr20loc = bitRead (adresse_locomotive2, 0);

  }//Fin du else
}// fin du if (compteur_touche_locomotive == 2)

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

if (compteur_touche_locomotive2 == 3)//Prise en compte du chiffre saisi pour trois entrées
{
   nbre_locomotive2 = ((toucheA_locomotive2-48) * 100) + ((toucheB_locomotive2-48)*10)
   +(toucheC_locomotive2-48);
   //nbre contient l'intégralité de la saisie

   if (nbre_locomotive2 > 0 && nbre_locomotive2 < 128)
   {
   compteur_touche_locomotive2 = 0;//Réinitialise compteur_touche

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"
```

```
   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
   loc2Selectionne = 1;//Sélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

   autorise_touche_7_et_9 = 1;
   //Autorise la prise en compte de la touche 7 ou 9 pour changer la locomotive sous contrôle

  autorise_touche_1_et_4 = 1;

   lcd.clear(); // Efface écran

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

   adresse_locomotive2 = nbre_locomotive2;
   //recopie dans adresse la valeur de nbre en binaire

   //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   adr27loc = 0;//Adresse courte
   adr26loc = bitRead (adresse_locomotive2, 6);//Met dans adr6, le bit 6 de la variable adresse
   adr25loc = bitRead (adresse_locomotive2, 5);
   adr24loc = bitRead (adresse_locomotive2, 4);
   adr23loc = bitRead (adresse_locomotive2, 3);
   adr22loc = bitRead (adresse_locomotive2, 2);
   adr21loc = bitRead (adresse_locomotive2, 1);
   adr20loc = bitRead (adresse_locomotive2, 0);

   }//fin if nbre_locomotive2 <128
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (nbre_locomotive2 > 127)
//Prise en compte du chiffre saisi supérieur à deux entrées ou nombre total supérieur à 9
 //pour plus de locomotive augmenter nbre
{
   compteur_touche_locomotive2 = 0;//Réinitialise compteur_touche

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   autorise_touche_7_et_9 = 1;
   //Autorise la prise en compte de la touche 7 ou 9 pour changer la locomotive sous contrôle
```

```
  autorise_touche_1_et_4 = 1;

   lecture_memoire_eeprom2 ();//Appel fonction lecture_memoire_eeprom2 ()

   }//Fin du If > 127
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (nbre_locomotive2 == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2


   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche
```

```
}//Fin de if nbre_locomotive == 0

}// fin if (compteur_touche_locomotive2 == 3)

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
if (compteur_touche_locomotive2 == 4)//Prise en compte du chiffre pour quatre entrées
{
   nbre_locomotive2 = ((toucheA_locomotive2-48) * 1000) + ((toucheB_locomotive2-48)*100)+
   ((toucheC_locomotive2-48)*10)+(toucheD_locomotive2-48);
   //nbre contient l'intégralité de la saisie

if (nbre_locomotive2 == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2


   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
```

```
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0

   else
   {
   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   lecture_memoire_eeprom2 ();//Appel fonction lecture_memoire_eeprom2 ()

   }//Fin du else

}//fin if compteur_touche_locomotive == 4
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
if (compteur_touche_locomotive2 == 5)//Prise en compte du chiffre pour quatre entrées
{
   nbre_locomotive2 = ((toucheA_locomotive2-48) * 100) + ((toucheB_locomotive2-48)*10)+
   (toucheC_locomotive2-48);

   nbre_locomotive2 = (nbre_locomotive2*100) + ((toucheD_locomotive2-48)*10)+(toucheE_locomotive2-48);
   //nbre contient l'intégralité de la saisie
   //Reconstitué en 2 fois sinon erreur dans la reconstitution

if (nbre_locomotive2 == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1
```

```
   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2



   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0

   else
   {
   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   lecture_memoire_eeprom2 ();//Appel fonction lecture_memoire_eeprom2 ())

   }//Fin du else

}//fin if compteur_touche_locomotive2 == 5
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
if (compteur_touche_locomotive2 == 6)//Prise en compte du chiffre pour quatre entrées
{
   nbre_locomotive2 = ((toucheA_locomotive2-48) * 1000) + ((toucheB_locomotive2-48)*100)+
   ((toucheC_locomotive2-48)*10)+(toucheD_locomotive2-48);

   nbre_locomotive2 = (nbre_locomotive2*100)+((toucheE_locomotive2-48)*10) + (toucheF_locomotive2-48);
```

```
  //nbre contient l'intégralité de la saisie
   //Reconstitué en 2 fois sinon erreur dans la reconstitution

if (nbre_locomotive2 == 0)//Refus du chiffre 0
{
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2


   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin de if nbre_locomotive == 0

   else
   {
   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"
```

```
   lecture_memoire_eeprom2 ();//Appel fonction lecture_memoire_eeprom2 ())

   }//Fin du else

}//fin if compteur_touche_locomotive2 == 6

}//Fin du VOID TRAITEMENT DES APPUI TOUCHE LOCOMOTIVE2

////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////// Génère un bit ZERO//////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
void bitzero()
{
PORTE &=~ (1<<4);//Equivalent à digitalWrite(sdcc,LOW), Sortie sdcc, état bas, broche 2
PORTE |= (1<<5);//Equivalent à digitalWrite(sdcc2,HIGH), Sortie sdcc2, état haut, broche 3
delayMicroseconds(105);       // Pause de 105 microsecondes
PORTE |= (1<<4);//Equivalent à digitalWrite(sdcc,HIGH), Sortie sdcc, état haut, broche 2
PORTE &=~ (1<<5);//Equivalent à digitalWrite(sdcc2,LOW), Sortie sdcc2, état bas, broche 3
delayMicroseconds(105);       // Pause de 105 microsecondes
}
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////// Génère un bit UN//////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
void bitun()   //Génère un bit à 1
{
PORTE &=~ (1<<4);//Equivalent à digitalWrite(sdcc,LOW), Sortie sdcc, état bas, broche 51
PORTE |= (1<<5);////Equivalent à digitalWrite(sdcc2,HIGH), Sortie sdcc2, état haut, broche 53
delayMicroseconds(58);       // Pause de 58 microsecondes
PORTE |= (1<<4);//Equivalent à digitalWrite(sdcc,HIGH), Sortie sdcc, état haut, broche 51
PORTE &=~ (1<<5);//Equivalent à digitalWrite(sdcc2,LOW), Sortie sdcc2, état bas, broche 53
delayMicroseconds(58);       // Pause de 58 microsecondes
}
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////// TRAME LOCOMOTIVE//////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
```

```
void trame_locomotive()
{
//Valeur Lue par le convertisseur analogique/numérique
//La tension est égale à : ((ValeurLue*5)/1024)
//en fonction de la position du potentiomètre de vitesse
ValeurLue = analogRead(A0); //la valeur lue sera comprise entre 0 et 1023
//(potentiomètre) sur la broche A2

lcd.setCursor(0,0);//Place le curseur colonne 0, ligne 1

autorisation_interrupteur = 1;//Autorise le fonctionnement des interrupteurs de fonction

lcd.print("Locomotive : "); // Affiche la chaîne texte

lcd.print (nbre_locomotive);// Affiche le numéro de locomotive
////////////////////////////////
////////////////////////////////
if (comm7 == 1)//Affichage fleche sens de march sur ecran LCD
{
  lcd.setCursor(19,2);
  lcd.print(char(1));//Sens avant
}
else
{
  lcd.setCursor(19,2);

  lcd.print(char(2));//Sens arriere
}
////////////////////////////////
////////////////////////////////
if (mode_manoeuvre == 0)
{
int ConversionValeurLue = (ValeurLue / 8);

lcd.setCursor(19,0);
lcd.print(char(0));//Affiche symbole Locomotive

if (ConversionValeurLue == 1 || ConversionValeurLue == 0)
{
  comm6 = 0;comm5 = 0;comm4 = 0;comm3 = 0;comm2 = 0;comm1 = 0;comm0 = 1;
```

```
lcd.setCursor (3,2);
lcd.print ("Vitesse : 000");

}

if (ConversionValeurLue > 1)
{
 int Conversionaffichage = ConversionValeurLue - 1;

comm6 = bitRead (ConversionValeurLue, 6);
comm5 = bitRead (ConversionValeurLue, 5);
comm4 = bitRead (ConversionValeurLue, 4);
comm3 = bitRead (ConversionValeurLue, 3);
comm2 = bitRead (ConversionValeurLue, 2);
comm1 = bitRead (ConversionValeurLue, 1);
comm0 = bitRead (ConversionValeurLue, 0);

if (ConversionValeurLue <= 10)
{
lcd.setCursor (3,2);
lcd.print ("Vitesse : 00");

lcd.setCursor (15,2);
lcd.print (Conversionaffichage);
}

if (ConversionValeurLue > 10 && ConversionValeurLue <= 100)
{
lcd.setCursor (3,2);
lcd.print ("Vitesse : 00");

lcd.setCursor (14,2);
lcd.print (Conversionaffichage);
}

if (ConversionValeurLue > 100)
{
lcd.setCursor (3,2);
lcd.print ("Vitesse : ");
```

```
lcd.setCursor (13,2);
lcd.print (Conversionaffichage);
}

}//Fin du else
}//Fin if (mode_manoeuvre == 0)
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
if (mode_manoeuvre == 1)
{
ConversionValeurLue = (ValeurLue / 8);//Convertie le resultat de la

lcd.setCursor(19,0);
lcd.print("M");//Affiche symbole Locomotive

int Conversionmodemanoeuvre = (ValeurLue / 16);//conversion numerique pour correspondre avec un pas de 128

if (ConversionValeurLue == 0 || ConversionValeurLue == 1)
// 0 et 1 = arret
{
comm6 = 0;comm5 = 0;comm4 = 0;comm3 = 0;comm2 = 0;comm1 = 0;comm0 = 1;

lcd.setCursor (3,2);
lcd.print ("Vitesse : 000");

}
if (ConversionValeurLue > 1)
{
Conversionaffichage = (ConversionValeurLue /2)-1;
//Permet de diviser l'affichage du pas de vitesse par 2
//Le -1 permet de commencer a l'affichage pas 1
comm6 = bitRead (Conversionmodemanoeuvre, 6);
comm5 = bitRead (Conversionmodemanoeuvre, 5);
comm4 = bitRead (Conversionmodemanoeuvre, 4);
comm3 = bitRead (Conversionmodemanoeuvre, 3);
comm2 = bitRead (Conversionmodemanoeuvre, 2);
comm1 = bitRead (Conversionmodemanoeuvre, 1);
comm0 = bitRead (Conversionmodemanoeuvre, 0);

if (ConversionValeurLue <= 20)
```

```
{
lcd.setCursor (3,2);
lcd.print ("Vitesse : 00");

lcd.setCursor (15,2);
lcd.print (Conversionaffichage);
}

if (ConversionValeurLue > 21 && ConversionValeurLue <= 150)
//Limite a 62 pas de vitesse en manoeuvre
//>21 pour eviter bug affichage
{
lcd.setCursor (3,2);
lcd.print ("Vitesse : 00");

lcd.setCursor (14,2);
lcd.print (Conversionaffichage);
}

}//Fin du else

}//Fin du if (mode_manoeuvre == 1)


lcd.noBlink();//Empêche le clignotement du curseur

//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
////////////////////// GENERE LA TRAME dcc LOCOMOTIVE////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
// Génération des paquets DCC
// Octet de synchronisation
bitun();bitun();bitun();bitun();
bitun();bitun();bitun();bitun();
bitun();bitun();bitun();bitun();
bitun();bitun();bitun();bitun();
////////////////////////////////////
////////////////////////////////////
bitzero(); // Bit à 0 de séparation
////////////////////////////////////
```

```
////////////////////////////////////
// Octet d'adresse 1
//La centrale transmet l'octet d'adresse
bitzero();//adr7
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 2 128 pas de vitesse
bitzero();
bitzero();
bitun();
bitun();
bitun();
bitun();
bitun();
bitun();
////////////////////////////////////
////////////////////////////////////
// Bit à zéro de séparation
bitzero();
////////////////////////////////////
////////////////////////////////////
// Octet de commande
//La centrale transmet l'octet de commande
```

```
if (comm7==1) bitun();  // Si bit à 1
if (comm7==0) bitzero(); // Si bit à 0
if (comm6==1) bitun();  // Si bit à 1
if (comm6==0) bitzero(); // Si bit à 0
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
//Calcul du OU Exclusif
cont7 = adr7 ^ comm7;  // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle
cont6 = adr6 ^ comm6;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont5 = adr5 ^ comm5 ^ 1;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont4 = adr4 ^ comm4 ^ 1;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont3 = adr3 ^ comm3 ^ 1;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont2 = adr2 ^ comm2 ^ 1;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont1 = adr1 ^ comm1 ^ 1;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont0 = adr0 ^ comm0 ^ 1;  // ^ Calcul du OU EXCLUSIF bit 0 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
////////////////////////////////////
////////////////////////////////////
// Bit à un de fin de transmission
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 26 zéros (5ms) avant nouvelle transmission
bitzero();bitzero();bitzero();bitzero();bitzero();
bitzero();bitzero();bitzero();bitzero();bitzero();
bitzero();bitzero();bitzero();bitzero();bitzero();
bitzero();bitzero();bitzero();bitzero();bitzero();
bitzero();bitzero();bitzero();bitzero();bitzero();
////////////////////////////////////
////////////////////////////////////
}//Fin trame dcc locomotive

//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
////////////////////// TRAME LOCOMOTIVE2//////////////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
void trame_locomotive2()
{
//Valeur Lue par le convertisseur analogique/numérique
//La tension est égale à : ((ValeurLue*5)/1024)
//en fonction de la position du potentiomètre de vitesse
ValeurLue2 = analogRead(A2); //la valeur lue sera comprise entre 0 et 1023
//(potentiomètre) sur la broche A2

lcd.setCursor(0,0);//Place le curseur colonne 0, ligne 1
```

```
autorisation_interrupteur = 1;
//Autorise le fonctionnement des interrupteurs de fonction

lcd.print("Locomotive2: "); // Affiche la chaîne texte

lcd.print (nbre_locomotive2);// Affiche le numéro de locomotive
////////////////////////////////
////////////////////////////////
if (comm27loc == 1)//Affichage fleche sens de marche sur ecran LCD
{
 lcd.setCursor(19,2);
 lcd.print(char(1));//Sens Avant
}
else
{
 lcd.setCursor(19,2);

 lcd.print(char(2));//Sens arriere
}
////////////////////////////////
////////////////////////////////
if (mode2_manoeuvre == 0)
{
ConversionValeurLue = (ValeurLue2 / 8);

lcd.setCursor(19,0);
lcd.print(char(0));//Affiche symbole Locomotive

if (ConversionValeurLue == 1 || ConversionValeurLue == 0)
{
 comm26loc = 0;comm25loc = 0;comm24loc = 0;comm23loc = 0;comm22loc = 0;comm21loc = 0;comm20loc = 1;

lcd.setCursor (2,2);
lcd.print ("Vitesse2 : 000");

}



if (ConversionValeurLue > 1)
{
Conversionaffichage = ConversionValeurLue - 1;
```

```
comm26loc = bitRead (ConversionValeurLue, 6);
comm25loc = bitRead (ConversionValeurLue, 5);
comm24loc = bitRead (ConversionValeurLue, 4);
comm23loc = bitRead (ConversionValeurLue, 3);
comm22loc = bitRead (ConversionValeurLue, 2);
comm21loc = bitRead (ConversionValeurLue, 1);
comm20loc = bitRead (ConversionValeurLue, 0);

if (ConversionValeurLue <= 10)
{
lcd.setCursor (2,2);
lcd.print ("Vitesse2 : 00");

lcd.setCursor (15,2);
lcd.print (Conversionaffichage);
}

if (ConversionValeurLue > 10 && ConversionValeurLue <= 100)
{
lcd.setCursor (2,2);
lcd.print ("Vitesse2 : 00");

lcd.setCursor (14,2);
lcd.print (Conversionaffichage);
}

if (ConversionValeurLue > 100)
{
lcd.setCursor (2,2);
lcd.print ("Vitesse2 : ");

lcd.setCursor (13,2);
lcd.print (Conversionaffichage);
}

}//Fin du else
}//Fin if (mode2_manoeuvre == 0)
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
```

```
if (mode2_manoeuvre == 1)

{

ConversionValeurLue = (ValeurLue2 / 8);//Convertie le resultat de la


lcd.setCursor(19,0);

lcd.print("M");//Affiche symbole Locomotive


Conversionmodemanoeuvre = (ValeurLue2 / 16);//conversion numerique pour correspondre avec un pas de 128


if (ConversionValeurLue == 0 || ConversionValeurLue == 1)

// 0 et 1 = arret

{

comm26loc = 0;comm25loc = 0;comm24loc = 0;comm23loc = 0;comm22loc = 0;comm21loc = 0;comm20loc = 1;


lcd.setCursor (2,2);

lcd.print ("Vitesse2 : 000");


}

if (ConversionValeurLue > 1)

{

Conversionaffichage = (ConversionValeurLue /2)-1;

//Permet de diviser l'affichage du pas de vitesse par 2

//Le -1 permet de commencer a l'affichage pas 1

comm26loc = bitRead (Conversionmodemanoeuvre, 6);

comm25loc = bitRead (Conversionmodemanoeuvre, 5);

comm24loc = bitRead (Conversionmodemanoeuvre, 4);

comm23loc = bitRead (Conversionmodemanoeuvre, 3);

comm22loc = bitRead (Conversionmodemanoeuvre, 2);

comm21loc = bitRead (Conversionmodemanoeuvre, 1);

comm20loc = bitRead (Conversionmodemanoeuvre, 0);


if (ConversionValeurLue <= 20)

{

lcd.setCursor (2,2);

lcd.print ("Vitesse2 : 00");


lcd.setCursor (15,2);

lcd.print (Conversionaffichage);

}


if (ConversionValeurLue > 21 && ConversionValeurLue <= 150)
```

```
//Limite a 62 pas de vitesse en manoeuvre
//>21 pour eviter bug affichage
{
lcd.setCursor (2,2);
lcd.print ("Vitesse2 : 00");

lcd.setCursor (14,2);
lcd.print (Conversionaffichage);
}

}//Fin du else

}//Fin du if (mode2_manoeuvre == 1)

lcd.noBlink();//Empêche le clignotement du curseur

////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////// GENERE LA TRAME dcc LOCOMOTIVE2///////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
// Génération des paquets DCC
// Octet de synchronisation
bitun();bitun();bitun();bitun();
bitun();bitun();bitun();bitun();
bitun();bitun();bitun();bitun();
bitun();bitun();bitun();bitun();
////////////////////////////////////
////////////////////////////////////
bitzero(); // Bit à 0 de séparation
////////////////////////////////////
////////////////////////////////////
// Octet d'adresse 1
//La centrale transmet l'octet d'adresse
bitzero();//adr7
if (adr26loc==1) bitun();  // Si bit à 1
if (adr26loc==0) bitzero(); // Si bit à 0
if (adr25loc==1) bitun();  // Si bit à 1
if (adr25loc==0) bitzero(); // Si bit à 0
if (adr24loc==1) bitun();  // Si bit à 1
if (adr24loc==0) bitzero(); // Si bit à 0
```

```
if (adr23loc==1) bitun();  // Si bit à 1
if (adr23loc==0) bitzero(); // Si bit à 0
if (adr22loc==1) bitun();  // Si bit à 1
if (adr22loc==0) bitzero(); // Si bit à 0
if (adr21loc==1) bitun();  // Si bit à 1
if (adr21loc==0) bitzero(); // Si bit à 0
if (adr20loc==1) bitun();  // Si bit à 1
if (adr20loc==0) bitzero(); // Si bit à 0
////////////////////////////////////
////////////////////////////////////
// Bit à zéro de séparation
bitzero();
////////////////////////////////////
////////////////////////////////////
//octet 2 128 pas de vitesse
bitzero();
bitzero();
bitun();
bitun();
bitun();
bitun();
bitun();
bitun();
////////////////////////////////////
////////////////////////////////////
// Bit à zéro de séparation
bitzero();
////////////////////////////////////
////////////////////////////////////
// Octet de commande
//La centrale transmet l'octet de commande
if (comm27loc==1) bitun();  // Si bit à 1
if (comm27loc==0) bitzero(); // Si bit à 0
if (comm26loc==1) bitun();  // Si bit à 1
if (comm26loc==0) bitzero(); // Si bit à 0
if (comm25loc==1) bitun();  // Si bit à 1
if (comm25loc==0) bitzero(); // Si bit à 0
if (comm24loc==1) bitun();  // Si bit à 1
if (comm24loc==0) bitzero(); // Si bit à 0
if (comm23loc==1) bitun();  // Si bit à 1
if (comm23loc==0) bitzero(); // Si bit à 0
```

```
if (comm22loc==1) bitun();  // Si bit à 1
if (comm22loc==0) bitzero(); // Si bit à 0
if (comm21loc==1) bitun();  // Si bit à 1
if (comm21loc==0) bitzero(); // Si bit à 0
if (comm20loc==1) bitun();  // Si bit à 1
if (comm20loc==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
//Calcul du OU Exclusif
cont27loc = adr27loc ^ comm27loc;  // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle
cont26loc = adr26loc ^ comm26loc;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont25loc = adr25loc ^ comm25loc ^ 1;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont24loc = adr24loc ^ comm24loc ^ 1;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont23loc = adr23loc ^ comm23loc ^ 1;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont22loc = adr22loc ^ comm22loc ^ 1;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont21loc = adr21loc ^ comm21loc ^ 1;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont20loc = adr20loc ^ comm20loc ^ 1;  // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle

if (cont27loc==1) bitun();  // Si bit à 1
if (cont27loc==0) bitzero(); // Si bit à 0
if (cont26loc==1) bitun();  // Si bit à 1
if (cont26loc==0) bitzero(); // Si bit à 0
if (cont25loc==1) bitun();  // Si bit à 1
if (cont25loc==0) bitzero(); // Si bit à 0
if (cont24loc==1) bitun();  // Si bit à 1
if (cont24loc==0) bitzero(); // Si bit à 0
if (cont23loc==1) bitun();  // Si bit à 1
if (cont23loc==0) bitzero(); // Si bit à 0
if (cont22loc==1) bitun();  // Si bit à 1
if (cont22loc==0) bitzero(); // Si bit à 0
if (cont21loc==1) bitun();  // Si bit à 1
if (cont21loc==0) bitzero(); // Si bit à 0
if (cont20loc==1) bitun();  // Si bit à 1
if (cont20loc==0) bitzero(); // Si bit à
////////////////////////////////////
```

```
////////////////////////////////////
// Bit à un de fin de transmission
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros (5ms) avant nouvelle transmission
bitzero();bitzero();bitzero();bitzero();bitzero();
bitzero();bitzero();bitzero();bitzero();bitzero();
bitzero();bitzero();bitzero();bitzero();bitzero();
bitzero();bitzero();bitzero();bitzero();bitzero();
bitzero();bitzero();bitzero();bitzero();bitzero();
////////////////////////////////////
////////////////////////////////////
}//Fin trame dcc locomotive2
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////Génère la trame IDLE//////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void trame_idle()//
{
  //Leds Sens de marche
  PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)
  PORTA &=~ (1<<1);//Equivalent à digitalWrite(Ledbleue,LOW)
  PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
  PORTK |=  (1<<4);//Equivalent à digitalWrite(Ledjaune,HIGH)

//Bit de Synchronisation 16 bit à 1
// Octet de synchronisation
 for ( i=0; i <= 15; i++)
 {bitun(); }    // La centrale transmet 16 bits à 1
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
bitzero();
////////////////////////////////////
////////////////////////////////////
//Octet de d'adresse 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 25 zéros (5 ms)avant nouvelle transmission
for ( i=0; i<=24; i++)
bitzero();
}// Fin trame_idle()
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
//////////////////GENERE La TRAME FONCTION//////////////////
///////////////////////INTERRUPTEUR///////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
void envoi_trame_fonction()//Interrupteurs de fonction
{
  PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)
  PORTA |=  (1<<1);//Equivalent à digitalWrite(Ledbleue,HIGH)
  PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
```

```
  PORTK &=~ (1<<4);//Equivalent à digitalWrite(Ledjaune,LOW)

//Octet 2 de commande locomotive 1
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
   comm7 = 1;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm6 = 0;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm5 = 0;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm4 = octetFonctionFL;
   comm3 = octetFonctionF4;
   comm2 = octetFonctionF3;
   comm1 = octetFonctionF2;
   comm0 = octetFonctionF1;

//Octet 2 de commande locomotive 2
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
   comm27loc = 1;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm26loc = 0;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm25loc = 0;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm24loc = octetFonctionFL;
   comm23loc = octetFonctionF4;
   comm22loc = octetFonctionF3;
   comm21loc = octetFonctionF2;
   comm20loc = octetFonctionF1;
 ////////////////////////////////////////////
 ////////////////////////////////////////////
for(int compteur = 0; compteur < 8; compteur++)
//Envoi 5 fois la Fonction pour sécuriser la réalisation de l'ordre
{
  //Bit de Synchronisation 20 bit à 1
// Octet de synchronisation
 for ( i=0; i <= 19; i++)
 {bitun();}    // La centrale transmet 20 bits à 1
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
/////////////////////////////////////////
bitzero();
//////////////////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////////////
/////////////La fonction est active pour la loc 1/////////////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
//Octet de données 1
if (locSelectionne == 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

//if (comm7==1) bitun();  // Si bit à 1
//if (comm7==0) bitzero(); // Si bit à 0
bitun();
//if (comm6==1) bitun();  // Si bit à 1
//if (comm6==0) bitzero(); // Si bit à 0
bitzero();
```

```
//if (comm5==1) bitun();  // Si bit à 1
//if (comm5==0) bitzero(); // Si bit à 0
bitzero();
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 du If (locSelectionne == 1)
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
/////////////La fonction est active pour la loc 2//////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
if (loc2Selectionne == 1)
{
if (adr27loc==1) bitun();  // Si bit à 1
if (adr27loc==0) bitzero(); // Si bit à 0
if (adr26loc==1) bitun();  // Si bit à 1
if (adr26loc==0) bitzero(); // Si bit à 0
if (adr25loc==1) bitun();  // Si bit à 1
if (adr25loc==0) bitzero(); // Si bit à 0
if (adr24loc==1) bitun();  // Si bit à 1
if (adr24loc==0) bitzero(); // Si bit à 0
if (adr23loc==1) bitun();  // Si bit à 1
if (adr23loc==0) bitzero(); // Si bit à 0
if (adr22loc==1) bitun();  // Si bit à 1
if (adr22loc==0) bitzero(); // Si bit à 0
if (adr21loc==1) bitun();  // Si bit à 1
if (adr21loc==0) bitzero(); // Si bit à 0
if (adr20loc==1) bitun();  // Si bit à 1
if (adr20loc==0) bitzero(); // Si bit à 0

////////////////////////////////////
////////////////////////////////////
// Bit à zéro de séparation
bitzero();
////////////////////////////////////
////////////////////////////////////
// Octet de commande
//La centrale transmet l'octet de commande
```

```
//if (comm27loc==1) bitun();  // Si bit à 1
//if (comm27loc==0) bitzero(); // Si bit à 0
bitun();
//if (comm26loc==1) bitun();  // Si bit à 1
//if (comm26loc==0) bitzero(); // Si bit à 0
bitzero();
//if (comm25loc==1) bitun();  // Si bit à 1
//if (comm25loc==0) bitzero(); // Si bit à 0
bitzero();
if (comm24loc==1) bitun();  // Si bit à 1
if (comm24loc==0) bitzero(); // Si bit à 0
if (comm23loc==1) bitun();  // Si bit à 1
if (comm23loc==0) bitzero(); // Si bit à 0
if (comm22loc==1) bitun();  // Si bit à 1
if (comm22loc==0) bitzero(); // Si bit à 0
if (comm21loc==1) bitun();  // Si bit à 1
if (comm21loc==0) bitzero(); // Si bit à 0
if (comm20loc==1) bitun();  // Si bit à 1
if (comm20loc==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
//Calcul du OU Exclusif
cont20loc = adr20loc ^ comm20loc;  // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont21loc = adr21loc ^ comm21loc;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont22loc = adr22loc ^ comm22loc;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont23loc = adr23loc ^ comm23loc;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont24loc = adr24loc ^ comm24loc;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont25loc = adr25loc ^ comm25loc;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont26loc = adr26loc ^ comm26loc;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont27loc = adr27loc ^ comm27loc;  // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle

if (cont27loc==1) bitun();  // Si bit à 1
if (cont27loc==0) bitzero(); // Si bit à 0
if (cont26loc==1) bitun();  // Si bit à 1
```

```
if (cont26loc==0) bitzero(); // Si bit à 0
if (cont25loc==1) bitun();  // Si bit à 1
if (cont25loc==0) bitzero(); // Si bit à 0
if (cont24loc==1) bitun();  // Si bit à 1
if (cont24loc==0) bitzero(); // Si bit à 0
if (cont23loc==1) bitun();  // Si bit à 1
if (cont23loc==0) bitzero(); // Si bit à 0
if (cont22loc==1) bitun();  // Si bit à 1
if (cont22loc==0) bitzero(); // Si bit à 0
if (cont21loc==1) bitun();  // Si bit à 1
if (cont21loc==0) bitzero(); // Si bit à 0
if (cont20loc==1) bitun();  // Si bit à 1
if (cont20loc==0) bitzero(); // Si bit à
}//Fin du If (loc2Selectionne == 1)
////////////////////////////////////
////////////////////////////////////
//Fin de transmission
//bit un
////////////////////////////////////////////
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros avant nouvelle transmission
////////////////////////////////////////////
for ( i=0; i<=24; i++)
bitzero();

//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
/////Permet l'affichage de la locmotive sélectionnée//////////
///////////////en fonction de la variable/////////////////////
////////////locSelectionne ou loc2Selectionne/////////////////
//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////

locSelectionne = Miseenmemoirelocselectionne;
//Copie dans locSelectionne la mémorisation de la variable pour permettre l'envoi
//de la trame locomotive afin de retrouver l'affichage de la vitesse de la locSelectionnée

loc2Selectionne = Miseenmemoireloc2selectionne;
//Copie dans locSelectionne la mémorisation de la variable pour permettre l'envoi
```

```
//de la trame locomotive afin de retrouver l'affichage de la vitesse de la locSelectionnée

autorise_touche_7_et_9 = 1;
//Autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle
//Remis à 1 après la fonction demandée pour que seule la touche 3 soit pris en compte

autorise_touche_1_et_4 = 1;

autorise_touche_fonction = 0;//Interdit la saisie de chiffre pour programmation locomotive
//Seule les touches 7 et 9 sont autorisées

}//Fin du for 5 fois fonction FL désactivée
}//Fin du void trame 100

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////Génère la trame URGENCE//////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
// Génère la trame d'urgence (Emergency Stop, bit 5 ignoré)
void TrameArretUrg()
{
//Bit de Synchronisation 16 bit à 1
bitun();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
bitzero();bitzero();bitzero();bitzero();
bitzero();bitzero();bitzero();bitzero();
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
```

```
bitzero();
////////////////////////////////////
////////////////////////////////////
//Octet de données 2
bitzero();bitun();bitzero();bitun();
bitzero();bitzero();bitzero();bitzero();
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
bitzero();
////////////////////////////////////
////////////////////////////////////
//Octet de données 3
bitzero();bitun();bitzero();bitun();
bitzero();bitzero();bitzero();bitzero();
////////////////////////////////////
////////////////////////////////////
//Fin de transmission
//bit un
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros (5ms) avant nouvelle transmission
 for ( i=0; i<=24; i++)
bitzero();
}// Fin Void TrameArretUrg()

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////// MISE EN MEMOIRE DES TOUCHES APPUYEES CV//////////////////////////
//////////////////////////TOUCHE APPUYEE NUMERIQUE//////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_cv() //Stockage touche appuyée
{
compteur_touche_cv = compteur_touche_cv + 1;//Incrémente la variable compteur_touche

if (compteur_touche_cv == 1) //Teste si compteur_touche = 1
{toucheAcv = touche;
lcd.blink ();
```

```
autorise_touche_Cv_e = 1;//Autorise la prise en compte par la touche E

}//Garde trace de la touche 1ere touche appuyée

if (compteur_touche_cv == 2) //Teste si compteur_touche = 2
{toucheBcv = touche;
lcd.blink ();}//Garde trace de la touche 2ième touche appuyée

if (compteur_touche_cv == 3) //Teste si compteur_touche = 3
{toucheCcv = touche;//Garde trace de la touche 3ième touche appuyée
lcd.noBlink ();

autorise_touche_Cv = 0;//Empêche la saisie de chiffre supplémentaire

}//Garde trace de la touche 3ième touche appuyée
}//Fin du VOID MISE EN MEMOIRE DES TOUCHES APPUYEES

////////////////////////////////////////////////////////////////////////////////////
////////////////////// TRAITEMENT DES TOUCHES APPUYEES PROGRAMMATION CV//////////////////
//////////////////////////////TRAITEMENT APPUI TOUCHE////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_cv()//Correspond à Appui touche 'E' et autorise_touche_cv
{

 if (compteur_touche_cv == 1)//Prise en compte  si 1 chiffre saisi
 {
    nbre_cv = (toucheAcv-48);//nbre contient l'intégralité de la saisie

if (nbre_cv == 0)//Prise en compte  si 1 chiffre saisi
 {
  lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1
```

```
    lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_programmation_accessoire == 0)

else
{

     envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_cv = 0;//Remet le compteur à 0 pour prochaine saisie

    adresse_cv = nbre_cv;//recopie dans adresse la valeur de nbre en binaire

    lcd.setCursor(9,2);
    //Place le curseur colonne 9, ligne 3 comme dans choix locomotive
    //pour que les chiffres se superposent
    lcd.print (adresse_cv);

    //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv = 0;//Adresse courte
    adr6cv = bitRead (adresse_cv, 6);
    adr5cv = bitRead (adresse_cv, 5);
    adr4cv = bitRead (adresse_cv, 4);
    adr3cv = bitRead (adresse_cv, 3);
    adr2cv = bitRead (adresse_cv, 2);
    adr1cv = bitRead (adresse_cv, 1);
```

```
   adr0cv = bitRead (adresse_cv, 0);

//Affiche le nombre adresse de départ en binaire
   lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
   lcd.print (adr7cv);
   lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
   lcd.print (adr6cv);
   lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
   lcd.print (adr5cv);
   lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
   lcd.print (adr4cv);
   lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
   lcd.print (adr3cv);
   lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
   lcd.print (adr2cv);
   lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
   lcd.print (adr1cv);
   lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
   lcd.print (adr0cv);

lcd.noBlink();

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_cv);

 ////Affiche le nombre adresse de départ en binaire 2ième affichage
   lcd.setCursor(6,1);//Place le curseur colonne 6, ligne 2
   lcd.print (adr7cv);
   lcd.setCursor(7,1);//Place le curseur colonne 7, ligne 2
   lcd.print (adr6cv);
   lcd.setCursor(8,1);//Place le curseur colonne 8, ligne 2
   lcd.print (adr5cv);
```

```
   lcd.setCursor(9,1);//Place le curseur colonne 9, ligne 2
   lcd.print (adr4cv);
   lcd.setCursor(10,1);//Place le curseur colonne 10, ligne 2
   lcd.print (adr3cv);
   lcd.setCursor(11,1);//Place le curseur colonne 11, ligne 2
   lcd.print (adr2cv);
   lcd.setCursor(12,1);//Place le curseur colonne 12, ligne 2
   lcd.print (adr1cv);
   lcd.setCursor(13,1);//Place le curseur colonne 13, ligne 2
   lcd.print (adr0cv);



  lcd.setCursor (0,2);
lcd.print ("Saisir la donnee :");
lcd.blink();
lcd.setCursor (8,3);

//Gestion des claviers
autorise_touche_Cv2 = 1;//Autorise la prise en compte de l'appui touche + et autorise_touche_Cv2 à 1
}//Fin du else
}// fin du if = 1
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
if (compteur_touche_cv == 2)
{
  nbre_cv = ((toucheAcv-48) * 10) + (toucheBcv-48);//nbre contient l'intégralité de la saisie

if (nbre_cv == 0)//Prise en compte  si 1 chiffre saisi
 {
  lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1
```

```
    lcd.print("entre 1 et 127") ; // Affiche la chaîne texte

    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_programmation_accessoire == 0)

else
{
    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_cv = 0;//Remet le compteur à 0 pour prochaine saisie

    adresse_cv = nbre_cv;//recopie dans adresse la valeur de nbre en binaire

    lcd.setCursor(9,2);
    //Place le curseur colonne 9, ligne 3 comme dans choix locomotive
    //pour que les chiffres se superposent
    lcd.print (adresse_cv);

    //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv = 0;//Adresse courte
    adr6cv = bitRead (adresse_cv, 6);
    adr5cv = bitRead (adresse_cv, 5);
    adr4cv = bitRead (adresse_cv, 4);
    adr3cv = bitRead (adresse_cv, 3);
    adr2cv = bitRead (adresse_cv, 2);
    adr1cv = bitRead (adresse_cv, 1);
    adr0cv = bitRead (adresse_cv, 0);
```

```
//Affiche le nombre adresse de départ en binaire
   lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
   lcd.print (adr7cv);
   lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
   lcd.print (adr6cv);
   lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
   lcd.print (adr5cv);
   lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
   lcd.print (adr4cv);
   lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
   lcd.print (adr3cv);
   lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
   lcd.print (adr2cv);
   lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
   lcd.print (adr1cv);
   lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
   lcd.print (adr0cv);

lcd.noBlink();

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_cv);

   lcd.setCursor(6,1);//Place le curseur colonne 6, ligne 2
   lcd.print (adr7cv);
   lcd.setCursor(7,1);//Place le curseur colonne 7, ligne 2
   lcd.print (adr6cv);
   lcd.setCursor(8,1);//Place le curseur colonne 8, ligne 2
   lcd.print (adr5cv);
   lcd.setCursor(9,1);//Place le curseur colonne 9, ligne 2
   lcd.print (adr4cv);
   lcd.setCursor(10,1);//Place le curseur colonne 10, ligne 2
```

```
   lcd.print (adr3cv);
   lcd.setCursor(11,1);//Place le curseur colonne 11, ligne 2
   lcd.print (adr2cv);
   lcd.setCursor(12,1);//Place le curseur colonne 12, ligne 2
   lcd.print (adr1cv);
   lcd.setCursor(13,1);//Place le curseur colonne 13, ligne 2
   lcd.print (adr0cv);

  lcd.setCursor (0,2);
lcd.print ("Saisir la donnee :");
lcd.blink();
lcd.setCursor (8,3);

//Gestion des claviers
autorise_touche_Cv2 = 1;
//Autorise la prise en compte de l'appui touche + et autorise_touche_Cv2 à 1
}//Fin du else
}//fin du if compteur touche = 2
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (compteur_touche_cv == 3)
{
  nbre_cv = ((toucheAcv-48) * 100) + ((toucheBcv-48)*10) + (toucheCcv-48);
  //nbre contient l'intégralité de la saisie}

if (nbre_cv == 0 || nbre_cv > 127)//Prise en compte  si 1 chiffre saisi
 {
  lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 127") ; // Affiche la chaîne texte
```

```
    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_programmation_accessoire == 0)

else
{

    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_cv = 0;//Remet le compteur à 0 pour prochaine saisie

    adresse_cv = nbre_cv;//recopie dans adresse la valeur de nbre en binaire

    lcd.setCursor(9,2);
    //Place le curseur colonne 9, ligne 3 comme
    //dans choix locomotive pour que les chiffres se superposent
    lcd.print (adresse_cv);

    //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv = 0;//Adresse courte
    adr6cv = bitRead (adresse_cv, 6);
    adr5cv = bitRead (adresse_cv, 5);
    adr4cv = bitRead (adresse_cv, 4);
    adr3cv = bitRead (adresse_cv, 3);
    adr2cv = bitRead (adresse_cv, 2);
    adr1cv = bitRead (adresse_cv, 1);
    adr0cv = bitRead (adresse_cv, 0);

//Affiche le nombre adresse de départ en binaire
```

```
   lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
   lcd.print (adr7cv);
   lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
   lcd.print (adr6cv);
   lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
   lcd.print (adr5cv);
   lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
   lcd.print (adr4cv);
   lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
   lcd.print (adr3cv);
   lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
   lcd.print (adr2cv);
   lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
   lcd.print (adr1cv);
   lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
   lcd.print (adr0cv);

lcd.noBlink();

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_cv);

   lcd.setCursor(6,1);//Place le curseur colonne 6, ligne 2
   lcd.print (adr7cv);
   lcd.setCursor(7,1);//Place le curseur colonne 7, ligne 2
   lcd.print (adr6cv);
   lcd.setCursor(8,1);//Place le curseur colonne 8, ligne 2
   lcd.print (adr5cv);
   lcd.setCursor(9,1);//Place le curseur colonne 9, ligne 2
   lcd.print (adr4cv);
   lcd.setCursor(10,1);//Place le curseur colonne 10, ligne 2
   lcd.print (adr3cv);
```

```
   lcd.setCursor(11,1);//Place le curseur colonne 11, ligne 2
   lcd.print (adr2cv);
   lcd.setCursor(12,1);//Place le curseur colonne 12, ligne 2
   lcd.print (adr1cv);
   lcd.setCursor(13,1);//Place le curseur colonne 13, ligne 2
   lcd.print (adr0cv);

  lcd.setCursor (0,2);
lcd.print ("Saisir donnee");
lcd.setCursor (8,3);
lcd.blink();

