

/*****

This program was produced by the
CodeWizardAVR V1.24.0 Standard
Automatic Program Generator
© Copyright 1998-2003 HP InfoTech s.r.l.
<http://www.hpinfotech.ro>
e-mail:office@hpinfotech.ro

Project : Recepteur DMX
Version : 1
Date : 12/02/2007
Author : Philippe Mariano
Company : LYCEE Pierre Emile MARTIN
Comments:
Réception de la trame DMX et affichage sur un LCD

Chip type : ATmega32
Program type : Application
Clock frequency : 16,000000 MHz
Memory model : Small
External SRAM size : 0
Data Stack size : 512
*****/

```
#include <mega32.h>
#include <delay.h>
#include <stdlib.h>
#include <string.h>
#include <tpavnum.h> // Table de transcodage du clavier numérique utilisé
// Cette table devra être adapté au clavier utilisé
#include <ssi.h> // Sous-programme de désérialisation et de décodage de la
// trame issue du clavier PS2 et sous-programme Affiche_LCD
#include <pavenum.h> // Gestion d'un clavier numérique type PS2
```

```
// Alphanumeric LCD Module functions
#asm
.equ __lcd_port 0x15
#endasm
#include <lcd.h>
```

```
// Modes de fonctionnement
#define Recepteur 1
#define Emetteur 2
#define Diagnostic 3
```

```
// Sous-menu du mode récepteur
#define Reglage_Add 11
```

```

#define Aff_Bargraph      12
#define Aff_Decimal 13
#define Aff_Pourcent      14
#define Aff_Hexa          15
#define Aff_Binaire       16
#define Aff_max_min       17

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define ERREUR_TRAME (UCSRA & (1<<FE))
#define DONNEE_RECUE UDR
#define NOMBRE_CANAUx 24
#define VRAI 1
#define FAUX 0

#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// Adresse CGRAM de l'afficheur LCD
// -----
#define Add_Caract0_CGRAM 0x40
#define Add_Caract1_CGRAM 0x48
#define Add_Caract2_CGRAM 0x50
#define Add_Caract3_CGRAM 0x58
#define Add_Caract4_CGRAM 0x60
#define Add_Caract5_CGRAM 0x68
#define Add_Caract6_CGRAM 0x70
#define Add_Caract7_CGRAM 0x78

// Declare your global variables here
/*-----*/
type          Désignation          Commentaires
/*-----*/

// Communication avec le clavier PS2
// Le programme principal se synchronise avec la routine d'interruption située dans la bibliothèque stk200.lib grace aux deux
// variables ci-dessous
unsigned char  g_key_ASCII 'E';          // Contient le code ASCII de la touche pressée (à initialiser avec un
// caractère n'existant pas sur le clavier utilisé)
unsigned char  g_flag_char = 1;          // g_flag_char = 1 lorsqu'une touche est appuyée
// g_flag_char = 0 lorsqu'une touche a été relâchée

