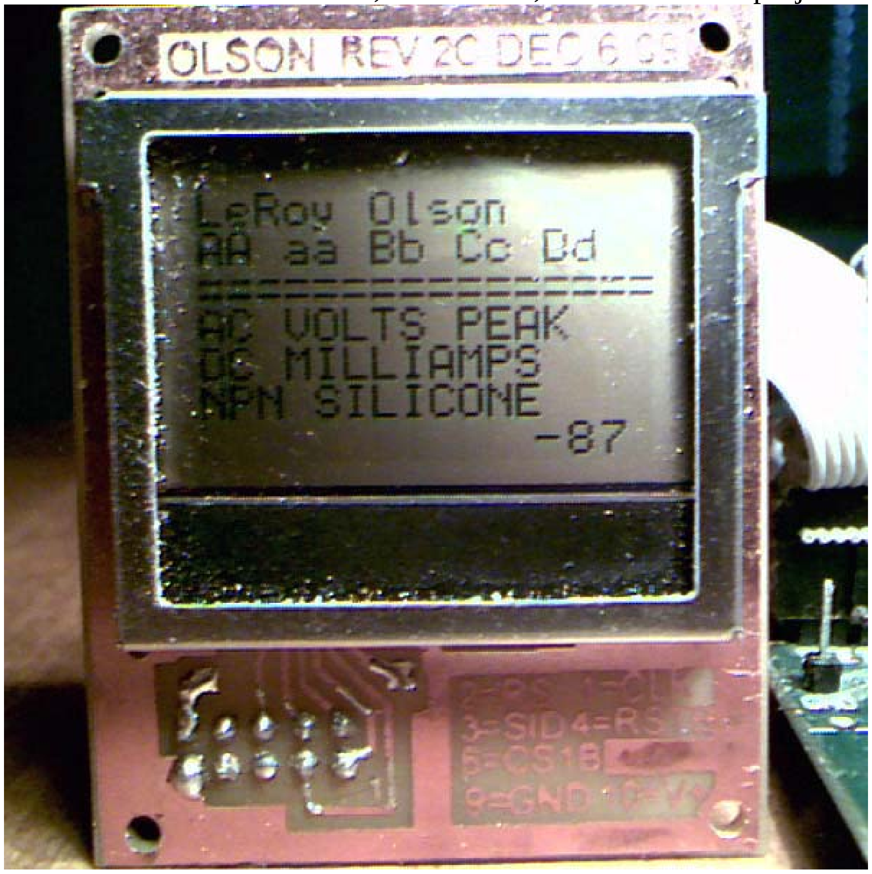
Text Displayed on GLCD with an ATtiny 26

This is a program I have compiled using mikroC to display text on an old Electronic Gold Mine GLCD 79294 SDI. These display's have been sold out but I had a number of them and wanted to find a use for them. They use a S6B0755 driver controller, I have modified the original header and c files from the site (Outguessing the machine) where the original files had been written for a Pic, they are now useable with an AVR chip. The font header file was created with MikroElectronic Font Creator as an mikroC GLCD lib. Then copy and paste to header file . The output to GLCD pin assignment is located in the s6b0755 header file and can easily be reconfigured for different pin arrangements. This all fit into an ATtiny26. It will display text strings, and put ascll characters on the screen, it will also reverse print direction, and screen black on white can be changed to white on black. In order to keep the file small I omitted all graphic functions as I did not need them. I compiled the files for a Tiny26 using the 8mhz internal osc.
Included are the c and h files, the hex file, and the Mikroc proj file.



```
//       fontdata6x8.h
//WARNING: This Font is usable only with MikroE GLCD Lib.
//         X-GLCD Lib does not handle this font.

//Font Generated by MikroElektronika GLCD Font Creator 1.2.0.0
//MikroeElektronika 2011
//http://www.mikroe.com
```

```
//GLCD FontName : font ter6x8
//GLCD FontSize : 6 x 8
#ifndef FONTDATA6x8_H_
#define FONTDATA6x8_H_

code char charmap[] = {
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00,      // Code for char
        0x00, 0x00, 0x06, 0x5F, 0x06, 0x00,      // Code for char !