//Gestion des claviers
autorise_touche_Cv2 = 1;
//Autorise la prise en compte de l'appui touche + et autorise_touche_Cv2 à 1

   }//Fin du else

}//fin du if compteur touche = 3

}//Fin du VOID TRAITEMENT DES TOUCHES APPUYEES
////////////////////////////////////////////////////////////////////////////////////
////////////////////////////// ////////////////////////////////////////////////////
////////////////////////// MISE EN MEMOIRE DES TOUCHES APPUYEES CV2//////////////////////
//////////////////////////////TOUCHE APPUYEE NUMERIQUE 2////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_cv2() //Stockage touche appuyée
{
 compteur_touche_cv2 = compteur_touche_cv2 + 1;//Incrémente la variable compteur_touche

if (compteur_touche_cv2 == 1) //Teste si compteur_touche = 1
{toucheAcv2 = touche;
lcd.blink ();

autorise_touche_Cv2_e = 1;//Autorise la prise en compte par la touche E

}//Garde trace de la touche 1ere touche appuyée

if (compteur_touche_cv2 == 2) //Teste si compteur_touche = 2
{toucheBcv2 = touche;
lcd.blink ();}//Garde trace de la touche 2ième touche appuyée
```

```
if (compteur_touche_cv2 == 3) //Teste si compteur_touche = 3
{toucheCcv2 = touche;//Garde trace de la touche 3ième touche appuyée
lcd.noBlink ();

autorise_touche_Cv2 = 0;//Empêche la saisie de nchiffre supplémentaire

}
}//Fin du VOID MISE EN MEMOIRE DES TOUCHES APPUYEES

////////////////////////////////////////////////////////////////////////////
////////////////////// TRAITEMENT DES TOUCHES APPUYEES PROGRAMMATION CV2//////////
//////////////////////////////TRAITEMENT APPUI TOUCHE//////////////////////////
////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_cv2()
{
 if (compteur_touche_cv2 == 1)//Prise en compte  si 1 chiffre saisi

{
   nbre_cv2 = (toucheAcv2-48);//nbre contient l'intégralité de la saisie

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_cv2 = 0;//Remet le compteur à 0 pour prochaine saisie

   adresse_cv2 = nbre_cv2;//recopie dans adresse la valeur de nbre en binaire

  lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_cv);//affiche la variable nbre_cv. Chiffre saisi pour l'adresse de départ

//Affiche le nombre adresse de départ en binaire
   lcd.setCursor(6,1);//Place le curseur colonne 6, ligne 2
   lcd.print (adr7cv);
   lcd.setCursor(7,1);//Place le curseur colonne 7, ligne 2
   lcd.print (adr6cv);
   lcd.setCursor(8,1);//Place le curseur colonne 8, ligne 2
   lcd.print (adr5cv);
   lcd.setCursor(9,1);//Place le curseur colonne 9, ligne 2
   lcd.print (adr4cv);
```

```
    lcd.setCursor(10,1);//Place le curseur colonne 10, ligne 2
    lcd.print (adr3cv);
    lcd.setCursor(11,1);//Place le curseur colonne 11, ligne 2
    lcd.print (adr2cv);
    lcd.setCursor(12,1);//Place le curseur colonne 12, ligne 2
    lcd.print (adr1cv);
    lcd.setCursor(13,1);//Place le curseur colonne 13, ligne 2
    lcd.print (adr0cv);

lcd.setCursor (0,2);
lcd.print ("Donnee saisie : ");
lcd.print(nbre_cv2);

//Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv2 = bitRead (adresse_cv2, 7);
    adr6cv2 = bitRead (adresse_cv2, 6);
    adr5cv2 = bitRead (adresse_cv2, 5);
    adr4cv2 = bitRead (adresse_cv2, 4);
    adr3cv2 = bitRead (adresse_cv2, 3);
    adr2cv2 = bitRead (adresse_cv2, 2);
    adr1cv2 = bitRead (adresse_cv2, 1);
    adr0cv2 = bitRead (adresse_cv2, 0);

    lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
    lcd.print (adr7cv2);
    lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
    lcd.print (adr6cv2);
    lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
    lcd.print (adr5cv2);
    lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
    lcd.print (adr4cv2);
    lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
    lcd.print (adr3cv2);
    lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
    lcd.print (adr2cv2);
    lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
    lcd.print (adr1cv2);
    lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
    lcd.print (adr0cv2);
```

```
lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (2,0);
lcd.print ("Numero de la Cv : ");

lcd.blink();//Permet le clignotement du curseur
lcd.setCursor (9,1);

//Gestion des claviers
autorise_touche_Cv4 = 1;

}// fin du if (compteur_touche_cv2 == 1)
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
if (compteur_touche_cv2 == 2)
{
  nbre_cv2 = ((toucheAcv2-48) * 10) + (toucheBcv2-48);
  //nbre contient l'intégralité de la saisie

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_cv2 = 0;//Remet le compteur à 0 pour prochaine saisie

   adresse_cv2 = nbre_cv2;//recopie dans adresse la valeur de nbre en binaire

  lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_cv);//affiche la variable nbre_cv

//Affiche le nombre adresse de départ en binaire
   lcd.setCursor(6,1);//Place le curseur colonne 6, ligne 2
```

```
    lcd.print (adr7cv);
    lcd.setCursor(7,1);//Place le curseur colonne 7, ligne 2
    lcd.print (adr6cv);
    lcd.setCursor(8,1);//Place le curseur colonne 8, ligne 2
    lcd.print (adr5cv);
    lcd.setCursor(9,1);//Place le curseur colonne 9, ligne 2
    lcd.print (adr4cv);
    lcd.setCursor(10,1);//Place le curseur colonne 10, ligne 2
    lcd.print (adr3cv);
    lcd.setCursor(11,1);//Place le curseur colonne 11, ligne 2
    lcd.print (adr2cv);
    lcd.setCursor(12,1);//Place le curseur colonne 12, ligne 2
    lcd.print (adr1cv);
    lcd.setCursor(13,1);//Place le curseur colonne 13, ligne 2
    lcd.print (adr0cv);

lcd.setCursor (0,2);
lcd.print ("Donnee saisie : ");
lcd.print(nbre_cv2);

//Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv2 = bitRead (adresse_cv2, 7);
    adr6cv2 = bitRead (adresse_cv2, 6);
    adr5cv2 = bitRead (adresse_cv2, 5);
    adr4cv2 = bitRead (adresse_cv2, 4);
    adr3cv2 = bitRead (adresse_cv2, 3);
    adr2cv2 = bitRead (adresse_cv2, 2);
    adr1cv2 = bitRead (adresse_cv2, 1);
    adr0cv2 = bitRead (adresse_cv2, 0);

    lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
    lcd.print (adr7cv2);
    lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
    lcd.print (adr6cv2);
    lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
    lcd.print (adr5cv2);
    lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
    lcd.print (adr4cv2);
    lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
    lcd.print (adr3cv2);
```

```
   lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
   lcd.print (adr2cv2);
   lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
   lcd.print (adr1cv2);
   lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
   lcd.print (adr0cv2);

lcd.noBlink();

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (2,0);
lcd.print ("Numero de la Cv : ");
lcd.setCursor (9,1);
lcd.blink();//Permet le clignotement du curseur

//Gestion des claviers
autorise_touche_Cv4 = 1;

}//fin du if (compteur_touche_cv2 == 2)
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (compteur_touche_cv2 == 3)
{
  nbre_cv2 = ((toucheAcv2-48) * 100) + ((toucheBcv2-48)*10) + (toucheCcv2-48);
  //nbre contient l'intégralité de la saisie}

if (nbre_cv2 < 256)
   {
   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_cv2 = 0;//Remet le compteur à 0 pour prochaine saisie

   adresse_cv2 = nbre_cv2;//recopie dans adresse la valeur de nbre en binaire
```

```
  lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_cv);//affiche la variable nbre_cv

//Affiche le nombre adresse de départ en binaire
   lcd.setCursor(6,1);//Place le curseur colonne 6, ligne 2
   lcd.print (adr7cv);
   lcd.setCursor(7,1);//Place le curseur colonne 7, ligne 2
   lcd.print (adr6cv);
   lcd.setCursor(8,1);//Place le curseur colonne 8, ligne 2
   lcd.print (adr5cv);
   lcd.setCursor(9,1);//Place le curseur colonne 9, ligne 2
   lcd.print (adr4cv);
   lcd.setCursor(10,1);//Place le curseur colonne 10, ligne 2
   lcd.print (adr3cv);
   lcd.setCursor(11,1);//Place le curseur colonne 11, ligne 2
   lcd.print (adr2cv);
   lcd.setCursor(12,1);//Place le curseur colonne 12, ligne 2
   lcd.print (adr1cv);
   lcd.setCursor(13,1);//Place le curseur colonne 13, ligne 2
   lcd.print (adr0cv);

lcd.setCursor (0,2);
lcd.print ("Donnee saisie : ");
lcd.print(nbre_cv2);

//Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   adr7cv2 = bitRead (adresse_cv2, 7);
   adr6cv2 = bitRead (adresse_cv2, 6);
   adr5cv2 = bitRead (adresse_cv2, 5);
   adr4cv2 = bitRead (adresse_cv2, 4);
   adr3cv2 = bitRead (adresse_cv2, 3);
   adr2cv2 = bitRead (adresse_cv2, 2);
   adr1cv2 = bitRead (adresse_cv2, 1);
   adr0cv2 = bitRead (adresse_cv2, 0);

   lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
   lcd.print (adr7cv2);
```

```
   lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
   lcd.print (adr6cv2);
   lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
   lcd.print (adr5cv2);
   lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
   lcd.print (adr4cv2);
   lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
   lcd.print (adr3cv2);
   lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
   lcd.print (adr2cv2);
   lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
   lcd.print (adr1cv2);
   lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
   lcd.print (adr0cv2);

lcd.noBlink();

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (2,0);
lcd.print ("Numero de la Cv : ");
lcd.setCursor (9,1);
lcd.blink();//Permet le clignotement du curseur

//Gestion des claviers
autorise_touche_Cv4 = 1;

}//fin du if (nbre_cv2 < 257)

if (compteur_touche_cv2 > 3 || nbre_cv2 > 255)
{
   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_cv2 = 0;//Remet le compteur à 0 pour prochaine saisie
```

```
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(0,0);//Place le curseur colonne 0, ligne 1

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
   lcd.print("Adresse Trop Grande") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 2
   lcd.print("doit etre comprise") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 3
   lcd.print("Entre 0 et 255") ; // Affiche la chaîne texte

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

   }//(compteur_touche_cv2 > 3 || nbre_cv2 > 255

}//fin if (compteur_touche_cv2 == 3)

}//Fin du VOID TRAITEMENT DES TOUCHES APPUYEES CV2

////////////////////////////////////////////////////////////////////////////////
////////////////////////// ////////////////////////////////////////////////////
////////////////////// MISE EN MEMOIRE DES TOUCHES APPUYEES CV4////////////////////////
//////////////////////////////TOUCHE APPUYEE NUMERIQUE CV4//////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_cv4() //Stockage touche appuyée
{
  compteur_touche_cv4 = compteur_touche_cv4 + 1;//Incrémente la variable compteur_touche

if (compteur_touche_cv4 == 1) //Teste si compteur_touche = 1
{toucheAcv4 = touche;
lcd.blink();

autorise_touche_Cv4_e = 1;//Autorise la prise en compte par la touche E
```

```
}//Garde trace de la touche 1ère touche appuyée

if (compteur_touche_cv4 == 2) //Teste si compteur_touche = 2
{toucheBcv4 = touche;
lcd.blink();}//Garde trace de la touche 2ième touche appuyée

if (compteur_touche_cv4 == 3) //Teste si compteur_touche = 3
{toucheCcv4 = touche;
lcd.blink();}//Garde trace de la touche 3ième touche appuyée

if (compteur_touche_cv4 == 4) //Teste si compteur_touche = 3
{toucheDcv4 = touche;//Garde trace de la touche 4ième touche appuyée
lcd.noBlink ();

autorise_touche_Cv4 = 0;//Empêche la saisie de chiffre supplémentaire

}
}//Fin du VOID TOUCHE APPUYEE NUMERIQUE CV4

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////// TRAITEMENT DES TOUCHES APPUYEES PROGRAMMATION CV4////////
///////////////////////////////////TRAITEMENT APPUI TOUCHE//////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_cv4()//Correspond à Appui touche 'E' et autorise_touche_cv4 = 1
{
 if (compteur_touche_cv4 == 1)//Prise en compte  si 1 chiffre saisi

{
   nbre_cv4 = (toucheAcv4-48);//nbre contient l'intégralité de la saisie

comm1cv = 0;//Pour adressage des 256 premières Cv
comm0cv = 0;//Pour adressage des 256 premières Cv

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_cv4 = 0;//Remet le compteur à 0 pour prochaine saisie

   adresse_cv4 = nbre_cv4-1;//recopie dans adresse la valeur de nbre en binaire
```

```
 //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   adr7cv4 = bitRead (adresse_cv4, 7);
   adr6cv4 = bitRead (adresse_cv4, 6);
   adr5cv4 = bitRead (adresse_cv4, 5);
   adr4cv4 = bitRead (adresse_cv4, 4);
   adr3cv4 = bitRead (adresse_cv4, 3);
   adr2cv4 = bitRead (adresse_cv4, 2);
   adr1cv4 = bitRead (adresse_cv4, 1);
   adr0cv4 = bitRead (adresse_cv4, 0);

//Affichage octet d'adressage des Cv et deux premiers bytes de l'adresse de la Cv en binaire
lcd.setCursor(1,2);//Place le curseur colonne 2, ligne 3
   lcd.print (comm7cv);
   lcd.setCursor(2,2);//Place le curseur colonne 3, ligne 3
   lcd.print (comm6cv);
   lcd.setCursor(3,2);//Place le curseur colonne 4, ligne 3
   lcd.print (comm5cv);
   lcd.setCursor(4,2);//Place le curseur colonne 5, ligne 3
   lcd.print (comm4cv);
   lcd.setCursor(5,2);//Place le curseur colonne 6, ligne 3
   lcd.print (comm3cv);
   lcd.setCursor(6,2);//Place le curseur colonne 7, ligne 3
   lcd.print (comm2cv);
   lcd.setCursor(7,2);//Place le curseur colonne 8, ligne 3
   lcd.print (comm1cv);
   lcd.setCursor(8,2);//Place le curseur colonne 9, ligne 3
   lcd.print (comm0cv);

//Affiche le numéro du reste de la Cv en binaire
   lcd.setCursor(11,2);//Place le curseur colonne 12, ligne 3
   lcd.print (adr7cv4);
   lcd.setCursor(12,2);//Place le curseur colonne 13, ligne 3
   lcd.print (adr6cv4);
   lcd.setCursor(13,2);//Place le curseur colonne 14, ligne 3
   lcd.print (adr5cv4);
   lcd.setCursor(14,2);//Place le curseur colonne 15, ligne 3
   lcd.print (adr4cv4);
   lcd.setCursor(15,2);//Place le curseur colonne 16, ligne 3
   lcd.print (adr3cv4);
```

```
   lcd.setCursor(16,2);//Place le curseur colonne 17, ligne 3
   lcd.print (adr2cv4);
   lcd.setCursor(17,2);//Place le curseur colonne 18, ligne 3
   lcd.print (adr1cv4);
   lcd.setCursor(18,2);//Place le curseur colonne 19, ligne 3
   lcd.print (adr0cv4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_cv();

}// fin du if (compteur_touche_cv4 == 1)
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (compteur_touche_cv4 == 2)
{

  nbre_cv4 = ((toucheAcv4-48) * 10) + (toucheBcv4-48);
```

```
  //nbre contient l'intégralité de la saisie

comm1cv = 0;//Pour adressage des 256 premières Cv
comm0cv = 0;//Pour adressage des 256 premières Cv

    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_cv4 = 0;//Remet le compteur à 0 pour prochaine saisie

    adresse_cv4 = nbre_cv4-1;//recopie dans adresse la valeur de nbre en binaire


 //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv4 = bitRead (adresse_cv4, 7);
    adr6cv4 = bitRead (adresse_cv4, 6);
    adr5cv4 = bitRead (adresse_cv4, 5);
    adr4cv4 = bitRead (adresse_cv4, 4);
    adr3cv4 = bitRead (adresse_cv4, 3);
    adr2cv4 = bitRead (adresse_cv4, 2);
    adr1cv4 = bitRead (adresse_cv4, 1);
    adr0cv4 = bitRead (adresse_cv4, 0);

//Affichage octet d'adressage des Cv et deux premiers bytes de l'adresse de la Cv en binaire
lcd.setCursor(1,2);//Place le curseur colonne 2, ligne 3
    lcd.print (comm7cv);
    lcd.setCursor(2,2);//Place le curseur colonne 3, ligne 3
    lcd.print (comm6cv);
    lcd.setCursor(3,2);//Place le curseur colonne 4, ligne 3
    lcd.print (comm5cv);
    lcd.setCursor(4,2);//Place le curseur colonne 5, ligne 3
    lcd.print (comm4cv);
    lcd.setCursor(5,2);//Place le curseur colonne 6, ligne 3
    lcd.print (comm3cv);
    lcd.setCursor(6,2);//Place le curseur colonne 7, ligne 3
    lcd.print (comm2cv);
    lcd.setCursor(7,2);//Place le curseur colonne 8, ligne 3
    lcd.print (comm1cv);
    lcd.setCursor(8,2);//Place le curseur colonne 9, ligne 3
    lcd.print (comm0cv);
```

```
//Affiche le numéro du reste de la Cv en binaire
   lcd.setCursor(11,2);//Place le curseur colonne 12, ligne 3
   lcd.print (adr7cv4);
   lcd.setCursor(12,2);//Place le curseur colonne 13, ligne 3
   lcd.print (adr6cv4);
   lcd.setCursor(13,2);//Place le curseur colonne 14, ligne 3
   lcd.print (adr5cv4);
   lcd.setCursor(14,2);//Place le curseur colonne 15, ligne 3
   lcd.print (adr4cv4);
   lcd.setCursor(15,2);//Place le curseur colonne 16, ligne 3
   lcd.print (adr3cv4);
   lcd.setCursor(16,2);//Place le curseur colonne 17, ligne 3
   lcd.print (adr2cv4);
   lcd.setCursor(17,2);//Place le curseur colonne 18, ligne 3
   lcd.print (adr1cv4);
   lcd.setCursor(18,2);//Place le curseur colonne 19, ligne 3
   lcd.print (adr0cv4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for
```

```
envoi_trame_programmation_cv();



}//fin du if (compteur_touche_cv4 == 2)

////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (compteur_touche_cv4 == 3)
{
  nbre_cv4 = ((toucheAcv4-48) * 100) + ((toucheBcv4-48)*10) + (toucheCcv4-48);
  //nbre contient l'intégralité de la saisie}

if (nbre_cv4 < 256)
    {

comm1cv = 0;//Pour adressage des 256 premières Cv
comm0cv = 0;//Pour adressage des 256 premières Cv

    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_cv4 = 0;//Remet le compteur à 0 pour prochaine saisie

    adresse_cv4 = nbre_cv4-1;//recopie dans adresse la valeur de nbre en binaire

  //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv4 = bitRead (adresse_cv4, 7);
    adr6cv4 = bitRead (adresse_cv4, 6);
    adr5cv4 = bitRead (adresse_cv4, 5);
    adr4cv4 = bitRead (adresse_cv4, 4);
    adr3cv4 = bitRead (adresse_cv4, 3);
    adr2cv4 = bitRead (adresse_cv4, 2);
    adr1cv4 = bitRead (adresse_cv4, 1);
    adr0cv4 = bitRead (adresse_cv4, 0);

//Affichage octet d'adressage des Cv et deux premiers bytes de l'adresse de la Cv en binaire
lcd.setCursor(1,2);//Place le curseur colonne 2, ligne 3
    lcd.print (comm7cv);
    lcd.setCursor(2,2);//Place le curseur colonne 3, ligne 3
```

```
   lcd.print (comm6cv);
   lcd.setCursor(3,2);//Place le curseur colonne 4, ligne 3
   lcd.print (comm5cv);
   lcd.setCursor(4,2);//Place le curseur colonne 5, ligne 3
   lcd.print (comm4cv);
   lcd.setCursor(5,2);//Place le curseur colonne 6, ligne 3
   lcd.print (comm3cv);
   lcd.setCursor(6,2);//Place le curseur colonne 7, ligne 3
   lcd.print (comm2cv);
   lcd.setCursor(7,2);//Place le curseur colonne 8, ligne 3
   lcd.print (comm1cv);
   lcd.setCursor(8,2);//Place le curseur colonne 9, ligne 3
   lcd.print (comm0cv);

//Affiche le numéro du reste de la Cv en binaire
   lcd.setCursor(11,2);//Place le curseur colonne 12, ligne 3
   lcd.print (adr7cv4);
   lcd.setCursor(12,2);//Place le curseur colonne 13, ligne 3
   lcd.print (adr6cv4);
   lcd.setCursor(13,2);//Place le curseur colonne 14, ligne 3
   lcd.print (adr5cv4);
   lcd.setCursor(14,2);//Place le curseur colonne 15, ligne 3
   lcd.print (adr4cv4);
   lcd.setCursor(15,2);//Place le curseur colonne 16, ligne 3
   lcd.print (adr3cv4);
   lcd.setCursor(16,2);//Place le curseur colonne 17, ligne 3
   lcd.print (adr2cv4);
   lcd.setCursor(17,2);//Place le curseur colonne 18, ligne 3
   lcd.print (adr1cv4);
   lcd.setCursor(18,2);//Place le curseur colonne 19, ligne 3
   lcd.print (adr0cv4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for
```

```
lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_cv();

}//fin du if (compteur_touche_cv4 <256)
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (nbre_cv4 > 255 && nbre_cv4 < 512)
   {

comm1cv = 0;//Pour adressage entre 256 à 511 Cv
comm0cv = 1;//Pour adressage entre 256 à 511 Cv

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_cv4 = 0;//Remet le compteur à 0 pour prochaine saisie

   adresse_cv4 = nbre_cv4-1;//recopie dans adresse la valeur de nbre en binaire

  //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   adr7cv4 = bitRead (adresse_cv4, 7);
   adr6cv4 = bitRead (adresse_cv4, 6);
   adr5cv4 = bitRead (adresse_cv4, 5);
   adr4cv4 = bitRead (adresse_cv4, 4);
   adr3cv4 = bitRead (adresse_cv4, 3);
   adr2cv4 = bitRead (adresse_cv4, 2);
```

```
    adr1cv4 = bitRead (adresse_cv4, 1);
    adr0cv4 = bitRead (adresse_cv4, 0);

//Affichage octet d'adressage des Cv et deux premiers bytes
//de l'adresse de la Cv en binaire
lcd.setCursor(1,2);//Place le curseur colonne 2, ligne 3
    lcd.print (comm7cv);
    lcd.setCursor(2,2);//Place le curseur colonne 3, ligne 3
    lcd.print (comm6cv);
    lcd.setCursor(3,2);//Place le curseur colonne 4, ligne 3
    lcd.print (comm5cv);
    lcd.setCursor(4,2);//Place le curseur colonne 5, ligne 3
    lcd.print (comm4cv);
    lcd.setCursor(5,2);//Place le curseur colonne 6, ligne 3
    lcd.print (comm3cv);
    lcd.setCursor(6,2);//Place le curseur colonne 7, ligne 3
    lcd.print (comm2cv);
    lcd.setCursor(7,2);//Place le curseur colonne 8, ligne 3
    lcd.print (comm1cv);
    lcd.setCursor(8,2);//Place le curseur colonne 9, ligne 3
    lcd.print (comm0cv);

//Affiche le numéro du reste de la Cv en binaire
    lcd.setCursor(11,2);//Place le curseur colonne 12, ligne 3
    lcd.print (adr7cv4);
    lcd.setCursor(12,2);//Place le curseur colonne 13, ligne 3
    lcd.print (adr6cv4);
    lcd.setCursor(13,2);//Place le curseur colonne 14, ligne 3
    lcd.print (adr5cv4);
    lcd.setCursor(14,2);//Place le curseur colonne 15, ligne 3
    lcd.print (adr4cv4);
    lcd.setCursor(15,2);//Place le curseur colonne 16, ligne 3
    lcd.print (adr3cv4);
    lcd.setCursor(16,2);//Place le curseur colonne 17, ligne 3
    lcd.print (adr2cv4);
    lcd.setCursor(17,2);//Place le curseur colonne 18, ligne 3
    lcd.print (adr1cv4);
    lcd.setCursor(18,2);//Place le curseur colonne 19, ligne 3
    lcd.print (adr0cv4);

lcd.noBlink();//Empêche le clignotement du curseur
```

```
for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_cv();

}//fin du if (nbre_cv4 > 255 && nbre_cv4 < 512)
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
if (nbre_cv4 > 511 && nbre_cv4 < 768)
   {

comm1cv = 1;//Pour adressage entre 512 à 767 Cv
comm0cv = 0;//Pour adressage entre 512 à 767 Cv

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_cv4 = 0;//Remet le compteur à 0 pour prochaine saisie

   adresse_cv4 = nbre_cv4-1;//recopie dans adresse la valeur de nbre en binaire
```

```
  //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv4 = bitRead (adresse_cv4, 7);
    adr6cv4 = bitRead (adresse_cv4, 6);
    adr5cv4 = bitRead (adresse_cv4, 5);
    adr4cv4 = bitRead (adresse_cv4, 4);
    adr3cv4 = bitRead (adresse_cv4, 3);
    adr2cv4 = bitRead (adresse_cv4, 2);
    adr1cv4 = bitRead (adresse_cv4, 1);
    adr0cv4 = bitRead (adresse_cv4, 0);

//Affichage octet d'adressage des Cv et deux premiers bytes de l'adresse de la Cv en binaire
lcd.setCursor(1,2);//Place le curseur colonne 2, ligne 3
    lcd.print (comm7cv);
    lcd.setCursor(2,2);//Place le curseur colonne 3, ligne 3
    lcd.print (comm6cv);
    lcd.setCursor(3,2);//Place le curseur colonne 4, ligne 3
    lcd.print (comm5cv);
    lcd.setCursor(4,2);//Place le curseur colonne 5, ligne 3
    lcd.print (comm4cv);
    lcd.setCursor(5,2);//Place le curseur colonne 6, ligne 3
    lcd.print (comm3cv);
    lcd.setCursor(6,2);//Place le curseur colonne 7, ligne 3
    lcd.print (comm2cv);
    lcd.setCursor(7,2);//Place le curseur colonne 8, ligne 3
    lcd.print (comm1cv);
    lcd.setCursor(8,2);//Place le curseur colonne 9, ligne 3
    lcd.print (comm0cv);

//Affiche le numéro du reste de la Cv en binaire
    lcd.setCursor(11,2);//Place le curseur colonne 12, ligne 3
    lcd.print (adr7cv4);
    lcd.setCursor(12,2);//Place le curseur colonne 13, ligne 3
    lcd.print (adr6cv4);
    lcd.setCursor(13,2);//Place le curseur colonne 14, ligne 3
    lcd.print (adr5cv4);
    lcd.setCursor(14,2);//Place le curseur colonne 15, ligne 3
    lcd.print (adr4cv4);
    lcd.setCursor(15,2);//Place le curseur colonne 16, ligne 3
    lcd.print (adr3cv4);
    lcd.setCursor(16,2);//Place le curseur colonne 17, ligne 3
```

```
   lcd.print (adr2cv4);
   lcd.setCursor(17,2);//Place le curseur colonne 18, ligne 3
   lcd.print (adr1cv4);
   lcd.setCursor(18,2);//Place le curseur colonne 19, ligne 3
   lcd.print (adr0cv4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_cv();

}//fin du if (nbre_cv4 > 511 && nbre_cv4 < 768)//3 chiffres
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (nbre_cv4 > 767 && nbre_cv4 < 1000)
   {

comm1cv = 1;//Pour adressage entre 769 à 999 Cv
comm0cv = 1;//Pour adressage entre 769 à 999 Cv
```

```
    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_cv4 = 0;//Remet le compteur à 0 pour prochaine saisie

    adresse_cv4 = nbre_cv4-1;//recopie dans adresse la valeur de nbre en binaire

  //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv4 = bitRead (adresse_cv4, 7);
    adr6cv4 = bitRead (adresse_cv4, 6);
    adr5cv4 = bitRead (adresse_cv4, 5);
    adr4cv4 = bitRead (adresse_cv4, 4);
    adr3cv4 = bitRead (adresse_cv4, 3);
    adr2cv4 = bitRead (adresse_cv4, 2);
    adr1cv4 = bitRead (adresse_cv4, 1);
    adr0cv4 = bitRead (adresse_cv4, 0);

//Affichage octet d'adressage des Cv et deux premiers bytes
//de l'adresse de la Cv en binaire
lcd.setCursor(1,2);//Place le curseur colonne 2, ligne 3
    lcd.print (comm7cv);
    lcd.setCursor(2,2);//Place le curseur colonne 3, ligne 3
    lcd.print (comm6cv);
    lcd.setCursor(3,2);//Place le curseur colonne 4, ligne 3
    lcd.print (comm5cv);
    lcd.setCursor(4,2);//Place le curseur colonne 5, ligne 3
    lcd.print (comm4cv);
    lcd.setCursor(5,2);//Place le curseur colonne 6, ligne 3
    lcd.print (comm3cv);
    lcd.setCursor(6,2);//Place le curseur colonne 7, ligne 3
    lcd.print (comm2cv);
    lcd.setCursor(7,2);//Place le curseur colonne 8, ligne 3
    lcd.print (comm1cv);
    lcd.setCursor(8,2);//Place le curseur colonne 9, ligne 3
    lcd.print (comm0cv);

//Affiche le numéro du reste de la Cv en binaire
    lcd.setCursor(11,2);//Place le curseur colonne 12, ligne 3
    lcd.print (adr7cv4);
    lcd.setCursor(12,2);//Place le curseur colonne 13, ligne 3
```

```
   lcd.print (adr6cv4);
   lcd.setCursor(13,2);//Place le curseur colonne 14, ligne 3
   lcd.print (adr5cv4);
   lcd.setCursor(14,2);//Place le curseur colonne 15, ligne 3
   lcd.print (adr4cv4);
   lcd.setCursor(15,2);//Place le curseur colonne 16, ligne 3
   lcd.print (adr3cv4);
   lcd.setCursor(16,2);//Place le curseur colonne 17, ligne 3
   lcd.print (adr2cv4);
   lcd.setCursor(17,2);//Place le curseur colonne 18, ligne 3
   lcd.print (adr1cv4);
   lcd.setCursor(18,2);//Place le curseur colonne 19, ligne 3
   lcd.print (adr0cv4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_cv();

}//fin du if (nbre_cv4 > 767 && nbre_cv4 < 1000)
```

```
}//Fin du If = 3

//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
if (compteur_touche_cv4 == 4)
{
  nbre_cv4 = ((toucheAcv4-48) * 1000) + ((toucheBcv4-48)*100)
  + ((toucheCcv4-48)*10) + (toucheDcv4-48);
  //nbre contient l'intégralité de la saisie}



if (nbre_cv4 > 999 && nbre_cv4 < 1024)
    {

comm1cv = 1;//Pour adressage entre 1000 à 1023 Cv
comm0cv = 1;//Pour adressage entre 1000 à 1023 Cv

    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"


    compteur_touche_cv4 = 0;//Remet le compteur à 0 pour prochaine saisie


    adresse_cv4 = nbre_cv4-1;//recopie dans adresse la valeur de nbre en binaire
    //moins 1 (protocole DCC)

  //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7cv4 = bitRead (adresse_cv4, 7);
    adr6cv4 = bitRead (adresse_cv4, 6);
    adr5cv4 = bitRead (adresse_cv4, 5);
    adr4cv4 = bitRead (adresse_cv4, 4);
    adr3cv4 = bitRead (adresse_cv4, 3);
    adr2cv4 = bitRead (adresse_cv4, 2);
    adr1cv4 = bitRead (adresse_cv4, 1);
    adr0cv4 = bitRead (adresse_cv4, 0);

//Affichage octet d'adressage des Cv et deux premiers bytes de l'adresse de la Cv en binaire
lcd.setCursor(1,2);//Place le curseur colonne 2, ligne 3
    lcd.print (comm7cv);
    lcd.setCursor(2,2);//Place le curseur colonne 3, ligne 3
```

```
   lcd.print (comm6cv);
   lcd.setCursor(3,2);//Place le curseur colonne 4, ligne 3
   lcd.print (comm5cv);
   lcd.setCursor(4,2);//Place le curseur colonne 5, ligne 3
   lcd.print (comm4cv);
   lcd.setCursor(5,2);//Place le curseur colonne 6, ligne 3
   lcd.print (comm3cv);
   lcd.setCursor(6,2);//Place le curseur colonne 7, ligne 3
   lcd.print (comm2cv);
   lcd.setCursor(7,2);//Place le curseur colonne 8, ligne 3
   lcd.print (comm1cv);
   lcd.setCursor(8,2);//Place le curseur colonne 9, ligne 3
   lcd.print (comm0cv);

//Affiche le numéro du reste de la Cv en binaire
   lcd.setCursor(11,2);//Place le curseur colonne 12, ligne 3
   lcd.print (adr7cv4);
   lcd.setCursor(12,2);//Place le curseur colonne 13, ligne 3
   lcd.print (adr6cv4);
   lcd.setCursor(13,2);//Place le curseur colonne 14, ligne 3
   lcd.print (adr5cv4);
   lcd.setCursor(14,2);//Place le curseur colonne 15, ligne 3
   lcd.print (adr4cv4);
   lcd.setCursor(15,2);//Place le curseur colonne 16, ligne 3
   lcd.print (adr3cv4);
   lcd.setCursor(16,2);//Place le curseur colonne 17, ligne 3
   lcd.print (adr2cv4);
   lcd.setCursor(17,2);//Place le curseur colonne 18, ligne 3
   lcd.print (adr1cv4);
   lcd.setCursor(18,2);//Place le curseur colonne 19, ligne 3
   lcd.print (adr0cv4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for
```

```
lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_cv();

}//fin du if (nbre_cv4 > 999 && nbre_cv4 < 1024)
}//Fin du If = 4
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
if (compteur_touche_cv4 > 4 || nbre_cv4 > 1023)//Dépassement du nombre de Cv
{

   compteur_touche_cv2 = 0; //Remet le compteur à 0 pour prochaine saisie

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_cv4 = 0;//Remet le compteur à 0 pour prochaine saisie

   lcd.clear(); // Efface écran si appui

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   lcd.setCursor(4,0);//Place le curseur colonne 0, ligne 1
   lcd.print("Numero de Cv") ; // Affiche la chaîne texte

   lcd.setCursor(4,1);//Place le curseur colonne 0, ligne 2
   lcd.print("Trop Grande") ; // Affiche la chaîne texte
```

```
   lcd.setCursor(1,2);//Place le curseur colonne 0, ligne 3
   lcd.print("doit etre comprise") ; // Affiche la chaîne texte

   lcd.setCursor(2,3);//Place le curseur colonne 0, ligne 4
   lcd.print("Entre 1 et 1023") ; // Affiche la chaîne texte

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Permet une temporisation

   {
    trame_idle();//Envoi trame idle
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//fin if (compteur_touche_cv4 > 4 || nbre_cv4 > 1023)

}//Fin du VOID TRAITEMENT DES TOUCHES APPUYEES
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////PROGRAMMATION ADRESSE ACCESSOIRE DEPART////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void programmation_accessoire_depart ()
{
retourdetrame = 0;

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   autorise_touche_programmation_accessoire = 1;
   //Interdit la saisie de chiffre pour programmation_accessoire à 0 pour prise en compte nouvel appui

   traitement_appui_touches_programmation_accessoire = 1;
   //Interdit la rentrée dans la boucle traitement_appui_touches_programmation_accessoire quand appui sur 'E'

   lcd.clear(); // Efface écran si appui = sinon  affiche touche
   lcd.setCursor(7,0);//Place le curseur colonne 7, ligne 1
   lcd.print ("ENTRER     P");
```

```
  lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
  lcd.print ("L'ADRESSE de");

  lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
  lcd.print ("L'ACCESSOIRE");

  lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4

  lcd.blink();//Autorise le clignotement du curseur

  //RAZ sélection loc
  locSelectionne = 0;//Empêche l'envoi de la trame locomotive et une erreur d'affichage
  loc2Selectionne = 0;//Empêche l'envoi de la trame locomotive2 et une erreur d'affichage


}//fin void programmation_accessoire_depart ()

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////////PROGRAMMATION MEMORISATION LOC/////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void memorisation_numero_loc ()
{
  retourdetrame = 0;

  prog_memo_loc = 0;//Autorise les accès direct des menus par les touches A, B, C, D

  envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

  autorise_touche_memorisation_loc = 1;
  //Autorise l'entrée et l'affichage de nombres dans la boucle mémorisation

  traitement_appui_touches_memorisation_loc = 1;
  //Permet la rentrée dans la boucle traitement_appui_touches_programmation_accessoire4 quand appui sur 'E'

  lcd.clear(); // Efface écran si appui = sinon  affiche touche
  lcd.setCursor(7,0);//Place le curseur colonne 7, ligne 1
  lcd.print ("ENTRER     ");

  lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
```

```
  lcd.print ("le numero de");

  lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
  lcd.print ("La locomotive");

  lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4

  lcd.blink();//Autorise le clignotement du curseur


  //RAZ sélection loc
  locSelectionne = 0;//Empêche l'envoi de la trame locomotive et une erreur d'affichage
  loc2Selectionne = 0;//Empêche l'envoi de la trame locomotive2 et une erreur d'affichage


}//fin void memorisation_numero_loc ()
////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////// ///////////////////////////////////////////////////////////////
//////////////MISE EN MEMOIRE DES TOUCHES APPUYEES PROGRAMMATION ACCESSOIRE//////////////////////
////////////////////////////////TOUCHE APPUYEE NUMERIQUE//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_programmation_accessoire() //Stockage touche appuyée
{
compteur_touche_programmation_accessoire = compteur_touche_programmation_accessoire + 1;
//Incrémente la variable compteur_touche

if (compteur_touche_programmation_accessoire == 1) //Teste si compteur_touche = 1
{toucheAprogrammation_accessoire = touche;
lcd.blink ();

autorise_touche_programmation_accessoire_e = 1;//Autorise la prise en compte par la touche E

}//Garde trace de la touche 1ere touche appuyée
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire == 2) //Teste si compteur_touche = 2
{toucheBprogrammation_accessoire = touche;
lcd.blink ();}//Garde trace de la touche 1ere touche appuyée
////////////////////////////////////////////////
```

```
////////////////////////////////////////////////
////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire == 3) //Teste si compteur_touche = 3
{toucheCprogrammation_accessoire = touche;//Garde trace de la touche 3ième touche appuyée

lcd.noBlink ();

autorise_touche_programmation_accessoire = 0;//Empêche la saisie de chiffre supplémentaire

}
}//Fin du VOID MISE EN MEMOIRE DES TOUCHES APPUYEES

////////////////////////////////////////////////////////////////////////////////////
////////////////////// TRAITEMENT DES TOUCHES APPUYEES PROGRAMMATION ACCESSOIRE/////////////////
//////////////////////////////TRAITEMENT APPUI TOUCHE/////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_programmation_accessoire()
//Correspond à Appui touche 'E' et autorise_touche_programmation_accessoire
{

 if (compteur_touche_programmation_accessoire == 1)//Prise en compte  si 1 chiffre saisi
 {
    nbre_programmation_accessoire = (toucheAprogrammation_accessoire-48);
     //nbre contient l'intégralité de la saisie

if (nbre_programmation_accessoire == 0)//Prise en compte  si 1 chiffre saisi
 {
  lcd.clear(); // Efface écran si appui
    lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

    lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

    lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

    lcd.print("doit etre compris") ; // Affiche la chaîne texte

    lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

    lcd.print("entre 1 et 510") ; // Affiche la chaîne texte

    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
```

```
  lcd.noBlink();//Empêche le clignotement du curseur

  for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
  //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
  //et permet une temporisation
  {
   trame_idle();
  }//fermeture du for

  retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_programmation_accessoire == 0)

else
{

  envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

  compteur_touche_programmation_accessoire = 0;//Remet le compteur à 0 pour prochaine saisie

  adresse_programmation_accessoire = nbre_programmation_accessoire;
  //recopie dans adresse la valeur de nbre en binaire

  lcd.setCursor(9,2);//Place le curseur colonne 9, ligne 3 comme dans
  //choix locomotive pour que les chiffres se superposent


  //Transforme la variable adresse en binaire et stocke dans les variables adr
  //Octet 1 d'adresse
  adr7programmation_accessoire = 1;//ACCESSOIRE
  adr6programmation_accessoire = 0;//ACCESSOIRE
  adr5programmation_accessoire = bitRead (adresse_programmation_accessoire, 5);
  adr4programmation_accessoire = bitRead (adresse_programmation_accessoire, 4);
  adr3programmation_accessoire = bitRead (adresse_programmation_accessoire, 3);
  adr2programmation_accessoire = bitRead (adresse_programmation_accessoire, 2);
  adr1programmation_accessoire = bitRead (adresse_programmation_accessoire, 1);
  adr0programmation_accessoire = bitRead (adresse_programmation_accessoire, 0);