```

```

unsigned char  g_Sens_transfert = 0;                // variable globale indiquant le sens de
                                                    // communication µC <---> clavier  que doit gérer la routine d'interruption
unsigned int g_Trame_TX;                            //

// Paramètres utilisés en mode réception
unsigned int Table_Donnee[NOMBRE_CANAUUX] ;        // Contient les données reçues sur la liaison DMX512
unsigned int compteur_canal = 0, canal_base = 1,nouveau_canal_base = 0, Table_canal[3]={0,0,0};
unsigned char drapeau_reception = FAUX;

// Constantes en flash
//-----
// Caractères personnalisés placés dans la mémoire CGRAM du LCD
const unsigned char flecheHaut[8] = {0x04, 0x0E, 0x15, 0x04, 0x04, 0x04, 0x04, 0x00};    // @0 CGRAM
const unsigned char flecheBas[8] = {};        // @1 CGRAM      à compléter
const unsigned char Symbole_Barre_Graphe[8] = {}; // @2 CGRAM  à compléter
const unsigned char Barre_graphe_niveau2[8] = {}; // @3 CGRAM  à compléter
const unsigned char Barre_graphe_niveau3[8] = {}; // @4 CGRAM  à compléter
const unsigned char Barre_graphe_niveau4[8] = {}; // @5 CGRAM  à compléter
const unsigned char Barre_graphe_niveau5[8] = {}; // @6 CGRAM  à compléter
const unsigned char Barre_graphe_niveau6[8] = {}; // @7 CGRAM  à compléter

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
//static unsigned int compteur_canal;
unsigned char Tampon_Donnee;
unsigned int position;

if ERREUR_TRAME {
    compteur_canal = 0;
    Tampon_Donnee = DONNEE_RECUE; // mise à zéro de RXC
}
else {
    if ((compteur_canal >= canal_base) && (compteur_canal <= canal_base + NOMBRE_CANAUUX - 1)){
        position = compteur_canal - canal_base;
        Tampon_Donnee = DONNEE_RECUE;
        Table_Donnee[position] = Tampon_Donnee;
        if (compteur_canal == canal_base + NOMBRE_CANAUUX - 1)
            drapeau_reception = VRAI;
    }
    else
        Tampon_Donnee = DONNEE_RECUE; // mise à zéro de RXC
    compteur_canal += 1;
}
}