        0x00, 0x07, 0x03, 0x00, 0x07, 0x03,      // Code for char "
        0x00, 0x24, 0x7E, 0x24, 0x7E, 0x24,      // Code for char #
        0x00, 0x24, 0x2B, 0x6A, 0x12, 0x00,      // Code for char $
        0x00, 0x63, 0x13, 0x08, 0x64, 0x63,      // Code for char %
        0x00, 0x36, 0x49, 0x56, 0x20, 0x50,      // Code for char &
        0x00, 0x00, 0x07, 0x03, 0x00, 0x00,      // Code for char '
        0x00, 0x00, 0x3E, 0x41, 0x00, 0x00,      // Code for char (
        0x00, 0x00, 0x41, 0x3E, 0x00, 0x00,      // Code for char )
        0x00, 0x08, 0x3E, 0x1C, 0x3E, 0x08,      // Code for char *
        0x00, 0x08, 0x08, 0x3E, 0x08, 0x08,      // Code for char +
        0x00, 0x00, 0xE0, 0x60, 0x00, 0x00,      // Code for char ,
        0x00, 0x08, 0x08, 0x08, 0x08, 0x08,      // Code for char -
        0x00, 0x00, 0x60, 0x60, 0x00, 0x00,      // Code for char .
        0x00, 0x20, 0x10, 0x08, 0x04, 0x02,      // Code for char /
        0x00, 0x3E, 0x51, 0x49, 0x45, 0x3E,      // Code for char 0
        0x00, 0x00, 0x42, 0x7F, 0x40, 0x00,      // Code for char 1
        0x00, 0x62, 0x51, 0x49, 0x49, 0x46,      // Code for char 2
        0x00, 0x22, 0x49, 0x49, 0x49, 0x36,      // Code for char 3
        0x00, 0x18, 0x14, 0x12, 0x7F, 0x10,      // Code for char 4
        0x00, 0x2F, 0x49, 0x49, 0x49, 0x31,      // Code for char 5
        0x00, 0x3C, 0x4A, 0x49, 0x49, 0x30,      // Code for char 6
        0x00, 0x01, 0x71, 0x09, 0x05, 0x03,      // Code for char 7
        0x00, 0x36, 0x49, 0x49, 0x49, 0x36,      // Code for char 8
        0x00, 0x06, 0x49, 0x49, 0x29, 0x1E,      // Code for char 9
        0x00, 0x00, 0x6C, 0x6C, 0x00, 0x00,      // Code for char :
        0x00, 0x00, 0xEC, 0x6C, 0x00, 0x00,      // Code for char ;
        0x00, 0x08, 0x14, 0x22, 0x41, 0x00,      // Code for char <
        0x00, 0x24, 0x24, 0x24, 0x24, 0x24,      // Code for char =
        0x00, 0x00, 0x41, 0x22, 0x14, 0x08,      // Code for char >
        0x00, 0x02, 0x01, 0x59, 0x09, 0x06,      // Code for char ?
        0x00, 0x3E, 0x41, 0x5D, 0x55, 0x1E,      // Code for char @
        0x00, 0x7E, 0x11, 0x11, 0x11, 0x7E,      // Code for char A
        0x00, 0x7F, 0x49, 0x49, 0x49, 0x36,      // Code for char B
        0x00, 0x3E, 0x41, 0x41, 0x41, 0x22,      // Code for char C
        0x00, 0x7F, 0x41, 0x41, 0x41, 0x3E,      // Code for char D
        0x00, 0x7F, 0x49, 0x49, 0x49, 0x41,      // Code for char E
        0x00, 0x7F, 0x09, 0x09, 0x09, 0x01,      // Code for char F
```

```
0x00, 0x3E, 0x41, 0x49, 0x49, 0x7A,      // Code for char G
0x00, 0x7F, 0x08, 0x08, 0x08, 0x7F,      // Code for char H
0x00, 0x00, 0x41, 0x7F, 0x41, 0x00,      // Code for char I
0x00, 0x30, 0x40, 0x40, 0x40, 0x3F,      // Code for char J
0x00, 0x7F, 0x08, 0x14, 0x22, 0x41,      // Code for char K
0x00, 0x7F, 0x40, 0x40, 0x40, 0x40,      // Code for char L
0x00, 0x7F, 0x02, 0x04, 0x02, 0x7F,      // Code for char M
0x00, 0x7F, 0x02, 0x04, 0x08, 0x7F,      // Code for char N
0x00, 0x3E, 0x41, 0x41, 0x41, 0x3E,      // Code for char O