  //Octet 2 d'adresse
  adr27programmation_accessoire = 1;//ACCESSOIRE
```

```
  adr26programmation_accessoire = bitRead (adresse_programmation_accessoire, 8);//ACCESSOIRE complément à 1
  adr25programmation_accessoire = bitRead (adresse_programmation_accessoire, 7);//ACCESSOIRE complément à 1
  adr24programmation_accessoire = bitRead (adresse_programmation_accessoire, 6);//ACCESSOIRE complément à 1
  adr23programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV
  adr22programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV
  adr21programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV
  adr20programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV

//Affiche le nombre adresse de départ en binaire

  lcd.setCursor(1,3);//Place le curseur colonne 6, ligne 4
  lcd.print (adr7programmation_accessoire);
  lcd.setCursor(2,3);//Place le curseur colonne 7, ligne 4
  lcd.print (adr6programmation_accessoire);
  lcd.setCursor(3,3);//Place le curseur colonne 8, ligne 4
  lcd.print (adr5programmation_accessoire);
  lcd.setCursor(4,3);//Place le curseur colonne 9, ligne 4
  lcd.print (adr4programmation_accessoire);
  lcd.setCursor(5,3);//Place le curseur colonne 10, ligne 4
  lcd.print (adr3programmation_accessoire);
  lcd.setCursor(6,3);//Place le curseur colonne 11, ligne 4
  lcd.print (adr2programmation_accessoire);
  lcd.setCursor(7,3);//Place le curseur colonne 12, ligne 4
  lcd.print (adr1programmation_accessoire);
  lcd.setCursor(8,3);//Place le curseur colonne 13, ligne 4
  lcd.print (adr0programmation_accessoire);

  lcd.setCursor(9,3);//Place le curseur colonne 13, ligne 4
  lcd.print (" ");//Efface le numéro d'adresse rentré

  lcd.setCursor(10,3);//Place le curseur colonne 6, ligne 4
  lcd.print (adr27programmation_accessoire);
  lcd.setCursor(11,3);//Place le curseur colonne 7, ligne 4
  lcd.print (adr26programmation_accessoire^1);
  lcd.setCursor(12,3);//Place le curseur colonne 8, ligne 4
  lcd.print (adr25programmation_accessoire^1);
  lcd.setCursor(13,3);//Place le curseur colonne 9, ligne 4
  lcd.print (adr24programmation_accessoire^1);
  lcd.setCursor(14,3);//Place le curseur colonne 10, ligne 4
  lcd.print (adr23programmation_accessoire);
  lcd.setCursor(15,3);//Place le curseur colonne 11, ligne 4
```

```
   lcd.print (adr22programmation_accessoire);
   lcd.setCursor(16,3);//Place le curseur colonne 12, ligne 4
   lcd.print (adr21programmation_accessoire);
   lcd.setCursor(17,3);//Place le curseur colonne 13, ligne 4
   lcd.print (adr20programmation_accessoire);

   lcd.noBlink();

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for
 lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_programmation_accessoire);
lcd.noBlink();

 ////Affiche le nombre adresse de départ en binaire 2ième affichage

   lcd.setCursor(1,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr7programmation_accessoire);
   lcd.setCursor(2,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr6programmation_accessoire);
   lcd.setCursor(3,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr5programmation_accessoire);
   lcd.setCursor(4,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr4programmation_accessoire);
   lcd.setCursor(5,1);//Place le curseur colonne 10, ligne 4
   lcd.print (adr3programmation_accessoire);
   lcd.setCursor(6,1);//Place le curseur colonne 11, ligne 4
   lcd.print (adr2programmation_accessoire);
   lcd.setCursor(7,1);//Place le curseur colonne 12, ligne 4
   lcd.print (adr1programmation_accessoire);
   lcd.setCursor(8,1);//Place le curseur colonne 13, ligne 4
   lcd.print (adr0programmation_accessoire);

   lcd.setCursor(10,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr27programmation_accessoire);
```

```
   lcd.setCursor(11,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr26programmation_accessoire^1);
   lcd.setCursor(12,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr25programmation_accessoire^1);
   lcd.setCursor(13,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr24programmation_accessoire^1);
   lcd.setCursor(14,1);//Place le curseur colonne 10, ligne 4
   lcd.print (adr23programmation_accessoire);
   lcd.setCursor(15,1);//Place le curseur colonne 11, ligne 4
   lcd.print (adr22programmation_accessoire);
   lcd.setCursor(16,1);//Place le curseur colonne 12, ligne 4
   lcd.print (adr21programmation_accessoire);
   lcd.setCursor(17,1);//Place le curseur colonne 13, ligne 4
   lcd.print (adr20programmation_accessoire);

  lcd.setCursor (0,2);
lcd.print ("Saisir la donnee :");
lcd.blink();
lcd.setCursor (8,3);

//Gestion des claviers
autorise_touche_programmation_accessoire2 = 1;
//Autorise la prise en compte de l'appui touche + et autorise_touche_Cv2 à 1
}//Fin du else
}// fin du if = 1
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire == 2)
{
  nbre_programmation_accessoire = ((toucheAprogrammation_accessoire-48) * 10) + (toucheBprogrammation_accessoire-48);
  //nbre contient l'intégralité de la saisie


 if (nbre_programmation_accessoire == 0)//Prise en compte  si 1 chiffre saisi
 {
  lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte
```

```
    lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

    lcd.print("doit etre compris") ; // Affiche la chaîne texte

    lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

    lcd.print("entre 1 et 510") ; // Affiche la chaîne texte

    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_programmation_accessoire == 0)

else
{

    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_programmation_accessoire = 0;//Remet le compteur à 0 pour prochaine saisie

    adresse_programmation_accessoire = nbre_programmation_accessoire;
    //recopie dans adresse la valeur de nbre en binaire

    lcd.setCursor(9,2);
    //Place le curseur colonne 9, ligne 3 comme dans choix locomotive pour que les chiffres se superposent



    //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse

    adr7programmation_accessoire = 1;//ACCESSOIRE
```

```
  adr6programmation_accessoire = 0;//ACCESSOIRE
  adr5programmation_accessoire = bitRead (adresse_programmation_accessoire, 5);
  adr4programmation_accessoire = bitRead (adresse_programmation_accessoire, 4);
  adr3programmation_accessoire = bitRead (adresse_programmation_accessoire, 3);
  adr2programmation_accessoire = bitRead (adresse_programmation_accessoire, 2);
  adr1programmation_accessoire = bitRead (adresse_programmation_accessoire, 1);
  adr0programmation_accessoire = bitRead (adresse_programmation_accessoire, 0);

  //Octet 2 d'adresse

  adr27programmation_accessoire = 1;//ACCESSOIRE
  adr26programmation_accessoire = bitRead (adresse_programmation_accessoire, 8);//ACCESSOIRE complément à 1
  adr25programmation_accessoire = bitRead (adresse_programmation_accessoire, 7);//ACCESSOIRE complément à 1
  adr24programmation_accessoire = bitRead (adresse_programmation_accessoire, 6);//ACCESSOIRE complément à 1
  adr23programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV
  adr22programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV
  adr21programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV
  adr20programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV


//Affiche le nombre adresse de départ en binaire

  lcd.setCursor(1,3);//Place le curseur colonne 6, ligne 4
  lcd.print (adr7programmation_accessoire);
  lcd.setCursor(2,3);//Place le curseur colonne 7, ligne 4
  lcd.print (adr6programmation_accessoire);
  lcd.setCursor(3,3);//Place le curseur colonne 8, ligne 4
  lcd.print (adr5programmation_accessoire);
  lcd.setCursor(4,3);//Place le curseur colonne 9, ligne 4
  lcd.print (adr4programmation_accessoire);
  lcd.setCursor(5,3);//Place le curseur colonne 10, ligne 4
  lcd.print (adr3programmation_accessoire);
  lcd.setCursor(6,3);//Place le curseur colonne 11, ligne 4
  lcd.print (adr2programmation_accessoire);
  lcd.setCursor(7,3);//Place le curseur colonne 12, ligne 4
  lcd.print (adr1programmation_accessoire);
  lcd.setCursor(8,3);//Place le curseur colonne 13, ligne 4
  lcd.print (adr0programmation_accessoire);

  lcd.setCursor(9,3);//Place le curseur colonne 13, ligne 4
  lcd.print (" ");//Efface le numéro d'adresse rentré
```

```
   lcd.setCursor(10,3);//Place le curseur colonne 6, ligne 4
   lcd.print (adr27programmation_accessoire);
   lcd.setCursor(11,3);//Place le curseur colonne 7, ligne 4
   lcd.print (adr26programmation_accessoire^1);
   lcd.setCursor(12,3);//Place le curseur colonne 8, ligne 4
   lcd.print (adr25programmation_accessoire^1);
   lcd.setCursor(13,3);//Place le curseur colonne 9, ligne 4
   lcd.print (adr24programmation_accessoire^1);
   lcd.setCursor(14,3);//Place le curseur colonne 10, ligne 4
   lcd.print (adr23programmation_accessoire);
   lcd.setCursor(15,3);//Place le curseur colonne 11, ligne 4
   lcd.print (adr22programmation_accessoire);
   lcd.setCursor(16,3);//Place le curseur colonne 12, ligne 4
   lcd.print (adr21programmation_accessoire);
   lcd.setCursor(17,3);//Place le curseur colonne 13, ligne 4
   lcd.print (adr20programmation_accessoire);

   lcd.noBlink();

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_programmation_accessoire);
lcd.noBlink();

   ////Affiche le nombre adresse de départ en binaire 2ième affichage

   lcd.setCursor(1,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr7programmation_accessoire);
   lcd.setCursor(2,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr6programmation_accessoire);
   lcd.setCursor(3,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr5programmation_accessoire);
```

```
  lcd.setCursor(4,1);//Place le curseur colonne 9, ligne 4
  lcd.print (adr4programmation_accessoire);
  lcd.setCursor(5,1);//Place le curseur colonne 10, ligne 4
  lcd.print (adr3programmation_accessoire);
  lcd.setCursor(6,1);//Place le curseur colonne 11, ligne 4
  lcd.print (adr2programmation_accessoire);
  lcd.setCursor(7,1);//Place le curseur colonne 12, ligne 4
  lcd.print (adr1programmation_accessoire);
  lcd.setCursor(8,1);//Place le curseur colonne 13, ligne 4
  lcd.print (adr0programmation_accessoire);

  lcd.setCursor(10,1);//Place le curseur colonne 6, ligne 4
  lcd.print (adr27programmation_accessoire);
  lcd.setCursor(11,1);//Place le curseur colonne 7, ligne 4
  lcd.print (adr26programmation_accessoire^1);
  lcd.setCursor(12,1);//Place le curseur colonne 8, ligne 4
  lcd.print (adr25programmation_accessoire^1);
  lcd.setCursor(13,1);//Place le curseur colonne 9, ligne 4
  lcd.print (adr24programmation_accessoire^1);
  lcd.setCursor(14,1);//Place le curseur colonne 10, ligne 4
  lcd.print (adr23programmation_accessoire);
  lcd.setCursor(15,1);//Place le curseur colonne 11, ligne 4
  lcd.print (adr22programmation_accessoire);
  lcd.setCursor(16,1);//Place le curseur colonne 12, ligne 4
  lcd.print (adr21programmation_accessoire);
  lcd.setCursor(17,1);//Place le curseur colonne 13, ligne 4
  lcd.print (adr20programmation_accessoire);


 lcd.setCursor (0,2);
lcd.print ("Saisir la donnee :");
lcd.blink();
lcd.setCursor (8,3);

//Gestion des claviers
autorise_touche_programmation_accessoire2 = 1;
//Autorise la prise en compte de l'appui touche + et autorise_touche_programmation_accessoire2 à 1