```

```
// Standard Input/Output functions
#include <stdio.h>
```

```
void main(void)
{
// Declare your local variables here
/*-----
type          Désignation          Commentaires
-----*/
char          display_buffer_ligne0[17];          // tampon ligne 0 de l'afficheur LCD
char          display_buffer_ligne1[17];          // tampon ligne 1 de l'afficheur LCD
unsigned char Menu Recepteur,Menu_1 Emetteur;          // Etats du graphe des transitions "Menu"
unsigned char i;
unsigned char *chaine; // espace[]=" ";
//unsigned int pos[] = {3,0,3,1,11,0,11,1};
unsigned char Saisie_code = VRAI, Code_différent = FAUX, pos_X = 12, barre = 4;

// Input/Output Ports initialization
// Port A initialization
// Func0 In Func1 In Func2 In Func3 In Func4 In Func5 In Func6 In Func7 In
// State0 T State1 T State2 T State3 T State4 T State5 T State6 T State7 T
PORTA 0x00;
DDRA 0x00;

// Port B initialization
// Func0 In Func1 In Func2 In Func3 In Func4 In Func5 In Func6 In Func7 In
// State0 T State1 T State2 T State3 T State4 T State5 T State6 T State7 T
PORTB 0x00;
DDRB 0x1f;

// Port C initialization
// Func0 In Func1 In Func2 In Func3 In Func4 In Func5 In Func6 In Func7 In
// State0 T State1 T State2 T State3 T State4 T State5 T State6 T State7 T
PORTC 0x00;
DDRC 0x00;

// Port D initialization
// Func0 In Func1 In Func2 In Func3 In Func4 In Func5 In Func6 In Func7 In
// State0 T State1 T State2 T State3 T State4 T State5 T State6 T State7 T
PORTD 0x00;
DDRD 0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK 0x00;
```

```
// USART initialization
// Communication Parameters: 8 Data, 2 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud rate: 250000
UCSRA 0x00;
UCSRB 0x90;
UCSRC 0x8E;
UBRRH 0x00;
UBRRL 0x03;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Low level
// INT1: Off
// INT2: Off
GICR| 0x40;
MCUCR 0x00;
MCUCSR 0x00;
GIFR 0x40;

// LCD module initialization
lcd_init(16);

// Chargement de la mémoire CGRAM de l'afficheur avec les
// caractères personnalisés
for (i 0;i<8;i++)
    lcd_write_byte(Add_Caract0_CGRAM + i, flecheHaut[i]);

à compléter

// Global enable interrupts
#asm("sei")

while (1)
{
    switch (Menu)
    {
// ----- Début Menus principaux -----
        case Recepteur : if (Menu_1 != Menu)
        {
            sprintf(display_buffer_ligne0,"Mode Recepteur ");
            sprintf(display_buffer_ligne1,"Entr>Acq ");
            Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);
            lcd_gotoxy(15,0); // Affichage flecheHaut
            lcd_putchar(0);
```

```
    lcd_gotoxy(15,1);    // Affichage flecheBas
    lcd_putchar(1);
    }

    if (g_flag_char == 1)
    {
        switch(g_key_ASCII)
        {
            case ':': Menu = Reglage_Add; g_key_ASCII = 'E'; break;
            case '8': Menu = Diagnostic; g_key_ASCII = 'E'; break;
            case '2': Menu = Emetteur; g_key_ASCII = 'E'; break;
            default: Menu = Recepteur;
        }
    }
    Menu_1 = Recepteur;
break;

case Emetteur : if (Menu_1 != Menu)
{
    sprintf(display_buffer_ligne0,"Mode Emetteur ");
    sprintf(display_buffer_ligne1,"Entr>Acq          ");
    Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);
    lcd_gotoxy(15,0);
    lcd_putchar(0);
    lcd_gotoxy(15,1);
    lcd_putchar(1);
}

    if (g_flag_char == 1)
    {
        switch(g_key_ASCII)
        {
            //case '.': g_key_ASCII = 'E'; break;
            case '8': Menu = Recepteur; g_key_ASCII = 'E'; break;
            case '2': Menu = Diagnostic; g_key_ASCII = 'E'; break;
            default: Menu = Emetteur;
        }
    }
    Menu_1 = Emetteur;
break;

case Diagnostic : if (Menu_1 != Menu)
{
    sprintf(display_buffer_ligne0,"Mode Diagnostic ");
    sprintf(display_buffer_ligne1,"Entr>Acq          ");
    Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);
    lcd_gotoxy(15,0);
    lcd_putchar(0);
    lcd_gotoxy(15,1);
```

```

    lcd_putchar(1);
}

    if (g_flag_char == 1)
    {
        switch(g_key_ASCII)
        {
            //case '.': PORTB.0 = 1; g_key_ASCII = 'E'; break;
            case '8': Menu = Emetteur; g_key_ASCII = 'E'; break;
            case '2': Menu = Recepteur; g_key_ASCII = 'E'; break;
            default: Menu = Diagnostic;
        }
    }
    Menu_1 = Diagnostic;
break;

// Fin Menus principaux
// -----
// Début Sous-menu en mode Recepteur

    case Reglage_Add : if (Menu_1 != Menu)
    {
        sprintf(display_buffer_ligne0,"Canal base=      ");
        sprintf(display_buffer_ligne1,"Etr>Acq D1>Esc  ");
        Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);

        for(i 12;i<15;i++)
        {
            lcd_gotoxy(i,0); lcd_putchar('*');
        }
        lcd_gotoxy(12,0);
        _lcd_ready();
        _lcd_write_data(displayOn3);
    }

    if (g_flag_char == 1)
    {
        // début if g_flag_char
        switch (Saisie_code)
        {
            case VRAI : if (g_key_ASCII == '\n') // Acquiescement du code par Enter
            {
                if (Code_différent == VRAI)
                {
                    nouveau_canal_base = 100*Table_canal[0] + 10*Table_canal[1] + Table_canal[2];
                    if ((nouveau_canal_base > 0) && (nouveau_canal_base<513)) // canal correct
                    {
                        canal_base = nouveau_canal_base;
                        lcd_gotoxy(15,0); lcd_putchar(0); // Affichage des flèches
                        lcd_gotoxy(15,1); lcd_putchar(1);
                    }
                }
            }
        }
    }