0x00, 0x7F, 0x09, 0x09, 0x09, 0x06,      // Code for char P
0x00, 0x3E, 0x41, 0x51, 0x21, 0x5E,      // Code for char Q
0x00, 0x7F, 0x09, 0x09, 0x19, 0x66,      // Code for char R
0x00, 0x26, 0x49, 0x49, 0x49, 0x32,      // Code for char S
0x00, 0x01, 0x01, 0x7F, 0x01, 0x01,      // Code for char T
0x00, 0x3F, 0x40, 0x40, 0x40, 0x3F,      // Code for char U
0x00, 0x1F, 0x20, 0x40, 0x20, 0x1F,      // Code for char V
0x00, 0x3F, 0x40, 0x3C, 0x40, 0x3F,      // Code for char W
0x00, 0x63, 0x14, 0x08, 0x14, 0x63,      // Code for char X
0x00, 0x07, 0x08, 0x70, 0x08, 0x07,      // Code for char Y
0x00, 0x71, 0x49, 0x45, 0x43, 0x00,      // Code for char Z
0x00, 0x00, 0x7F, 0x41, 0x41, 0x00,      // Code for char [
0x00, 0x02, 0x04, 0x08, 0x10, 0x20,      // Code for char BackSlash
0x00, 0x00, 0x41, 0x41, 0x7F, 0x00,      // Code for char ]
0x00, 0x04, 0x02, 0x01, 0x02, 0x04,      // Code for char ^
0x80, 0x80, 0x80, 0x80, 0x80, 0x80,      // Code for char _
0x00, 0x00, 0x03, 0x07, 0x00, 0x00,      // Code for char `
0x00, 0x20, 0x54, 0x54, 0x54, 0x78,      // Code for char a
0x00, 0x7F, 0x44, 0x44, 0x44, 0x38,      // Code for char b
0x00, 0x38, 0x44, 0x44, 0x44, 0x28,      // Code for char c
0x00, 0x38, 0x44, 0x44, 0x44, 0x7F,      // Code for char d
0x00, 0x38, 0x54, 0x54, 0x54, 0x08,      // Code for char e
0x00, 0x08, 0x7E, 0x09, 0x09, 0x00,      // Code for char f
0x00, 0x18, 0xA4, 0xA4, 0xA4, 0x7C,      // Code for char g
0x00, 0x7F, 0x04, 0x04, 0x78, 0x00,      // Code for char h
0x00, 0x00, 0x00, 0x7D, 0x40, 0x00,      // Code for char i
0x00, 0x40, 0x80, 0x84, 0x7D, 0x00,      // Code for char j
0x00, 0x7F, 0x10, 0x28, 0x44, 0x00,      // Code for char k
0x00, 0x00, 0x00, 0x7F, 0x40, 0x00,      // Code for char l
0x00, 0x7C, 0x04, 0x18, 0x04, 0x78,      // Code for char m
0x00, 0x7C, 0x04, 0x04, 0x78, 0x00,      // Code for char n
0x00, 0x38, 0x44, 0x44, 0x44, 0x38,      // Code for char o
0x00, 0xFC, 0x44, 0x44, 0x44, 0x38,      // Code for char p
0x00, 0x38, 0x44, 0x44, 0x44, 0xFC,      // Code for char q
0x00, 0x44, 0x78, 0x44, 0x04, 0x08,      // Code for char r
0x00, 0x08, 0x54, 0x54, 0x54, 0x20,      // Code for char s
0x00, 0x04, 0x3E, 0x44, 0x24, 0x00,      // Code for char t
```

```
        0x00, 0x3C, 0x40, 0x20, 0x7C, 0x00,      // Code for char u
        0x00, 0x1C, 0x20, 0x40, 0x20, 0x1C,      // Code for char v
        0x00, 0x3C, 0x60, 0x30, 0x60, 0x3C,      // Code for char w
        0x00, 0x6C, 0x10, 0x10, 0x6C, 0x00,      // Code for char x
        0x00, 0x9C, 0xA0, 0x60, 0x3C, 0x00,      // Code for char y
        0x00, 0x64, 0x54, 0x54, 0x4C, 0x00,      // Code for char z
        0x00, 0x08, 0x3E, 0x41, 0x41, 0x00,      // Code for char {
        0x00, 0x00, 0x00, 0x77, 0x00, 0x00,      // Code for char |
        0x00, 0x00, 0x41, 0x41, 0x3E, 0x08,      // Code for char }
        0x00, 0x02, 0x01, 0x02, 0x01, 0x00,      // Code for char ~
        0x00, 0x3C, 0x26, 0x23, 0x26, 0x3C       // Code for char
        };
        #endif


/****************** g15623.c *****************************