}//Fin du else
}//fin du if compteur touche = 2
```

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

if (compteur_touche_programmation_accessoire == 3)
{
  nbre_programmation_accessoire = ((toucheAprogrammation_accessoire-48) * 100) + ((toucheBprogrammation_accessoire-48)*10)
  + toucheCprogrammation_accessoire-48;
  //nbre contient l'intégralité de la saisie


 if (nbre_programmation_accessoire == 0 || nbre_programmation_accessoire > 510)//Prise en compte  si 1 chiffre saisi
 {
  lcd.clear(); // Efface écran si appui
    lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

    lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

    lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

    lcd.print("doit etre compris") ; // Affiche la chaîne texte

    lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

    lcd.print("entre 1 et 510") ; // Affiche la chaîne texte

    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_programmation_accessoire == 0)
```

```
else
{

  envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

  compteur_touche_programmation_accessoire = 0;//Remet le compteur à 0 pour prochaine saisie

  adresse_programmation_accessoire = nbre_programmation_accessoire;
  //recopie dans adresse la valeur de nbre en binaire

  lcd.setCursor(9,2);
  //Place le curseur colonne 9, ligne 3 comme dans choix locomotive pour que les chiffres se superposent


  //Transforme la variable adresse en binaire et stocke dans les variables adr
  //Octet 1 d'adresse

  adr7programmation_accessoire = 1;//ACCESSOIRE
  adr6programmation_accessoire = 0;//ACCESSOIRE
  adr5programmation_accessoire = bitRead (adresse_programmation_accessoire, 5);
  adr4programmation_accessoire = bitRead (adresse_programmation_accessoire, 4);
  adr3programmation_accessoire = bitRead (adresse_programmation_accessoire, 3);
  adr2programmation_accessoire = bitRead (adresse_programmation_accessoire, 2);
  adr1programmation_accessoire = bitRead (adresse_programmation_accessoire, 1);
  adr0programmation_accessoire = bitRead (adresse_programmation_accessoire, 0);

  //Octet 2 d'adresse

  adr27programmation_accessoire = 1;//ACCESSOIRE
  adr26programmation_accessoire = bitRead (adresse_programmation_accessoire, 8);//ACCESSOIRE complément à 1
  adr25programmation_accessoire = bitRead (adresse_programmation_accessoire, 7);//ACCESSOIRE complément à 1
  adr24programmation_accessoire = bitRead (adresse_programmation_accessoire, 6);//ACCESSOIRE complément à 1
  adr23programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV
  adr22programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV
  adr21programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV
  adr20programmation_accessoire = 0;//ACCESSOIRE Permet d'adresser toutes les CV


//Affiche le nombre adresse de départ en binaire
```

```
  lcd.setCursor(1,3);//Place le curseur colonne 6, ligne 4
  lcd.print (adr7programmation_accessoire);
  lcd.setCursor(2,3);//Place le curseur colonne 7, ligne 4
  lcd.print (adr6programmation_accessoire);
  lcd.setCursor(3,3);//Place le curseur colonne 8, ligne 4
  lcd.print (adr5programmation_accessoire);
  lcd.setCursor(4,3);//Place le curseur colonne 9, ligne 4
  lcd.print (adr4programmation_accessoire);
  lcd.setCursor(5,3);//Place le curseur colonne 10, ligne 4
  lcd.print (adr3programmation_accessoire);
  lcd.setCursor(6,3);//Place le curseur colonne 11, ligne 4
  lcd.print (adr2programmation_accessoire);
  lcd.setCursor(7,3);//Place le curseur colonne 12, ligne 4
  lcd.print (adr1programmation_accessoire);
  lcd.setCursor(8,3);//Place le curseur colonne 13, ligne 4
  lcd.print (adr0programmation_accessoire);

  lcd.setCursor(9,3);//Place le curseur colonne 13, ligne 4
  lcd.print (" ");//Efface le numéro d'adresse rentré

  lcd.setCursor(10,3);//Place le curseur colonne 6, ligne 4
  lcd.print (adr27programmation_accessoire);
  lcd.setCursor(11,3);//Place le curseur colonne 7, ligne 4
  lcd.print (adr26programmation_accessoire^1);
  lcd.setCursor(12,3);//Place le curseur colonne 8, ligne 4
  lcd.print (adr25programmation_accessoire^1);
  lcd.setCursor(13,3);//Place le curseur colonne 9, ligne 4
  lcd.print (adr24programmation_accessoire^1);
  lcd.setCursor(14,3);//Place le curseur colonne 10, ligne 4
  lcd.print (adr23programmation_accessoire);
  lcd.setCursor(15,3);//Place le curseur colonne 11, ligne 4
  lcd.print (adr22programmation_accessoire);
  lcd.setCursor(16,3);//Place le curseur colonne 12, ligne 4
  lcd.print (adr21programmation_accessoire);
  lcd.setCursor(17,3);//Place le curseur colonne 13, ligne 4
  lcd.print (adr20programmation_accessoire);

  lcd.noBlink();

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
```

```
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_programmation_accessoire);
lcd.noBlink();

   ////Affiche le nombre adresse de départ en binaire 2ième affichage

   lcd.setCursor(1,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr7programmation_accessoire);
   lcd.setCursor(2,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr6programmation_accessoire);
   lcd.setCursor(3,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr5programmation_accessoire);
   lcd.setCursor(4,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr4programmation_accessoire);
   lcd.setCursor(5,1);//Place le curseur colonne 10, ligne 4
   lcd.print (adr3programmation_accessoire);
   lcd.setCursor(6,1);//Place le curseur colonne 11, ligne 4
   lcd.print (adr2programmation_accessoire);
   lcd.setCursor(7,1);//Place le curseur colonne 12, ligne 4
   lcd.print (adr1programmation_accessoire);
   lcd.setCursor(8,1);//Place le curseur colonne 13, ligne 4
   lcd.print (adr0programmation_accessoire);

   lcd.setCursor(10,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr27programmation_accessoire);
   lcd.setCursor(11,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr26programmation_accessoire^1);
   lcd.setCursor(12,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr25programmation_accessoire^1);
   lcd.setCursor(13,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr24programmation_accessoire^1);
   lcd.setCursor(14,1);//Place le curseur colonne 10, ligne 4
   lcd.print (adr23programmation_accessoire);
   lcd.setCursor(15,1);//Place le curseur colonne 11, ligne 4
```

```
   lcd.print (adr22programmation_accessoire);
   lcd.setCursor(16,1);//Place le curseur colonne 12, ligne 4
   lcd.print (adr21programmation_accessoire);
   lcd.setCursor(17,1);//Place le curseur colonne 13, ligne 4
   lcd.print (adr20programmation_accessoire);



  lcd.setCursor (0,2);
lcd.print ("Saisir la donnee :");
lcd.blink();
lcd.setCursor (8,3);

//Gestion des claviers
autorise_touche_programmation_accessoire2 = 1;
//Autorise la prise en compte de l'appui touche + et autorise_touche_programmation_accessoire2 à 1



}//Fin du else



}//fin du if compteur touche = 3
}//Fin du VOID TRAITEMENT DES TOUCHES APPUYEES
////////////////////////////////////////////////////////////////////////////////////
//////////////////////////// ////////////////////////////////////////////////////////
////////////MISE EN MEMOIRE DES TOUCHES APPUYEES PROGRAMMATION ACCESSOIRE2////////////
//////////////////////////TOUCHE APPUYEE NUMERIQUE 2//////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_programmation_accessoire2() //Stockage touche appuyée
{
 compteur_touche_programmation_accessoire2 = compteur_touche_programmation_accessoire2 + 1;
 //Incrémente la variable compteur_touche

if (compteur_touche_programmation_accessoire2 == 1) //Teste si compteur_touche = 1
{toucheAprogrammation_accessoire2 = touche;
lcd.blink ();

autorise_touche_programmation_accessoire2_e = 1;//Autorise la prise en compte par la touche E

}//Garde trace de la touche 1ere touche appuyée

if (compteur_touche_programmation_accessoire2 == 2) //Teste si compteur_touche = 2
```

```
{toucheBprogrammation_accessoire2 = touche;
lcd.blink ();}//Garde trace de la touche 1ere touche appuyée

if (compteur_touche_programmation_accessoire2 == 3) //Teste si compteur_touche = 3
{toucheCprogrammation_accessoire2 = touche;//Garde trace de la touche 3ième touche appuyée
lcd.noBlink ();

autorise_touche_programmation_accessoire2 = 0;//Empêche la saisie de chiffre supplémentaire

}//Garde trace de la touche 1ere touche appuyée
}//Fin du VOID MISE EN MEMOIRE DES TOUCHES APPUYEES

////////////////////////////////////////////////////////////////////////////////////
////////////////////// TRAITEMENT DES TOUCHES APPUYEES PROGRAMMATION ACCESSOIRE2//////////
/////////////////////////////////TRAITEMENT APPUI TOUCHE//////////////////////////////
////////////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_programmation_accessoire2()
{
 if (compteur_touche_programmation_accessoire2 == 1)//Prise en compte  si 1 chiffre saisi

{
   nbre_programmation_accessoire2 = (toucheAprogrammation_accessoire2-48);
   //nbre contient l'intégralité de la saisie

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_programmation_accessoire2 = 0;//Remet le compteur à 0 pour prochaine saisie

   donnee_programmation_accessoire2 = nbre_programmation_accessoire2;
   //recopie dans adresse la valeur de nbre en binaire

  lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_programmation_accessoire);
//affiche la variable nbre_programmation_accessoire. Chiffre saisi pour l'adresse de départ

//Affiche le nombre adresse de départ en binaire
   lcd.setCursor(1,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr7programmation_accessoire);
   lcd.setCursor(2,1);//Place le curseur colonne 7, ligne 4
```

```
  lcd.print (adr6programmation_accessoire);
  lcd.setCursor(3,1);//Place le curseur colonne 8, ligne 4
  lcd.print (adr5programmation_accessoire);
  lcd.setCursor(4,1);//Place le curseur colonne 9, ligne 4
  lcd.print (adr4programmation_accessoire);
  lcd.setCursor(5,1);//Place le curseur colonne 10, ligne 4
  lcd.print (adr3programmation_accessoire);
  lcd.setCursor(6,1);//Place le curseur colonne 11, ligne 4
  lcd.print (adr2programmation_accessoire);
  lcd.setCursor(7,1);//Place le curseur colonne 12, ligne 4
  lcd.print (adr1programmation_accessoire);
  lcd.setCursor(8,1);//Place le curseur colonne 13, ligne 4
  lcd.print (adr0programmation_accessoire);

  lcd.setCursor(10,1);//Place le curseur colonne 6, ligne 4
  lcd.print (adr27programmation_accessoire);
  lcd.setCursor(11,1);//Place le curseur colonne 7, ligne 4
  lcd.print (adr26programmation_accessoire^1);
  lcd.setCursor(12,1);//Place le curseur colonne 8, ligne 4
  lcd.print (adr25programmation_accessoire^1);
  lcd.setCursor(13,1);//Place le curseur colonne 9, ligne 4
  lcd.print (adr24programmation_accessoire^1);
  lcd.setCursor(14,1);//Place le curseur colonne 10, ligne 4
  lcd.print (adr23programmation_accessoire);
  lcd.setCursor(15,1);//Place le curseur colonne 11, ligne 4
  lcd.print (adr22programmation_accessoire);
  lcd.setCursor(16,1);//Place le curseur colonne 12, ligne 4
  lcd.print (adr21programmation_accessoire);
  lcd.setCursor(17,1);//Place le curseur colonne 13, ligne 4
  lcd.print (adr20programmation_accessoire);

lcd.setCursor (0,2);
lcd.print ("Donnee saisie :");
lcd.print(nbre_programmation_accessoire2);
lcd.noBlink();

//Transforme la variable adresse en binaire et stocke dans les variables adr
  //Octet 1 d'adresse

  donnee7programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 7);
  donnee6programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 6);
```

```
   donnee5programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 5);
   donnee4programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 4);
   donnee3programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 3);
   donnee2programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 2);
   donnee1programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 1);
   donnee0programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 0);

   lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
   lcd.print (donnee7programmation_accessoire2);
   lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
   lcd.print (donnee6programmation_accessoire2);
   lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
   lcd.print (donnee5programmation_accessoire2);
   lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
   lcd.print (donnee4programmation_accessoire2);
   lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
   lcd.print (donnee3programmation_accessoire2);
   lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
   lcd.print (donnee2programmation_accessoire2);
   lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
   lcd.print (donnee1programmation_accessoire2);
   lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
   lcd.print (donnee0programmation_accessoire2);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Numero de la Cv : ");

lcd.blink();//Permet le clignotement du curseur
lcd.setCursor (9,1);

//Gestion des claviers
```

```
autorise_touche_programmation_accessoire4 = 1;

}// fin du if (compteur_touche_programmation_accessoire2 == 1)
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire2 == 2)
{
  nbre_programmation_accessoire2 = ((toucheAprogrammation_accessoire2-48) * 10)
  + (toucheBprogrammation_accessoire2-48);
  //nbre contient l'intégralité de la saisie

    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_programmation_accessoire2 = 0;//Remet le compteur à 0 pour prochaine saisie

    donnee_programmation_accessoire2 = nbre_programmation_accessoire2;
    //recopie dans adresse la valeur de nbre en binaire

  lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_programmation_accessoire);
//affiche la variable nbre_programmation_accessoire. Chiffre saisi pour l'adresse de départ

//Affiche le nombre adresse de départ en binaire
    lcd.setCursor(1,1);//Place le curseur colonne 6, ligne 4
    lcd.print (adr7programmation_accessoire);
    lcd.setCursor(2,1);//Place le curseur colonne 7, ligne 4
    lcd.print (adr6programmation_accessoire);
    lcd.setCursor(3,1);//Place le curseur colonne 8, ligne 4
    lcd.print (adr5programmation_accessoire);
    lcd.setCursor(4,1);//Place le curseur colonne 9, ligne 4
    lcd.print (adr4programmation_accessoire);
    lcd.setCursor(5,1);//Place le curseur colonne 10, ligne 4
    lcd.print (adr3programmation_accessoire);
    lcd.setCursor(6,1);//Place le curseur colonne 11, ligne 4
    lcd.print (adr2programmation_accessoire);
    lcd.setCursor(7,1);//Place le curseur colonne 12, ligne 4
    lcd.print (adr1programmation_accessoire);
    lcd.setCursor(8,1);//Place le curseur colonne 13, ligne 4
```

```
    lcd.print (adr0programmation_accessoire);


    lcd.setCursor(10,1);//Place le curseur colonne 6, ligne 4
    lcd.print (adr27programmation_accessoire);
    lcd.setCursor(11,1);//Place le curseur colonne 7, ligne 4
    lcd.print (adr26programmation_accessoire^1);
    lcd.setCursor(12,1);//Place le curseur colonne 8, ligne 4
    lcd.print (adr25programmation_accessoire^1);
    lcd.setCursor(13,1);//Place le curseur colonne 9, ligne 4
    lcd.print (adr24programmation_accessoire^1);
    lcd.setCursor(14,1);//Place le curseur colonne 10, ligne 4
    lcd.print (adr23programmation_accessoire);
    lcd.setCursor(15,1);//Place le curseur colonne 11, ligne 4
    lcd.print (adr22programmation_accessoire);
    lcd.setCursor(16,1);//Place le curseur colonne 12, ligne 4
    lcd.print (adr21programmation_accessoire);
    lcd.setCursor(17,1);//Place le curseur colonne 13, ligne 4
    lcd.print (adr20programmation_accessoire);

lcd.setCursor (0,2);
lcd.print ("Donnee saisie :");
lcd.print(nbre_programmation_accessoire2);
lcd.noBlink();

//Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    donnee7programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 7);
    donnee6programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 6);
    donnee5programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 5);
    donnee4programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 4);
    donnee3programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 3);
    donnee2programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 2);
    donnee1programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 1);
    donnee0programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 0);


    lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
    lcd.print (donnee7programmation_accessoire2);
    lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
    lcd.print (donnee6programmation_accessoire2);
    lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
    lcd.print (donnee5programmation_accessoire2);
```

```
   lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
   lcd.print (donnee4programmation_accessoire2);
   lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
   lcd.print (donnee3programmation_accessoire2);
   lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
   lcd.print (donnee2programmation_accessoire2);
   lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
   lcd.print (donnee1programmation_accessoire2);
   lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
   lcd.print (donnee0programmation_accessoire2);

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Numero de la Cv : ");
lcd.setCursor (9,1);
lcd.blink();

//Gestion des claviers
autorise_touche_programmation_accessoire4 = 1;

}//fin du if (compteur_touche_programmation_accessoire2 == 2)
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire2 == 3)
{
  nbre_programmation_accessoire2 = ((toucheAprogrammation_accessoire2-48) * 100)
  + ((toucheBprogrammation_accessoire2-48)*10) + (toucheCprogrammation_accessoire2-48);
  //nbre contient l'intégralité de la saisie}

if (nbre_programmation_accessoire2 < 256)
   {
   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"
```

```
   compteur_touche_programmation_accessoire2 = 0;//Remet le compteur à 0 pour prochaine saisie

   donnee_programmation_accessoire2 = nbre_programmation_accessoire2;
   //recopie dans adresse la valeur de nbre en binaire

  lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_programmation_accessoire);
//affiche la variable nbre_programmation_accessoire. Chiffre saisi pour l'adresse de départ

//Affiche le nombre adresse de départ en binaire
   lcd.setCursor(1,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr7programmation_accessoire);
   lcd.setCursor(2,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr6programmation_accessoire);
   lcd.setCursor(3,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr5programmation_accessoire);
   lcd.setCursor(4,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr4programmation_accessoire);
   lcd.setCursor(5,1);//Place le curseur colonne 10, ligne 4
   lcd.print (adr3programmation_accessoire);
   lcd.setCursor(6,1);//Place le curseur colonne 11, ligne 4
   lcd.print (adr2programmation_accessoire);
   lcd.setCursor(7,1);//Place le curseur colonne 12, ligne 4
   lcd.print (adr1programmation_accessoire);
   lcd.setCursor(8,1);//Place le curseur colonne 13, ligne 4
   lcd.print (adr0programmation_accessoire);

   lcd.setCursor(10,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr27programmation_accessoire);
   lcd.setCursor(11,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr26programmation_accessoire^1);
   lcd.setCursor(12,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr25programmation_accessoire^1);
   lcd.setCursor(13,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr24programmation_accessoire^1);
   lcd.setCursor(14,1);//Place le curseur colonne 10, ligne 4
   lcd.print (adr23programmation_accessoire);
   lcd.setCursor(15,1);//Place le curseur colonne 11, ligne 4
```

```
    lcd.print (adr22programmation_accessoire);
    lcd.setCursor(16,1);//Place le curseur colonne 12, ligne 4
    lcd.print (adr21programmation_accessoire);
    lcd.setCursor(17,1);//Place le curseur colonne 13, ligne 4
    lcd.print (adr20programmation_accessoire);

lcd.setCursor (0,2);
lcd.print ("Donnee saisie :");
lcd.print(nbre_programmation_accessoire2);
lcd.noBlink();

//Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    donnee7programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 7);
    donnee6programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 6);
    donnee5programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 5);
    donnee4programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 4);
    donnee3programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 3);
    donnee2programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 2);
    donnee1programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 1);
    donnee0programmation_accessoire2 = bitRead (donnee_programmation_accessoire2, 0);

    lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
    lcd.print (donnee7programmation_accessoire2);
    lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
    lcd.print (donnee6programmation_accessoire2);
    lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
    lcd.print (donnee5programmation_accessoire2);
    lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
    lcd.print (donnee4programmation_accessoire2);
    lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
    lcd.print (donnee3programmation_accessoire2);
    lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
    lcd.print (donnee2programmation_accessoire2);
    lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
    lcd.print (donnee1programmation_accessoire2);
    lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
    lcd.print (donnee0programmation_accessoire2);

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
```

```
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Numero de la Cv : ");
lcd.setCursor (9,1);
lcd.blink();

//Gestion des claviers
autorise_touche_programmation_accessoire4 = 1;

}//fin du if (nbre_programmation_accessoire2 < 256)
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire2 > 3 || nbre_programmation_accessoire2 > 255)
{
   compteur_touche_programmation_accessoire2 = 0; //Remet le compteur à 0 pour prochaine saisie

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_programmation_accessoire2 = 0;//Remet le compteur à 0 pour prochaine saisie

   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 2

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(2,2);//Place le curseur colonne 0, ligne 3

   lcd.print("entre 0 et 255") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
```

```
   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

   }//(compteur_touche_programmation_accessoire2 > 3 || nbre_programmation_accessoire2 > 255

}//fin if (compteur_touche_programmation_accessoire2 == 3)

}//Fin du VOID TRAITEMENT DES TOUCHES APPUYEES CV2

////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////// ////////////////////////////////////////////////////////////
////////MISE EN MEMOIRE DES TOUCHES APPUYEES PROGRAMMATION ACCESSOIRE4//////////////////////
//////////////////////TOUCHE APPUYEE NUMERIQUE ACCESSOIRE4////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_programmation_accessoire4() //Stockage touche appuyée
{
  compteur_touche_programmation_accessoire4 = compteur_touche_programmation_accessoire4 + 1;
  //Incrémente la variable compteur_touche

if (compteur_touche_programmation_accessoire4 == 1) //Teste si compteur_touche = 1
{toucheAprogrammation_accessoire4 = touche;
lcd.blink();

autorise_touche_programmation_accessoire4_e = 1;//Autorise la prise en compte par la touche E

}//Garde trace de la touche 1ère touche appuyée

if (compteur_touche_programmation_accessoire4 == 2) //Teste si compteur_touche = 2
{toucheBprogrammation_accessoire4 = touche;
lcd.blink();}//Garde trace de la touche 2ième touche appuyée

if (compteur_touche_programmation_accessoire4 == 3) //Teste si compteur_touche = 3
{toucheCprogrammation_accessoire4 = touche;
```

```
lcd.blink();}//Garde trace de la touche 3ième touche appuyée

if (compteur_touche_programmation_accessoire4 == 4) //Teste si compteur_touche = 4
{toucheDprogrammation_accessoire4 = touche;//Garde trace de la touche 4ième touche appuyée
lcd.noBlink ();

autorise_touche_programmation_accessoire4 = 0;//Empêche la saisie de chiffre supplémentaire



}//Garde trace de la touche 4ième touche appuyée
}//Fin du VOID TOUCHE APPUYEE NUMERIQUE CV4

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
/////////////////TRAITEMENT DES TOUCHES APPUYEES PROGRAMMATION ACCESSOIRE4////////
////////////////////////////TRAITEMENT APPUI TOUCHE/////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_programmation_accessoire4()
//Correspond à Appui touche 'E' et autorise_touche_cv4 = 1
{
 if (compteur_touche_programmation_accessoire4 == 1)//Prise en compte  si 1 chiffre saisi

{
   nbre_programmation_accessoire4 = (toucheAprogrammation_accessoire4-48);
   //nbre contient l'intégralité de la saisie

commcv9programmation_accessoire4 = 0;//Pour adressage des 256 premières Cv accessoire
commcv8programmation_accessoire4 = 0;//Pour adressage des 256 premières Cv accessoire

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_programmation_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

   commcv_programmation_accessoire4 = nbre_programmation_accessoire4-1;
   //recopie dans adresse la valeur de nbre en binaire

 //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commcv7programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 7);
   commcv6programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 6);
```

```
  commcv5programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 5);
  commcv4programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 4);
  commcv3programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 3);
  commcv2programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 2);
  commcv1programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 1);
  commcv0programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 0);

//Affichage octet d'adressage des programmation_accessoire et
//deux premiers bytes de l'adresse de la Cv en binaire
  lcd.setCursor(1,2);//Place le curseur colonne 1, ligne 3
  lcd.print (comm15programmation_accessoire4);
  lcd.setCursor(2,2);//Place le curseur colonne 2, ligne 3
  lcd.print (comm14programmation_accessoire4);
  lcd.setCursor(3,2);//Place le curseur colonne 3, ligne 3
  lcd.print (comm13programmation_accessoire4);
  lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
  lcd.print (comm12programmation_accessoire4);
  lcd.setCursor(5,2);//Place le curseur colonne 5, ligne 3
  lcd.print (comm11programmation_accessoire4);
  lcd.setCursor(6,2);//Place le curseur colonne 6, ligne 3
  lcd.print (comm10programmation_accessoire4);
  lcd.setCursor(7,2);//Place le curseur colonne 7, ligne 3
  lcd.print (commcv9programmation_accessoire4);
  lcd.setCursor(8,2);//Place le curseur colonne 8, ligne 3
  lcd.print (commcv8programmation_accessoire4);


  lcd.setCursor(10,2);//Place le curseur colonne 10, ligne 3
  lcd.print (commcv7programmation_accessoire4);
  lcd.setCursor(11,2);//Place le curseur colonne 11, ligne 3
  lcd.print (commcv6programmation_accessoire4);
  lcd.setCursor(12,2);//Place le curseur colonne 12, ligne 3
  lcd.print (commcv5programmation_accessoire4);
  lcd.setCursor(13,2);//Place le curseur colonne 13, ligne 3
  lcd.print (commcv4programmation_accessoire4);
  lcd.setCursor(14,2);//Place le curseur colonne 14, ligne 3
  lcd.print (commcv3programmation_accessoire4);
  lcd.setCursor(15,2);//Place le curseur colonne 15, ligne 3
  lcd.print (commcv2programmation_accessoire4);
  lcd.setCursor(16,2);//Place le curseur colonne 16, ligne 3
  lcd.print (commcv1programmation_accessoire4);
```

```
   lcd.setCursor(17,2);//Place le curseur colonne 17, ligne 3
   lcd.print (commcv0programmation_accessoire4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for
lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_accessoire();

}// fin du if (compteur_touche_programmation_accessoire4 == 1)
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire4 == 2)
{

  nbre_programmation_accessoire4 = ((toucheAprogrammation_accessoire4-48) * 10)
  + (toucheBprogrammation_accessoire4-48);
  //nbre contient l'intégralité de la saisie

commcv9programmation_accessoire4 = 0;//Pour adressage des 256 premières Cv accessoire
commcv8programmation_accessoire4 = 0;//Pour adressage des 256 premières Cv accessoire
```

```
    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_programmation_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

    commcv_programmation_accessoire4 = nbre_programmation_accessoire4-1;
    //recopie dans adresse la valeur de nbre en binaire

  //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    commcv7programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 7);
    commcv6programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 6);
    commcv5programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 5);
    commcv4programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 4);
    commcv3programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 3);
    commcv2programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 2);
    commcv1programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 1);
    commcv0programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 0);

 //Affichage octet d'adressage des programmation_accessoire
 //et deux premiers bytes de l'adresse de la Cv en binaire
    lcd.setCursor(1,2);//Place le curseur colonne 1, ligne 3
    lcd.print (comm15programmation_accessoire4);
    lcd.setCursor(2,2);//Place le curseur colonne 2, ligne 3
    lcd.print (comm14programmation_accessoire4);
    lcd.setCursor(3,2);//Place le curseur colonne 3, ligne 3
    lcd.print (comm13programmation_accessoire4);
    lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
    lcd.print (comm12programmation_accessoire4);
    lcd.setCursor(5,2);//Place le curseur colonne 5, ligne 3
    lcd.print (comm11programmation_accessoire4);
    lcd.setCursor(6,2);//Place le curseur colonne 6, ligne 3
    lcd.print (comm10programmation_accessoire4);
    lcd.setCursor(7,2);//Place le curseur colonne 7, ligne 3
    lcd.print (commcv9programmation_accessoire4);
    lcd.setCursor(8,2);//Place le curseur colonne 8, ligne 3
    lcd.print (commcv8programmation_accessoire4);


    lcd.setCursor(10,2);//Place le curseur colonne 10, ligne 3
    lcd.print (commcv7programmation_accessoire4);
```

```
   lcd.setCursor(11,2);//Place le curseur colonne 11, ligne 3
   lcd.print (commcv6programmation_accessoire4);
   lcd.setCursor(12,2);//Place le curseur colonne 12, ligne 3
   lcd.print (commcv5programmation_accessoire4);
   lcd.setCursor(13,2);//Place le curseur colonne 13, ligne 3
   lcd.print (commcv4programmation_accessoire4);
   lcd.setCursor(14,2);//Place le curseur colonne 14, ligne 3
   lcd.print (commcv3programmation_accessoire4);
   lcd.setCursor(15,2);//Place le curseur colonne 15, ligne 3
   lcd.print (commcv2programmation_accessoire4);
   lcd.setCursor(16,2);//Place le curseur colonne 16, ligne 3
   lcd.print (commcv1programmation_accessoire4);
   lcd.setCursor(17,2);//Place le curseur colonne 17, ligne 3
   lcd.print (commcv0programmation_accessoire4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_accessoire();
```

```
}//fin du if (compteur_touche_programmation_accessoire4 == 2)

////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire4 == 3)
{
  nbre_programmation_accessoire4 = ((toucheAprogrammation_accessoire4-48) * 100)
  + ((toucheBprogrammation_accessoire4-48)*10) + (toucheCprogrammation_accessoire4-48);
  //nbre contient l'intégralité de la saisie}

if (nbre_programmation_accessoire4 < 256)
   {

commcv9programmation_accessoire4 = 0;//Pour adressage de 0 à 255 Cv accessoire
commcv8programmation_accessoire4 = 0;//Pour adressage de 0 à 255 Cv accessoire

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_programmation_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

   commcv_programmation_accessoire4 = nbre_programmation_accessoire4-1;
  //recopie dans adresse la valeur de nbre en binaire

 //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commcv7programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 7);
   commcv6programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 6);
   commcv5programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 5);
   commcv4programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 4);
   commcv3programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 3);
   commcv2programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 2);
   commcv1programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 1);
   commcv0programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 0);

//Affichage octet d'adressage des programmation_accessoire
//et deux premiers bytes de l'adresse de la Cv en binaire
   lcd.setCursor(1,2);//Place le curseur colonne 1, ligne 3
   lcd.print (comm15programmation_accessoire4);
   lcd.setCursor(2,2);//Place le curseur colonne 2, ligne 3
```

```
  lcd.print (comm14programmation_accessoire4);
  lcd.setCursor(3,2);//Place le curseur colonne 3, ligne 3
  lcd.print (comm13programmation_accessoire4);
  lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
  lcd.print (comm12programmation_accessoire4);
  lcd.setCursor(5,2);//Place le curseur colonne 5, ligne 3
  lcd.print (comm11programmation_accessoire4);
  lcd.setCursor(6,2);//Place le curseur colonne 6, ligne 3
  lcd.print (comm10programmation_accessoire4);
  lcd.setCursor(7,2);//Place le curseur colonne 7, ligne 3
  lcd.print (commcv9programmation_accessoire4);
  lcd.setCursor(8,2);//Place le curseur colonne 8, ligne 3
  lcd.print (commcv8programmation_accessoire4);


  lcd.setCursor(10,2);//Place le curseur colonne 10, ligne 3
  lcd.print (commcv7programmation_accessoire4);
  lcd.setCursor(11,2);//Place le curseur colonne 11, ligne 3
  lcd.print (commcv6programmation_accessoire4);
  lcd.setCursor(12,2);//Place le curseur colonne 12, ligne 3
  lcd.print (commcv5programmation_accessoire4);
  lcd.setCursor(13,2);//Place le curseur colonne 13, ligne 3
  lcd.print (commcv4programmation_accessoire4);
  lcd.setCursor(14,2);//Place le curseur colonne 14, ligne 3
  lcd.print (commcv3programmation_accessoire4);
  lcd.setCursor(15,2);//Place le curseur colonne 15, ligne 3
  lcd.print (commcv2programmation_accessoire4);
  lcd.setCursor(16,2);//Place le curseur colonne 16, ligne 3
  lcd.print (commcv1programmation_accessoire4);
  lcd.setCursor(17,2);//Place le curseur colonne 17, ligne 3
  lcd.print (commcv0programmation_accessoire4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
  {
   trame_idle();
  }//fermeture du for
```

```
lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
  {
   trame_idle();
  }//fermeture du for

envoi_trame_programmation_accessoire();
}//fin du if (nbre_programmation_accessoire4 < 256)
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (nbre_programmation_accessoire4 > 255 && nbre_programmation_accessoire4 < 512)
  {

commcv9programmation_accessoire4 = 0;//Pour adressage entre 256 à 511 Cv accessoire
commcv8programmation_accessoire4 = 1;//Pour adressage entre 256 à 511 Cv accessoire

  envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

  compteur_touche_programmation_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

  commcv_programmation_accessoire4 = nbre_programmation_accessoire4-1;
  //recopie dans adresse la valeur de nbre en binaire

 //Transforme la variable adresse en binaire et stocke dans les variables adr
  //Octet 1 d'adresse
  commcv7programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 7);
  commcv6programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 6);
  commcv5programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 5);
  commcv4programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 4);
  commcv3programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 3);
  commcv2programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 2);
```

```
  commcv1programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 1);
  commcv0programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 0);

//Affichage octet d'adressage des programmation_accessoire
//et deux premiers bytes de l'adresse de la Cv en binaire
  lcd.setCursor(1,2);//Place le curseur colonne 1, ligne 3
  lcd.print (comm15programmation_accessoire4);
  lcd.setCursor(2,2);//Place le curseur colonne 2, ligne 3
  lcd.print (comm14programmation_accessoire4);
  lcd.setCursor(3,2);//Place le curseur colonne 3, ligne 3
  lcd.print (comm13programmation_accessoire4);
  lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
  lcd.print (comm12programmation_accessoire4);
  lcd.setCursor(5,2);//Place le curseur colonne 5, ligne 3
  lcd.print (comm11programmation_accessoire4);
  lcd.setCursor(6,2);//Place le curseur colonne 6, ligne 3
  lcd.print (comm10programmation_accessoire4);
  lcd.setCursor(7,2);//Place le curseur colonne 7, ligne 3
  lcd.print (commcv9programmation_accessoire4);
  lcd.setCursor(8,2);//Place le curseur colonne 8, ligne 3
  lcd.print (commcv8programmation_accessoire4);


  lcd.setCursor(10,2);//Place le curseur colonne 10, ligne 3
  lcd.print (commcv7programmation_accessoire4);
  lcd.setCursor(11,2);//Place le curseur colonne 11, ligne 3
  lcd.print (commcv6programmation_accessoire4);
  lcd.setCursor(12,2);//Place le curseur colonne 12, ligne 3
  lcd.print (commcv5programmation_accessoire4);
  lcd.setCursor(13,2);//Place le curseur colonne 13, ligne 3
  lcd.print (commcv4programmation_accessoire4);
  lcd.setCursor(14,2);//Place le curseur colonne 14, ligne 3
  lcd.print (commcv3programmation_accessoire4);
  lcd.setCursor(15,2);//Place le curseur colonne 15, ligne 3
  lcd.print (commcv2programmation_accessoire4);
  lcd.setCursor(16,2);//Place le curseur colonne 16, ligne 3
  lcd.print (commcv1programmation_accessoire4);
  lcd.setCursor(17,2);//Place le curseur colonne 17, ligne 3
  lcd.print (commcv0programmation_accessoire4);

lcd.noBlink();//Empêche le clignotement du curseur
```

```
for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_accessoire ();
}//fin du if (nbre_programmation_accessoire4 > 255 && nbre_programmation_accessoire4 < 512)
////////////////////////////////////////////////////
////////////////////////////////////////////////////
////////////////////////////////////////////////////
if (nbre_programmation_accessoire4 > 511 && nbre_programmation_accessoire4 < 768)
   {

commcv9programmation_accessoire4 = 1;//Pour adressage entre 512 à 767 Cv accessoire
commcv8programmation_accessoire4 = 0;//Pour adressage entre 512 à 767 Cv accessoire

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   compteur_touche_programmation_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

   commcv_programmation_accessoire4 = nbre_programmation_accessoire4-1;
   //recopie dans adresse la valeur de nbre en binaire
```

```
 //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commcv7programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 7);
   commcv6programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 6);
   commcv5programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 5);
   commcv4programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 4);
   commcv3programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 3);
   commcv2programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 2);
   commcv1programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 1);
   commcv0programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 0);

//Affichage octet d'adressage des programmation_accessoire et
//deux premiers bytes de l'adresse de la Cv en binaire
   lcd.setCursor(1,2);//Place le curseur colonne 1, ligne 3
   lcd.print (comm15programmation_accessoire4);
   lcd.setCursor(2,2);//Place le curseur colonne 2, ligne 3
   lcd.print (comm14programmation_accessoire4);
   lcd.setCursor(3,2);//Place le curseur colonne 3, ligne 3
   lcd.print (comm13programmation_accessoire4);
   lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
   lcd.print (comm12programmation_accessoire4);
   lcd.setCursor(5,2);//Place le curseur colonne 5, ligne 3
   lcd.print (comm11programmation_accessoire4);
   lcd.setCursor(6,2);//Place le curseur colonne 6, ligne 3
   lcd.print (comm10programmation_accessoire4);
   lcd.setCursor(7,2);//Place le curseur colonne 7, ligne 3
   lcd.print (commcv9programmation_accessoire4);
   lcd.setCursor(8,2);//Place le curseur colonne 8, ligne 3
   lcd.print (commcv8programmation_accessoire4);


   lcd.setCursor(10,2);//Place le curseur colonne 10, ligne 3
   lcd.print (commcv7programmation_accessoire4);
   lcd.setCursor(11,2);//Place le curseur colonne 11, ligne 3
   lcd.print (commcv6programmation_accessoire4);
   lcd.setCursor(12,2);//Place le curseur colonne 12, ligne 3
   lcd.print (commcv5programmation_accessoire4);
   lcd.setCursor(13,2);//Place le curseur colonne 13, ligne 3
   lcd.print (commcv4programmation_accessoire4);
   lcd.setCursor(14,2);//Place le curseur colonne 14, ligne 3
   lcd.print (commcv3programmation_accessoire4);
```

```
   lcd.setCursor(15,2);//Place le curseur colonne 15, ligne 3
   lcd.print (commcv2programmation_accessoire4);
   lcd.setCursor(16,2);//Place le curseur colonne 16, ligne 3
   lcd.print (commcv1programmation_accessoire4);
   lcd.setCursor(17,2);//Place le curseur colonne 17, ligne 3
   lcd.print (commcv0programmation_accessoire4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_accessoire();
}//fin du if (nbre_programmation_accessoire4 > 511 && nbre_programmation_accessoire4 < 768)
////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (nbre_programmation_accessoire4 > 767 && nbre_programmation_accessoire4 < 1000)//3 chiffres
   {

commcv9programmation_accessoire4 = 1;//Pour adressage entre 769 à 999 Cv accessoire
commcv8programmation_accessoire4 = 1;//Pour adressage entre 769 à 999 Cv accessoire
```

```
    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_programmation_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

    commcv_programmation_accessoire4 = nbre_programmation_accessoire4-1;
    //recopie dans adresse la valeur de nbre en binaire

  //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    commcv7programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 7);
    commcv6programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 6);
    commcv5programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 5);
    commcv4programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 4);
    commcv3programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 3);
    commcv2programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 2);
    commcv1programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 1);
    commcv0programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 0);

 //Affichage octet d'adressage des programmation_accessoire et
 //deux premiers bytes de l'adresse de la Cv en binaire
    lcd.setCursor(1,2);//Place le curseur colonne 1, ligne 3
    lcd.print (comm15programmation_accessoire4);
    lcd.setCursor(2,2);//Place le curseur colonne 2, ligne 3
    lcd.print (comm14programmation_accessoire4);
    lcd.setCursor(3,2);//Place le curseur colonne 3, ligne 3
    lcd.print (comm13programmation_accessoire4);
    lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
    lcd.print (comm12programmation_accessoire4);
    lcd.setCursor(5,2);//Place le curseur colonne 5, ligne 3
    lcd.print (comm11programmation_accessoire4);
    lcd.setCursor(6,2);//Place le curseur colonne 6, ligne 3
    lcd.print (comm10programmation_accessoire4);
    lcd.setCursor(7,2);//Place le curseur colonne 7, ligne 3
    lcd.print (commcv9programmation_accessoire4);
    lcd.setCursor(8,2);//Place le curseur colonne 8, ligne 3
    lcd.print (commcv8programmation_accessoire4);


    lcd.setCursor(10,2);//Place le curseur colonne 10, ligne 3
    lcd.print (commcv7programmation_accessoire4);
```

```
   lcd.setCursor(11,2);//Place le curseur colonne 11, ligne 3
   lcd.print (commcv6programmation_accessoire4);
   lcd.setCursor(12,2);//Place le curseur colonne 12, ligne 3
   lcd.print (commcv5programmation_accessoire4);
   lcd.setCursor(13,2);//Place le curseur colonne 13, ligne 3
   lcd.print (commcv4programmation_accessoire4);
   lcd.setCursor(14,2);//Place le curseur colonne 14, ligne 3
   lcd.print (commcv3programmation_accessoire4);
   lcd.setCursor(15,2);//Place le curseur colonne 15, ligne 3
   lcd.print (commcv2programmation_accessoire4);
   lcd.setCursor(16,2);//Place le curseur colonne 16, ligne 3
   lcd.print (commcv1programmation_accessoire4);
   lcd.setCursor(17,2);//Place le curseur colonne 17, ligne 3
   lcd.print (commcv0programmation_accessoire4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_accessoire();
}//fin du if (nbre_programmation_accessoire4 > 768 && nbre_programmation_accessoire4 < 1000)
```

```
}//Fin du If = 3

////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire4 == 4)//4 chiffres
{
  nbre_programmation_accessoire4 = ((toucheAprogrammation_accessoire4-48) * 1000)
  + ((toucheBprogrammation_accessoire4-48)*100) + ((toucheCprogrammation_accessoire4-48)*10)
  + (toucheDprogrammation_accessoire4-48);
  //nbre contient l'intégralité de la saisie}

////////////////////////////////////////////////
////////////////////////////////////////////////
////////////////////////////////////////////////
if (nbre_programmation_accessoire4 > 999 && nbre_programmation_accessoire4 < 1024)
    {

commcv9programmation_accessoire4 = 1;//Pour adressage entre 1000 à 1023 Cv accessoire
commcv8programmation_accessoire4 = 1;//Pour adressage entre 1000 à 1023 Cv accessoire

    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    compteur_touche_programmation_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

    commcv_programmation_accessoire4 = nbre_programmation_accessoire4-1;
    //recopie dans adresse la valeur de nbre en binaire

 //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    commcv7programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 7);
    commcv6programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 6);
    commcv5programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 5);
    commcv4programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 4);
    commcv3programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 3);
    commcv2programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 2);
    commcv1programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 1);
    commcv0programmation_accessoire4 = bitRead (commcv_programmation_accessoire4, 0);

//Affichage octet d'adressage des programmation_accessoire et
```

```
//deux premiers bytes de l'adresse de la Cv en binaire
   lcd.setCursor(1,2);//Place le curseur colonne 1, ligne 3
   lcd.print (comm15programmation_accessoire4);
   lcd.setCursor(2,2);//Place le curseur colonne 2, ligne 3
   lcd.print (comm14programmation_accessoire4);
   lcd.setCursor(3,2);//Place le curseur colonne 3, ligne 3
   lcd.print (comm13programmation_accessoire4);
   lcd.setCursor(4,2);//Place le curseur colonne 4, ligne 3
   lcd.print (comm12programmation_accessoire4);
   lcd.setCursor(5,2);//Place le curseur colonne 5, ligne 3
   lcd.print (comm11programmation_accessoire4);
   lcd.setCursor(6,2);//Place le curseur colonne 6, ligne 3
   lcd.print (comm10programmation_accessoire4);
   lcd.setCursor(7,2);//Place le curseur colonne 7, ligne 3
   lcd.print (commcv9programmation_accessoire4);
   lcd.setCursor(8,2);//Place le curseur colonne 8, ligne 3
   lcd.print (commcv8programmation_accessoire4);


   lcd.setCursor(10,2);//Place le curseur colonne 10, ligne 3
   lcd.print (commcv7programmation_accessoire4);
   lcd.setCursor(11,2);//Place le curseur colonne 11, ligne 3
   lcd.print (commcv6programmation_accessoire4);
   lcd.setCursor(12,2);//Place le curseur colonne 12, ligne 3
   lcd.print (commcv5programmation_accessoire4);
   lcd.setCursor(13,2);//Place le curseur colonne 13, ligne 3
   lcd.print (commcv4programmation_accessoire4);
   lcd.setCursor(14,2);//Place le curseur colonne 14, ligne 3
   lcd.print (commcv3programmation_accessoire4);
   lcd.setCursor(15,2);//Place le curseur colonne 15, ligne 3
   lcd.print (commcv2programmation_accessoire4);
   lcd.setCursor(16,2);//Place le curseur colonne 16, ligne 3
   lcd.print (commcv1programmation_accessoire4);
   lcd.setCursor(17,2);//Place le curseur colonne 17, ligne 3
   lcd.print (commcv0programmation_accessoire4);

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
```

```
   {
    trame_idle();
   }//fermeture du for

lcd.clear();
lcd.setCursor (4,1);
lcd.print ("Programmation");
lcd.setCursor (6,2);
lcd.print ("en cours");

lcd.noBlink();//Empêche le clignotement du curseur

for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

envoi_trame_programmation_accessoire();
}//fin du if (nbre_programmation_accessoire4 > 999 && nbre_programmation_accessoire4 < 1024)
}//Fin du If = 4
//////////////////////////////////////////////////
//////////////////////////////////////////////////
//////////////////////////////////////////////////
if (compteur_touche_programmation_accessoire4 > 4 || nbre_programmation_accessoire4 > 1023)
//Dépassement du nombre de Cv accessoire
{

   compteur_touche_programmation_accessoire4 = 0; //Remet le compteur à 0 pour prochaine saisie

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

   lcd.clear(); // Efface écran si appui
   lcd.setCursor(7,0);//Place le curseur colonne 0, ligne 2
   lcd.print("Taille") ; // Affiche la chaîne texte

   lcd.setCursor(9,1);//Place le curseur colonne 0, ligne 2
   lcd.print("ou") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 2
```

```
   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Permet une temporisation

   {
    trame_idle();//Envoi trame idle
   }//fermeture du for
   retourdetrame = 1;//En attente appui touche

}//fin if (compteur_touche_programmation_accessoire4 > 4 || nbre_programmation_accessoire4 > 1023)

}//Fin du VOID TRAITEMENT DES TOUCHES APPUYEES

////////////////////////////////////////////////////////////////////////////////
////////////////////////////// ////////////////////////////////////////////////
////////////////////////// MISE EN MEMOIRE DES TOUCHES APPUYEES FONCTION////////////////
//////////////////////////////TOUCHE APPUYEE NUMERIQUE FONCTION//////////////////////
////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_fonction() //Stockage touche appuyée
{
 compteur_touche_fonction = compteur_touche_fonction + 1;
 //Incrémente la variable compteur_touche

if (compteur_touche_fonction == 1) //Teste si compteur_touche = 1
{toucheAfonction = touche;
lcd.blink();

autorise_touche_fonction_e = 1;//Autorise la prise en compte par la touche E

}//Garde trace de la touche 1ere touche appuyée

if (compteur_touche_fonction == 2) //Teste si compteur_touche = 2
{toucheBfonction = touche;
lcd.blink();}//Garde trace de la touche 2ième touche appuyée

if (compteur_touche_fonction == 3) //Teste si compteur_touche = 3
{toucheCfonction = touche;
lcd.noBlink();
```

```
autorise_touche_fonction = 0;//Empêche la saisie de chiffre supplémentaire

}//Garde trace de la touche 3ième touche appuyée

}//Fin du VOID MISE EN MEMOIRE DES TOUCHES APPUYEES
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
///////////////void traitement_appui_touche_fonction()//////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
void traitement_appui_touche_fonction()
//Correspond à Appui touche 'E' et traitement_appui_touche_fonction = 1
{



/*



if (compteur_touche_fonction == 1)//Prise en compte du chiffre saisi pour une entrée
{
nbre_fonction = (toucheAfonction-48);//nbre contient l'intégralité de la saisie



if (nbre_fonction == 0)
{
  octetFonction = bitClear (octetFonction, 4);//Désactive la fonction
  trame_fonction_clavier_100();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 1)
{
  octetFonction = bitSet (octetFonction, 4);//Active la fonction
   trame_fonction_clavier_100();
}
}//compteur_touche_fonction == 1
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = FL/////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
```

```
*/
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////FONCTION REMISE à ZEROS TTES FONCTIONS/////////////////
//////////////////////sf FL, F1, F2, F3, F4/////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (compteur_touche_fonction == 1)//Prise en compte du chiffre saisi pour une entrée
{
nbre_fonction = (toucheAfonction-48);//nbre contient l'intégralité de la saisie



if (nbre_fonction == 3)
{

  trame_fonction_raz();
}
} //compteur_touche_fonction == 1 et nbre_fonction == 9

////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////FONCTION REMISE à ZEROS TTES FONCTIONS/////////////////
//////////////////////sf FL, F1, F2, F3, F4/////////////////////////
////////////////////////////////////////////////////////////////////////



////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////// FONCTION = F1///////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (compteur_touche_fonction == 2)//Prise en compte du chiffre saisi pour une entrée
{
nbre_fonction = ((toucheAfonction-48) * 10) + (toucheBfonction-48);
//nbre contient l'intégralité de la saisie

/*
if (nbre_fonction == 10)
{
  octetFonction = bitClear (octetFonction, 0);//Désactive la fonction
```

```
  trame_fonction_clavier_100();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 11)
{
  octetFonction = bitSet (octetFonction, 0);//Active la fonction
   trame_fonction_clavier_100();
 }

//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F1////////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F2////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 20)
{
  octetFonction = bitClear (octetFonction, 1);//Désactive la fonction
  trame_fonction_clavier_100();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 21)
{
  octetFonction = bitSet (octetFonction, 1);//Active la fonction
   trame_fonction_clavier_100();
 }
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F2////////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F3////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 30)
{
  octetFonction = bitClear (octetFonction, 2);//Désactive la fonction
  trame_fonction_clavier_100();
}
//////////////////////////////////////////////////////////////////////
```

```
if (nbre_fonction == 31)
{
  octetFonction = bitSet (octetFonction, 2);//Active la fonction
   trame_fonction_clavier_100();
}
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////Fin FONCTION = F3/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F4/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 40)
{
  octetFonction = bitClear (octetFonction, 3);//Désactive la fonction
  trame_fonction_clavier_100();
}
 ////////////////////////////////////////////////////////////////////
if (nbre_fonction == 41)
{
  octetFonction = bitSet (octetFonction, 3);//Active la fonction
   trame_fonction_clavier_100();
 }
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////Fin FONCTION = F4/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
*/

////////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F5/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 50)
{
  octetFonction = bitClear (octetFonction1011, 0);//Désactive la fonction
  trame_fonction_clavier_1011();
}
```

```
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 51)
{
  octetFonction = bitSet (octetFonction1011, 0);//Active la fonction
   trame_fonction_clavier_1011();
}
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F5////////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F6////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 60)
{
  octetFonction = bitClear (octetFonction1011, 1);//Désactive la fonction
  trame_fonction_clavier_1011();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 61)
{
  octetFonction = bitSet (octetFonction1011, 1);//Active la fonction
   trame_fonction_clavier_1011();
 }
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F6////////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F7////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 70)
{
  octetFonction = bitClear (octetFonction1011, 2);//Désactive la fonction
  trame_fonction_clavier_1011();
}

//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 71)
```

```
{
  octetFonction = bitSet (octetFonction1011, 2);//Active la fonction
   trame_fonction_clavier_1011();
}
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F7/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F8/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 80)
{
  octetFonction = bitClear (octetFonction1011, 3);//Désactive la fonction
  trame_fonction_clavier_1011();
}
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 81)
{
  octetFonction = bitSet (octetFonction1011, 3);//Active la fonction
   trame_fonction_clavier_1011();
 }
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F8/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F9/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 90)
{
  octetFonction = bitClear (octetFonction1010, 0);//Désactive la fonction
  trame_fonction_clavier_1010();
}
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 91)
{
  octetFonction = bitSet (octetFonction1010, 0);//Active la fonction
   trame_fonction_clavier_1010();
```

```
 }
}//fin du if 2
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F9/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F10/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (compteur_touche_fonction == 3)//Prise en compte du chiffre saisi pour une entrée
{
nbre_fonction = ((toucheAfonction-48) * 100) + ((toucheBfonction-48)*10)
+ (toucheCfonction-48);
//nbre contient l'intégralité de la saisie

if (nbre_fonction == 100)
{
  octetFonction = bitClear (octetFonction1010, 1);//Désactive la fonction
  trame_fonction_clavier_1010();
}
////////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 101)
{
  octetFonction = bitSet (octetFonction1010, 1);//Active la fonction
   trame_fonction_clavier_1010();
 }
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F10/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F11/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 110)
{
  octetFonction = bitClear (octetFonction1010, 2);//Désactive la fonction
  trame_fonction_clavier_1010();
}
////////////////////////////////////////////////////////////////////////
```

```
if (nbre_fonction == 111)
{
  octetFonction = bitSet (octetFonction1010, 2);//Active la fonction
   trame_fonction_clavier_1010();
 }
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
//////////////////Fin FONCTION = F11/////////////////////////////////
////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F12/////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
if (nbre_fonction == 120)
{
  octetFonction = bitClear (octetFonction1010, 3);//Désactive la fonction
  trame_fonction_clavier_1010();
}
////////////////////////////////////////////////////////////////////
if (nbre_fonction == 121)
{
  octetFonction = bitSet (octetFonction1010, 3);//Active la fonction
   trame_fonction_clavier_1010();
 }
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
//////////////////Fin FONCTION = F12/////////////////////////////////
////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F13/////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
if (nbre_fonction == 130)
{
  octetFonction = bitClear (octetFonction11011110, 0);//Désactive la fonction
  trame_fonction_clavier_11011110();
}
////////////////////////////////////////////////////////////////////
if (nbre_fonction == 131)
{
  octetFonction = bitSet (octetFonction11011110, 0);//Active la fonction
```

```
   trame_fonction_clavier_11011110();
  }
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F13/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F14/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 140)
{
  octetFonction = bitClear (octetFonction11011110, 1);//Désactive la fonction
  trame_fonction_clavier_11011110();
}
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 141)
{
  octetFonction = bitSet (octetFonction11011110, 1);//Active la fonction
   trame_fonction_clavier_11011110();
  }
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F14/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F15/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 150)
{
  octetFonction = bitClear (octetFonction11011110, 2);//Désactive la fonction
  trame_fonction_clavier_11011110();
}
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 151)
{
  octetFonction = bitSet (octetFonction11011110, 2);//Active la fonction
   trame_fonction_clavier_11011110();
  }
////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F15/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F16/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 160)
{
  octetFonction = bitClear (octetFonction11011110, 3);//Désactive la fonction
  trame_fonction_clavier_11011110();
}
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 161)
{
  octetFonction = bitSet (octetFonction11011110, 3);//Active la fonction
   trame_fonction_clavier_11011110();
 }
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F16/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F17/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 170)
{
  octetFonction = bitClear (octetFonction11011110, 4);//Désactive la fonction
  trame_fonction_clavier_11011110();
}
////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 171)
{
  octetFonction = bitSet (octetFonction11011110, 4);//Active la fonction
   trame_fonction_clavier_11011110();
 }
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F17/////////////////////////////////////
////////////////////////////////////////////////////////////////////////
```

```
//////////////////////FONCTION = F18////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 180)
{
  octetFonction = bitClear (octetFonction11011110, 5);//Désactive la fonction
  trame_fonction_clavier_11011110();
}
//////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 181)
{
  octetFonction = bitSet (octetFonction11011110, 5);//Active la fonction
   trame_fonction_clavier_11011110();
 }
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////Fin FONCTION = F18////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F19////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 190)
{
  octetFonction = bitClear (octetFonction11011110, 6);//Désactive la fonction
  trame_fonction_clavier_11011110();
}
 //////////////////////////////////////////////////////////////////////////
if (nbre_fonction == 191)
{
  octetFonction = bitSet (octetFonction11011110, 6);//Active la fonction
   trame_fonction_clavier_11011110();
 }
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////Fin FONCTION = F19////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////FONCTION = F20////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 200)
{
  octetFonction = bitClear (octetFonction11011110, 7);//Désactive la fonction
  trame_fonction_clavier_11011110();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 201)
{
  octetFonction = bitSet (octetFonction11011110, 7);//Active la fonction
   trame_fonction_clavier_11011110();
 }
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F20///////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F21///////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 210)
{
  octetFonction = bitClear (octetFonction11011111, 0);//Désactive la fonction
  trame_fonction_clavier_11011111();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 211)
{
  octetFonction = bitSet (octetFonction11011111, 0);//Active la fonction
   trame_fonction_clavier_11011111();
 }
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F21///////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F22///////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 220)
{
```

```
  octetFonction = bitClear (octetFonction11011111, 1);//Désactive la fonction
  trame_fonction_clavier_11011111();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 221)
{
  octetFonction = bitSet (octetFonction11011111, 1);//Active la fonction
   trame_fonction_clavier_11011111();
 }
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F22///////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F23///////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 230)
{
  octetFonction = bitClear (octetFonction11011111, 2);//Désactive la fonction
  trame_fonction_clavier_11011111();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 231)
{
  octetFonction = bitSet (octetFonction11011111, 2);//Active la fonction
   trame_fonction_clavier_11011111();
 }
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F23///////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F24///////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 240)
{
  octetFonction = bitClear (octetFonction11011111, 3);//Désactive la fonction
  trame_fonction_clavier_11011111();
}
```

```
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 241)
{
  octetFonction = bitSet (octetFonction11011111, 3);//Active la fonction
   trame_fonction_clavier_11011111();
 }
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F24///////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F25///////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 250)
{
  octetFonction = bitClear (octetFonction11011111, 4);//Désactive la fonction
  trame_fonction_clavier_11011111();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 251)
{
  octetFonction = bitSet (octetFonction11011111, 4);//Active la fonction
   trame_fonction_clavier_11011111();
 }
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F25///////////////////////////////////
//////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F26///////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 260)
{
  octetFonction = bitClear (octetFonction11011111, 5);//Désactive la fonction
  trame_fonction_clavier_11011111();
}
//////////////////////////////////////////////////////////////////////
if (nbre_fonction == 261)
{
```

```
  octetFonction = bitSet (octetFonction11011111, 5);//Active la fonction
   trame_fonction_clavier_11011111();
 }
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F26/////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F27/////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
if (nbre_fonction == 270)
{
  octetFonction = bitClear (octetFonction11011111, 6);//Désactive la fonction
  trame_fonction_clavier_11011111();
}
////////////////////////////////////////////////////////////////////
if (nbre_fonction == 271)
{
  octetFonction = bitSet (octetFonction11011111, 6);//Active la fonction
   trame_fonction_clavier_11011111();
 }
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
/////////////////Fin FONCTION = F27/////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////FONCTION = F28/////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
if (nbre_fonction == 280)
{
  octetFonction = bitClear (octetFonction11011111, 7);//Désactive la fonction
  trame_fonction_clavier_11011111();
}
////////////////////////////////////////////////////////////////////
if (nbre_fonction == 281)
{
  octetFonction = bitSet (octetFonction11011111, 7);//Active la fonction
   trame_fonction_clavier_11011111();
 }
```

```
  }//Fin du if compteur touche fonction = 3
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////Fin FONCTION = F28//////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////Détermine si le nombre saisi pour la fonction/////////
////////////////////////////est trop grand//////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (compteur_touche_fonction > 3)
{
  nbre_fonction = ((toucheAfonction-48) * 1000) + ((toucheBfonction-48)*100)
  + ((toucheCfonction-48)*10) + (toucheDfonction-48);
  //nbre contient l'intégralité de la saisie}
compteur_touche_fonction = 0; //Remet le compteur à 0 pour prochaine saisie

    //Gestion des claviers
    autorise_touche_locomotive = 0;
    //Interdit la prise en compte de l'appui touche E et programmation locomotive à 0
    autorise_touche_fonction = 0;
    //Interdit la prise en compte de l'appui touche E et programmation fonction à 0
    autorise_touche_Cv = 0;
    //Interdit la prise en compte de l'appui touche E et programmation cv à 0
    autorise_touche_Cv2 = 0;
    //Interdit la prise en compte de l'appui touche E et programmation cv2 à 0
    autorise_touche_Cv4 = 0;
    //Interdit la prise en compte de l'appui touche E et programmation cv4 à 0

    toucheAfonction = 0;//Réinitialise toucheA fonction
    toucheBfonction = 0;//Réinitialise toucheB fonction
    toucheCfonction = 0;//Réinitialise toucheC fonction
    toucheDfonction = 0;//Réinitialise toucheD fonction
    toucheEfonction = 0;//Réinitialise toucheE fonction

    lcd.clear(); // Efface écran si appui

    lcd.setCursor(7,0);//Place le curseur colonne 0, ligne 1
    lcd.print("Taille") ; // Affiche la chaîne texte
```

```
   lcd.setCursor(9,1);//Place le curseur colonne 0, ligne 2
   lcd.print("ou") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 3
   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   for(int compteurtempo = 0; compteurtempo < 120; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation

   {
    trame_idle();
   }//fermeture du for
   retourdetrame = 1;//En attente appui touche

}//fin if (compteur_touche_fonction > 3)
////////////////////////////////////////////////////////////////////////
//////////////////////////////FIN DE////////////////////////////////////
/////////////////Détermine si le nombre saisi pour la fonction/////////
////////////////////////////est trop grand//////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////////Détermine si le nombre saisi pour la fonction/////////
////////////////////////////n'est pas compatible//////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if ((nbre_fonction == 50)||(nbre_fonction == 51)
||(nbre_fonction == 60)||(nbre_fonction == 61)
||(nbre_fonction == 70)||(nbre_fonction == 71)
||(nbre_fonction == 80)||(nbre_fonction == 81)
||(nbre_fonction == 90)||(nbre_fonction == 91)
||(nbre_fonction == 100)||(nbre_fonction == 101)
||(nbre_fonction == 110)||(nbre_fonction == 111)
||(nbre_fonction == 120)||(nbre_fonction == 121)
||(nbre_fonction == 130)||(nbre_fonction == 131)
||(nbre_fonction == 140)||(nbre_fonction == 141)
||(nbre_fonction == 150)||(nbre_fonction == 151)
||(nbre_fonction == 160)||(nbre_fonction == 161)
```

```
||(nbre_fonction == 170)||(nbre_fonction == 171)
||(nbre_fonction == 180)||(nbre_fonction == 181)
||(nbre_fonction == 190)||(nbre_fonction == 191)
||(nbre_fonction == 200)||(nbre_fonction == 201)
||(nbre_fonction == 210)||(nbre_fonction == 211)
||(nbre_fonction == 220)||(nbre_fonction == 221)
||(nbre_fonction == 230)||(nbre_fonction == 231)
||(nbre_fonction == 240)||(nbre_fonction == 241)
||(nbre_fonction == 250)||(nbre_fonction == 251)
||(nbre_fonction == 260)||(nbre_fonction == 261)
||(nbre_fonction == 270)||(nbre_fonction == 271)
||(nbre_fonction == 280)||(nbre_fonction == 281)
||(nbre_fonction == 3))//3 pour raz ttes fonctions sauf FL, F1, F2, F3, F4
/*
 if ((nbre_fonction == 0)||(nbre_fonction == 1)
||(nbre_fonction == 10)||(nbre_fonction == 11)
||(nbre_fonction == 20)||(nbre_fonction == 21)
||(nbre_fonction == 30)||(nbre_fonction == 31)
||(nbre_fonction == 40)||(nbre_fonction == 41)
||(nbre_fonction == 50)||(nbre_fonction == 51)
||(nbre_fonction == 60)||(nbre_fonction == 61)
||(nbre_fonction == 70)||(nbre_fonction == 71)
||(nbre_fonction == 80)||(nbre_fonction == 81)
||(nbre_fonction == 90)||(nbre_fonction == 91)
||(nbre_fonction == 100)||(nbre_fonction == 101)
||(nbre_fonction == 110)||(nbre_fonction == 111)
||(nbre_fonction == 120)||(nbre_fonction == 121)
||(nbre_fonction == 130)||(nbre_fonction == 131)
||(nbre_fonction == 140)||(nbre_fonction == 141)
||(nbre_fonction == 150)||(nbre_fonction == 151)
||(nbre_fonction == 160)||(nbre_fonction == 161)
||(nbre_fonction == 170)||(nbre_fonction == 171)
||(nbre_fonction == 180)||(nbre_fonction == 181)
||(nbre_fonction == 190)||(nbre_fonction == 191)
||(nbre_fonction == 200)||(nbre_fonction == 201)
||(nbre_fonction == 210)||(nbre_fonction == 211)
||(nbre_fonction == 220)||(nbre_fonction == 221)
||(nbre_fonction == 230)||(nbre_fonction == 231)
||(nbre_fonction == 240)||(nbre_fonction == 241)
||(nbre_fonction == 250)||(nbre_fonction == 251)
||(nbre_fonction == 260)||(nbre_fonction == 261)
```

```
||(nbre_fonction == 270)||(nbre_fonction == 271)
||(nbre_fonction == 280)||(nbre_fonction == 281)
||(nbre_fonction == 3))//3 pour raz ttes fonctions sauf FL, F1, F2, F3, F4
 */
//Prise en compte du chiffre saisi pour deux entrées
{lcd.clear();}
//efface l'écran pour permettre réaffichage loc sélectionnée correctement (évite les bug d'affichage)
//Et éviter d'effacer la chaine "En attente appui touche" du setup

else
{
   compteur_touche_fonction = 0; //Remet le compteur à 0 pour prochaine saisie

    locSelectionne = Miseenmemoirelocselectionne;
    //Permet de retrouver la locomotive sélectionnée après appui sur la touche "D" de fonction
    loc2Selectionne = Miseenmemoireloc2selectionne;
    //Car lors de l'appui de cette touche les deux variables loc (2)Selectionne sont remises à zéros

   //Gestion des claviers
   autorise_touche_locomotive = 0;
   //Interdit la prise en compte de l'appui touche E et programmation locomotive à 0
   autorise_touche_fonction = 0;
   //Interdit la prise en compte de l'appui touche E et programmation fonction à 0
   autorise_touche_Cv = 0;//Interdit la prise en compte de l'appui touche E et programmation cv à 0
   autorise_touche_Cv2 = 0;//Interdit la prise en compte de l'appui touche E et programmation cv2 à 0
   autorise_touche_Cv4 = 0;//Interdit la prise en compte de l'appui touche E et programmation cv4 à 0

   toucheAfonction = 0;//Réinitialise toucheA fonction
   toucheBfonction = 0;//Réinitialise toucheB fonction
   toucheCfonction = 0;//Réinitialise toucheC fonction
   toucheDfonction = 0;//Réinitialise toucheD fonction
   toucheEfonction = 0;//Réinitialise toucheE fonction

   autorise_touche_7_et_9 = 1;
   //Ré-autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle
   //Remis à 1 après la fonction demandée pour que seule la touche 3 soit pris en compte

  autorise_touche_1_et_4 = 1;

   autorise_touche_fonction = 0;//Interdit la saisie de chiffre pour programmation locomotive
   //Seule les touches 7 et 9 sont autorisées
```

```
   lcd.clear(); // Efface écran si appui

   lcd.setCursor(7,0);//Place le curseur colonne 0, ligne 1
   lcd.print("Taille") ; // Affiche la chaîne texte

   lcd.setCursor(9,1);//Place le curseur colonne 0, ligne 2
   lcd.print("ou") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 3
   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   for(int compteurtempo = 0; compteurtempo < 120; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for


    lcd.clear();//efface l'écran pour permettre réaffichage loc sélectionnée correctement
    //(évite les bug d'affichage)


}//fin else
}//Fin traitement_appui_touche_fonction()

////////////////////////////////////////////////////////////////////////
//////////////////////////////FIN DE////////////////////////////////////
/////////////////Détermine si le nombre saisi pour la fonction/////////
////////////////////////////n'est pas compatible//////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
void envoi_trame_programmation_cv()
{
  PORTK |= (1<<5);//Equivalent à digitalWrite(Ledrouge,HIGH)
  PORTA |= (1<<1);//Equivalent à digitalWrite(Ledbleue,HIGH)
  PORTA |= (1<<3);//Equivalent à digitalWrite(Ledverte,HIGH)
  PORTK |= (1<<4);//Equivalent à digitalWrite(Ledjaune,HIGH)
```

```
 for(int compteur = 0; compteur < 4; compteur++)
 //Envoi 5 fois la Fonction pour sécuriser la réalisation de l'ordre
{
 //Bit de Synchronisation 16 bit à 1
// Octet de synchronisation
for ( i=0; i <= 15; i++)
 {bitun();}    // La centrale transmet 16 bits à 1
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
////////////////////////////////////////
bitzero(); // Bit à 0 de séparation
////////////////////////////////////
////////////////////////////////////
//Octet de d'adresse de départ
if (adr7cv==1) bitun();  // Si bit à 1
if (adr7cv==0) bitzero(); // Si bit à 0

if (adr6cv==1) bitun();  // Si bit à 1
if (adr6cv==0) bitzero(); // Si bit à 0
if (adr5cv==1) bitun();  // Si bit à 1
if (adr5cv==0) bitzero(); // Si bit à 0
if (adr4cv==1) bitun();  // Si bit à 1
if (adr4cv==0) bitzero(); // Si bit à 0
if (adr3cv==1) bitun();  // Si bit à 1
if (adr3cv==0) bitzero(); // Si bit à 0
if (adr2cv==1) bitun();  // Si bit à 1
if (adr2cv==0) bitzero(); // Si bit à 0
if (adr1cv==1) bitun();  // Si bit à 1
if (adr1cv==0) bitzero(); // Si bit à 0
if (adr0cv==1) bitun();  // Si bit à 1
if (adr0cv==0) bitzero(); // Si bit à 0
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
/////////////////////////////////////////////////////
bitzero();
////////////////////////////////////
```

```
////////////////////////////////////
//Octet de commande
comm7cv = 1;
comm6cv = 1;
comm5cv = 1;
comm4cv = 0;
comm3cv = 1;
comm2cv = 1;

bitun();  // bit à 1: les 3 premiers bits à 1 pour programmation décodeur
bitun(); // bit à 1 : les 3 premiers bits à 1 pour programmation décodeur
bitun();  // bit à 1: les 3 premiers bits à 1 pour programmation décodeur
bitzero(); // bit à 0 : pour pouvoir adresser toutes les cv
bitun();  // bit à 1: les deux bits à 1 permettent d'écrire dans la cv
bitun();  // bit à 1: les deux bits à 1 permettent d'écrire dans la cv

//choix de la cv
if (comm1cv==1) bitun();//Pour adressage 1024 Cv, valeur déterminé lors du choix de la Cv
if (comm1cv==0) bitzero();//Pour adressage 1024 Cv, valeur déterminé lors du choix de la Cv

if (comm0cv==1) bitun();//Pour adressage 1024 Cv, valeur déterminé lors du choix de la Cv
if (comm0cv==0) bitzero();//Pour adressage 1024 Cv, valeur déterminé lors du choix de la Cv

////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
bitzero();
////////////////////////////////////
////////////////////////////////////
//Octet du numéro de la cv
if (adr7cv4==1) bitun();  // Si bit à 1
if (adr7cv4==0) bitzero(); // Si bit à 0

if (adr6cv4==1) bitun();  // Si bit à 1
if (adr6cv4==0) bitzero(); // Si bit à 0
if (adr5cv4==1) bitun();  // Si bit à 1
if (adr5cv4==0) bitzero(); // Si bit à 0
if (adr4cv4==1) bitun();  // Si bit à 1
if (adr4cv4==0) bitzero(); // Si bit à 0
if (adr3cv4==1) bitun();  // Si bit à 1
```

```
if (adr3cv4==0) bitzero(); // Si bit à 0
if (adr2cv4==1) bitun();  // Si bit à 1
if (adr2cv4==0) bitzero(); // Si bit à 0
if (adr1cv4==1) bitun();  // Si bit à 1
if (adr1cv4==0) bitzero(); // Si bit à 0
if (adr0cv4==1) bitun();  // Si bit à 1
if (adr0cv4==0) bitzero(); // Si bit à 0
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
bitzero();
////////////////////////////////////
//Octet de données à entrer dans la cv
if (adr7cv2==1) bitun();  // Si bit à 1
if (adr7cv2==0) bitzero(); // Si bit à 0

if (adr6cv2==1) bitun();  // Si bit à 1
if (adr6cv2==0) bitzero(); // Si bit à 0
if (adr5cv2==1) bitun();  // Si bit à 1
if (adr5cv2==0) bitzero(); // Si bit à 0
if (adr4cv2==1) bitun();  // Si bit à 1
if (adr4cv2==0) bitzero(); // Si bit à 0
if (adr3cv2==1) bitun();  // Si bit à 1
if (adr3cv2==0) bitzero(); // Si bit à 0
if (adr2cv2==1) bitun();  // Si bit à 1
if (adr2cv2==0) bitzero(); // Si bit à 0
if (adr1cv2==1) bitun();  // Si bit à 1
if (adr1cv2==0) bitzero(); // Si bit à 0
if (adr0cv2==1) bitun();  // Si bit à 1
if (adr0cv2==0) bitzero(); // Si bit à 0
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
////////////////////////////////////
bitzero();
////////////////////////////////////
////////////////////////////////////
//Octet de données 3
```

```
//Calcul du OU Exclusif
cont7cv = adr7cv ^ comm7cv ^ adr7cv4 ^ adr7cv2 ; // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle
cont6cv = adr6cv ^ comm6cv ^ adr6cv4 ^ adr6cv2 ; // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont5cv = adr5cv ^ comm5cv ^ adr5cv4 ^ adr5cv2 ; // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont4cv = adr4cv ^ comm4cv ^ adr4cv4 ^ adr4cv2 ; // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont3cv = adr3cv ^ comm3cv ^ adr3cv4 ^ adr3cv2 ; // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont2cv = adr2cv ^ comm2cv ^ adr2cv4 ^ adr2cv2 ; // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont1cv = adr1cv ^ comm1cv ^ adr1cv4 ^ adr1cv2 ; // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont0cv = adr0cv ^ comm0cv ^ adr0cv4 ^ adr0cv2 ; // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle

// Envoi de l'octet de contrôle
//La centrale transmet l'octet de contrôle
if (cont7cv==1) bitun();  // Si bit à 1
if (cont7cv==0) bitzero(); // Si bit à 0
if (cont6cv==1) bitun();  // Si bit à 1
if (cont6cv==0) bitzero(); // Si bit à 0
if (cont5cv==1) bitun();  // Si bit à 1
if (cont5cv==0) bitzero(); // Si bit à 0
if (cont4cv==1) bitun();  // Si bit à 1
if (cont4cv==0) bitzero(); // Si bit à 0
if (cont3cv==1) bitun();  // Si bit à 1
if (cont3cv==0) bitzero(); // Si bit à 0
if (cont2cv==1) bitun();  // Si bit à 1
if (cont2cv==0) bitzero(); // Si bit à 0
if (cont1cv==1) bitun();  // Si bit à 1
if (cont1cv==0) bitzero(); // Si bit à 0
if (cont0cv==1) bitun();  // Si bit à 1
if (cont0cv==0) bitzero(); // Si bit à 0
////////////////////////////////////
////////////////////////////////////
//Fin de transmission
//bit un
////////////////////////////////////////////
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros avant nouvelle transmission
////////////////////////////////////////////
for ( i=0; i<=24; i++)
{bitzero();}
```

```
}//Fermeture du for

lcd.clear(); // Efface écran si appui
   lcd.setCursor(0,0);//Place le curseur colonne 0, ligne 1
   lcd.print("   PROGRAMMATION") ; // Affiche la chaîne texte
   lcd.setCursor(0,2);//Place le curseur colonne 0, ligne 3
   lcd.print("     EFFECTUEE") ; // Affiche la chaîne texte

   lcd.noBlink();//Empêche le clignotement du curseur

  for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
  //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

retourdetrame = 1;//retour à comme une pression sur la touche *

}//Fin void envoi_trame_programmation_cv
/*
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////// Génère la trame FONCTION 100/////////////////////
//////////////////////////////CLAVIER///////////////////////////////
////////////////////////////////////////////////////////////////////
void trame_fonction_clavier_100()
{
  locSelectionne = Miseenmemoirelocselectionne;
  //Permet de retrouver la locomotive sélectionnée après appui sur la touche "D" de fonction
  loc2Selectionne = Miseenmemoireloc2selectionne;
  //Car lors de l'appui de cette touche les deux variables loc (2)Selectionne sont remises à zéros

  PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)
  PORTA |=  (1<<1);//Equivalent à digitalWrite(Ledbleue,HIGH)
  PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
  PORTK &=~ (1<<4);//Equivalent à digitalWrite(Ledjaune,LOW)

//Octet 2 de commande locomotive 1
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
```

```
   comm7 = 1;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm6 = 0;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm5 = 0;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm4 = bitRead (octetFonction, 4);
   comm3 = bitRead (octetFonction, 3);
   comm2 = bitRead (octetFonction, 2);
   comm1 = bitRead (octetFonction, 1);
   comm0 = bitRead (octetFonction, 0);

//Octet 2 de commande locomotive 2
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
   comm27loc = 1;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm26loc = 0;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm25loc = 0;//Les 3 bits comandent les fonctions FL, F1, F2, F3, F4
   comm24loc = bitRead (octetFonction, 4);
   comm23loc = bitRead (octetFonction, 3);
   comm22loc = bitRead (octetFonction, 2);
   comm21loc = bitRead (octetFonction, 1);
   comm20loc = bitRead (octetFonction, 0);
////////////////////////////////////////////
 ////////////////////////////////////////////
for(int compteur = 0; compteur < 4; compteur++)
//Envoi 4 fois la Fonction pour sécuriser la réalisation de l'ordre
{
  //Bit de Synchronisation 16 bit à 1
// Octet de synchronisation
 for ( i=0; i <= 20; i++)
 {bitun();}    // La centrale transmet 20 bits à 1
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
////////////////////////////////////////
bitzero();




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

