```
/*-------------------------------------------------------*/
//      Mc32DriverLcd.c
/*-------------------------------------------------------*/
//Description :Driver pour LCD PIC 32
//
//
//Auteur : Gomes Andres
//
//Version:V1.2
//Compilateur:XC32 V1.33 & harmony 1.00
//
//      MODIFICATIONS   :
//          CHR 15.05.2014   retouche lcd_putc
//          CHR 15.05.2014   ajout lcd_printf
//          CHR 09.09.2014   utilisation de bsp_cponfig.h
//          SCA 11.10.2016   utilisation des délais core timer
//
/*-------------------------------------------------------*/


#include "GenericTypeDefs.h"
#include "bsp.h"
#include "Mc32DriverLcd.h"
#include "Mc32Delays.h"

/*-------------------------------------------------------*/
// Définition du tableau pour l'adresse des lignes
/*-------------------------------------------------------*/
const BYTE taddrLines[5] = { 0,    // ligne 0 pas utilisé
                             0,    // ligne 1 commence au caractère 0
                             64,    // ligne 2 commence au caractère 64 (0x40)
                             20,    // ligne 3 commence au caractère 20
                             84 };  // ligne 4 commence au caractère 84 (0x40+20)
/*-------------------------------------------------------*/
// Fonctions
/*-------------------------------------------------------*/
BYTE lcd_read_byte( void )
{
      UINT8_BITS lcd_read_byte;
      uint8_t i;
  LCD_DB4_T = 1; // 1=input
     LCD_DB5_T = 1;
  LCD_DB6_T = 1;
  LCD_DB7_T = 1;

      LCD_RW_W = 1;
      for(i = 0; i < 4 ; i++)
      {
          delay500nsCt();//ds0066 demande 0.5us
      }
      LCD_E_W = 1;
   for(i = 0; i < 4 ; i++)
      {
          delay500nsCt();//ds0066 demande 0.5us
      }
```

```
      }
      lcd_read_byte.bits.b7 = LCD_DB7_R;
      lcd_read_byte.bits.b6 = LCD_DB6_R;
      lcd_read_byte.bits.b5 = LCD_DB5_R;
      lcd_read_byte.bits.b4 = LCD_DB4_R;
      LCD_E_W = 0; // attention e pulse min = 500ns à 1 et autant à 0
   for(i = 0; i < 4 ; i++)
      {
          delay500nsCt();//ds0066 demande 0.5us
      }
      LCD_E_W = 1;
       for(i = 0; i < 4 ; i++)
      {
          delay500nsCt();//ds0066 demande 0.5us
      }
      lcd_read_byte.bits.b3 = LCD_DB7_R;
      lcd_read_byte.bits.b2 = LCD_DB6_R;
      lcd_read_byte.bits.b1 = LCD_DB5_R;
      lcd_read_byte.bits.b0 = LCD_DB4_R;
      LCD_E_W = 0;
       for(i = 0; i < 4 ; i++)
      {
          delay500nsCt();//ds0066 demande 0.5us
      }
      LCD_DB4_T = 0; // 0=output
  LCD_DB5_T = 0;
  LCD_DB6_T = 0;
  LCD_DB7_T = 0;
      return(lcd_read_byte.Val);
}

/*-------------------------------------------------------*/
void lcd_send_nibble( BYTE n )
{
   UINT8_BITS NibbleToWrite;

   NibbleToWrite.Val = n;
   LCD_DB7_W = NibbleToWrite.bits.b3;
   LCD_DB6_W = NibbleToWrite.bits.b2;
   LCD_DB5_W = NibbleToWrite.bits.b1;
   LCD_DB4_W = NibbleToWrite.bits.b0;
   delay500nsCt();
   LCD_E_W = 1;
   delay500nsCt(); // E pulse width min = 450ns pour le 1!
   LCD_E_W = 0;
   delay500nsCt(); // E pulse width min = 450ns également pour le 0!