```

```

        Saisie_code = FAUX;
    }
    else // Si canal faux
    {
        for(i 12;i<15;i++)
        {
            Table_canal[i-12] = 0;
            lcd_gotoxy(i,0); lcd_putchar('*');
        }
        lcd_gotoxy(12,0);pos_X 12;
    }
    Code_different = FAUX;
}
else if (g_key_ASCII == '.') //Echappement par Del
{
    for(i 12;i<15;i++)
    {
        Table_canal[i-12] = 0;
        lcd_gotoxy(i,0); lcd_putchar('*');
    }
    lcd_gotoxy(15,0); lcd_putchar(0);
    lcd_gotoxy(15,1); lcd_putchar(1);
    Code_different = FAUX; Saisie_code = FAUX; pos_X 12;
}
else if ((g_key_ASCII >= 0x30) && (g_key_ASCII < 0x39)) // Saisie du code
{
    lcd_putchar(g_key_ASCII);Table_canal[pos_X-12] = g_key_ASCII - 0x30;
    if (pos_X<14) pos_X++; else pos_X 12;
    lcd_gotoxy(pos_X,0);
    Code_different = VRAI;
}
g_key_ASCII = 'E';
break;

case FAUX : switch (g_key_ASCII)
{
    case '.': Saisie_code = VRAI; Menu = Recepteur; g_key_ASCII = 'E'; break;
    case '8': Saisie_code = VRAI; Menu = Aff_max_min; g_key_ASCII = 'E'; break;
    case '2': Saisie_code = VRAI; Menu = Aff_Bargraph; g_key_ASCII = 'E'; lcd_clear();
              itoa(canal_base,chaine); lcd_puts(chaine);
    break;
    default: Menu = Reglage_Add;
}

break;

} // fin switch Saisie_code
} //fin du if g_flag_char

```



```

    Menu_1 = Reglage_Add;
break;

case Aff_Bargraph : if (Menu_1 != Menu)
    {
        lcd_gotoxy(15,0);lcd_putchar(0);
        lcd_gotoxy(15,1);lcd_putchar(1);
        lcd_gotoxy(0,1);lcd_putchar(7);
    }

    while (drapeau_reception == FAUX) ;

    for (i 0;i<24;i++)
    {
        barre = (float)(Table_Donnee[i])/32;
        if (i<12) lcd_gotoxy(i+3,0);
        else lcd_gotoxy(i-9,1);
        if (barre == 0) lcd_putchar(0x20);
        else if (barre == 1) lcd_putchar(0x5F);
        else if (barre == 7) lcd_putchar(0xFF);
        else lcd_putchar(barre);
    }

    if (g_flag_char == 1)
    {
        switch(g_key_ASCII)
        {
            //case '.': PORTB.0 = 1; g_key_ASCII = 'E'; break;
            case '8': Menu = Reglage_Add; g_key_ASCII = 'E'; break;
            case '2': Menu = Aff_max_min; g_key_ASCII = 'E'; break;
            default: Menu = Aff_Bargraph;
        }
    }
    Menu_1 = Aff_Bargraph;
break;

case Aff_max_min : if (Menu_1 != Menu)
    {
        sprintf(display_buffer_ligne0,"");
        sprintf(display_buffer_ligne1,"Entr>Acq      ");
        Affiche_LCD(display_buffer_ligne0,display_buffer_ligne1);
        lcd_gotoxy(15,0);
        lcd_putchar(0);
        lcd_gotoxy(15,1);
        lcd_putchar(1);
    }

    if (g_flag_char == 1)

```

```

        {
            switch(g_key_ASCII)
            {
                //case '.': PORTB.0 = 1; g_key_ASCII = 'E'; break;
                case '8': Menu = Aff_Bargraph; g_key_ASCII = 'E'; break;
                case '2': Menu = Reglage_Add; g_key_ASCII = 'E'; break;
                default: Menu = Aff_max_min;
            }
        }
        Menu_1 = Aff_max_min;
    break;

/*
while (drapeau_reception == FAUX) ;

//if (Table_Donnee[0] != 20)
{
    //asm ("cli")
    PORTB.0 = 1;

    for (i 0;i<4;i++)
    {
        itoa(Table_Donnee[i],chaine);
        lcd_gotoxy(pos[2*i],pos[2*i+1]);
        lcd_puts(espace);
        lcd_gotoxy(pos[2*i],pos[2*i+1]);
        lcd_puts(chaine);
        Table_Donnee_1[i] = Table_Donnee[i];
    }

    drapeau_reception = FAUX;
    PORTB.0 = 0;
    //asm("sei")
}
*/
}

}

}

```