```
//////////////La fonction est active pour la loc 1//////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
//Octet de données 1
if (locSelectionne == 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
if (comm7==1) bitun();  // Si bit à 1
if (comm7==0) bitzero(); // Si bit à 0
if (comm6==1) bitun();  // Si bit à 1
if (comm6==0) bitzero(); // Si bit à 0
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 du if (locSelectionne == 1)
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
//////////////La fonction est active pour la loc 2//////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
if (loc2Selectionne == 1)
{
if (adr27loc==1) bitun();  // Si bit à 1
if (adr27loc==0) bitzero(); // Si bit à 0
if (adr26loc==1) bitun();  // Si bit à 1
if (adr26loc==0) bitzero(); // Si bit à 0
if (adr25loc==1) bitun();  // Si bit à 1
if (adr25loc==0) bitzero(); // Si bit à 0
if (adr24loc==1) bitun();  // Si bit à 1
if (adr24loc==0) bitzero(); // Si bit à 0
if (adr23loc==1) bitun();  // Si bit à 1
if (adr23loc==0) bitzero(); // Si bit à 0
if (adr22loc==1) bitun();  // Si bit à 1
if (adr22loc==0) bitzero(); // Si bit à 0
if (adr21loc==1) bitun();  // Si bit à 1
if (adr21loc==0) bitzero(); // Si bit à 0
if (adr20loc==1) bitun();  // Si bit à 1
if (adr20loc==0) bitzero(); // Si bit à 0

////////////////////////////////////
////////////////////////////////////
// Bit à zéro de séparation
bitzero();
////////////////////////////////////
////////////////////////////////////
// Octet de commande
//La centrale transmet l'octet de commande

if (comm27loc==1) bitun();  // Si bit à 1
if (comm27loc==0) bitzero(); // Si bit à 0
if (comm26loc==1) bitun();  // Si bit à 1
if (comm26loc==0) bitzero(); // Si bit à 0
```

```
if (comm25loc==1) bitun();  // Si bit à 1
if (comm25loc==0) bitzero(); // Si bit à 0
if (comm24loc==1) bitun();  // Si bit à 1
if (comm24loc==0) bitzero(); // Si bit à 0
if (comm23loc==1) bitun();  // Si bit à 1
if (comm23loc==0) bitzero(); // Si bit à 0
if (comm22loc==1) bitun();  // Si bit à 1
if (comm22loc==0) bitzero(); // Si bit à 0
if (comm21loc==1) bitun();  // Si bit à 1
if (comm21loc==0) bitzero(); // Si bit à 0
if (comm20loc==1) bitun();  // Si bit à 1
if (comm20loc==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
//Calcul du OU Exclusif
cont20loc = adr20loc ^ comm20loc;  // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont21loc = adr21loc ^ comm21loc;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont22loc = adr22loc ^ comm22loc;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont23loc = adr23loc ^ comm23loc;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont24loc = adr24loc ^ comm24loc;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont25loc = adr25loc ^ comm25loc;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont26loc = adr26loc ^ comm26loc;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont27loc = adr27loc ^ comm27loc;  // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle

if (cont27loc==1) bitun();  // Si bit à 1
if (cont27loc==0) bitzero(); // Si bit à 0
if (cont26loc==1) bitun();  // Si bit à 1
if (cont26loc==0) bitzero(); // Si bit à 0
if (cont25loc==1) bitun();  // Si bit à 1
if (cont25loc==0) bitzero(); // Si bit à 0
if (cont24loc==1) bitun();  // Si bit à 1
if (cont24loc==0) bitzero(); // Si bit à 0
if (cont23loc==1) bitun();  // Si bit à 1
if (cont23loc==0) bitzero(); // Si bit à 0
if (cont22loc==1) bitun();  // Si bit à 1
```

```
if (cont22loc==0) bitzero(); // Si bit à 0
if (cont21loc==1) bitun();  // Si bit à 1
if (cont21loc==0) bitzero(); // Si bit à 0
if (cont20loc==1) bitun();  // Si bit à 1
if (cont20loc==0) bitzero(); // Si bit à
}//Fin du if (loc2Selectionne == 1)

////////////////////////////////////
////////////////////////////////////
//Fin de transmission
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros (5 ms)avant nouvelle transmission
for ( i=0; i<=25; i++)
{bitzero();}
//Variables FL0, FL10, FL20, FL30, FL40 pour éviter la réactivation
//de la fonction si l'interrupteur correspondant est sur HIGH


}//Fin du for 4 fois fonctions FL, F1, F2, F3, F4

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
/////Permet l'affichage de la locmotive sélectionnée//////////
//////////////en fonction de la variable//////////////////////
////////////locSelectionne ou loc2Selectionne/////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
//lcd.clear();//efface l'écran pour permettre réaffichage loc sélectionnée correctement

autorise_touche_7_et_9 = 1;//Ré-autorise l'entrée dans la boucle
//de la touche 7 ou 9 pour changer la locomotive sous contrôle
//Remis à 1 après la fonction demandée pour que seule la touche 3 soit pris en compte

autorise_touche_fonction = 0;//Interdit la saisie de chiffre pour programmation locomotive
//Seule les touches 7 et 9 sont autorisées


//Au cas ou aucune loc n'est sélectionnée
if (locSelectionne == 0 && loc2Selectionne == 0)
```

```
{
lcd.clear();//Place le curseur colonne 0, ligne 1
lcd.setCursor(3,0);//Place le curseur colonne 3, ligne 1
lcd.print("Locomotive non"); // Affiche la chaîne texte
lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
lcd.print ("selectionnee");

//Emet 1 bip pour appuyer l'affichage
for(int compteurbuzzer = 0; compteurbuzzer < dureetempobuzzer; compteurbuzzer++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
   }//fermeture du for buzzer

//Emet la trame idle pendant l'affichage "locomotive non sélectionnée" pour garder la puissance
for(int compteurtempo = 0; compteurtempo < dureetempofonction; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
//et permet une temporisation
   {
    trame_idle();
   }//fermeture du for idle

retourdetrame = 1;//En attente appui touche
}//Fin du if (locSelectionne == 0 && loc2Selectionne == 0)

}//Fin du void trame 100
*/
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////// Génère la trame FONCTION 1011//////////////////
////////////////////////////CLAVIER/////////////////////////////////
////////////////////////////////////////////////////////////////////
void trame_fonction_clavier_1011()
{

  locSelectionne = Miseenmemoirelocselectionne;
  //Permet de retrouver la locomotive sélectionnée après appui sur la touche "D" de fonction
  loc2Selectionne = Miseenmemoireloc2selectionne;
  //Car lors de l'appui de cette touche les deux variables loc (2)Selectionne sont remises à zéros
```

```
  PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)
  PORTA |=  (1<<1);//Equivalent à digitalWrite(Ledbleue,HIGH)
  PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
  PORTK &=~ (1<<4);//Equivalent à digitalWrite(Ledjaune,LOW)

//Octet 2 de commande locomotive 1
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
    comm7 = 1;//Les 3 bits commandent les fonctions F8, F7, F6, F5
    comm6 = 0;//Les 3 bits commandent les fonctions F8, F7, F6, F5
    comm5 = 1;//Les 3 bits commandent les fonctions F8, F7, F6, F5
    comm4 = 1;//Les 3 bits commandent les fonctions F8, F7, F6, F5
    comm3 = bitRead (octetFonction1011, 3);
    comm2 = bitRead (octetFonction1011, 2);
    comm1 = bitRead (octetFonction1011, 1);
    comm0 = bitRead (octetFonction1011, 0);

//Octet 2 de commande locomotive 2
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
    comm27loc = 1;//Les 3 bits commandent les fonctions F8, F7, F6, F5
    comm26loc = 0;//Les 3 bits commandent les fonctions F8, F7, F6, F5
    comm25loc = 1;//Les 3 bits commandent les fonctions F8, F7, F6, F5
    comm24loc = 1;//Les 3 bits commandent les fonctions F8, F7, F6, F5
    comm23loc = bitRead (octetFonction1011, 3);
    comm22loc = bitRead (octetFonction1011, 2);
    comm21loc = bitRead (octetFonction1011, 1);
    comm20loc = bitRead (octetFonction1011, 0);
 ////////////////////////////////////////////
 ////////////////////////////////////////////
for(int compteur = 0; compteur < 4; compteur++)
//Envoi 5 fois la Fonction pour sécuriser la réalisation de l'ordre
{
  //Bit de Synchronisation 20 bit à 1
// Octet de synchronisation
 for ( i=0; i <= 19; i++)
 bitun();    // La centrale transmet 20 bits à 1
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
```

```
//////////////////////////////////////
bitzero();
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
/////////////La fonction est active pour la loc 1/////////////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
//Octet de données 1
if (locSelectionne == 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
//if (comm7==1) bitun();  // Si bit à 1
//if (comm7==0) bitzero(); // Si bit à 0
bitun();
//if (comm6==1) bitun();  // Si bit à 1
```

```
//if (comm6==0) bitzero(); // Si bit à 0
bitzero();
//if (comm5==1) bitun();  // Si bit à 1
//if (comm5==0) bitzero(); // Si bit à 0
bitun();
//if (comm4==1) bitun();  // Si bit à 1
//if (comm4==0) bitzero(); // Si bit à 0
bitun();
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 du if (locSelectionne == 1)
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////La fonction est active pour la loc 2//////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (loc2Selectionne == 1)
{
if (adr27loc==1) bitun();  // Si bit à 1
if (adr27loc==0) bitzero(); // Si bit à 0
if (adr26loc==1) bitun();  // Si bit à 1
if (adr26loc==0) bitzero(); // Si bit à 0
if (adr25loc==1) bitun();  // Si bit à 1
if (adr25loc==0) bitzero(); // Si bit à 0
if (adr24loc==1) bitun();  // Si bit à 1
if (adr24loc==0) bitzero(); // Si bit à 0
if (adr23loc==1) bitun();  // Si bit à 1
if (adr23loc==0) bitzero(); // Si bit à 0
if (adr22loc==1) bitun();  // Si bit à 1
if (adr22loc==0) bitzero(); // Si bit à 0
if (adr21loc==1) bitun();  // Si bit à 1
if (adr21loc==0) bitzero(); // Si bit à 0
if (adr20loc==1) bitun();  // Si bit à 1
if (adr20loc==0) bitzero(); // Si bit à 0

////////////////////////////////////
////////////////////////////////////
// Bit à zéro de séparation
bitzero();
////////////////////////////////////
```

```
////////////////////////////////////
// Octet de commande
//La centrale transmet l'octet de commande

//if (comm27loc==1) bitun();  // Si bit à 1
//if (comm27loc==0) bitzero(); // Si bit à 0
bitun();
//if (comm26loc==1) bitun();  // Si bit à 1
//if (comm26loc==0) bitzero(); // Si bit à 0
bitzero;
//if (comm25loc==1) bitun();  // Si bit à 1
//if (comm25loc==0) bitzero(); // Si bit à 0
bitun();
//if (comm24loc==1) bitun();  // Si bit à 1
//if (comm24loc==0) bitzero(); // Si bit à 0
bitun();
if (comm23loc==1) bitun();  // Si bit à 1
if (comm23loc==0) bitzero(); // Si bit à 0
if (comm22loc==1) bitun();  // Si bit à 1
if (comm22loc==0) bitzero(); // Si bit à 0
if (comm21loc==1) bitun();  // Si bit à 1
if (comm21loc==0) bitzero(); // Si bit à 0
if (comm20loc==1) bitun();  // Si bit à 1
if (comm20loc==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
//Calcul du OU Exclusif
cont20loc = adr20loc ^ comm20loc;  // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont21loc = adr21loc ^ comm21loc;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont22loc = adr22loc ^ comm22loc;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont23loc = adr23loc ^ comm23loc;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont24loc = adr24loc ^ comm24loc;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont25loc = adr25loc ^ comm25loc;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont26loc = adr26loc ^ comm26loc;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont27loc = adr27loc ^ comm27loc;  // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle
```

```
if (cont27loc==1) bitun();  // Si bit à 1
if (cont27loc==0) bitzero(); // Si bit à 0
if (cont26loc==1) bitun();  // Si bit à 1
if (cont26loc==0) bitzero(); // Si bit à 0
if (cont25loc==1) bitun();  // Si bit à 1
if (cont25loc==0) bitzero(); // Si bit à 0
if (cont24loc==1) bitun();  // Si bit à 1
if (cont24loc==0) bitzero(); // Si bit à 0
if (cont23loc==1) bitun();  // Si bit à 1
if (cont23loc==0) bitzero(); // Si bit à 0
if (cont22loc==1) bitun();  // Si bit à 1
if (cont22loc==0) bitzero(); // Si bit à 0
if (cont21loc==1) bitun();  // Si bit à 1
if (cont21loc==0) bitzero(); // Si bit à 0
if (cont20loc==1) bitun();  // Si bit à 1
if (cont20loc==0) bitzero(); // Si bit à
}//Fin du if (loc2Selectionne == 1)
////////////////////////////////////
////////////////////////////////////
//Fin de transmission
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros (5 ms)avant nouvelle transmission
for ( i=0; i<=24; i++)
bitzero();

}//Fin du for 4 fois fonctions F8, F7, F6, F5
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
/////Permet l'affichage de la locomotive sélectionnée//////////
//////////////en fonction de la variable//////////////////////
////////////locSelectionne ou loc2Selectionne/////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//lcd.clear();//efface l'écran pour permettre réaffichage loc sélectionnée correctement

autorise_touche_7_et_9 = 1;
//Ré-autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle
   //Remis à 1 après la fonction demandée pour que seule la touche 3 soit pris en compte
```

```
  autorise_touche_1_et_4 = 1;

autorise_touche_fonction = 0;//Interdit la saisie de chiffre pour programmation locomotive
//Seule les touches 7 et 9 sont autorisées

/*
//Au cas ou aucune loc n'est sélectionnée
if (locSelectionne == 0 && loc2Selectionne == 0)
{
lcd.clear();//Place le curseur colonne 0, ligne 1
lcd.setCursor(3,0);//Place le curseur colonne 3, ligne 1
lcd.print("Locomotive non"); // Affiche la chaîne texte
lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
lcd.print ("selectionnee");

//Emet 1 bip pour appuyer l'affichage
for(int compteurbuzzer = 0; compteurbuzzer < dureetempobuzzer; compteurbuzzer++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
   }//fermeture du for buzzer

//Emet la trame idle pendant l'affichage "locomotive non sélectionnée" pour garder la puissance
for(int compteurtempo = 0; compteurtempo < dureetempofonction; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for idle

retourdetrame = 1;//En attente appui touche
}//Fin du if (locSelectionne == 0 && loc2Selectionne == 0)
*/
}//Fin du void trame 1011
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////// Génère la trame FONCTION 1010//////////////////
//////////////////////////////CLAVIER///////////////////////////////
////////////////////////////////////////////////////////////////////
```

```
// Génère la trame FONCTION 1010
void trame_fonction_clavier_1010()
{
  locSelectionne = Miseenmemoirelocselectionne;
  //Permet de retrouver la locomotive sélectionnée après appui sur la touche "D" de fonction
  loc2Selectionne = Miseenmemoireloc2selectionne;
  //Car lors de l'appui de cette touche les deux variables loc (2)Selectionne sont remises à zéros

  PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)
  PORTA |=  (1<<1);//Equivalent à digitalWrite(Ledbleue,HIGH)
  PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
  PORTK &=~ (1<<4);//Equivalent à digitalWrite(Ledjaune,LOW)

//Octet 2 de commande locomotive 1
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
    comm7 = 1;//Les 3 bits commandent les fonctions F12, F11, F10, F9
    comm6 = 0;//Les 3 bits commandent les fonctions F12, F11, F10, F9
    comm5 = 1;//Les 3 bits commandent les fonctions F12, F11, F10, F9
    comm4 = 0;//Les 3 bits commandent les fonctions F12, F11, F10, F9
    comm3 = bitRead (octetFonction1010, 3);
    comm2 = bitRead (octetFonction1010, 2);
    comm1 = bitRead (octetFonction1010, 1);
    comm0 = bitRead (octetFonction1010, 0);

//Octet 2 de commande locomotive 2
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
    comm27loc = 1;//Les 3 bits commandent les fonctions F12, F11, F10, F9
    comm26loc = 0;//Les 3 bits commandent les fonctions F12, F11, F10, F9
    comm25loc = 1;//Les 3 bits commandent les fonctions F12, F11, F10, F9
    comm24loc = 0;//Les 3 bits commandent les fonctions F12, F11, F10, F9
    comm23loc = bitRead (octetFonction1010, 3);
    comm22loc = bitRead (octetFonction1010, 2);
    comm21loc = bitRead (octetFonction1010, 1);
    comm20loc = bitRead (octetFonction1010, 0);
 ////////////////////////////////////////////
 ////////////////////////////////////////////
for(int compteur = 0; compteur < 4; compteur++)
//Envoi 5 fois la Fonction pour sécuriser la réalisation de l'ordre
{
```

```
  //Bit de Synchronisation 20 bit à 1
// Octet de synchronisation
 for ( i=0; i <= 19; i++)
 bitun();    // La centrale transmet 20 bits à 1
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
/////////////////////////////////////////
bitzero();
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////La fonction est active pour la loc 1////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//Octet de données 1
if (locSelectionne == 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
//if (comm7==1) bitun();  // Si bit à 1
//if (comm7==0) bitzero(); // Si bit à 0
bitun();
//if (comm6==1) bitun();  // Si bit à 1
//if (comm6==0) bitzero(); // Si bit à 0
bitzero();
//if (comm5==1) bitun();  // Si bit à 1
//if (comm5==0) bitzero(); // Si bit à 0
bitun();
//if (comm4==1) bitun();  // Si bit à 1
//if (comm4==0) bitzero(); // Si bit à 0
bitzero();
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 du if (locSelectionne == 1)
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////La fonction est active pour la loc 2//////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (loc2Selectionne == 1)
{
if (adr27loc==1) bitun();  // Si bit à 1
if (adr27loc==0) bitzero(); // Si bit à 0
if (adr26loc==1) bitun();  // Si bit à 1
if (adr26loc==0) bitzero(); // Si bit à 0
if (adr25loc==1) bitun();  // Si bit à 1
if (adr25loc==0) bitzero(); // Si bit à 0
if (adr24loc==1) bitun();  // Si bit à 1
if (adr24loc==0) bitzero(); // Si bit à 0
if (adr23loc==1) bitun();  // Si bit à 1
if (adr23loc==0) bitzero(); // Si bit à 0
if (adr22loc==1) bitun();  // Si bit à 1
if (adr22loc==0) bitzero(); // Si bit à 0
if (adr21loc==1) bitun();  // Si bit à 1
if (adr21loc==0) bitzero(); // Si bit à 0
```

```
if (adr20loc==1) bitun();  // Si bit à 1
if (adr20loc==0) bitzero(); // Si bit à 0

////////////////////////////////////
////////////////////////////////////
// Bit à zéro de séparation
bitzero();
////////////////////////////////////
////////////////////////////////////
// Octet de commande
//La centrale transmet l'octet de commande

//if (comm27loc==1) bitun();  // Si bit à 1
//if (comm27loc==0) bitzero(); // Si bit à 0
bitun();
//if (comm26loc==1) bitun();  // Si bit à 1
//if (comm26loc==0) bitzero(); // Si bit à 0
bitzero();
//if (comm25loc==1) bitun();  // Si bit à 1
//if (comm25loc==0) bitzero(); // Si bit à 0
bitun();
//if (comm24loc==1) bitun();  // Si bit à 1
//if (comm24loc==0) bitzero(); // Si bit à 0
bitun();

if (comm23loc==1) bitun();  // Si bit à 1
if (comm23loc==0) bitzero(); // Si bit à 0
if (comm22loc==1) bitun();  // Si bit à 1
if (comm22loc==0) bitzero(); // Si bit à 0
if (comm21loc==1) bitun();  // Si bit à 1
if (comm21loc==0) bitzero(); // Si bit à 0
if (comm20loc==1) bitun();  // Si bit à 1
if (comm20loc==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
```

```
//Calcul du OU Exclusif
cont20loc = adr20loc ^ comm20loc;  // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont21loc = adr21loc ^ comm21loc;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont22loc = adr22loc ^ comm22loc;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont23loc = adr23loc ^ comm23loc;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont24loc = adr24loc ^ comm24loc;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont25loc = adr25loc ^ comm25loc;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont26loc = adr26loc ^ comm26loc;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont27loc = adr27loc ^ comm27loc;  // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle

if (cont27loc==1) bitun();  // Si bit à 1
if (cont27loc==0) bitzero(); // Si bit à 0
if (cont26loc==1) bitun();  // Si bit à 1
if (cont26loc==0) bitzero(); // Si bit à 0
if (cont25loc==1) bitun();  // Si bit à 1
if (cont25loc==0) bitzero(); // Si bit à 0
if (cont24loc==1) bitun();  // Si bit à 1
if (cont24loc==0) bitzero(); // Si bit à 0
if (cont23loc==1) bitun();  // Si bit à 1
if (cont23loc==0) bitzero(); // Si bit à 0
if (cont22loc==1) bitun();  // Si bit à 1
if (cont22loc==0) bitzero(); // Si bit à 0
if (cont21loc==1) bitun();  // Si bit à 1
if (cont21loc==0) bitzero(); // Si bit à 0
if (cont20loc==1) bitun();  // Si bit à 1
if (cont20loc==0) bitzero(); // Si bit à
}//Fin du if (loc2Selectionne == 1)
////////////////////////////////////
////////////////////////////////////
//Fin de transmission
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros (5 ms)avant nouvelle transmission
for ( i=0; i<=24; i++)
bitzero();

}//Fin du for 4 fois fonctions F12, F11, F10, F9
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
/////Permet l'affichage de la locomotive sélectionnée/////////
```

```
//////////////en fonction de la variable//////////////////////
////////////locSelectionne ou loc2Selectionne/////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
//lcd.clear();//efface l'écran pour permettre réaffichage loc sélectionnée correctement

autorise_touche_7_et_9 = 1;
//Ré-autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle
   //Remis à 1 après la fonction demandée pour que seule la touche 3 soit pris en compte

  autorise_touche_1_et_4 = 1;

autorise_touche_fonction = 0;//Interdit la saisie de chiffre pour programmation locomotive
//Seule les touches 7 et 9 sont autorisées

/*
//Au cas ou aucune loc n'est sélectionnée
if (locSelectionne == 0 && loc2Selectionne == 0)
{
lcd.clear();//Place le curseur colonne 0, ligne 1
lcd.setCursor(3,0);//Place le curseur colonne 3, ligne 1
lcd.print("Locomotive non"); // Affiche la chaîne texte
lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
lcd.print ("selectionnee");

//Emet 1 bip pour appuyer l'affichage
for(int compteurbuzzer = 0; compteurbuzzer < dureetempobuzzer; compteurbuzzer++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
   }//fermeture du for buzzer

//Emet la trame idle pendant l'affichage "locomotive non sélectionnée" pour garder la puissance
for(int compteurtempo = 0; compteurtempo < dureetempofonction; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for idle
```

```
retourdetrame = 1;//En attente appui touche
}//Fin du if (locSelectionne == 0 && loc2Selectionne == 0)
*/
}//Fin du void trame 1010
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
/////////////// Génère la trame FONCTION 11011110//////////////////
////////////////////////////CLAVIER/////////////////////////////////
////////////////////////////////////////////////////////////////////
void trame_fonction_clavier_11011110()
{
 locSelectionne = Miseenmemoirelocselectionne;
 //Permet de retrouver la locomotive sélectionnée après appui sur la touche "D" de fonction
 loc2Selectionne = Miseenmemoireloc2selectionne;
 //Car lors de l'appui de cette touche les deux variables loc (2)Selectionne sont remises à zéros

 PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)
 PORTA |=  (1<<1);//Equivalent à digitalWrite(Ledbleue,HIGH)
 PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
 PORTK &=~ (1<<4);//Equivalent à digitalWrite(Ledjaune,LOW)

//Octet 2 de commande locomotive 1
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
   comm7 = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm6 = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm5 = 0;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm4 = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm3 = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm2 = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm1 = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm0 = 0;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13

//Octet 2 de commande locomotive 2
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
   comm27loc = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm26loc = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm25loc = 0;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm24loc = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm23loc = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
```

```
   comm22loc = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm21loc = 1;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
   comm20loc = 0;//L'octet commande les fonctions F20, F19, F18, F17, F16, F15, F14, F13
 ////////////////////////////////////////////
 ////////////////////////////////////////////
for(int compteur = 0; compteur < 4; compteur++)
//Envoi 5 fois la Fonction pour sécuriser la réalisation de l'ordre
{
  //Bit de Synchronisation 20 bit à 1
// Octet de synchronisation
 for ( i=0; i <= 19; i++)
 {bitun();}    // La centrale transmet 20 bits à 1
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
////////////////////////////////////////
bitzero();
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////La fonction est active pour la loc 1//////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//Octet de données 1
if (locSelectionne == 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 11011110
/*
if (comm7==1) bitun();  // Si bit à 1
if (comm7==0) bitzero(); // Si bit à 0
if (comm6==1) bitun();  // Si bit à 1
if (comm6==0) bitzero(); // Si bit à 0
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
*/
bitun();bitun();bitzero();bitun();bitun();bitun();bitun();bitzero();
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
bitzero();

////////////////////////////////////
////////////////////////////////////
//Octet de fonction

fonc17 = bitRead (octetFonction11011110, 7);
```

```
fonc16 = bitRead (octetFonction11011110, 6);
fonc15 = bitRead (octetFonction11011110, 5);
fonc14 = bitRead (octetFonction11011110, 4);
fonc13 = bitRead (octetFonction11011110, 3);
fonc12 = bitRead (octetFonction11011110, 2);
fonc11 = bitRead (octetFonction11011110, 1);
fonc10 = bitRead (octetFonction11011110, 0);

if (fonc17==1) bitun();  // Si bit à 1
if (fonc17==0) bitzero(); // Si bit à 0
if (fonc16==1) bitun();  // Si bit à 1
if (fonc16==0) bitzero(); // Si bit à 0
if (fonc15==1) bitun();  // Si bit à 1
if (fonc15==0) bitzero(); // Si bit à 0
if (fonc14==1) bitun();  // Si bit à 1
if (fonc14==0) bitzero(); // Si bit à 0
if (fonc13==1) bitun();  // Si bit à 1
if (fonc13==0) bitzero(); // Si bit à 0
if (fonc12==1) bitun();  // Si bit à 1
if (fonc12==0) bitzero(); // Si bit à 0
if (fonc11==1) bitun();  // Si bit à 1
if (fonc11==0) bitzero(); // Si bit à 0
if (fonc10==1) bitun();  // Si bit à 1
if (fonc10==0) bitzero(); // Si bit à 0

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

////////////////////////////////////
////////////////////////////////////
//Octet de données 3
//Calcul du OU Exclusif
cont0 = adr0 ^ comm0 ^ fonc10;  // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont1 = adr1 ^ comm1 ^ fonc11;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont2 = adr2 ^ comm2 ^ fonc12;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont3 = adr3 ^ comm3 ^ fonc13;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont4 = adr4 ^ comm4 ^ fonc14;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont5 = adr5 ^ comm5 ^ fonc15;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont6 = adr6 ^ comm6 ^ fonc16;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
```

```
cont7 = adr7 ^ comm7 ^ fonc17;  // ^ Calcul du OU EXCLUSIF bit 7 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 du if (locSelectionne == 1)

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

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

/////////////La fonction est active pour la loc 2////////////////////

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

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

if (loc2Selectionne == 1)

{

if (adr27loc==1) bitun();  // Si bit à 1

if (adr27loc==0) bitzero(); // Si bit à 0

if (adr26loc==1) bitun();  // Si bit à 1

if (adr26loc==0) bitzero(); // Si bit à 0

if (adr25loc==1) bitun();  // Si bit à 1

if (adr25loc==0) bitzero(); // Si bit à 0

if (adr24loc==1) bitun();  // Si bit à 1

if (adr24loc==0) bitzero(); // Si bit à 0

if (adr23loc==1) bitun();  // Si bit à 1

if (adr23loc==0) bitzero(); // Si bit à 0

if (adr22loc==1) bitun();  // Si bit à 1

if (adr22loc==0) bitzero(); // Si bit à 0
```

```
if (adr21loc==1) bitun();  // Si bit à 1
if (adr21loc==0) bitzero(); // Si bit à 0
if (adr20loc==1) bitun();  // Si bit à 1
if (adr20loc==0) bitzero(); // Si bit à 0

////////////////////////////////////
////////////////////////////////////
// Bit à zéro de séparation
bitzero();
////////////////////////////////////
////////////////////////////////////
// Octet de commande 11011110
//La centrale transmet l'octet de commande
/*
if (comm27loc==1) bitun();  // Si bit à 1
if (comm27loc==0) bitzero(); // Si bit à 0
if (comm26loc==1) bitun();  // Si bit à 1
if (comm26loc==0) bitzero(); // Si bit à 0
if (comm25loc==1) bitun();  // Si bit à 1
if (comm25loc==0) bitzero(); // Si bit à 0
if (comm24loc==1) bitun();  // Si bit à 1
if (comm24loc==0) bitzero(); // Si bit à 0
if (comm23loc==1) bitun();  // Si bit à 1
if (comm23loc==0) bitzero(); // Si bit à 0
if (comm22loc==1) bitun();  // Si bit à 1
if (comm22loc==0) bitzero(); // Si bit à 0
if (comm21loc==1) bitun();  // Si bit à 1
if (comm21loc==0) bitzero(); // Si bit à 0
if (comm20loc==1) bitun();  // Si bit à 1
if (comm20loc==0) bitzero(); // Si bit à 0
*/
bitun();bitun();bitzero();bitun();bitun();bitun();bitun();bitzero();
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
bitzero();

////////////////////////////////////
////////////////////////////////////
//Octet de fonction
```

```
fonc27 = bitRead (octetFonction11011110, 7);
fonc26 = bitRead (octetFonction11011110, 6);
fonc25 = bitRead (octetFonction11011110, 5);
fonc24 = bitRead (octetFonction11011110, 4);
fonc23 = bitRead (octetFonction11011110, 3);
fonc22 = bitRead (octetFonction11011110, 2);
fonc21 = bitRead (octetFonction11011110, 1);
fonc20 = bitRead (octetFonction11011110, 0);

if (fonc27==1) bitun();  // Si bit à 1
if (fonc27==0) bitzero(); // Si bit à 0
if (fonc26==1) bitun();  // Si bit à 1
if (fonc26==0) bitzero(); // Si bit à 0
if (fonc25==1) bitun();  // Si bit à 1
if (fonc25==0) bitzero(); // Si bit à 0
if (fonc24==1) bitun();  // Si bit à 1
if (fonc24==0) bitzero(); // Si bit à 0
if (fonc23==1) bitun();  // Si bit à 1
if (fonc23==0) bitzero(); // Si bit à 0
if (fonc22==1) bitun();  // Si bit à 1
if (fonc22==0) bitzero(); // Si bit à 0
if (fonc21==1) bitun();  // Si bit à 1
if (fonc21==0) bitzero(); // Si bit à 0
if (fonc20==1) bitun();  // Si bit à 1
if (fonc20==0) bitzero(); // Si bit à 0
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
bitzero();

////////////////////////////////////
////////////////////////////////////
// Envoi de l'octet de contrôle
//La centrale transmet l'octet de contrôle
//Calcul du OU Exclusif
cont20loc = adr20loc ^ comm20loc ^ fonc20;  // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont21loc = adr21loc ^ comm21loc ^ fonc21;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont22loc = adr22loc ^ comm22loc ^ fonc22;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont23loc = adr23loc ^ comm23loc ^ fonc23;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont24loc = adr24loc ^ comm24loc ^ fonc24;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont25loc = adr25loc ^ comm25loc ^ fonc25;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
```

```
cont26loc = adr26loc ^ comm26loc ^ fonc26;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont27loc = adr27loc ^ comm27loc ^ fonc27;  // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle

if (cont27loc==1) bitun();  // Si bit à 1
if (cont27loc==0) bitzero(); // Si bit à 0
if (cont26loc==1) bitun();  // Si bit à 1
if (cont26loc==0) bitzero(); // Si bit à 0
if (cont25loc==1) bitun();  // Si bit à 1
if (cont25loc==0) bitzero(); // Si bit à 0
if (cont24loc==1) bitun();  // Si bit à 1
if (cont24loc==0) bitzero(); // Si bit à 0
if (cont23loc==1) bitun();  // Si bit à 1
if (cont23loc==0) bitzero(); // Si bit à 0
if (cont22loc==1) bitun();  // Si bit à 1
if (cont22loc==0) bitzero(); // Si bit à 0
if (cont21loc==1) bitun();  // Si bit à 1
if (cont21loc==0) bitzero(); // Si bit à 0
if (cont20loc==1) bitun();  // Si bit à 1
if (cont20loc==0) bitzero(); // Si bit à

////////////////////////////////////
////////////////////////////////////
}//Fin du if (loc2Selectionne == 1)
////////////////////////////////////
////////////////////////////////////
//Fin de transmission
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros (5 ms)avant nouvelle transmission
for ( i=0; i<=24; i++)
{bitzero();}