}

/*-------------------------------------------------------*/
void lcd_send_byte( BYTE address, BYTE n )
{
LCD_RS_W = 0;
while ( (lcd_read_byte() & 0x80) == 0x80 ) ;
    LCD_RS_W = address;
    LCD_RW_W = 0;
```

```
    //LCD_E déjà à 0
    lcd_send_nibble(n >> 4);
    lcd_send_nibble(n & 0xf);
}

/*-------------------------------------------------------*/
void lcd_init(void)
{
  // on va effectuer exactement ce que demande le ST0066U
  // on repositionne LCD_E tout pour un démarrage correct
    LCD_E_W = 0;
    delay_usCt(10); // si LCD_E était à 1, on attend
    LCD_RS_W = 0;  // demandé pour une commande
    LCD_RW_W = 0;

    delay_msCt(50);

    lcd_send_nibble(0x03); // correspond à 0x30, interface 4 bits
    delay_msCt(50);
    lcd_send_nibble(0x03); // correspond à 0x30, interface 4 bits
    delay_msCt(50);
    lcd_send_nibble(0x03); // correspond à 0x30, interface 4 bits
    delay_msCt(50);

    lcd_send_nibble(0x02); // correspond à 0x02, interface 4 bits

    delay_msCt(50);


    lcd_send_byte(0,0x28); //rs=0, 2 lines mode, display off
    delay_usCt(400); //ds0066 demande >390us
    lcd_send_byte(0,0x10); //rs=0, display on, cursor off, blink off
    delay_usCt(400); //ds0066 demande >390us
    lcd_send_byte(0,0x0F); //rs=0, display clear  ///0F
    delay_msCt(200); //ds0066 demande >199ms
    lcd_send_byte(0,0x06); //rs=0, increment mode, entire shift off
    delay_usCt(400); //ds0066 demande >390us
    lcd_send_byte(0,0x0C);//cursor off
    delay_usCt(400);
    lcd_send_byte(0,0x01); //clear display
}
//     suivant comment l'interfaçage avec le LCD s'est arrêté, il faut tout remettre à plat
//     selon les notes d'applications, il faut envoyer 3 fois un nibble 0x3
//     pour lui faire croire que nous sommes en interface8 bits
//     chaque envoi doit être séparé de 5ms!!
//    lcd_send_nibble(0x03); // correspond à 0x30, interface 8 bits
//    delay_msCt(5);
//    lcd_send_nibble(0x02); // correspond à 0x30, interface 8 bits
//    delay_msCt(5);
//    lcd_send_nibble(0x02); // correspond à 0x30, interface 8 bits
//    delay_msCt(5);
//    // maintenant, on peut configurer notre LCD en interface 4 bits
//    LCD_RS_W = 0;  // demandé pour une commande (on assure!)
//    lcd_send_nibble(2);// 4 bits interface
//    lcd_send_byte(0,0b00001000); //rs=0, 2 lines mode, display off
```

```
//    delay_usCt(40); //ds0066 demande >39us
//    lcd_send_byte(0,0b00000001); //rs=0, display on, cursor off, blink off
//    delay_usCt(40); //ds0066 demande >39us
//    lcd_send_byte(0,0b00000001); //rs=0, display clear
//    delay_msCt(2); //ds0066 demande >1,53ms
//    lcd_send_byte(0,0b00000110); //rs=0, increment mode, entire shift off
//    delay_usCt(40); //ds0066 demande >39us


/*-------------------------------------------------------*/
void lcd_gotoxy( BYTE x, BYTE y)
{
   BYTE address;
   address = taddrLines[y];
   address+=x-1;

   lcd_send_byte(0,0x80|address);
}

/*-------------------------------------------------------*/
// Modif du 15.05.2014 C. Huber
void lcd_putc( BYTE c)
{
switch (c)
{
    case '\f'   : lcd_send_byte(0,1);break;  // modif du (1,1) en (0,1)
     case '\n'   : lcd_gotoxy(1,2);        break;
     case '\b'   : lcd_send_byte(0,0x10);break;
     default     : lcd_send_byte(1,c);break;
   }
}

/*-------------------------------------------------------*/
void lcd_put_string_ram( char *ptr_char )
{
while (*ptr_char != 0)
{
    lcd_putc(*ptr_char);
    ptr_char++;
  }
}

/*-------------------------------------------------------*/
void lcd_put_string_rom( const char *ptr_char )
{
while (*ptr_char != 0)
{
    lcd_putc(*ptr_char);
    ptr_char++;
  }
}

/*-------------------------------------------------------*/
char lcd_getc( BYTE x, BYTE y)
```

```
{
BYTE value;
    lcd_gotoxy(x,y);
    while ( lcd_read_byte() & 0x80 ); // wait until busy flag is low
    LCD_RS_W = 1;
    value = lcd_read_byte();

    LCD_RS_W = 0;
    return(value);
}

/*-----------------------------------------------------------*/
void lcd_bl_on( void )
{
LCD_BL_T = 0;
LCD_BL_W = 1;
}

/*-----------------------------------------------------------*/
void lcd_bl_off( void )
{
LCD_BL_T = 0;
LCD_BL_W = 0;
}

// printf_lcd   New pour migration
// Auteur C. Huber 15.05.2014
void printf_lcd( const char *format,  ...)
{
    char Buffer[21];
    va_list args;
    va_start(args, format);

    vsprintf(Buffer, format, args);
    lcd_put_string_ram(Buffer);

    va_end(args);
}

// Ajout a la demande des utilisateurs
// Auteur C. Huber 02.12.2014
void lcd_ClearLine( unsigned char NoLine)
{
    int i;
    if (NoLine >= 1 && NoLine <= 4)  {

        lcd_gotoxy( 1, NoLine) ;
        for (i = 0 ; i < 20 ; i++)
        {
            lcd_send_byte(1,0x20);
        }

     }
}
```