}//Fin du for 4 fois fonctions F20, F19, F18, F17, F16, F15, F14, F13
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
/////Permet l'affichage de la locmotive sélectionnée/////////
//////////////en fonction de la variable//////////////////////
///////////locSelectionne ou loc2Selectionne/////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

```
//lcd.clear();//efface l'écran pour permettre réaffichage loc sélectionnée correctement

autorise_touche_7_et_9 = 1;
//Ré-autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle
   //Remis à 1 après la fonction demandée pour que seule la touche 3 soit pris en compte

  autorise_touche_1_et_4 = 1;

autorise_touche_fonction = 0;//Interdit la saisie de chiffre pour programmation locomotive
//Seule les touches 7 et 9 sont autorisées

/*
//Au cas ou aucune loc n'est sélectionnée
if (locSelectionne == 0 && loc2Selectionne == 0)
{
lcd.clear();//Place le curseur colonne 0, ligne 1
lcd.setCursor(3,0);//Place le curseur colonne 3, ligne 1
lcd.print("Locomotive non"); // Affiche la chaîne texte
lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
lcd.print ("selectionnee");

//Emet 1 bip pour appuyer l'affichage
for(int compteurbuzzer = 0; compteurbuzzer < dureetempofonction; compteurbuzzer++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
   }//fermeture du for buzzer

//Emet la trame idle pendant l'affichage "locomotive non sélectionnée" pour garder la puissance
for(int compteurtempo = 0; compteurtempo < dureetempofonction; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for idle

retourdetrame = 1;//En attente appui touche
}//Fin du if (locSelectionne == 0 && loc2Selectionne == 0)
*/
}//Fin du void trame 11011110
```

```
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
//////////////Génère la trame FONCTION 11011111/////////////////////
////////////////////////////CLAVIER/////////////////////////////////
//////////////////////////////////////////////////////////////////
void trame_fonction_clavier_11011111()
{
  locSelectionne = Miseenmemoirelocselectionne;
  //Permet de retrouver la locomotive sélectionnée après appui sur la touche "D" de fonction
  loc2Selectionne = Miseenmemoireloc2selectionne;
  //Car lors de l'appui de cette touche les deux variables loc (2)Selectionne sont remises à zéros

  PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)
  PORTA |=  (1<<1);//Equivalent à digitalWrite(Ledbleue,HIGH)
  PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
  PORTK &=~ (1<<4);//Equivalent à digitalWrite(Ledjaune,LOW)

//Octet 2 de commande locomotive 1
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
    comm7 = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm6 = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm5 = 0;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm4 = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm3 = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm2 = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm1 = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm0 = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21

//Octet 2 de commande locomotive 2
//Lecture des bits et positionnement
//Placé ici pour éviter de les retrouver dans la boucle : Variable lue 1 fois
    comm27loc = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm26loc = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm25loc = 0;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm24loc = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm23loc = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm22loc = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm21loc = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
    comm20loc = 1;//L'octet commande les fonctions F28, F27, F26, F25, F24, F23, F22, F21
 ////////////////////////////////////////////
```

```
 ////////////////////////////////////////////
for(int compteur = 0; compteur < 4; compteur++)
//Envoi 4 fois la Fonction pour sécuriser la réalisation de l'ordre
{
  //Bit de Synchronisation 20 bit à 1
// Octet de synchronisation
 for ( i=0; i <= 19; i++)
 {bitun();}    // La centrale transmet 20 bits à 1
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
////////////////////////////////////////
bitzero();
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
/////////////La fonction est active pour la loc 1//////////////////////
//////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
//Octet de données 1
if (locSelectionne == 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 11011111
/*
if (comm7==1) bitun();  // Si bit à 1
if (comm7==0) bitzero(); // Si bit à 0
if (comm6==1) bitun();  // Si bit à 1
if (comm6==0) bitzero(); // Si bit à 0
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
*/
bitun();bitun();bitzero();bitun();bitun();bitun();bitun();bitun();
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
bitzero();

////////////////////////////////////
////////////////////////////////////
//Octet de fonction

fonc17 = bitRead (octetFonction11011111, 7);
fonc16 = bitRead (octetFonction11011111, 6);
fonc15 = bitRead (octetFonction11011111, 5);
fonc14 = bitRead (octetFonction11011111, 4);
fonc13 = bitRead (octetFonction11011111, 3);
```

```
fonc12 = bitRead (octetFonction11011111, 2);
fonc11 = bitRead (octetFonction11011111, 1);
fonc10 = bitRead (octetFonction11011111, 0);

if (fonc17==1) bitun();  // Si bit à 1
if (fonc17==0) bitzero(); // Si bit à 0
if (fonc16==1) bitun();  // Si bit à 1
if (fonc16==0) bitzero(); // Si bit à 0
if (fonc15==1) bitun();  // Si bit à 1
if (fonc15==0) bitzero(); // Si bit à 0
if (fonc14==1) bitun();  // Si bit à 1
if (fonc14==0) bitzero(); // Si bit à 0
if (fonc13==1) bitun();  // Si bit à 1
if (fonc13==0) bitzero(); // Si bit à 0
if (fonc12==1) bitun();  // Si bit à 1
if (fonc12==0) bitzero(); // Si bit à 0
if (fonc11==1) bitun();  // Si bit à 1
if (fonc11==0) bitzero(); // Si bit à 0
if (fonc10==1) bitun();  // Si bit à 1
if (fonc10==0) bitzero(); // Si bit à 0

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

////////////////////////////////////
////////////////////////////////////
//Octet de données 3
//Calcul du OU Exclusif
cont0 = adr0 ^ comm0 ^ fonc10;  // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont1 = adr1 ^ comm1 ^ fonc11;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont2 = adr2 ^ comm2 ^ fonc12;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont3 = adr3 ^ comm3 ^ fonc13;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont4 = adr4 ^ comm4 ^ fonc14;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont5 = adr5 ^ comm5 ^ fonc15;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont6 = adr6 ^ comm6 ^ fonc16;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont7 = adr7 ^ comm7 ^ fonc17;  // ^ Calcul du OU EXCLUSIF bit 7 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 du if (locSelectionne == 1)
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
/////////////La fonction est active pour la loc 2//////////////////////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
if (loc2Selectionne == 1)
{
if (adr27loc==1) bitun();  // Si bit à 1
if (adr27loc==0) bitzero(); // Si bit à 0
if (adr26loc==1) bitun();  // Si bit à 1
if (adr26loc==0) bitzero(); // Si bit à 0
if (adr25loc==1) bitun();  // Si bit à 1
if (adr25loc==0) bitzero(); // Si bit à 0
if (adr24loc==1) bitun();  // Si bit à 1
if (adr24loc==0) bitzero(); // Si bit à 0
if (adr23loc==1) bitun();  // Si bit à 1
if (adr23loc==0) bitzero(); // Si bit à 0
if (adr22loc==1) bitun();  // Si bit à 1
if (adr22loc==0) bitzero(); // Si bit à 0
if (adr21loc==1) bitun();  // Si bit à 1
if (adr21loc==0) bitzero(); // Si bit à 0
if (adr20loc==1) bitun();  // Si bit à 1
if (adr20loc==0) bitzero(); // Si bit à 0

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

```
////////////////////////////////////
// Bit à zéro de séparation
bitzero();
////////////////////////////////////
////////////////////////////////////
// Octet de commande 11011111
//La centrale transmet l'octet de commande
/*
if (comm27loc==1) bitun();  // Si bit à 1
if (comm27loc==0) bitzero(); // Si bit à 0
if (comm26loc==1) bitun();  // Si bit à 1
if (comm26loc==0) bitzero(); // Si bit à 0
if (comm25loc==1) bitun();  // Si bit à 1
if (comm25loc==0) bitzero(); // Si bit à 0
if (comm24loc==1) bitun();  // Si bit à 1
if (comm24loc==0) bitzero(); // Si bit à 0
if (comm23loc==1) bitun();  // Si bit à 1
if (comm23loc==0) bitzero(); // Si bit à 0
if (comm22loc==1) bitun();  // Si bit à 1
if (comm22loc==0) bitzero(); // Si bit à 0
if (comm21loc==1) bitun();  // Si bit à 1
if (comm21loc==0) bitzero(); // Si bit à 0
if (comm20loc==1) bitun();  // Si bit à 1
if (comm20loc==0) bitzero(); // Si bit à 0
*/
bitun();bitun();bitzero();bitun();bitun();bitun();bitun();bitun();
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
bitzero();

////////////////////////////////////
////////////////////////////////////
//Octet de fonction

fonc27 = bitRead (octetFonction11011111, 7);
fonc26 = bitRead (octetFonction11011111, 6);
fonc25 = bitRead (octetFonction11011111, 5);
fonc24 = bitRead (octetFonction11011111, 4);
fonc23 = bitRead (octetFonction11011111, 3);
fonc22 = bitRead (octetFonction11011111, 2);
```

```
fonc21 = bitRead (octetFonction11011111, 1);
fonc20 = bitRead (octetFonction11011111, 0);

if (fonc27==1) bitun();  // Si bit à 1
if (fonc27==0) bitzero(); // Si bit à 0
if (fonc26==1) bitun();  // Si bit à 1
if (fonc26==0) bitzero(); // Si bit à 0
if (fonc25==1) bitun();  // Si bit à 1
if (fonc25==0) bitzero(); // Si bit à 0
if (fonc24==1) bitun();  // Si bit à 1
if (fonc24==0) bitzero(); // Si bit à 0
if (fonc23==1) bitun();  // Si bit à 1
if (fonc23==0) bitzero(); // Si bit à 0
if (fonc22==1) bitun();  // Si bit à 1
if (fonc22==0) bitzero(); // Si bit à 0
if (fonc21==1) bitun();  // Si bit à 1
if (fonc21==0) bitzero(); // Si bit à 0
if (fonc20==1) bitun();  // Si bit à 1
if (fonc20==0) bitzero(); // Si bit à 0
////////////////////////////////////
////////////////////////////////////
//Bit de séparation
bitzero();

////////////////////////////////////
////////////////////////////////////
// Envoi de l'octet de contrôle
//La centrale transmet l'octet de contrôle
//Calcul du OU Exclusif
cont20loc = adr20loc ^ comm20loc ^ fonc20;  // ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle
cont21loc = adr21loc ^ comm21loc ^ fonc21;  // ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle
cont22loc = adr22loc ^ comm22loc ^ fonc22;  // ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle
cont23loc = adr23loc ^ comm23loc ^ fonc23;  // ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
cont24loc = adr24loc ^ comm24loc ^ fonc24;  // ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle
cont25loc = adr25loc ^ comm25loc ^ fonc25;  // ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle
cont26loc = adr26loc ^ comm26loc ^ fonc26;  // ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle
cont27loc = adr27loc ^ comm27loc ^ fonc27;  // ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle

if (cont27loc==1) bitun();  // Si bit à 1
if (cont27loc==0) bitzero(); // Si bit à 0
if (cont26loc==1) bitun();  // Si bit à 1
```

```
if (cont26loc==0) bitzero(); // Si bit à 0
if (cont25loc==1) bitun();  // Si bit à 1
if (cont25loc==0) bitzero(); // Si bit à 0
if (cont24loc==1) bitun();  // Si bit à 1
if (cont24loc==0) bitzero(); // Si bit à 0
if (cont23loc==1) bitun();  // Si bit à 1
if (cont23loc==0) bitzero(); // Si bit à 0
if (cont22loc==1) bitun();  // Si bit à 1
if (cont22loc==0) bitzero(); // Si bit à 0
if (cont21loc==1) bitun();  // Si bit à 1
if (cont21loc==0) bitzero(); // Si bit à 0
if (cont20loc==1) bitun();  // Si bit à 1
if (cont20loc==0) bitzero(); // Si bit à

}//Fin du if (loc2Selectionne == 1)
////////////////////////////////////
////////////////////////////////////
//Fin de transmission
bitun();
////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros (5 ms)avant nouvelle transmission
for ( i=0; i<=24; i++)
{bitzero();}

}//Fin du for 4 fois fonctions F28, F27, F26, F25, F24, F23, F22, F21
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
/////Permet l'affichage de la locmotive sélectionnée//////////
//////////////en fonction de la variable/////////////////////
////////////locSelectionne ou loc2Selectionne/////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//lcd.clear();//efface l'écran pour permettre réaffichage loc sélectionnée correctement

autorise_touche_7_et_9 = 1;
//Ré-autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle
   //Remis à 1 après la fonction demandée pour que seule la touche 3 soit pris en compte

  autorise_touche_1_et_4 = 1;
```

```
autorise_touche_fonction = 0;//Interdit la saisie de chiffre pour programmation locomotive
//Seule les touches 7 et 9 sont autorisées

/*
//Au cas ou aucune loc n'est sélectionnée
if (locSelectionne == 0 && loc2Selectionne == 0)
{



lcd.clear();//Place le curseur colonne 0, ligne 1
lcd.setCursor(3,0);//Place le curseur colonne 3, ligne 1
lcd.print("Locomotive non"); // Affiche la chaîne texte
lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
lcd.print ("selectionnee");

//Emet 1 bip pour appuyer l'affichage
for(int compteurbuzzer = 0; compteurbuzzer < dureetempobuzzer; compteurbuzzer++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
   }//fermeture du for buzzer

//Emet la trame idle pendant l'affichage "locomotive non sélectionnée" pour garder la puissance
for(int compteurtempo = 0; compteurtempo < dureetempofonction; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for idle

retourdetrame = 1;//En attente appui touche
}//Fin du if (locSelectionne == 0 && loc2Selectionne == 0)
*/
}//Fin du void trame 11011111

////////////////////////////////////
////////////////////////////////////
//Remet toutes les fonctions à 0 sf Fl, F2, F3, F4 (Accès par interrupteurs)
void trame_fonction_raz()
{
```

```
locSelectionne = Miseenmemoirelocselectionne;
//Copie dans locSelectionne la mémorisation de la variable pour permettre l'envoi de la trame locomotive
//afin de retrouver l'affichage de la vitesse de la locSelectionnée
loc2Selectionne = Miseenmemoireloc2selectionne;
//Copie dans locSelectionne la mémorisation de la variable pour permettre l'envoi de la trame locomotive
//afin de retrouver l'affichage de la vitesse de la locSelectionnée

bitClear (octetFonction11011111, 7);//F28
bitClear (octetFonction11011111, 6);//F27
bitClear (octetFonction11011111, 5);//F26
bitClear (octetFonction11011111, 4);//F25
bitClear (octetFonction11011111, 3);//F24
bitClear (octetFonction11011111, 2);//F23
bitClear (octetFonction11011111, 1);//F22
bitClear (octetFonction11011111, 0);//F21

bitClear (octetFonction11011110, 7);//F20
bitClear (octetFonction11011110, 6);//F19
bitClear (octetFonction11011110, 5);//F18
bitClear (octetFonction11011110, 4);//F17
bitClear (octetFonction11011110, 3);//F16
bitClear (octetFonction11011110, 2);//F15
bitClear (octetFonction11011110, 1);//F14
bitClear (octetFonction11011110, 0);//F13

bitClear (octetFonction1010, 3);//F12
bitClear (octetFonction1010, 2);//F11
bitClear (octetFonction1010, 1);//F10
bitClear (octetFonction1010, 0);//F9

bitClear (octetFonction1011, 3);//F8
bitClear (octetFonction1011, 2);//F7
bitClear (octetFonction1011, 1);//F6
bitClear (octetFonction1011, 0);//F5

trame_fonction_clavier_1010();
trame_fonction_clavier_1011();
trame_fonction_clavier_11011110();
trame_fonction_clavier_11011111();
```

```
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
/////Permet le retour automatique au contrôle de la Locomotive////
///////après utilisation de la touche de fonction//////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//lcd.clear();//efface l'écran pour permettre réaffichage loc sélectionnée correctement

autorise_touche_7_et_9 = 1;
//Ré-autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle
   //Remis à 1 après la fonction demandée pour que seule la touche 3 soit pris en compte

  autorise_touche_1_et_4 = 1;

autorise_touche_fonction = 0;//Interdit la saisie de chiffre pour programmation locomotive
//Seule les touches 7 et 9 sont autorisées

/*
//Au cas ou aucune loc n'est sélectionnée
if (locSelectionne == 0 && loc2Selectionne == 0)
{
lcd.clear();//Place le curseur colonne 0, ligne 1
lcd.setCursor(3,0);//Place le curseur colonne 3, ligne 1
lcd.print("Locomotive non"); // Affiche la chaîne texte
lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
lcd.print ("selectionnee");

for(int compteurtempo = 0; compteurtempo < dureetempofonction; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

retourdetrame = 1;//En attente appui touche
}//Fin du if (locSelectionne == 0 && loc2Selectionne == 0)
*/
}//Fin du void trame fonction raz

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

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
//////////////////////Void RAZ//////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
void envoi_raz() //Envoi_raz

{
 retourdetrame = 0;
 prog_memo_loc = 0;//Permet l'utilisation direct des menus par les touches A, B, C, D


  compteur_touche_locomotive = 0;
  //remet compteur_touche_locomotive à 0 pour prise en compte nouvel appui
  compteur_touche_locomotive2 = 0;
  //remet compteur_touche_locomotive2 à 0 pour prise en compte nouvel appui
  compteur_touche_fonction = 0;
  //remet compteur_touche_fonction à 0 pour prise en compte nouvel appui
  compteur_touche_cv = 0;//remet compteur_touche_cv à 0 pour prise en compte nouvel appui
  compteur_touche_cv2 = 0;//remet compteur_touche_cv2 à 0 pour prise en compte nouvel appui
  compteur_touche_cv4 = 0;//remet compteur_touche_cv4 à 0 pour prise en compte nouvel appui
  compteur_touche_accessoire = 0;
  //remet compteur_touche_accessoire à 0 pour prise en compte nouvel appui
  compteur_touche_accessoire4 = 0;
  //remet compteur_touche_accessoire2 à 0 pour prise en compte nouvel appui
  compteur_touche_programmation_accessoire = 0;
  //remet compteur_touche_programmation_accessoire à 0 pour prise en compte nouvel appui
  compteur_touche_programmation_accessoire2 = 0;
  //remet compteur_touche_programmation_accessoire2 à 0 pour prise en compte nouvel appui
  compteur_touche_programmation_accessoire4 = 0;
  //remet compteur_touche_programmation_accessoire4 à 0 pour prise en compte nouvel appui

  //Gestion des claviers
  autorise_touche_locomotive = 0;//Interdit l'entrée et l'affichage de nombres dans la boucle locomotive
  autorise_touche_locomotive2 = 0;//Interdit l'entrée et l'affichage de nombres dans la boucle locomotive
  autorise_touche_7_et_9 = 0;
  //Interdit l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle


 autorise_touche_1_et_4 = 0;
```

```
autorise_touche_choix = 0;//Interdit l'entrée des chiffres 1 ou 4 uniquement
autorise_touche_programmation_accessoire = 0;
//Interdit ou Autorise la saisie de chiffre pour programmation_accessoire

autorise_touche_fonction = 0;//Interdit l'entrée et l'affichage de nombres dans la boucle locomotive
autorise_touche_Cv = 0;//Interdit l'entrée et l'affichage de nombres dans la boucle programmationCv
autorise_touche_Cv2 = 0;//Interdit la saisie de chiffre pour programmation_cv2
autorise_touche_Cv4 = 0;//Interdit la saisie de chiffre pour programmation_Cv4
autorise_touche_accessoire = 0;//Interdit la saisie de chiffre pour accessoire
autorise_touche_accessoire4 = 0;
//Interdit la prise en compte de l'appui touche pour la touche 'D' de fonction, si 0 accessoire saisi si 1
autorise_touche_accessoire44 = 0;
//Autorise (Autorise confirmé)  la prise en compte de l'appui touche pour la touche 'E' de fonction
autorise_touche_programmation_accessoire = 0;
//Interdit la saisie de chiffre pour programmation_accessoire à 0 pour prise en compte nouvel appui
autorise_touche_programmation_accessoire2 = 0;
//Interdit la saisie de chiffre pour programmation_accessoire2 à 0 pour prise en compte nouvel appui
autorise_touche_programmation_accessoire4 = 0;
//Interdit la saisie de chiffre pour programmation_accessoire4 à 0 pour prise en compte nouvel appui
autorise_touche_memorisation_loc = 0;
//Interdit l'entrée et l'affichage de nombres dans la boucle mémorisation

autorise_touche_memorisation_adresse_loc = 0;
//Interdit l'entrée et l'affichage de nombres dans la boucle mémorisation

autorise_touche_A = 0;//Autorise l'entrée et l'affichage de nombres dans la boucle mémorisation

autorise_touche_locomotive_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_locomotive2_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_Cv_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_Cv2_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_Cv4_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_fonction_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_accessoire_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_accessoire4_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_programmation_accessoire_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_programmation_accessoire2_e = 0;//Interdit Prise en compte pour validation par touche E
autorise_touche_programmation_accessoire4_e = 0;//Interdit Prise en compte pour validation par touche E



//Autorise le traitement appui touche locomotive
```

```
traitement_appui_touches_locomotive = 0;
//Interdit de rentrer dans la boucle traitement_appui_touches_locomotive quand appui sur 'E'
traitement_appui_touches_locomotive2 = 0;
//Interdit la rentrée dans la boucle traitement_appui_touches_locomotive2 quand appui sur 'E'
traitement_appui_touches_fonction = 0;
//Interdit la rentrée dans la boucle traitement_appui_touches_fonction quand appui sur 'E'
traitement_appui_touches_accessoire = 0;
//Interdit la rentrée dans la boucle traitement_appui_touches_accessoire quand appui sur 'E'
traitement_appui_touches_accessoire4 = 0;
//Interdit la rentrée dans la boucle traitement_appui_touches_accessoire4 quand appui sur 'E'
traitement_appui_touches_programmation_accessoire = 0;
//Interdit la rentrée dans la boucle traitement_appui_touches_programmation_accessoire quand appui sur 'E'
traitement_appui_touches_programmation_accessoire2 = 0;
//Interdit la rentrée dans la boucle traitement_appui_touches_programmation_accessoire2 quand appui sur 'E'
traitement_appui_touches_programmation_accessoire4 = 0;
//Interdit la rentrée dans la boucle traitement_appui_touches_programmation_accessoire4 quand appui sur 'E'

//Remise à zéro des copies de touche saisies
toucheA_locomotive = 0;//Réinitialise toucheA
toucheB_locomotive = 0;//Réinitialise toucheB
toucheC_locomotive = 0;//Réinitialise toucheC
toucheD_locomotive = 0;//Réinitialise toucheD
toucheE_locomotive = 0;//Réinitialise toucheE
toucheF_locomotive = 0;//Réinitialise toucheF

//Remise à zéro des copies de touche saisies
toucheA_locomotive2 = 0;//Réinitialise toucheA
toucheB_locomotive2 = 0;//Réinitialise toucheB
toucheC_locomotive2 = 0;//Réinitialise toucheC
toucheD_locomotive2 = 0;//Réinitialise toucheD
toucheE_locomotive2 = 0;//Réinitialise toucheE
toucheF_locomotive2 = 0;//Réinitialise toucheF

toucheAcv = 0;//Réinitialise toucheA
toucheBcv = 0;//Réinitialise toucheB
toucheCcv = 0;//Réinitialise toucheC

toucheAcv2 = 0;//Réinitialise toucheA
toucheBcv2 = 0;//Réinitialise toucheB
toucheCcv2 = 0;//Réinitialise toucheC
```

```
  toucheAcv4 = 0;//Réinitialise toucheA
  toucheBcv4 = 0;//Réinitialise toucheB
  toucheCcv4 = 0;//Réinitialise toucheC
  toucheDcv4 = 0;//Réinitialise toucheD

  toucheAfonction = 0;//Réinitialise toucheA fonction
  toucheBfonction = 0;//Réinitialise toucheB fonction
  toucheCfonction = 0;//Réinitialise toucheC fonction
  toucheDfonction = 0;//Réinitialise toucheD fonction
  toucheEfonction = 0;//Réinitialise toucheE fonction

  toucheAaccessoire = 0;//Utilisée pour enregistrer la première touche appuyée
  toucheBaccessoire = 0;//Utilisée pour enregistrer la deuxième touche appuyée
  toucheCaccessoire = 0;//Utilisée pour enregistrer la troisième touche appuyée

  toucheAaccessoire4 = 0;//Utilisée pour enregistrer la première touche appuyée
  toucheBaccessoire4 = 0;//Utilisée pour enregistrer la deuxième touche appuyée
  toucheCaccessoire4 = 0;//Utilisée pour enregistrer la troisième touche appuyée

  toucheAprogrammation_accessoire = 0;//Utilisée pour enregistrer la première touche appuyée
  toucheBprogrammation_accessoire = 0;//Utilisée pour enregistrer la deuxième touche appuyée
  toucheCprogrammation_accessoire = 0;//Utilisée pour enregistrer la troisième touche appuyée

  toucheAprogrammation_accessoire4 = 0;//Utilisée pour enregistrer la première touche appuyée
  toucheBprogrammation_accessoire4 = 0;//Utilisée pour enregistrer la deuxième touche appuyée
  toucheCprogrammation_accessoire4 = 0;//Utilisée pour enregistrer la troisième touche appuyée
  toucheDprogrammation_accessoire4 = 0;//Utilisée pour enregistrer la quatrième touche appuyée

  confirme_raz = 0;

  //RAZ sélection des loc
  locSelectionne = 0;//Empêche l'envoi de la trame locomotive et une erreur d'affichage
  loc2Selectionne = 0;//Empêche l'envoi de la trame locomotive2 et une erreur d'affichage

 }//Fin du void envoi_raz()

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
void envoi_raz_retourdetrame()//envoi raz_retourdetrame
```

```
{

  envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    lcd.clear(); // Efface écran si appui
    lcd.setCursor(0,0);//Place le curseur colonne 1, ligne 1

    lcd.print("     En ATTENTE") ; // Affiche la chaîne texte
    lcd.setCursor(0,2);//Place le curseur colonne 0, ligne 3
    lcd.print("    APPUI TOUCHE") ; // Affiche la chaîne texte

    lcd.noBlink();//Empêche le clignotement du curseur

    //RAZ sélection des loc
   locSelectionne = 0;//Empêche l'envoi de la trame locomotive et une erreur d'affichage
    loc2Selectionne = 0;//Empêche l'envoi de la trame locomotive2 et une erreur d'affichage

} //Fin void envoi_raz_retourdetrame()
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
void Cv() // Envoi programmation Cv
{
  envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

autorise_touche_Cv = 1;//Interdit l'entrée et l'affichage de nombres dans la boucle programmationCv

    lcd.clear(); // Efface écran si appui = sinon  affiche touche
    lcd.setCursor(2,0);//Place le curseur colonne 2, ligne 1
    lcd.print("Programmation Cv"); // Affiche la chaîne texte

    lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 2
    lcd.print("Atte Adr decodeur"); // Affiche la chaîne texte
    lcd.setCursor(9,2);//Place le curseur colonne 9, ligne 3, prépositionne le chiffre qui va être saisi

    lcd.blink();//Autorise le clignotement du curseur

    //RAZ sélection loc
    locSelectionne = 0;//Empêche l'envoi de la trame locomotive et une erreur d'affichage
    loc2Selectionne = 0;//Empêche l'envoi de la trame locomotive2 et une erreur d'affichage
```

```
} //Fin du void Cv Fin if (touche == 'C')

////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////// ////////////////////////////////////////////////////////////////
/////////////////MISE EN MEMOIRE DES TOUCHES APPUYEES COMMANDE ACCESSOIRE////////////////////
//////////////////////////////TOUCHE APPUYEE NUMERIQUE//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_accessoire() //Stockage touche appuyée
{
compteur_touche_accessoire = compteur_touche_accessoire + 1;//Incrémente la variable compteur_touche

if (compteur_touche_accessoire == 1) //Teste si compteur_touche = 1
{toucheAaccessoire = touche;
lcd.blink ();

autorise_touche_accessoire_e = 1;//Autorise la prise en compte par la touche E

}//Garde trace de la touche 1ere touche appuyée

if (compteur_touche_accessoire == 2) //Teste si compteur_touche = 2
{toucheBaccessoire = touche;
lcd.blink ();

}//Garde trace de la touche 2ième touche appuyée

if (compteur_touche_accessoire == 3) //Teste si compteur_touche = 3
{toucheCaccessoire = touche;//Garde trace de la touche 3ième touche appuyée
lcd.noBlink ();

autorise_touche_accessoire = 0;//Empêche la saisie de chiffre supplémentaire

}//Garde trace de la touche 3ieme touche appuyée

}//Fin du VOID MISE EN MEMOIRE DES TOUCHES APPUYEES
////////////////////////////////////////////////////////////////////////////////////////////
/////////////////MISE EN MEMOIRE DES TOUCHES APPUYEES COMMANDE ACCESSOIRE////////////////////
//////////////////////////////TRAITEMENT APPUI TOUCHE//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_accessoire()//Correspond à Appui touche 'E' et autorise_touche_accessoire
```

```
{

 if (compteur_touche_accessoire == 1)//Prise en compte  si 1 chiffre saisi
 {
     nbre_accessoire = (toucheAaccessoire-48);//nbre contient l'intégralité de la saisie

if (nbre_accessoire == 0)//Prise en compte  si 1 chiffre saisi
 {
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 510") ; // Affiche la chaîne texte

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_programmation_accessoire == 0)

else
{
   compteur_touche_accessoire = 0;//Remet le compteur à 0 pour prochaine saisie

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"
```

```
    adresse_accessoire = nbre_accessoire;//recopie dans adresse la valeur de nbre en binaire

    //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7accessoire = 1;//Adresse accessoire
    adr6accessoire = 0;//Adresse accessoire
    adr5accessoire = bitRead (adresse_accessoire, 5);
    adr4accessoire = bitRead (adresse_accessoire, 4);
    adr3accessoire = bitRead (adresse_accessoire, 3);
    adr2accessoire = bitRead (adresse_accessoire, 2);
    adr1accessoire = bitRead (adresse_accessoire, 1);
    adr0accessoire = bitRead (adresse_accessoire, 0);

//Octet 2 d'adresse

    adr27accessoire = 1;//ACCESSOIRE
    adr26accessoire = bitRead (adresse_accessoire, 8);//ACCESSOIRE complément à 1
    adr25accessoire = bitRead (adresse_accessoire, 7);//ACCESSOIRE complément à 1
    adr24accessoire = bitRead (adresse_accessoire, 6);//ACCESSOIRE complément à 1


//Affiche le nombre adresse de départ en binaire
    lcd.setCursor(1,3);//Place le curseur colonne 2, ligne 4
    lcd.print (adr7accessoire);
    lcd.setCursor(2,3);//Place le curseur colonne 3, ligne 4
    lcd.print (adr6accessoire);
    lcd.setCursor(3,3);//Place le curseur colonne 4, ligne 4
    lcd.print (adr5accessoire);
    lcd.setCursor(4,3);//Place le curseur colonne 5, ligne 4
    lcd.print (adr4accessoire);
    lcd.setCursor(5,3);//Place le curseur colonne 6, ligne 4
    lcd.print (adr3accessoire);
    lcd.setCursor(6,3);//Place le curseur colonne 7, ligne 4
    lcd.print (adr2accessoire);
    lcd.setCursor(7,3);//Place le curseur colonne 8, ligne 4
    lcd.print (adr1accessoire);
    lcd.setCursor(8,3);//Place le curseur colonne 9, ligne 4
    lcd.print (adr0accessoire);

    lcd.setCursor(9,3);//Place le curseur colonne 10, ligne 4
```

```
   lcd.print (" ");//Efface le numéro d'adresse rentré


   lcd.setCursor(10,3);//Place le curseur colonne 11, ligne 4
   lcd.print (adr27accessoire);
   lcd.setCursor(11,3);//Place le curseur colonne 12, ligne 4
   lcd.print (adr26accessoire^1);
   lcd.setCursor(12,3);//Place le curseur colonne 13, ligne 4
   lcd.print (adr25accessoire^1);
   lcd.setCursor(13,3);//Place le curseur colonne 14, ligne 4
   lcd.print (adr24accessoire^1);
   lcd.setCursor(14,3);//Place le curseur colonne 15, ligne 4
   lcd.print (adr23accessoire);
   lcd.setCursor(15,3);//Place le curseur colonne 16, ligne 4
   lcd.print (adr22accessoire);
   lcd.setCursor(16,3);//Place le curseur colonne 17, ligne 4
   lcd.print (adr21accessoire);
   lcd.setCursor(17,3);//Place le curseur colonne 18, ligne 4
   lcd.print (adr20accessoire);


for(int compteurtempo = 0; compteurtempo < dureetempoaccessoire; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for


 lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_accessoire);


 ////Affiche le nombre adresse de départ en binaire 2ième affichage
   lcd.setCursor(1,1);//Place le curseur colonne 2, ligne 4
   lcd.print (adr7accessoire);
   lcd.setCursor(2,1);//Place le curseur colonne 3, ligne 4
   lcd.print (adr6accessoire);
   lcd.setCursor(3,1);//Place le curseur colonne 4, ligne 4
   lcd.print (adr5accessoire);
   lcd.setCursor(4,1);//Place le curseur colonne 5, ligne 4
   lcd.print (adr4accessoire);
   lcd.setCursor(5,1);//Place le curseur colonne 6, ligne 4
```

```
   lcd.print (adr3accessoire);
   lcd.setCursor(6,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr2accessoire);
   lcd.setCursor(7,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr1accessoire);
   lcd.setCursor(8,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr0accessoire);

   lcd.setCursor(9,1);//Place le curseur colonne 10, ligne 4
   lcd.print (" ");//Efface le numéro d'adresse rentré

   lcd.setCursor(10,1);//Place le curseur colonne 11, ligne 4
   lcd.print (adr27accessoire);
   lcd.setCursor(11,1);//Place le curseur colonne 12, ligne 4
   lcd.print (adr26accessoire^1);
   lcd.setCursor(12,1);//Place le curseur colonne 13, ligne 4
   lcd.print (adr25accessoire^1);
   lcd.setCursor(13,1);//Place le curseur colonne 14, ligne 4
   lcd.print (adr24accessoire^1);
   lcd.setCursor(14,1);//Place le curseur colonne 15, ligne 4
   lcd.print (adr23accessoire);
   lcd.setCursor(15,1);//Place le curseur colonne 16, ligne 4
   lcd.print (adr22accessoire);
   lcd.setCursor(16,1);//Place le curseur colonne 17, ligne 4
   lcd.print (adr21accessoire);
   lcd.setCursor(17,1);//Place le curseur colonne 18, ligne 4
   lcd.print (adr20accessoire);

  lcd.setCursor (0,2);
lcd.print ("Saisir donnee :   D");//Rappel appui touche D pour
lcd.setCursor (9,3);
lcd.blink ();

//Gestion des claviers


autorise_touche_accessoire4 = 1;//Autorise la prise en compte de l'appui touche + et autorise_touche_accessoire4 à 1

}//Fin du else
}// fin du if = 1
//////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
if (compteur_touche_accessoire == 2)
{
  nbre_accessoire = ((toucheAaccessoire-48) * 10) + (toucheBaccessoire-48);//nbre contient l'intégralité de la saisie}

if (nbre_accessoire == 0)//Prise en compte  si 1 chiffre saisi
 {
  lcd.clear(); // Efface écran si appui
    lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

    lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

    lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

    lcd.print("doit etre compris") ; // Affiche la chaîne texte

    lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

    lcd.print("entre 1 et 510") ; // Affiche la chaîne texte

    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_programmation_accessoire == 0)

else
{

    compteur_touche_accessoire = 0;//Remet le compteur à 0 pour prochaine saisie
```

```
    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    adresse_accessoire = nbre_accessoire;//recopie dans adresse la valeur de nbre en binaire

    //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7accessoire = 1;//Adresse accessoire
    adr6accessoire = 0;//Adresse accessoire
    adr5accessoire = bitRead (adresse_accessoire, 5);
    adr4accessoire = bitRead (adresse_accessoire, 4);
    adr3accessoire = bitRead (adresse_accessoire, 3);
    adr2accessoire = bitRead (adresse_accessoire, 2);
    adr1accessoire = bitRead (adresse_accessoire, 1);
    adr0accessoire = bitRead (adresse_accessoire, 0);

//Affiche le nombre adresse de départ en binaire
    lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
    lcd.print (adr7accessoire);
    lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
    lcd.print (adr6accessoire);
    lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
    lcd.print (adr5accessoire);
    lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
    lcd.print (adr4accessoire);
    lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
    lcd.print (adr3accessoire);
    lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
    lcd.print (adr2accessoire);
    lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
    lcd.print (adr1accessoire);
    lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
    lcd.print (adr0accessoire);


//Octet 2 d'adresse

    adr27accessoire = 1;//ACCESSOIRE
    adr26accessoire = bitRead (adresse_accessoire, 8);//ACCESSOIRE complément à 1
    adr25accessoire = bitRead (adresse_accessoire, 7);//ACCESSOIRE complément à 1
    adr24accessoire = bitRead (adresse_accessoire, 6);//ACCESSOIRE complément à 1
```

```
//Affiche le nombre adresse de départ en binaire
   lcd.setCursor(1,3);//Place le curseur colonne 2, ligne 4
   lcd.print (adr7accessoire);
   lcd.setCursor(2,3);//Place le curseur colonne 3, ligne 4
   lcd.print (adr6accessoire);
   lcd.setCursor(3,3);//Place le curseur colonne 4, ligne 4
   lcd.print (adr5accessoire);
   lcd.setCursor(4,3);//Place le curseur colonne 5, ligne 4
   lcd.print (adr4accessoire);
   lcd.setCursor(5,3);//Place le curseur colonne 6, ligne 4
   lcd.print (adr3accessoire);
   lcd.setCursor(6,3);//Place le curseur colonne 7, ligne 4
   lcd.print (adr2accessoire);
   lcd.setCursor(7,3);//Place le curseur colonne 8, ligne 4
   lcd.print (adr1accessoire);
   lcd.setCursor(8,3);//Place le curseur colonne 9, ligne 4
   lcd.print (adr0accessoire);

   lcd.setCursor(9,3);//Place le curseur colonne 10, ligne 4
   lcd.print (" ");//Efface le numéro d'adresse rentré

   lcd.setCursor(10,3);//Place le curseur colonne 11, ligne 4
   lcd.print (adr27accessoire);
   lcd.setCursor(11,3);//Place le curseur colonne 12, ligne 4
   lcd.print (adr26accessoire^1);
   lcd.setCursor(12,3);//Place le curseur colonne 13, ligne 4
   lcd.print (adr25accessoire^1);
   lcd.setCursor(13,3);//Place le curseur colonne 14, ligne 4
   lcd.print (adr24accessoire^1);
   lcd.setCursor(14,3);//Place le curseur colonne 15, ligne 4
   lcd.print (adr23accessoire);
   lcd.setCursor(15,3);//Place le curseur colonne 16, ligne 4
   lcd.print (adr22accessoire);
   lcd.setCursor(16,3);//Place le curseur colonne 17, ligne 4
   lcd.print (adr21accessoire);
   lcd.setCursor(17,3);//Place le curseur colonne 18, ligne 4
   lcd.print (adr20accessoire);

for(int compteurtempo = 0; compteurtempo < dureetempoaccessoire; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
```

```
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

 lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_accessoire);

 ////Affiche le nombre adresse de départ en binaire 2ième affichage
   lcd.setCursor(1,1);//Place le curseur colonne 2, ligne 4
   lcd.print (adr7accessoire);
   lcd.setCursor(2,1);//Place le curseur colonne 3, ligne 4
   lcd.print (adr6accessoire);
   lcd.setCursor(3,1);//Place le curseur colonne 4, ligne 4
   lcd.print (adr5accessoire);
   lcd.setCursor(4,1);//Place le curseur colonne 5, ligne 4
   lcd.print (adr4accessoire);
   lcd.setCursor(5,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr3accessoire);
   lcd.setCursor(6,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr2accessoire);
   lcd.setCursor(7,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr1accessoire);
   lcd.setCursor(8,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr0accessoire);

   lcd.setCursor(9,1);//Place le curseur colonne 10, ligne 4
   lcd.print (" ");//Efface le numéro d'adresse rentré

   lcd.setCursor(10,1);//Place le curseur colonne 11, ligne 4
   lcd.print (adr27accessoire);
   lcd.setCursor(11,1);//Place le curseur colonne 12, ligne 4
   lcd.print (adr26accessoire^1);
   lcd.setCursor(12,1);//Place le curseur colonne 13, ligne 4
   lcd.print (adr25accessoire^1);
   lcd.setCursor(13,1);//Place le curseur colonne 14, ligne 4
   lcd.print (adr24accessoire^1);
   lcd.setCursor(14,1);//Place le curseur colonne 15, ligne 4
   lcd.print (adr23accessoire);
```

```
   lcd.setCursor(15,1);//Place le curseur colonne 16, ligne 4
   lcd.print (adr22accessoire);
   lcd.setCursor(16,1);//Place le curseur colonne 17, ligne 4
   lcd.print (adr21accessoire);
   lcd.setCursor(17,1);//Place le curseur colonne 18, ligne 4
   lcd.print (adr20accessoire);

  lcd.setCursor (0,2);
lcd.print ("Saisir donnee :   D");//Rappel appui touche D pour Valider

lcd.setCursor (9,3);
lcd.blink ();

//Gestion des claviers
autorise_touche_accessoire4 = 1;//Autorise la prise en compte de l'appui touche + et autorise_touche_accessoire4 à 1

   }//fin du else

}//Fin du if compteur_touche_accessoire == 2
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
if (compteur_touche_accessoire == 3)
{
  nbre_accessoire = ((toucheAaccessoire-48) * 100) + ((toucheBaccessoire-48)* 10)
  + toucheCaccessoire-48;//nbre contient l'intégralité de la saisie}

if (nbre_accessoire == 0 || nbre_accessoire > 510)//Prise en compte  si 1 chiffre saisi
 {
   lcd.clear(); // Efface écran si appui
   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

   lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

   lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

   lcd.print("doit etre compris") ; // Affiche la chaîne texte

   lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

   lcd.print("entre 1 et 510") ; // Affiche la chaîne texte
```

```
    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    lcd.noBlink();//Empêche le clignotement du curseur

    for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
    //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
     trame_idle();
    }//fermeture du for

    retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_programmation_accessoire == 0)

else
{

    compteur_touche_accessoire = 0;//Remet le compteur à 0 pour prochaine saisie

    envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

    adresse_accessoire = nbre_accessoire;//recopie dans adresse la valeur de nbre en binaire

    //Transforme la variable adresse en binaire et stocke dans les variables adr
    //Octet 1 d'adresse
    adr7accessoire = 1;//Adresse accessoire
    adr6accessoire = 0;//Adresse accessoire
    adr5accessoire = bitRead (adresse_accessoire, 5);
    adr4accessoire = bitRead (adresse_accessoire, 4);
    adr3accessoire = bitRead (adresse_accessoire, 3);
    adr2accessoire = bitRead (adresse_accessoire, 2);
    adr1accessoire = bitRead (adresse_accessoire, 1);
    adr0accessoire = bitRead (adresse_accessoire, 0);

//Octet 2 d'adresse

    adr27accessoire = 1;//ACCESSOIRE
    adr26accessoire = bitRead (adresse_accessoire, 8);//ACCESSOIRE complément à 1
    adr25accessoire = bitRead (adresse_accessoire, 7);//ACCESSOIRE complément à 1
```

```
    adr24accessoire = bitRead (adresse_accessoire, 6);//ACCESSOIRE complément à 1



//Affiche le nombre adresse de départ en binaire
    lcd.setCursor(1,3);//Place le curseur colonne 2, ligne 4
    lcd.print (adr7accessoire);
    lcd.setCursor(2,3);//Place le curseur colonne 3, ligne 4
    lcd.print (adr6accessoire);
    lcd.setCursor(3,3);//Place le curseur colonne 4, ligne 4
    lcd.print (adr5accessoire);
    lcd.setCursor(4,3);//Place le curseur colonne 5, ligne 4
    lcd.print (adr4accessoire);
    lcd.setCursor(5,3);//Place le curseur colonne 6, ligne 4
    lcd.print (adr3accessoire);
    lcd.setCursor(6,3);//Place le curseur colonne 7, ligne 4
    lcd.print (adr2accessoire);
    lcd.setCursor(7,3);//Place le curseur colonne 8, ligne 4
    lcd.print (adr1accessoire);
    lcd.setCursor(8,3);//Place le curseur colonne 9, ligne 4
    lcd.print (adr0accessoire);

    lcd.setCursor(9,3);//Place le curseur colonne 10, ligne 4
    lcd.print (" ");//Efface le numéro d'adresse rentré

    lcd.setCursor(10,3);//Place le curseur colonne 11, ligne 4
    lcd.print (adr27accessoire);
    lcd.setCursor(11,3);//Place le curseur colonne 12, ligne 4
    lcd.print (adr26accessoire^1);
    lcd.setCursor(12,3);//Place le curseur colonne 13, ligne 4
    lcd.print (adr25accessoire^1);
    lcd.setCursor(13,3);//Place le curseur colonne 14, ligne 4
    lcd.print (adr24accessoire^1);
    lcd.setCursor(14,3);//Place le curseur colonne 15, ligne 4
    lcd.print (adr23accessoire);
    lcd.setCursor(15,3);//Place le curseur colonne 16, ligne 4
    lcd.print (adr22accessoire);
    lcd.setCursor(16,3);//Place le curseur colonne 17, ligne 4
    lcd.print (adr21accessoire);
    lcd.setCursor(17,3);//Place le curseur colonne 18, ligne 4
    lcd.print (adr20accessoire);
```

```
for(int compteurtempo = 0; compteurtempo < dureetempoaccessoire; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

 lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_accessoire);

 ////Affiche le nombre adresse de départ en binaire 2ième affichage
   lcd.setCursor(1,1);//Place le curseur colonne 2, ligne 4
   lcd.print (adr7accessoire);
   lcd.setCursor(2,1);//Place le curseur colonne 3, ligne 4
   lcd.print (adr6accessoire);
   lcd.setCursor(3,1);//Place le curseur colonne 4, ligne 4
   lcd.print (adr5accessoire);
   lcd.setCursor(4,1);//Place le curseur colonne 5, ligne 4
   lcd.print (adr4accessoire);
   lcd.setCursor(5,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr3accessoire);
   lcd.setCursor(6,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr2accessoire);
   lcd.setCursor(7,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr1accessoire);
   lcd.setCursor(8,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr0accessoire);

   lcd.setCursor(9,1);//Place le curseur colonne 10, ligne 4
   lcd.print (" ");//Efface le numéro d'adresse rentré

   lcd.setCursor(10,1);//Place le curseur colonne 11, ligne 4
   lcd.print (adr27accessoire);
   lcd.setCursor(11,1);//Place le curseur colonne 12, ligne 4
   lcd.print (adr26accessoire^1);
   lcd.setCursor(12,1);//Place le curseur colonne 13, ligne 4
   lcd.print (adr25accessoire^1);
   lcd.setCursor(13,1);//Place le curseur colonne 14, ligne 4
   lcd.print (adr24accessoire^1);
```

```
   lcd.setCursor(14,1);//Place le curseur colonne 15, ligne 4
   lcd.print (adr23accessoire);
   lcd.setCursor(15,1);//Place le curseur colonne 16, ligne 4
   lcd.print (adr22accessoire);
   lcd.setCursor(16,1);//Place le curseur colonne 17, ligne 4
   lcd.print (adr21accessoire);
   lcd.setCursor(17,1);//Place le curseur colonne 18, ligne 4
   lcd.print (adr20accessoire);

  lcd.setCursor (0,2);
lcd.print ("Saisir donnee :   D");//Rappel appui touche D pour Valider

lcd.setCursor (9,3);
lcd.blink ();

//Gestion des claviers
autorise_touche_accessoire4 = 1;//Autorise la prise en compte de l'appui touche + et autorise_touche_accessoire4 à 1

   }//fin du else

}//Fin if (compteur_touche_accessoire == 3)
}//Fin du VOID TRAITEMENT APPUI TOUCHE ACCESSOIRE
////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////// ////////////////////////////////////////////////////////////
////////////////////////// MISE EN MEMOIRE DES TOUCHES APPUYEES ACCESSOIRE2////////////////////////////
//////////////////////////////TOUCHE APPUYEE NUMERIQUE/////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_accessoire4() //Stockage touche appuyée
{

compteur_touche_accessoire4 = compteur_touche_accessoire4 + 1;//Incrémente la variable compteur_touche

if (compteur_touche_accessoire4 == 1) //Teste si compteur_touche = 1
{toucheAaccessoire4 = touche;
lcd.blink ();

autorise_touche_accessoire4_e = 1;//Autorise la prise en compte par la touche E
autorise_touche_accessoire44 = 1;//Autorise la rentrée dans le menu commande accessoire touche D

}//Garde trace de la touche 1ere touche appuyée
```

```
if (compteur_touche_accessoire4 == 2) //Teste si compteur_touche = 2
{toucheBaccessoire4 = touche;
lcd.noBlink ();

autorise_touche_accessoire4 = 0;

}//Garde trace de la touche 2ième touche appuyée

}//Fin du VOID MISE EN MEMOIRE DES TOUCHES APPUYEES

////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////Choix de la Commande de l'aiguille et de la sortie//////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
 void traitement_appui_touche_accessoire4()
{
  if (compteur_touche_accessoire4 == 1)//nbre_accessoire4 contient l'intégralité de la saisie si 1 chiffre
   {
    nbre_accessoire4 = (toucheAaccessoire4-48);
    compteur_touche_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

if (nbre_accessoire4 == 0)//Prise en compte  si 1 chiffre saisi
 {
    lcd.clear(); // Efface écran si appui
    lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

    lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

    lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

    lcd.print("doit etre compris") ; // Affiche la chaîne texte

    lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

    lcd.print("entre 1 et 510") ; // Affiche la chaîne texte

    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

    lcd.noBlink();//Empêche le clignotement du curseur
```

```
  for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
  //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
  //et permet une temporisation
  {
   trame_idle();
  }//fermeture du for

  retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_accessoire4 == 0)
}//Fin du if (compteur_touche_accessoire4 == 1)



 if (compteur_touche_accessoire4 == 2)
 {
  nbre_accessoire4 = ((toucheAaccessoire4-48) * 10) + (toucheBaccessoire4-48);
 //nbre_accessoire4 contient l'intégralité de la saisie si 2 chiffres

  if (nbre_accessoire4 == 0)//Prise en compte  si 1 chiffre saisi
{
  lcd.clear(); // Efface écran si appui
  lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1

  lcd.print("Nbre Interdit") ; // Affiche la chaîne texte

  lcd.setCursor(1,1);//Place le curseur colonne 0, ligne 1

  lcd.print("doit etre compris") ; // Affiche la chaîne texte

  lcd.setCursor(3,2);//Place le curseur colonne 0, ligne 1

  lcd.print("entre 1 et 510") ; // Affiche la chaîne texte

  tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

  lcd.noBlink();//Empêche le clignotement du curseur

  for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
  //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
  //et permet une temporisation
```

```
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin de if (nbre_accessoire == 0)

else
{

   compteur_touche_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

   envoi_raz();//Envoi raz sans "En ATTENTE APPUI TOUCHE"

 commande_accessoire4 = nbre_accessoire4;//recopie dans adresse la valeur de nbre en binaire

  lcd.clear();
lcd.setCursor (0,0);
lcd.print ("Adresse depart : ");
lcd.print(nbre_accessoire);
//affiche la variable nbre_accessoire. Chiffre saisi pour l'adresse de départ

////Affiche le nombre adresse de départ en binaire 2ième affichage
   lcd.setCursor(1,1);//Place le curseur colonne 2, ligne 4
   lcd.print (adr7accessoire);
   lcd.setCursor(2,1);//Place le curseur colonne 3, ligne 4
   lcd.print (adr6accessoire);
   lcd.setCursor(3,1);//Place le curseur colonne 4, ligne 4
   lcd.print (adr5accessoire);
   lcd.setCursor(4,1);//Place le curseur colonne 5, ligne 4
   lcd.print (adr4accessoire);
   lcd.setCursor(5,1);//Place le curseur colonne 6, ligne 4
   lcd.print (adr3accessoire);
   lcd.setCursor(6,1);//Place le curseur colonne 7, ligne 4
   lcd.print (adr2accessoire);
   lcd.setCursor(7,1);//Place le curseur colonne 8, ligne 4
   lcd.print (adr1accessoire);
   lcd.setCursor(8,1);//Place le curseur colonne 9, ligne 4
   lcd.print (adr0accessoire);
```

```
  lcd.setCursor(9,1);//Place le curseur colonne 10, ligne 4
  lcd.print (" ");//Efface le numéro d'adresse rentré

  lcd.setCursor(10,1);//Place le curseur colonne 11, ligne 4
  lcd.print (adr27accessoire);
  lcd.setCursor(11,1);//Place le curseur colonne 12, ligne 4
  lcd.print (adr26accessoire^1);
  lcd.setCursor(12,1);//Place le curseur colonne 13, ligne 4
  lcd.print (adr25accessoire^1);
  lcd.setCursor(13,1);//Place le curseur colonne 14, ligne 4
  lcd.print (adr24accessoire^1);
  lcd.setCursor(14,1);//Place le curseur colonne 15, ligne 4
  lcd.print (adr23accessoire);
  lcd.setCursor(15,1);//Place le curseur colonne 16, ligne 4
  lcd.print (adr22accessoire);
  lcd.setCursor(16,1);//Place le curseur colonne 17, ligne 4
  lcd.print (adr21accessoire);
  lcd.setCursor(17,1);//Place le curseur colonne 18, ligne 4
  lcd.print (adr20accessoire);

lcd.setCursor (0,2);
lcd.print ("Donnee saisie :");
lcd.setCursor (16,2);
lcd.print(nbre_accessoire4);

//Transforme la donnée saisie en binaire et stocke dans les variables

  commande7accessoire4 = bitRead (commande_accessoire4, 7);
  commande6accessoire4 = bitRead (commande_accessoire4, 6);
  commande5accessoire4 = bitRead (commande_accessoire4, 5);
  commande4accessoire4 = bitRead (commande_accessoire4, 4);
  commande3accessoire4 = bitRead (commande_accessoire4, 3);
  commande2accessoire4 = bitRead (commande_accessoire4, 2);
  commande1accessoire4 = bitRead (commande_accessoire4, 1);
  commande0accessoire4 = bitRead (commande_accessoire4, 0);

  //Affichage de nbre_accessoire4 en binaire pour le Fun. La commande est après transformée.
  lcd.setCursor(6,3);//Place le curseur colonne 6, ligne 4
  lcd.print (commande7accessoire4);
  lcd.setCursor(7,3);//Place le curseur colonne 7, ligne 4
  lcd.print (commande6accessoire4);
```

```
   lcd.setCursor(8,3);//Place le curseur colonne 8, ligne 4
   lcd.print (commande5accessoire4);
   lcd.setCursor(9,3);//Place le curseur colonne 9, ligne 4
   lcd.print (commande4accessoire4);
   lcd.setCursor(10,3);//Place le curseur colonne 10, ligne 4
   lcd.print (commande3accessoire4);
   lcd.setCursor(11,3);//Place le curseur colonne 11, ligne 4
   lcd.print (commande2accessoire4);
   lcd.setCursor(12,3);//Place le curseur colonne 12, ligne 4
   lcd.print (commande1accessoire4);
   lcd.setCursor(13,3);//Place le curseur colonne 13, ligne 4
   lcd.print (commande0accessoire4);

for(int compteurtempo = 0; compteurtempo < dureetempoaccessoire; compteurtempo++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for
}
}

if (nbre_accessoire4 == 10)//Aiguille = 1, Active = 1, Position = Déviée
{
//Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commande7accessoire4 = 1;//Adresse codée sur 9 bits

   commande3accessoire4 = 1;//Active la sortie
   commande2accessoire4 = 0;//Sélectionne le groupe 1 du décodeur
   commande1accessoire4 = 0;//Sélectionne le groupe 1 du décodeur
   commande0accessoire4 = bitRead (commande_accessoire4, 0);//Sélection de la sortie

envoi_trame_commande_accessoire ();//Pour que l'ordre soit exécuté immédiatement
//Ensuite à la fin du void commade_accessoire le programme revient ici pour affichage
//De la commande exécutée

lcd.clear();
lcd.setCursor (3,0);
lcd.print ("ACCESSOIRE : 1");
lcd.setCursor (4,1);
```

```
lcd.print ("Sortie OUT A");
lcd.setCursor (7,2);
lcd.print ("Active");
lcd.noBlink ();
/*
lcd.clear();
lcd.setCursor (4,0);
lcd.print ("Aiguille = 1");
lcd.setCursor (6,1);
lcd.print ("Active = 1");
lcd.setCursor (0,2);
lcd.print ("Position = Deviee");
lcd.noBlink ();
*/
for(int compteur = 0; compteur < dureetempoaccessoire; compteur++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear ();// Efface l'écran pour préparer l'affichage au retour de la loc sélectionnée

selection_automatique_loc_apres_commande_accessoire ();
//Appel du void pour retour à la commande de la loc initiale (avant la commande de fonction)

}//Fin du if (nbre_accessoire4 == 10)


if (nbre_accessoire4 == 11)//Aiguille = 1, Active = 1, Position = Droit
{
//Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commande7accessoire4 = 1;//Adresse codée sur 9 bits

   commande3accessoire4 = 1;//Active la sortie
   commande2accessoire4 = 0;//Sélectionne le groupe 1 du décodeur
   commande1accessoire4 = 0;//Sélectionne le groupe 1 du décodeur
   commande0accessoire4 = bitRead (commande_accessoire4, 0);//Sélection de la sortie

envoi_trame_commande_accessoire ();//Pour que l'ordre soit exécuté immédiatement
```

```
//Ensuite à la fin du void commade_accessoire le programme revient ici pour affichage
//De la commande exécutée

lcd.clear();
lcd.setCursor (3,0);
lcd.print ("ACCESSOIRE : 1");
lcd.setCursor (4,1);
lcd.print ("Sortie OUT B");
lcd.setCursor (7,2);
lcd.print ("Active");
lcd.noBlink ();
/*
lcd.clear();
lcd.setCursor (4,0);
lcd.print ("Aiguille = 1");
lcd.setCursor (6,1);
lcd.print ("Active = 1");
lcd.setCursor (0,2);
lcd.print ("Position = Deviee");
lcd.noBlink ();
*/

for(int compteur = 0; compteur < dureetempoaccessoire; compteur++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear ();// Efface l'écran pour préparer l'affichage au retour de la loc sélectionnée

selection_automatique_loc_apres_commande_accessoire ();
//Appel du void pour retour à la commande de la loc initiale (avant la commande de fonction)

}//Fin du if (nbre_accessoire4 == 11)



if (nbre_accessoire4 == 20)//Aiguille = 2, Active = 1, Position = Déviée
{
//Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
```

```
   commande7accessoire4 = 1;//Adresse codée sur 9 bits

   commande3accessoire4 = 1;//Active la sortie
   commande2accessoire4 = 0;//Sélectionne le groupe 1 du décodeur
   commande1accessoire4 = 1;//Sélectionne le groupe 1 du décodeur
   commande0accessoire4 = bitRead (commande_accessoire4, 0);//Sélection de la sortie

envoi_trame_commande_accessoire ();//Pour que l'ordre soit exécuté immédiatement
//Ensuite à la fin du void commade_accessoire le programme revient ici pour affichage
//De la commande exécutée

lcd.clear();
lcd.setCursor (3,0);
lcd.print ("ACCESSOIRE : 2");
lcd.setCursor (4,1);
lcd.print ("Sortie OUT A");
lcd.setCursor (7,2);
lcd.print ("Active");
lcd.noBlink ();
/*
lcd.clear();
lcd.setCursor (4,0);
lcd.print ("Aiguille = 1");
lcd.setCursor (6,1);
lcd.print ("Active = 1");
lcd.setCursor (0,2);
lcd.print ("Position = Deviee");
lcd.noBlink ();
*/

for(int compteur = 0; compteur < dureetempoaccessoire; compteur++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear ();// Efface l'écran pour préparer l'affichage au retour de la loc sélectionnée

selection_automatique_loc_apres_commande_accessoire ();
//Appel du void pour retour à la commande de la loc initiale (avant la commande de fonction)
```

```
}//Fin du if (nbre_accessoire4 == 20)



if (nbre_accessoire4 == 21)//Aiguille = 2, Active = 1, Position = Droit
{
//Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commande7accessoire4 = 1;//Adresse codée sur 9 bits

   commande3accessoire4 = 1;//Active la sortie
   commande2accessoire4 = 0;//Sélectionne le groupe 1 du décodeur
   commande1accessoire4 = 1;//Sélectionne le groupe 1 du décodeur
   commande0accessoire4 = bitRead (commande_accessoire4, 0);//Sélection de la sortie

envoi_trame_commande_accessoire ();//Pour que l'ordre soit exécuté immédiatement
//Ensuite à la fin du void commade_accessoire le programme revient ici pour affichage
//De la commande exécutée

lcd.clear();
lcd.setCursor (3,0);
lcd.print ("ACCESSOIRE : 2");
lcd.setCursor (4,1);
lcd.print ("Sortie OUT B");
lcd.setCursor (7,2);
lcd.print ("Active");
lcd.noBlink ();
/*
lcd.clear();
lcd.setCursor (4,0);
lcd.print ("Aiguille = 1");
lcd.setCursor (6,1);
lcd.print ("Active = 1");
lcd.setCursor (0,2);
lcd.print ("Position = Deviee");
lcd.noBlink ();
*/

for(int compteur = 0; compteur < dureetempoaccessoire; compteur++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
```

```
   {
    trame_idle();
   }//fermeture du for

lcd.clear ();// Efface l'écran pour préparer l'affichage au retour de la loc sélectionnée

selection_automatique_loc_apres_commande_accessoire ();
//Appel du void pour retour à la commande de la loc initiale (avant la commande de fonction)

}//Fin du if (nbre_accessoire4 == 21)


if (nbre_accessoire4 == 30)//Aiguille = 3, Active = 1, Position = Déviée
{
//Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commande7accessoire4 = 1;//Adresse codée sur 9 bits

   commande3accessoire4 = 1;//Active la sortie
   commande2accessoire4 = 1;//Sélectionne le groupe 1 du décodeur
   commande1accessoire4 = 0;//Sélectionne le groupe 1 du décodeur
   commande0accessoire4 = bitRead (commande_accessoire4, 0);//Sélection de la sortie

envoi_trame_commande_accessoire ();//Pour que l'ordre soit exécuté immédiatement
//Ensuite à la fin du void commade_accessoire le programme revient ici pour affichage
//De la commande exécutée

lcd.clear();
lcd.setCursor (3,0);
lcd.print ("ACCESSOIRE : 3");
lcd.setCursor (4,1);
lcd.print ("Sortie OUT A");
lcd.setCursor (7,2);
lcd.print ("Active");
lcd.noBlink ();
/*
lcd.clear();
lcd.setCursor (4,0);
lcd.print ("Aiguille = 1");
lcd.setCursor (6,1);
lcd.print ("Active = 1");
```

```
lcd.setCursor (0,2);
lcd.print ("Position = Deviee");
lcd.noBlink ();
*/

for(int compteur = 0; compteur < dureetempoaccessoire; compteur++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear ();// Efface l'écran pour préparer l'affichage au retour de la loc sélectionnée

selection_automatique_loc_apres_commande_accessoire ();
//Appel du void pour retour à la commande de la loc initiale (avant la commande de fonction)

}//Fin du if (nbre_accessoire4 == 30)


if (nbre_accessoire4 == 31)//Aiguille = 3, Active = 1, Position = Droit
{
//Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commande7accessoire4 = 1;//Adresse codée sur 9 bits

   commande3accessoire4 = 1;//Active la sortie
   commande2accessoire4 = 1;//Sélectionne le groupe 1 du décodeur
   commande1accessoire4 = 0;//Sélectionne le groupe 1 du décodeur
   commande0accessoire4 = bitRead (commande_accessoire4, 0);//Sélection de la sortie

envoi_trame_commande_accessoire ();//Pour que l'ordre soit exécuté immédiatement
//Ensuite à la fin du void commade_accessoire le programme revient ici pour affichage
//De la commande exécutée

lcd.clear();
lcd.setCursor (3,0);
lcd.print ("ACCESSOIRE : 3");
lcd.setCursor (4,1);
lcd.print ("Sortie OUT B");
lcd.setCursor (7,2);
```

```
lcd.print ("Active");
lcd.noBlink ();
/*
lcd.clear();
lcd.setCursor (4,0);
lcd.print ("Aiguille = 1");
lcd.setCursor (6,1);
lcd.print ("Active = 1");
lcd.setCursor (0,2);
lcd.print ("Position = Deviee");
lcd.noBlink ();
*/

for(int compteur = 0; compteur < dureetempoaccessoire; compteur++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear ();// Efface l'écran pour préparer l'affichage au retour de la loc sélectionnée

selection_automatique_loc_apres_commande_accessoire ();
//Appel du void pour retour à la commande de la loc initiale (avant la commande de fonction)

}//Fin du if (nbre_accessoire4 == 31)

if (nbre_accessoire4 == 40)//Aiguille = 4, Active = 1, Position = Déviée
{
//Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commande7accessoire4 = 1;//Adresse codée sur 9 bits

   commande3accessoire4 = 1;//Active la sortie
   commande2accessoire4 = 1;//Sélectionne le groupe 1 du décodeur
   commande1accessoire4 = 1;//Sélectionne le groupe 1 du décodeur
   commande0accessoire4 = bitRead (commande_accessoire4, 0);//Sélection de la sortie

envoi_trame_commande_accessoire ();//Pour que l'ordre soit exécuté immédiatement
//Ensuite à la fin du void commade_accessoire le programme revient ici pour affichage
//De la commande exécutée
```

```
lcd.clear();
lcd.setCursor (3,0);
lcd.print ("ACCESSOIRE : 4");
lcd.setCursor (4,1);
lcd.print ("Sortie OUT A");
lcd.setCursor (7,2);
lcd.print ("Active");
lcd.noBlink ();
/*
lcd.clear();
lcd.setCursor (4,0);
lcd.print ("Aiguille = 1");
lcd.setCursor (6,1);
lcd.print ("Active = 1");
lcd.setCursor (0,2);
lcd.print ("Position = Deviee");
lcd.noBlink ();
*/

for(int compteur = 0; compteur < dureetempoaccessoire; compteur++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear ();// Efface l'écran pour préparer l'affichage au retour de la loc sélectionnée

selection_automatique_loc_apres_commande_accessoire ();
//Appel du void pour retour à la commande de la loc initiale (avant la commande de fonction)

}//Fin du if (nbre_accessoire4 == 40)



if (nbre_accessoire4 == 41)//Aiguille = 4, Active = 1, Position = Droit
{
//Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   commande7accessoire4 = 1;//Adresse codée sur 9 bits
```

```
   commande3accessoire4 = 1;//Active la sortie
   commande2accessoire4 = 1;//Sélectionne le groupe 1 du décodeur
   commande1accessoire4 = 1;//Sélectionne le groupe 1 du décodeur
   commande0accessoire4 = bitRead (commande_accessoire4, 0);//Sélection de la sortie

envoi_trame_commande_accessoire ();//Pour que l'ordre soit exécuté immédiatement
//Ensuite à la fin du void commade_accessoire le programme revient ici pour affichage
//De la commande exécutée

lcd.clear();
lcd.setCursor (3,0);
lcd.print ("ACCESSOIRE : 4");
lcd.setCursor (4,1);
lcd.print ("Sortie OUT B");
lcd.setCursor (7,2);
lcd.print ("Active");
lcd.noBlink ();
/*
lcd.clear();
lcd.setCursor (4,0);
lcd.print ("Aiguille = 1");
lcd.setCursor (6,1);
lcd.print ("Active = 1");
lcd.setCursor (0,2);
lcd.print ("Position = Deviee");
lcd.noBlink ();
*/

for(int compteur = 0; compteur < dureetempoaccessoire; compteur++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear ();// Efface l'écran pour préparer l'affichage au retour de la loc sélectionnée

selection_automatique_loc_apres_commande_accessoire ();
//Appel du void pour retour à la commande de la loc initiale (avant la commande de fonction)

}//Fin du if (nbre_accessoire4 == 41)
```

```
//}//Fin du If (compteur_touche_accessoire4 == 3)

if (nbre_accessoire4 != 0)//Si different de zéro
{
if (nbre_accessoire4 != 10 && nbre_accessoire4 != 11 && nbre_accessoire4 != 20
&& nbre_accessoire4 != 21 && nbre_accessoire4 != 30 && nbre_accessoire4 != 31
&& nbre_accessoire4 != 40 && nbre_accessoire4 != 41)

{
   autorise_touche_locomotive = 0;
   //Interdit la prise en compte de l'appui touche E et programmation locomotiv à 0
   autorise_touche_fonction = 0;
   //Interdit la prise en compte de l'appui touche E et programmation fonction à 0
   //autorise_touche_Cv = 0;//Interdit la prise en compte de l'appui touche E et programmation cv à 0
   //autorise_touche_Cv2 = 0;//Interdit la prise en compte de l'appui touche E et programmation cv2 à 0
   //autorise_touche_Cv4 = 0;//Interdit la prise en compte de l'appui touche E et programmation cv4 à 0
   autorise_touche_accessoire = 0;
   //Interdit la prise en compte de l'appui touche E et programmation accessoire à 0
   autorise_touche_accessoire4 = 0;
   //Autorise la prise en compte de l'appui touche pour la touche 'D' de fonction

   compteur_touche_accessoire4 = 0;//Remet le compteur à 0 pour prochaine saisie

   toucheAaccessoire4 = 0;//Réinitialise toucheA
   toucheBaccessoire4 = 0;//Réinitialise toucheB
   toucheCaccessoire4 = 0;//Réinitialise toucheC
   toucheDaccessoire4 = 0;//Réinitialise toucheD

   lcd.clear(); // Efface écran si appui
   lcd.setCursor(0,1);//Place le curseur colonne 0, ligne 1

   lcd.print("Taille Nbre Interdit") ; // Affiche la chaîne texte
   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempoaccessoire; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
```

```
   }//fermeture du for
  selection_automatique_loc_apres_commande_accessoire ();
  //Appel du void pour retour à la commande de la loc initiale (avant la commande de fonction)

}//Fin du if (nbre_accessoire4 != 10 || nbre_accessoire4 != 11...
}//Fin du if (nbre_accessoire4 != 0)//Si different de zéro
}// Fin du void traitement_appui_touche_accessoire4()

////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
void envoi_trame_programmation_accessoire()
{
  PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)
  PORTA |=  (1<<1);//Equivalent à digitalWrite(Ledbleue,HIGH)
  PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
  PORTK |=  (1<<4);//Equivalent à digitalWrite(Ledjaune,HIGH)

  for(int compteur = 0; compteur < 4; compteur++)
  //Envoi 5 fois la Fonction pour sécuriser la réalisation de l'ordre
{
    //Bit de Synchronisation 16 bit à 1
// Octet de synchronisation
 for ( i=0; i <= 15; i++)
 {
  bitun();    // La centrale transmet 16 bits à 1
  }

//////////////////////////////////////
//////////////////////////////////////
//Bit de séparation
//bit zero
//////////////////////////////////////////
bitzero(); // Bit à 0 de séparation

//////////////////////////////////////
//////////////////////////////////////
//Octet de d'adresse de départ

if (adr7programmation_accessoire==1) bitun();  // Si bit à 1
```

```
if (adr7programmation_accessoire==0) bitzero(); // Si bit à 0

if (adr6programmation_accessoire==1) bitun();  // Si bit à 1
if (adr6programmation_accessoire==0) bitzero(); // Si bit à 0
if (adr5programmation_accessoire==1) bitun();  // Si bit à 1
if (adr5programmation_accessoire==0) bitzero(); // Si bit à 0
if (adr4programmation_accessoire==1) bitun();  // Si bit à 1
if (adr4programmation_accessoire==0) bitzero(); // Si bit à 0
if (adr3programmation_accessoire==1) bitun();  // Si bit à 1
if (adr3programmation_accessoire==0) bitzero(); // Si bit à 0
if (adr2programmation_accessoire==1) bitun();  // Si bit à 1
if (adr2programmation_accessoire==0) bitzero(); // Si bit à 0
if (adr1programmation_accessoire==1) bitun();  // Si bit à 1
if (adr1programmation_accessoire==0) bitzero(); // Si bit à 0
if (adr0programmation_accessoire==1) bitun();  // Si bit à 1
if (adr0programmation_accessoire==0) bitzero(); // Si bit à 0

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

////////////////////////////////////
////////////////////////////////////
//Octet de d'adresse de départ

if (adr27programmation_accessoire==1) bitun();  // Si bit à 1
if (adr27programmation_accessoire==0) bitzero(); // Si bit à 0

if (((adr26programmation_accessoire)^1) == 1) bitun();  // Si bit à 1
if (((adr26programmation_accessoire)^1) == 0) bitzero(); // Si bit à 0
if (((adr25programmation_accessoire)^1) == 1) bitun();  // Si bit à 1
if (((adr25programmation_accessoire)^1) == 0) bitzero(); // Si bit à 0
if (((adr24programmation_accessoire)^1) == 1) bitun();  // Si bit à 1
if (((adr24programmation_accessoire)^1) == 0) bitzero(); // Si bit à 0
if (adr23programmation_accessoire==1) bitun();  // Si bit à 1
if (adr23programmation_accessoire==0) bitzero(); // Si bit à 0
if (adr22programmation_accessoire==1) bitun();  // Si bit à 1
if (adr22programmation_accessoire==0) bitzero(); // Si bit à 0
```

```
if (adr21programmation_accessoire==1) bitun();  // Si bit à 1
if (adr21programmation_accessoire==0) bitzero(); // Si bit à 0
if (adr20programmation_accessoire==1) bitun();  // Si bit à 1
if (adr20programmation_accessoire==0) bitzero(); // Si bit à 0

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

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

//Octet 1 de commande

bitun();  // bit à 1: les 3 premiers bits à 1 pour programmation décodeur accessoire
bitun(); // bit à 1 : les 3 premiers bits à 1 pour programmation décodeur accessoire
bitun();  // bit à 1: les 3 premiers bits à 1 pour programmation décodeur accessoire
bitzero(); // bit à 0 : pour pouvoir adresser toutes les cv accessoire
bitun();  // bit à 1: les deux bits à 1 permettent d'écrire dans la cv accessoire
bitun();  // bit à 1: les deux bits à 1 permettent d'écrire dans la cv accessoire

//choix de la cv

if (commcv9programmation_accessoire4==1) bitun();//Pour adressage 1024 Cv,
//valeur déterminé lors du choix de la Cv accessoire
if (commcv9programmation_accessoire4==0) bitzero();//Pour adressage 1024 Cv,
//valeur déterminé lors du choix de la Cv accessoire

if (commcv8programmation_accessoire4==1) bitun();//Pour adressage 1024 Cv,
//valeur déterminé lors du choix de la Cv accessoire
if (commcv8programmation_accessoire4==0) bitzero();//Pour adressage 1024 Cv,
//valeur déterminé lors du choix de la Cv accessoire

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

```
////////////////////////////////////
////////////////////////////////////
//Octet du numéro de la cv accessoire

if (commcv7programmation_accessoire4==1) bitun();  // Si bit à 1
if (commcv7programmation_accessoire4==0) bitzero(); // Si bit à 0

if (commcv6programmation_accessoire4==1) bitun();  // Si bit à 1
if (commcv6programmation_accessoire4==0) bitzero(); // Si bit à 0
if (commcv5programmation_accessoire4==1) bitun();  // Si bit à 1
if (commcv5programmation_accessoire4==0) bitzero(); // Si bit à 0
if (commcv4programmation_accessoire4==1) bitun();  // Si bit à 1
if (commcv4programmation_accessoire4==0) bitzero(); // Si bit à 0
if (commcv3programmation_accessoire4==1) bitun();  // Si bit à 1
if (commcv3programmation_accessoire4==0) bitzero(); // Si bit à 0
if (commcv2programmation_accessoire4==1) bitun();  // Si bit à 1
if (commcv2programmation_accessoire4==0) bitzero(); // Si bit à 0
if (commcv1programmation_accessoire4==1) bitun();  // Si bit à 1
if (commcv1programmation_accessoire4==0) bitzero(); // Si bit à 0
if (commcv0programmation_accessoire4==1) bitun();  // Si bit à 1
if (commcv0programmation_accessoire4==0) bitzero(); // Si bit à 0

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

////////////////////////////////////
//Octet de données à entrer dans la cv accessoire

if (donnee7programmation_accessoire2==1) bitun();  // Si bit à 1
if (donnee7programmation_accessoire2==0) bitzero(); // Si bit à 0

if (donnee6programmation_accessoire2==1) bitun();  // Si bit à 1
if (donnee6programmation_accessoire2==0) bitzero(); // Si bit à 0
if (donnee5programmation_accessoire2==1) bitun();  // Si bit à 1
if (donnee5programmation_accessoire2==0) bitzero(); // Si bit à 0
if (donnee4programmation_accessoire2==1) bitun();  // Si bit à 1
```

```
if (donnee4programmation_accessoire2==0) bitzero(); // Si bit à 0
if (donnee3programmation_accessoire2==1) bitun();  // Si bit à 1
if (donnee3programmation_accessoire2==0) bitzero(); // Si bit à 0
if (donnee2programmation_accessoire2==1) bitun();  // Si bit à 1
if (donnee2programmation_accessoire2==0) bitzero(); // Si bit à 0
if (donnee1programmation_accessoire2==1) bitun();  // Si bit à 1
if (donnee1programmation_accessoire2==0) bitzero(); // Si bit à 0
if (donnee0programmation_accessoire2==1) bitun();  // Si bit à 1
if (donnee0programmation_accessoire2==0) bitzero(); // Si bit à 0

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

////////////////////////////////////
////////////////////////////////////
//Octet de données 3
//Calcul du OU Exclusif
cont7programmation_accessoire4 = adr7programmation_accessoire ^ adr27programmation_accessoire ^
comm15programmation_accessoire4 ^ commcv7programmation_accessoire4 ^ donnee7programmation_accessoire2 ;
// ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle

cont6programmation_accessoire4 = adr6programmation_accessoire ^ ((adr26programmation_accessoire)^1) ^
comm14programmation_accessoire4 ^ commcv6programmation_accessoire4 ^ donnee6programmation_accessoire2 ;
// ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle

cont5programmation_accessoire4 = adr5programmation_accessoire ^ ((adr25programmation_accessoire)^1) ^
comm13programmation_accessoire4 ^ commcv5programmation_accessoire4 ^ donnee5programmation_accessoire2 ;
// ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle

cont4programmation_accessoire4 = adr4programmation_accessoire ^ ((adr24programmation_accessoire)^1) ^
comm12programmation_accessoire4 ^ commcv4programmation_accessoire4 ^ donnee4programmation_accessoire2 ;
// ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle

cont3programmation_accessoire4 = adr3programmation_accessoire ^ adr23programmation_accessoire ^
comm11programmation_accessoire4 ^ commcv3programmation_accessoire4 ^ donnee3programmation_accessoire2 ;
// ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle
```

```
cont2programmation_accessoire4 = adr2programmation_accessoire ^ adr22programmation_accessoire ^
comm10programmation_accessoire4 ^ commcv2programmation_accessoire4 ^ donnee2programmation_accessoire2 ;
// ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle

cont1programmation_accessoire4 = adr1programmation_accessoire ^ adr21programmation_accessoire ^
commcv9programmation_accessoire4 ^ commcv1programmation_accessoire4 ^ donnee1programmation_accessoire2 ;
// ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle

cont0programmation_accessoire4 = adr0programmation_accessoire ^ adr20programmation_accessoire ^
commcv8programmation_accessoire4 ^ commcv0programmation_accessoire4 ^ donnee0programmation_accessoire2 ;
// ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle



// Envoi de l'octet de contrôle
//La centrale transmet l'octet de contrôle

if (cont7programmation_accessoire4==1) bitun();  // Si bit à 1
if (cont7programmation_accessoire4==0) bitzero(); // Si bit à 0
if (cont6programmation_accessoire4==1) bitun();  // Si bit à 1
if (cont6programmation_accessoire4==0) bitzero(); // Si bit à 0
if (cont5programmation_accessoire4==1) bitun();  // Si bit à 1
if (cont5programmation_accessoire4==0) bitzero(); // Si bit à 0
if (cont4programmation_accessoire4==1) bitun();  // Si bit à 1
if (cont4programmation_accessoire4==0) bitzero(); // Si bit à 0
if (cont3programmation_accessoire4==1) bitun();  // Si bit à 1
if (cont3programmation_accessoire4==0) bitzero(); // Si bit à 0
if (cont2programmation_accessoire4==1) bitun();  // Si bit à 1
if (cont2programmation_accessoire4==0) bitzero(); // Si bit à 0
if (cont1programmation_accessoire4==1) bitun();  // Si bit à 1
if (cont1programmation_accessoire4==0) bitzero(); // Si bit à 0
if (cont0programmation_accessoire4==1) bitun();  // Si bit à 1
if (cont0programmation_accessoire4==0) bitzero(); // Si bit à 0

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

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

```
////////////////////////////////////
// Pause émission de 25 zéros avant nouvelle transmission
////////////////////////////////////////////////


for ( i=0; i<=24; i++)
bitzero();

}//Fermeture du for

lcd.clear(); // Efface écran si appui
   lcd.setCursor(0,0);//Place le curseur colonne 0, ligne 1
   lcd.print("   PROGRAMMATION") ; // Affiche la chaîne texte
   lcd.setCursor(0,2);//Place le curseur colonne 0, ligne 3
   lcd.print("     EFFECTUEE") ; // Affiche la chaîne texte

   lcd.noBlink();//Empêche le clignotement du curseur


   for(int compteurtempo = 0; compteurtempo < dureetempoprogaccess; compteurtempo++)
   //Permet une temporisation
   {
    trame_idle();//Envoi trame idle
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fin void envoi_trame_programmation_accessoire

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
void envoi_trame_commande_accessoire ()
{

  PORTK &=~ (1<<5);//Equivalent à digitalWrite(Ledrouge,LOW)
  PORTA |= (1<<1);//Equivalent à digitalWrite(Ledbleue,HIGH)
  PORTA &=~ (1<<3);//Equivalent à digitalWrite(Ledverte,LOW)
  PORTK |= (1<<4);//Equivalent à digitalWrite(Ledjaune,HIGH)
```

```
 for(int compteur = 0; compteur <6; compteur++)
 //Envoi 5 fois la Fonction pour sécuriser la réalisation de l'ordre
{
   //Bit de Synchronisation 16 bit à 1
// Octet de synchronisation
 for ( i=0; i <= 15; i++)
 {
  bitun();    // La centrale transmet 16 bits à 1
  }

////////////////////////////////////
////////////////////////////////////
//Bit de séparation
//bit zero
////////////////////////////////////////
bitzero(); // Bit à 0 de séparation

////////////////////////////////////
////////////////////////////////////
//Octet de d'adresse de départ

if (adr7accessoire==1) bitun();  // Si bit à 1
if (adr7accessoire==0) bitzero(); // Si bit à 0

if (adr6accessoire==1) bitun();  // Si bit à 1
if (adr6accessoire==0) bitzero(); // Si bit à 0
if (adr5accessoire==1) bitun();  // Si bit à 1
if (adr5accessoire==0) bitzero(); // Si bit à 0
if (adr4accessoire==1) bitun();  // Si bit à 1
if (adr4accessoire==0) bitzero(); // Si bit à 0
if (adr3accessoire==1) bitun();  // Si bit à 1
if (adr3accessoire==0) bitzero(); // Si bit à 0
if (adr2accessoire==1) bitun();  // Si bit à 1
if (adr2accessoire==0) bitzero(); // Si bit à 0
if (adr1accessoire==1) bitun();  // Si bit à 1
if (adr1accessoire==0) bitzero(); // Si bit à 0
if (adr0accessoire==1) bitun();  // Si bit à 1
if (adr0accessoire==0) bitzero(); // Si bit à 0

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

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

////////////////////////////////////
////////////////////////////////////
//Octet 1 de commande

bitun();// bit à 1: Adresse codée sur 9 bits
if (((adr26accessoire)^1) == 1) bitun();  // Si bit à 1
if (((adr26accessoire)^1) == 0) bitzero(); // Si bit à 0
if (((adr25accessoire)^1) == 1) bitun();  // Si bit à 1
if (((adr25accessoire)^1) == 0) bitzero(); // Si bit à 0
if (((adr24accessoire)^1) == 1) bitun();  // Si bit à 1
if (((adr24accessoire)^1) == 0) bitzero(); // Si bit à 0



//choix de la commande
if (commande3accessoire4 == 1) bitun();  //Active la sortie
if (commande3accessoire4 == 0) bitzero();  //Désactive la sortie
if (commande2accessoire4 == 1) bitun();  //Sélectionne le groupe du décodeur
if (commande2accessoire4 == 0) bitzero();  //Sélectionne le groupe du décodeur
if (commande1accessoire4 == 1) bitun();//Sélectionne le groupe du décodeur
if (commande1accessoire4 == 0) bitzero();//Sélectionne le groupe du décodeur
if (commande0accessoire4 == 1) bitun ();//Sélection de la sortie
if (commande0accessoire4 == 0) bitzero ();//Sélection de la sortie

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

////////////////////////////////////
////////////////////////////////////
//Octet de contrôle
//Calcul du OU Exclusif
cont7accessoire = adr7accessoire ^ commande7accessoire4;
// ^ Calcul du OU EXCLUSIF bit 7 Octet de contrôle
```

```
cont6accessoire = adr6accessoire ^ ((adr26accessoire)^1);
// ^ Calcul du OU EXCLUSIF bit 6 Octet de contrôle

cont5accessoire = adr5accessoire ^ ((adr25accessoire)^1);
// ^ Calcul du OU EXCLUSIF bit 5 Octet de contrôle

cont4accessoire = adr4accessoire ^ ((adr24accessoire)^1);
// ^ Calcul du OU EXCLUSIF bit 4 Octet de contrôle

cont3accessoire = adr3accessoire ^ commande3accessoire4;
// ^ Calcul du OU EXCLUSIF bit 3 Octet de contrôle

cont2accessoire = adr2accessoire ^ commande2accessoire4;
// ^ Calcul du OU EXCLUSIF bit 2 Octet de contrôle

cont1accessoire = adr1accessoire ^ commande1accessoire4;
// ^ Calcul du OU EXCLUSIF bit 1 Octet de contrôle

cont0accessoire = adr0accessoire ^ commande0accessoire4;
// ^ Calcul du OU EXCLUSIF bit 0 Octet de contrôle



// Envoi de l'octet de contrôle
//La centrale transmet l'octet de contrôle
if (cont7accessoire==1) bitun();  // Si bit à 1
if (cont7accessoire==0) bitzero(); // Si bit à 0
if (cont6accessoire==1) bitun();  // Si bit à 1
if (cont6accessoire==0) bitzero(); // Si bit à 0
if (cont5accessoire==1) bitun();  // Si bit à 1
if (cont5accessoire==0) bitzero(); // Si bit à 0
if (cont4accessoire==1) bitun();  // Si bit à 1
if (cont4accessoire==0) bitzero(); // Si bit à 0
if (cont3accessoire==1) bitun();  // Si bit à 1
if (cont3accessoire==0) bitzero(); // Si bit à 0
if (cont2accessoire==1) bitun();  // Si bit à 1
if (cont2accessoire==0) bitzero(); // Si bit à 0
if (cont1accessoire==1) bitun();  // Si bit à 1
if (cont1accessoire==0) bitzero(); // Si bit à 0
if (cont0accessoire==1) bitun();  // Si bit à 1
if (cont0accessoire==0) bitzero(); // Si bit à 0
```

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

////////////////////////////////////
////////////////////////////////////
// Pause émission de 25 zéros avant nouvelle transmission
/////////////////////////////////////////////
for ( i=0; i<=24; i++)
bitzero();

}//Fermeture du for(int compteur = 0; compteur < 5; compteur++)



lcd.clear(); // Efface écran si appui

   lcd.setCursor(6,0);//Place le curseur colonne 6, ligne 1
   lcd.print("COMMANDE") ; // Affiche la chaîne texte
   lcd.setCursor(6,2);//Place le curseur colonne 6, ligne 3
   lcd.print("EXECUTEE") ; // Affiche la chaîne texte

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempocommaccess; compteurtempo++)
   //Permet une temporisation
   {
   trame_idle();//Envoi trame idle

   }//fermeture du for

}//Fin void envoi_trame_commande_accessoire

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
/////Permet l'affichage de la locmotive sélectionnée//////////
//////////////en fonction de la variable/////////////////////
////////////locSelectionne ou loc2Selectionne/////////////////
////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////////
void selection_automatique_loc_apres_commande_accessoire ()
{
locSelectionne = Miseenmemoirelocselectionne;
//Permet de retrouver la locomotive sélectionnée après appui sur la touche "D" de fonction
  loc2Selectionne = Miseenmemoireloc2selectionne;
//Car lors de l'appui de cette touche les deux variables loc (2)Selectionne sont remises à zéros



autorise_touche_7_et_9 = 1;
//Ré-autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle
    //Remis à 1 après la fonction demandée pour que seule la touche 3 soit pris en compte


   autorise_touche_1_et_4 = 1;


autorise_touche_accessoire = 0;//Interdit la saisie de chiffre pour commande accessoire
//Seule les touches 7 et 9 sont autorisées

lcd.clear();//Efface l'ecran pour éviter bug affichage

//Au cas ou aucune loc n'est sélectionnée
if (locSelectionne == 0 && loc2Selectionne == 0)
{
lcd.clear();//Place le curseur colonne 0, ligne 1
lcd.setCursor(3,0);//Place le curseur colonne 3, ligne 1
lcd.print("Locomotive non"); // Affiche la chaîne texte
lcd.setCursor(4,1);//Place le curseur colonne 4, ligne 2
lcd.print ("selectionnee");

//Emet 1 bip pour appuyer l'affichage
for(int compteurbuzzer = 0; compteurbuzzer < dureetempobuzzer; compteurbuzzer++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
    {
    tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
    }//fermeture du for buzzer

//Emet la trame idle pendant l'affichage "locomotive non sélectionnée" pour garder la puissance
for(int compteur = 0; compteur < 150; compteur++)
//Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
    //et permet une temporisation
```

```
  {
   trame_idle();
  }//fermeture du for idle

retourdetrame = 1;//En attente appui touche
}//Fin du if (locSelectionne == 0 && loc2Selectionne == 0)
}//Fin du void selection_automatique_loc_apres_commande_accessoire()

////////////////////////////////////////////////////////////////////////
//////////////////////////// ////////////////////////////////////////////
////////MISE EN MEMOIRE DES TOUCHES MEMORISATION LOC//////////////////////
/////////////////POUR MISE EN MEMOIRE EEPROM//////////////////////////////
////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_memorisation_loc() //Stockage touche appuyée
{
 compteur_touche_memorisation_loc = compteur_touche_memorisation_loc + 1;
 //Incrémente la variable compteur_touche

if (compteur_touche_memorisation_loc == 1) //Teste si compteur_touche = 1
{toucheAprogrammation_memorisation_loc = touche;
lcd.blink();

autorise_touche_memorisation_loc_e = 1;//Autorise

}//Garde trace de la touche 1ère touche appuyée

if (compteur_touche_memorisation_loc == 2) //Teste si compteur_touche = 2
{toucheBprogrammation_memorisation_loc = touche;
lcd.blink();}//Garde trace de la touche 2ième touche appuyée

if (compteur_touche_memorisation_loc == 3) //Teste si compteur_touche = 3
{toucheCprogrammation_memorisation_loc = touche;
lcd.blink();}//Garde trace de la touche 3ième touche appuyée

if (compteur_touche_memorisation_loc == 4) //Teste si compteur_touche = 4
{toucheDprogrammation_memorisation_loc = touche;
lcd.blink();}//Garde trace de la touche 4ième touche appuyée

if (compteur_touche_memorisation_loc == 5) //Teste si compteur_touche = 5
{toucheEprogrammation_memorisation_loc = touche;
lcd.blink();}//Garde trace de la touche 5ième touche appuyée
```

```
if (compteur_touche_memorisation_loc == 6) //Teste si compteur_touche = 6
{toucheFprogrammation_memorisation_loc = touche;//Garde trace de la touche 6ième touche appuyée
lcd.noBlink ();

autorise_touche_memorisation_loc = 0;//Interdit la saisie de chiffre supplémentaire

}

}//touche_appuyee_numerique_memorisation_loc

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////// TRAITEMENT DES TOUCHES MEMORISATION LOC//////////////////
//////////////////////////POUR MISE EN MEMOIRE EEPROM///////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_memorisation_loc ()
//Correspond à Appui touche 'E' et tritement_appui_touche_memorisation_loc = 1
{

if (compteur_touche_memorisation_loc == 1)//Prise en compte du chiffre saisi pour une entrée
{
 locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
 loc2Selectionne = 0;//Sélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

 compteur_touche_memorisation_loc = 0;//Réinitialise compteur_touche

  autorise_touche_memorisation_loc = 0;//Interdit la prise en compte de l'appui touche

  toucheAprogrammation_memorisation_loc = 0;//Réinitialise toucheA
  toucheBprogrammation_memorisation_loc = 0;//Réinitialise toucheB
  toucheCprogrammation_memorisation_loc = 0;//Réinitialise toucheC
  toucheDprogrammation_memorisation_loc = 0;//Réinitialise toucheD
  toucheEprogrammation_memorisation_loc = 0;//Réinitialise toucheE
  toucheFprogrammation_memorisation_loc = 0;//Réinitialise toucheF

  lcd.clear(); // Efface écran si appui
  tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

  lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1
```

```
   lcd.print("Nbre INTERDIT") ; // Affiche la chaîne texte

   lcd.setCursor(5,1);//Place le curseur colonne 0, ligne 1
   lcd.print("DOIT ETRE") ; // Affiche la chaîne texte

   lcd.setCursor(2,2);//Place le curseur colonne 0, ligne 1
   lcd.print("SUPERIEUR A 127") ; // Affiche la chaîne texte

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempomemorisationloc; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

} //Fin du if (compteur_touche_memorisation_loc == 1)

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (compteur_touche_memorisation_loc == 2)//Prise en compte du chiffre saisi pour une entrée
{
  locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
  loc2Selectionne = 0;//DéSélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

  compteur_touche_memorisation_loc = 0;//Réinitialise compteur_touche

   autorise_touche_memorisation_loc = 0;//Interdit la prise en compte de l'appui touche

   toucheAprogrammation_memorisation_loc = 0;//Réinitialise toucheA
   toucheBprogrammation_memorisation_loc = 0;//Réinitialise toucheB
   toucheCprogrammation_memorisation_loc = 0;//Réinitialise toucheC
   toucheDprogrammation_memorisation_loc = 0;//Réinitialise toucheD
   toucheEprogrammation_memorisation_loc = 0;//Réinitialise toucheE
   toucheFprogrammation_memorisation_loc = 0;//Réinitialise toucheF
```

```
   lcd.clear(); // Efface écran si appui
   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1
   lcd.print("Nbre INTERDIT") ; // Affiche la chaîne texte

   lcd.setCursor(5,1);//Place le curseur colonne 0, ligne 1
   lcd.print("DOIT ETRE") ; // Affiche la chaîne texte

   lcd.setCursor(2,2);//Place le curseur colonne 0, ligne 1
   lcd.print("SUPERIEUR A 127") ; // Affiche la chaîne texte

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempomemorisationloc; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

} //Fin du if (compteur_touche_memorisation_loc == 2)


////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (compteur_touche_memorisation_loc == 3)//Prise en compte du chiffre saisi pour une entrée
{

  nbre_memorisation_loc = ((toucheAprogrammation_memorisation_loc-48)*100)
  + ((toucheBprogrammation_memorisation_loc-48)*10)
  + (toucheCprogrammation_memorisation_loc-48);
  //nbre contient l'intégralité de la saisie


if (nbre_memorisation_loc < 128)
  {
```

```
 locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
 loc2Selectionne = 0;//DéSélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

 compteur_touche_memorisation_loc = 0;//Réinitialise compteur_touche

  autorise_touche_memorisation_loc = 0;//Interdit la prise en compte de l'appui touche

  toucheAprogrammation_memorisation_loc = 0;//Réinitialise toucheA
  toucheBprogrammation_memorisation_loc = 0;//Réinitialise toucheB
  toucheCprogrammation_memorisation_loc = 0;//Réinitialise toucheC
  toucheDprogrammation_memorisation_loc = 0;//Réinitialise toucheD
  toucheEprogrammation_memorisation_loc = 0;//Réinitialise toucheE
  toucheFprogrammation_memorisation_loc = 0;//Réinitialise toucheF

  lcd.clear(); // Efface écran si appui
  tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

  lcd.setCursor(3,0);//Place le curseur colonne 0, ligne 1
  lcd.print("Nbre INTERDIT") ; // Affiche la chaîne texte

  lcd.setCursor(5,1);//Place le curseur colonne 0, ligne 1
  lcd.print("DOIT ETRE") ; // Affiche la chaîne texte

  lcd.setCursor(2,2);//Place le curseur colonne 0, ligne 1
  lcd.print("SUPERIEUR A 127") ; // Affiche la chaîne texte

  lcd.noBlink();//Empêche le clignotement du curseur

  for(int compteurtempo = 0; compteurtempo < dureetempomemorisationloc; compteurtempo++)
  //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
  //et permet une temporisation
  {
   trame_idle();
  }//fermeture du for

  retourdetrame = 1;//En attente appui touche
  }//if (nbre_memorisation_loc < 128)



if (nbre_memorisation_loc > 127)
```

```
{

  locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive

  loc2Selectionne = 0;//DéSélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

  compteur_touche_memorisation_loc = 0;//Réinitialise compteur_touche

    autorise_touche_memorisation_loc = 0;//Interdit la prise en compte de l'appui touche

    toucheAprogrammation_memorisation_loc = 0;//Réinitialise toucheA
    toucheBprogrammation_memorisation_loc = 0;//Réinitialise toucheB
    toucheCprogrammation_memorisation_loc = 0;//Réinitialise toucheC
    toucheDprogrammation_memorisation_loc = 0;//Réinitialise toucheD
    toucheEprogrammation_memorisation_loc = 0;//Réinitialise toucheE
    toucheFprogrammation_memorisation_loc = 0;//Réinitialise toucheF

  lcd.clear(); // Efface écran

lcd.setCursor (0,0);
lcd.print ("N locomotive : ");
lcd.setCursor (8,1);
lcd.print(nbre_memorisation_loc);

  lcd.setCursor (0,2);
lcd.print ("Saisir l'adresse :");

lcd.setCursor (8,3);

//Gestion des claviers
autorise_touche_memorisation_adresse_loc = 1;
//Autorise la prise en compte de l'appui touche + et autorise_touche_memorisation_adresse_loc à 1

}//if (nbre_memorisation_loc > 127)
} //Fin du if (compteur_touche_memorisation_loc == 3)

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (compteur_touche_memorisation_loc == 4)//Prise en compte du chiffre saisi pour une entrée
{
```

```
 nbre_memorisation_loc = ((toucheAprogrammation_memorisation_loc-48)*1000)
 + ((toucheBprogrammation_memorisation_loc-48)*100)  + ((toucheCprogrammation_memorisation_loc-48)*10)
 + (toucheDprogrammation_memorisation_loc-48);
 //nbre contient l'intégralité de la saisie

 locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
 loc2Selectionne = 0;//DéSélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

 compteur_touche_memorisation_loc = 0;//Réinitialise compteur_touche

  autorise_touche_memorisation_loc = 0;//Interdit la prise en compte de l'appui touche

  toucheAprogrammation_memorisation_loc = 0;//Réinitialise toucheA
  toucheBprogrammation_memorisation_loc = 0;//Réinitialise toucheB
  toucheCprogrammation_memorisation_loc = 0;//Réinitialise toucheC
  toucheDprogrammation_memorisation_loc = 0;//Réinitialise toucheD
  toucheEprogrammation_memorisation_loc = 0;//Réinitialise toucheE
  toucheFprogrammation_memorisation_loc = 0;//Réinitialise toucheF

 lcd.clear(); // Efface écran

lcd.setCursor (0,0);
lcd.print ("N locomotive : ");
lcd.setCursor (8,1);
lcd.print(nbre_memorisation_loc);

 lcd.setCursor (0,2);
lcd.print ("Saisir l'adresse :");

lcd.setCursor (8,3);

//Gestion des claviers
autorise_touche_memorisation_adresse_loc = 1;//Autorise la prise en compte de l'appui touche +
//et autorise_touche_memorisation_adresse_loc à 1

} //Fin du if (compteur_touche_memorisation_loc == 4)
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (compteur_touche_memorisation_loc == 5)//Prise en compte du chiffre saisi pour une entrée
```

```
{
 nbre_memorisation_loc = ((toucheAprogrammation_memorisation_loc-48)*10000)
 + ((toucheBprogrammation_memorisation_loc-48)*1000) + ((toucheCprogrammation_memorisation_loc-48)*100)
 + ((toucheDprogrammation_memorisation_loc-48)*10) + (toucheEprogrammation_memorisation_loc-48);
 //nbre contient l'intégralité de la saisie

 locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
 loc2Selectionne = 0;//DéSélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

 compteur_touche_memorisation_loc = 0;//Réinitialise compteur_touche

  autorise_touche_memorisation_loc = 0;//Interdit la prise en compte de l'appui touche

  toucheAprogrammation_memorisation_loc = 0;//Réinitialise toucheA
  toucheBprogrammation_memorisation_loc = 0;//Réinitialise toucheB
  toucheCprogrammation_memorisation_loc = 0;//Réinitialise toucheC
  toucheDprogrammation_memorisation_loc = 0;//Réinitialise toucheD
  toucheEprogrammation_memorisation_loc = 0;//Réinitialise toucheE
  toucheFprogrammation_memorisation_loc = 0;//Réinitialise toucheF

 lcd.clear(); // Efface écran

lcd.setCursor (0,0);
lcd.print ("N locomotive : ");
lcd.setCursor (8,1);
lcd.print(nbre_memorisation_loc);

 lcd.setCursor (0,2);
lcd.print ("Saisir l'adresse :");

lcd.setCursor (8,3);

//Gestion des claviers
autorise_touche_memorisation_adresse_loc = 1;//Autorise la prise en compte de l'appui touche +
//et autorise_touche_memorisation_adresse_loc à 1

} //Fin du if (compteur_touche_memorisation_loc == 5)
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
//////////////////////////// ////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
if (compteur_touche_memorisation_loc == 6)//Prise en compte du chiffre saisi pour une entrée
{
  nbre_memorisation_loc = ((toucheAprogrammation_memorisation_loc-48)*10000) +
  ((toucheBprogrammation_memorisation_loc-48)*1000) + ((toucheCprogrammation_memorisation_loc-48)*100)
 + ((toucheDprogrammation_memorisation_loc-48)*10) + (toucheEprogrammation_memorisation_loc-48);

  nbre_memorisation_loc = (nbre_memorisation_loc*10)+ (toucheFprogrammation_memorisation_loc-48);
//nbre contient l'intégralité de la saisie
//Reconstitué en 2 fois sinon erreur dans la reconstitution


  locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
  loc2Selectionne = 0;//DéSélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

  compteur_touche_memorisation_loc = 0;//Réinitialise compteur_touche

    autorise_touche_memorisation_loc = 0;//Interdit la prise en compte de l'appui touche

    toucheAprogrammation_memorisation_loc = 0;//Réinitialise toucheA
    toucheBprogrammation_memorisation_loc = 0;//Réinitialise toucheB
    toucheCprogrammation_memorisation_loc = 0;//Réinitialise toucheC
    toucheDprogrammation_memorisation_loc = 0;//Réinitialise toucheD
    toucheEprogrammation_memorisation_loc = 0;//Réinitialise toucheE
    toucheFprogrammation_memorisation_loc = 0;//Réinitialise toucheF

  lcd.clear(); // Efface écran

lcd.setCursor (0,0);
lcd.print ("N locomotive : ");
lcd.setCursor (8,1);
lcd.print(nbre_memorisation_loc);

 lcd.setCursor (0,2);
lcd.print ("Saisir l'adresse :");

lcd.setCursor (8,3);
```

```
//Gestion des claviers
autorise_touche_memorisation_adresse_loc = 1;//Autorise la prise en compte de l'appui touche +
//et autorise_touche_memorisation_adresse_loc à 1

} //Fin du if (compteur_touche_memorisation_loc == 6)

}//Fin du void traitement_appui_touche_memorisation_loc ()

////////////////////////////////////////////////////////////////////////
//////////////////////////// ////////////////////////////////////////////
////////MISE EN MEMOIRE DES TOUCHES MEMORISATION LOC2//////////////////////
////////////////POUR MISE EN MEMOIRE EEPROM/////////////// ////////////////////
////////////////////////////////////////////////////////////////////////
void touche_appuyee_numerique_memorisation_adresse_loc() //Stockage touche appuyée
{
  compteur_touche_memorisation_adresse_loc = compteur_touche_memorisation_adresse_loc + 1;
  //Incrémente la variable compteur_touche

if (compteur_touche_memorisation_adresse_loc == 1)
//Teste si compteur_touche_memorisation_adresse_loc = 1
{toucheAprogrammation_memorisation_adresse_loc = touche;
lcd.blink();

autorise_touche_memorisation_adresse_loc_e = 1;//Autorise l'entrée du chiffre par la touche E

}//Garde trace de la touche 1ère touche appuyée

if (compteur_touche_memorisation_adresse_loc == 2)//Teste si compteur_touche_memorisation_adresse_loc = 2
{toucheBprogrammation_memorisation_adresse_loc = touche;
lcd.blink();}//Garde trace de la touche 2ième touche appuyée

if (compteur_touche_memorisation_adresse_loc == 3)//Teste si compteur_touche_memorisation_adresse_loc = 3
{toucheCprogrammation_memorisation_adresse_loc = touche;//Garde trace de la touche 3ième touche appuyée
lcd.noBlink ();

autorise_touche_memorisation_adresse_loc = 0;//Interdit la saisie de chiffre supplémentaire

}

}//touche_appuyee_numerique_memorisation_loc
```

```
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////// TRAITEMENT DES TOUCHES MEMORISATION LOC/////////////////
////////////////////////////POUR MISE EN MEMOIRE EEPROM/////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
void traitement_appui_touche_memorisation_memorisation_loc ()
//Correspond à Appui touche 'E' et tritement_appui_touche_memorisation_loc2 = 1
{

if (compteur_touche_memorisation_adresse_loc == 1)//Prise en compte du chiffre saisi pour une entrée
{
  adresse_memorisation_loc = (toucheAprogrammation_memorisation_adresse_loc-48);
  //toucheA contient l'intégralité de la saisie

  locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
  loc2Selectionne = 0;//DéSélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

  compteur_touche_memorisation_adresse_loc = 0;//Réinitialise compteur_touche

  autorise_touche_memorisation_adresse_loc = 0;//Interdit la prise en compte de l'appui touche


    toucheAprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheA
    toucheBprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheB
    toucheCprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheC


lcd.clear(); // Efface écran
  lcd.setCursor (0,0);
lcd.print ("N locomotive:");
lcd.setCursor (14,0);
lcd.print(nbre_memorisation_loc);

 lcd.setCursor (5,1);
lcd.print ("Adresse:");

lcd.setCursor (15,1);
lcd.print(adresse_memorisation_loc);
```

```
lcd.setCursor (0,3);

prog_memo_loc = 0;//Autorise les touches de fonction A, B, C, D
autorise_touche_A = 1;//Autorise la prise en compte de l'appui touche

lcd.print ("Confirmer: 1");
lcd.setCursor (14,3);

//Gestion des claviers

}//Fermeture du if (compteur_touche_memorisation_adresse_loc == 1)
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
if (compteur_touche_memorisation_adresse_loc == 2)//Prise en compte du chiffre saisi pour une entrée
{
 adresse_memorisation_loc = ((toucheAprogrammation_memorisation_adresse_loc-48)*10)
 + (toucheBprogrammation_memorisation_adresse_loc-48);
 //toucheA contient l'intégralité de la saisie

 locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
 loc2Selectionne = 0;//DéSélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

 compteur_touche_memorisation_adresse_loc = 0;//Réinitialise compteur_touche

 autorise_touche_memorisation_adresse_loc = 0;//Interdit la prise en compte de l'appui touche


  toucheAprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheA
  toucheBprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheB
  toucheCprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheC


lcd.clear(); // Efface écran
lcd.setCursor (0,0);
lcd.print ("N locomotive:");
lcd.setCursor (14,0);
lcd.print(nbre_memorisation_loc);

lcd.setCursor (5,1);
```

```
lcd.print ("Adresse:");

lcd.setCursor (15,1);
lcd.print(adresse_memorisation_loc);



lcd.setCursor (0,3);

prog_memo_loc = 0;//Autorise les touches de fonction A, B, C, D
autorise_touche_A = 1;//Autorise la prise en compte de l'appui touche

lcd.print ("Confirmer: 1");
lcd.setCursor (14,3);



}//Fermeture du if (compteur_touche_memorisation_adresse_loc == 2)

////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
if (compteur_touche_memorisation_adresse_loc == 3)//Prise en compte du chiffre saisi pour une entrée
{
 adresse_memorisation_loc = ((toucheAprogrammation_memorisation_adresse_loc-48)*100)
 + ((toucheBprogrammation_memorisation_adresse_loc-48)*10)
 + (toucheCprogrammation_memorisation_adresse_loc-48);//toucheA contient l'intégralité de la saisie



if (adresse_memorisation_loc < 128)
{
 locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
 loc2Selectionne = 0;//DéSélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

 compteur_touche_memorisation_adresse_loc = 0;//Réinitialise compteur_touche

 autorise_touche_memorisation_adresse_loc = 0;//Interdit la prise en compte de l'appui touche


   toucheAprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheA
   toucheBprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheB
   toucheCprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheC
```

```
lcd.clear(); // Efface écran
lcd.setCursor (0,0);
lcd.print ("N locomotive:");
lcd.setCursor (14,0);
lcd.print(nbre_memorisation_loc);

lcd.setCursor (5,1);
lcd.print ("Adresse:");

lcd.setCursor (15,1);
lcd.print(adresse_memorisation_loc);

lcd.setCursor (0,3);

prog_memo_loc = 0;//Autorise les touches de fonction A, B, C, D
autorise_touche_A = 1;//Autorise la prise en compte de l'appui touche

lcd.print ("Confirmer: 1");
lcd.setCursor (14,3);

}//Fermeture du if adresse_memorisation_adresse_loc < 128)

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

if (adresse_memorisation_loc > 127)
{
  locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive
  loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2

  compteur_touche_memorisation_adresse_loc = 0;//Réinitialise compteur_touche

  autorise_touche_memorisation_adresse_loc = 0;//Interdit la prise en compte de l'appui touche

    toucheAprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheA
    toucheBprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheB
    toucheCprogrammation_memorisation_adresse_loc = 0;//Réinitialise toucheC
```

```
   lcd.blink();//Autorise le clignotement du curseur

   lcd.clear(); // Efface écran si appui
   lcd.setCursor(0,0);//Place le curseur colonne 0, ligne 1

   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran
   lcd.print("Adresse Trop Grande") ; // Affiche la chaîne texte

   lcd.setCursor(1,2);//Place le curseur colonne 0, ligne 1
   lcd.print("doit etre comprise") ; // Affiche la chaîne texte

   lcd.setCursor(3,3);//Place le curseur colonne 0, ligne 1
   lcd.print("Entre 1 et 127") ; // Affiche la chaîne texte

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

   retourdetrame = 1;//En attente appui touche

}//Fermeture du if adresse_memorisation_loc > 127)



}//Fermeture du if (compteur_touche_memorisation_adresse_loc == 3)

}//void traitement_appui_touche_memorisation_adresse_loc ()

////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
void traitement_appui_touche_memorisation_loc_eeprom ()
{
adresse = adresse_memorisation_loc *3;
//Adresse_memorisation_loc contient la valeur de l'adresse choisie lors de la saisie
```

```
octetmsb = nbre_memorisation_loc >> 16;//Contient l'octet 1 Octet de poids fort
octetmidsb = nbre_memorisation_loc >> 8;//Contient l'octet 2 Octet de poids intermédiaire
octetlsb = nbre_memorisation_loc >> 0;//Contient l'octet 3 Octet de poids faible

lcd.clear(); // Efface écran

lcd.setCursor (0,0);
lcd.print ("Octet 1: ");
lcd.print (octetmsb);
lcd.setCursor (0,1);
lcd.print ("Octet 2: ");
lcd.print (octetmidsb);
lcd.setCursor (0,2);
lcd.print ("Octet 3: ");
lcd.print (octetlsb);

reconstitueoctet = octetmsb << 8;//Décale octet de poids fort de 8 rangs sur la gauche
reconstitueoctet2 = reconstitueoctet | octetmidsb;//Rajout de l'octet de poids intermédiaire
reconstitueoctet3 = reconstitueoctet2 << 8;//Décale le résultat de 8 rangs sur la gauche
reconstituefin = reconstitueoctet3 | octetlsb;
//Rajout de l'octet de poids intermédiaire et l'octet est reconstitué

lcd.setCursor (0,3);
lcd.print ("Loc: ");
lcd.print (reconstituefin);
lcd.setCursor (12,3);
lcd.print ("Adr: ");
lcd.print(adresse_memorisation_loc);

 for(int compteurtempo = 0; compteurtempo < dureetempomemoloceeprom; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for

lcd.clear(); // Efface écran

lcd.setCursor (4,0);
lcd.print ("Memorisation");
```

```
lcd.setCursor (5,2);
lcd.print ("Effectuee");

 for(int compteurtempo = 0; compteurtempo < dureetempomemoloceeprom; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
   {
    trame_idle();
   }//fermeture du for


EEPROM.write (adresse, octetmsb);//Ecriture octet 1 msb

//Visualisation sur moniteur série
////////////////////////////////////
Serial.println (adresse_memorisation_loc);//Possibilité
Serial.print (adresse);
Serial.print(" ");Serial.print(" ");
Serial.println (octetmsb);
////////////////////////////////////

adresse = adresse + 1;

EEPROM.write (adresse, octetmidsb);//Ecriture octet 2 midsb
//Visualisation sur moniteur série
////////////////////////////////////
Serial.print (adresse);
Serial.print(" ");Serial.print(" ");
Serial.println (octetmidsb);
////////////////////////////////////
adresse = adresse + 1;

EEPROM.write (adresse, octetlsb);//Ecriture octet 3 lsb
//Visualisation sur moniteur série
////////////////////////////////////
Serial.print (adresse);
Serial.print(" ");Serial.print(" ");
Serial.println (octetlsb);

Serial.print (nbre_memorisation_loc);
```

```
Serial.print(" ");Serial.print(" ");Serial.print(" ");Serial.print(" ");
Serial.println (adresse_memorisation_loc);

Serial.println ("Memorisation");
Serial.println ("Effectuee");
//////////////////////////////////////

retourdetrame=1;//retour à comme une pression sur la touche *

}//Fermeture du void

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
/////////////////void lecture_memoire_eeprom ()//////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
void lecture_memoire_eeprom ()
{
locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2

adresse_depart = 1;//Point de départ
//Serial.println(adresse_depart);//A garder visualisation sur moniteur série

for (int i=0; i<128; i++)
{
adresse_recherche = adresse_depart *3;
//Serial.println(adresse_recherche);//A garder visualisation sur moniteur série

resultatoctetmsb = EEPROM.read (adresse_recherche);
adresse_recherche = adresse_recherche + 1;

//Serial.println(adresse_recherche);//A garder visualisation sur moniteur série
resultatoctetmidsb = EEPROM.read (adresse_recherche);

adresse_recherche = adresse_recherche + 1;
//Serial.println(adresse_recherche);//A garder visualisation sur moniteur série
resultatoctetlsb = EEPROM.read (adresse_recherche);

reconstitueoctet = resultatoctetmsb << 8;//Décale octet de poids fort de 8 rangs sur la gauche
reconstitueoctet2 = reconstitueoctet | resultatoctetmidsb;//Rajout de l'octet de poids intermédiaire
```

```
reconstitueoctet3 = reconstitueoctet2 << 8;//Décale le résultat de 8 rangs sur la gauche
reconstituefin = reconstitueoctet3 | resultatoctetlsb;
//Rajout de l'octet de poids intermédiaire et l'octet est reconstitué

adresse_depart = adresse_depart + 1;
//Serial.println(adresse_depart);//A garder visualisation sur moniteur série



if (nbre_locomotive == reconstituefin)
    {
      //Serial.println ("if nbreloco=reconstituefin");

      adresse_locomotive = ((adresse_recherche-2)/3);
      //Si 22 rentrée (22-2=20)/3 = 10:  10 début d'emplacement du numéro de la locomotive

      compteur_touche_locomotive = 0;//Réinitialise compteur_touche

   locSelectionne = 1;//Sélectionne la loc 1 et permet l'envoi de la trame_locomotive 1
   loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2

   autorise_touche_7_et_9 = 1;
   //Autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle

  autorise_touche_1_et_4 = 1;

   lcd.clear(); // Efface écran

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

   //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   adr7 = 0;//Adresse courte
   adr6 = bitRead (adresse_locomotive, 6);//Met dans adr6, le bit 6 de la variable adresse_locomotive
   adr5 = bitRead (adresse_locomotive, 5);
   adr4 = bitRead (adresse_locomotive, 4);
   adr3 = bitRead (adresse_locomotive, 3);
   adr2 = bitRead (adresse_locomotive, 2);
   adr1 = bitRead (adresse_locomotive, 1);
   adr0 = bitRead (adresse_locomotive, 0);
```

```
    }//(nbre_locomotive = reconstituefin;)

}//Fermeture du for (int i=0; i<128; i++)

 //Si aucune loc trouvée
if (locSelectionne == 0 && loc2Selectionne == 0)
//test pour éviter une rentrée dans forcée dans la boucle
{
Serial.println ("sortie du for");

lcd.clear(); // Efface écran si appui
   lcd.setCursor(4,0);//Place le curseur colonne 0, ligne 1
   lcd.print("Nbre Inconnu") ; // Affiche la chaîne texte

   lcd.setCursor(5,1);//Place le curseur colonne 0, ligne 1
   lcd.print(" dans la") ; // Affiche la chaîne texte

   lcd.setCursor(5,2);//Place le curseur colonne 0, ligne 1
   lcd.print(" MEMOIRE") ; // Affiche la chaîne texte


   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
    {
    trame_idle();
    }//fermeture du for


 retourdetrame = 1;//En attente appui touche

}//if (locSelectionne == 0 && loc2Selectionne == 0)

}//Fermeture du void lecture_memoire_eeprom
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
/////////////Fin void lecture_memoire_eeprom ()/////////////
```

```
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
/////////////////void lecture_memoire_eeprom2 ()//////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
void lecture_memoire_eeprom2 ()
{
locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2

adresse_depart = 1;//Point de départ
//Serial.println(adresse_depart);//A garder visualisation sur moniteur série

for (int i=0; i<128; i++)
{
adresse_recherche = adresse_depart *3;
//Serial.println(adresse_recherche);//A garder visualisation sur moniteur série

resultatoctetmsb = EEPROM.read (adresse_recherche);
adresse_recherche = adresse_recherche + 1;

//Serial.println(adresse_recherche);//A garder visualisation sur moniteur série
resultatoctetmidsb = EEPROM.read (adresse_recherche);

adresse_recherche = adresse_recherche + 1;
//Serial.println(adresse_recherche);//A garder visualisation sur moniteur série
resultatoctetlsb = EEPROM.read (adresse_recherche);

reconstitueoctet = resultatoctetmsb << 8;//Décale octet de poids fort de 8 rangs sur la gauche
reconstitueoctet2 = reconstitueoctet | resultatoctetmidsb;//Rajout de l'octet de poids intermédiaire
reconstitueoctet3 = reconstitueoctet2 << 8;//Décale le résultat de 8 rangs sur la gauche
reconstituefin = reconstitueoctet3 | resultatoctetlsb;
//Rajout de l'octet de poids faible et l'octet est reconstitué

adresse_depart = adresse_depart + 1;
//Serial.println(adresse_depart);//A garder visualisation sur moniteur série



if (nbre_locomotive2 == reconstituefin)
```

```
    {

      adresse_locomotive2 = ((adresse_recherche-2)/3);
      compteur_touche_locomotive2 = 0;//Réinitialise compteur_touche

   locSelectionne = 0;//DéSélectionne la loc 1 interdit l'envoi de la trame_locomotive 1
   loc2Selectionne = 1;//Sélectionne la loc 2 et permet l'envoi de la trame_locomotive 2

  autorise_touche_7_et_9 = 1;
   //Autorise l'entrée dans la boucle de la touche 7 ou 9 pour changer la locomotive sous contrôle

  autorise_touche_1_et_4 = 1;

   lcd.clear(); // Efface écran

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

   //Transforme la variable adresse en binaire et stocke dans les variables adr
   //Octet 1 d'adresse
   adr27loc = 0;//Adresse courte
   adr26loc = bitRead (adresse_locomotive2, 6);//Met dans adr6, le bit 6 de la variable adresse_locomotive
   adr25loc = bitRead (adresse_locomotive2, 5);
   adr24loc = bitRead (adresse_locomotive2, 4);
   adr23loc = bitRead (adresse_locomotive2, 3);
   adr22loc = bitRead (adresse_locomotive2, 2);
   adr21loc = bitRead (adresse_locomotive2, 1);
   adr20loc = bitRead (adresse_locomotive2, 0);

    }//(nbre_locomotive = reconstituefin;)

}//Fermeture du for (int i=0; i<128; i++)

 //Si aucune loc trouvée
if (locSelectionne == 0 && loc2Selectionne == 0)//test pour éviter une rentrée forcée dans la boucle
{
Serial.println ("sortie du for");

lcd.clear(); // Efface écran si appui
   lcd.setCursor(4,0);//Place le curseur colonne 0, ligne 1
   lcd.print("Nbre Inconnu") ; // Affiche la chaîne texte
```

```
   lcd.setCursor(5,1);//Place le curseur colonne 0, ligne 1
   lcd.print(" dans la") ; // Affiche la chaîne texte

   lcd.setCursor(5,2);//Place le curseur colonne 0, ligne 1
   lcd.print(" MEMOIRE") ; // Affiche la chaîne texte


   tone(buzzer, 440, 20);//Emet un bip pour appuyer l'affichage de l'écran

   lcd.noBlink();//Empêche le clignotement du curseur

   for(int compteurtempo = 0; compteurtempo < dureetempo; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
    {
    trame_idle();
    }//fermeture du for


 retourdetrame = 1;//En attente appui touche

}//if (locSelectionne == 0 && loc2Selectionne == 0)

}//Fermeture du void lecture_memoire_eeprom2
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
/////////Fin void lecture_memoire_eeprom2 ()/////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
void lecture_entiere_memoire_eeprom ()
{
lcd.clear();
lcd.setCursor (2,0);
lcd.print ("Lecture entiere");

lcd.setCursor (7,1);
lcd.print ("EEPROM");

lcd.setCursor (8,2);
lcd.print ("Sur");
```

```
lcd.setCursor (3,3);
lcd.print ("Moniteur serie");

locSelectionne = 0;//DéSélectionne la loc 1 et interdit l'envoi de la trame_locomotive 1
loc2Selectionne = 0;//DéSélectionne la loc 2 et interdit l'envoi de la trame_locomotive 2

adresse_depart = 1;//Point de départ
//Serial.println(adresse_depart);

for (int i=0; i<127; i++)
{
adresse_recherche = adresse_depart *3;
//Serial.println(adresse_recherche);

resultatoctetmsb = EEPROM.read (adresse_recherche);
adresse_recherche = adresse_recherche + 1;

//Serial.println(adresse_recherche);
resultatoctetmidsb = EEPROM.read (adresse_recherche);
adresse_recherche = adresse_recherche + 1;

//Serial.println(adresse_recherche);
resultatoctetlsb = EEPROM.read (adresse_recherche);

reconstitueoctet = resultatoctetmsb << 8;//Décale octet de poids fort de 8 rangs sur la gauche
reconstitueoctet2 = reconstitueoctet | resultatoctetmidsb;//Rajout de l'octet de poids intermédiaire
reconstitueoctet3 = reconstitueoctet2 << 8;//Décale le résultat de 8 rangs sur la gauche
reconstituefin = reconstitueoctet3 | resultatoctetlsb;
//Rajout de l'octet de poids faible et l'octet est reconstitué

//Vue sur moniteur série
Serial.print(adresse_depart);
Serial.print(":");Serial.print(" ");Serial.print(" ");

Serial.print(reconstituefin);

Serial.println();

adresse_depart = adresse_depart + 1;
```

```
}//Fermeture du for

for(int compteurtempo = 0; compteurtempo < dureetempolectureeeprom; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
    {
    trame_idle();
    }//fermeture du for

//La lecture de l'eeprom est déjà faite

retourdetrame = 1;//En attente appui touche

}
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
void raz_eeprom()
{
autorise_touche_A = 0;//Interdit la prise en compte de l'appui touche
adresse = 0;//Point de départ

lcd.clear();
lcd.setCursor (8,0);
lcd.print ("RAZ");

lcd.setCursor (6,2);
lcd.print ("EEPROM");

for (int i=0; i<500; i++)
{
EEPROM.write (adresse, 0);
/*
//Vue sur moniteur série si les instructions sont dégrisées
Serial.print (adresse);
Serial.print(" ");Serial.print(" ");
Serial.print("0");
Serial.println();
*/
```

```
adresse = adresse + 1;


}


lcd.clear();
lcd.setCursor (8,0);
lcd.print ("RAZ");
lcd.setCursor (5,2);
lcd.print ("EFFECTUEE");


for(int compteurtempo = 0; compteurtempo < dureetemporazeeprom; compteurtempo++)
   //Envoi trame idle pour ne pas interrompre l'envoi de trame dcc
   //et permet une temporisation
    {
    trame_idle();
    }//fermeture du for


retourdetrame=1;//retour à comme une pression sur la touche *


}
```