*************** ATtiny 26 *******************************
**************** communication with glcd ******************
****** LeRoy Olson *********  Mar 1 2012 *****************/


#include "s6b0755.h"
//#include "fontdata.h"

//void switch_select(void);
void display_init(void);
void switch_select();

void lcd_reset(void);
void display_cr(void);
void lcd_write(unsigned short mdata);
void lcd_set_invert(unsigned short val);
void display_clear(void);
void Display_cr(void);
void display_text(code char *addr);
void display_set_line(unsigned short line);
void display_set_col(unsigned short col);
void display_user_putc(int cc);

int cc;
unsigned int value;

void InitMain()
{
   dt3;
   dt3;
   dt3;
         display_init(void);
         display_clear();
         lcd_set_invert(0);//0 white screen black text use 1 to reverse
  dt3;
        display_set_line(1);
        display_text("LeRoy Olson");
        display_set_line(2);
        display_text("AA aa Bb Cc Dd");
        display_set_line(3);
        display_text("================");
        display_set_line(4);
        display_text("AC VOLTS PEAK");
        display_set_line(5);
        display_text("DC MILLIAMPS");
        display_set_line(6);
        display_text("NPN SILICONE");
  dt1;
   display_set_line(7);
        display_set_col(70);
        cc=45;
        display_user_putc(cc);
        display_set_line(7);
        display_set_col(77);
        cc=56;
        display_user_putc(cc);
        display_set_line(7);
        display_set_col(84);
        cc=55;
        display_user_putc(cc);
  dt3;
}

void main()
{
 LCD_DDR_SET; // Set as output Pins 0-3,and 5
 LCD_PORT_SET;  //Set Pins 0-3 and 5 low


  InitMain();                              // Perform main initialization
}


//************ lcd reset ********************
void lcd_reset(void)
{
  LCD_RESETB=0;
  LCD_CS1B=1;
  LCD_SCK=1;
  LCD_SID=1;

  dt1;
  LCD_RESETB=1;
}
//************ lcd write *******************
void lcd_write(unsigned short mdata)
{

     short lp;
     LCD_SCK=1;
     LCD_SID=1;
     dt2;
     LCD_CS1B=0;
     dt2;
    for(lp=0; lp<8; lp++)
     {
               LCD_SID=(mdata & 0B10000000)>>7;
               LCD_SCK=0;
               dt2;
               LCD_SCK=1;
               dt2;
               mdata=mdata<<0X01;
    }
     LCD_SID=1;
     LCD_CS1B=1;
}

//****************** lcd send cmd **********************
void lcd_send_cmd(unsigned short mdata)
{
  LCD_RS=0;
  lcd_write(mdata);
  LCD_RS=1;
}

//****************** lcd send data *********************
void lcd_send_data(unsigned short mdata)
{
  LCD_RS=1;
  lcd_write(mdata);
}
```

```
//***************** lcd init *************************
void lcd_init(void)
{
    lcd_reset();
    lcd_send_cmd(SOFT_RESET);
    dt1;
    lcd_send_cmd(SET_DUTY_1);
    lcd_send_cmd(SET_DUTY_2);
    dt1;

    lcd_send_cmd(SET_BIAS);
    dt1;
    //**** Application setup done *******
    lcd_send_cmd(SET_OSC_ON);
    dt1;
    lcd_send_cmd(DC_STEP_UP);
    dt1;

    lcd_send_cmd(REG_RESISTOR | REG_RESISTOR_VAL);
    dt1;

    lcd_send_cmd(SET_EVR_1);
    lcd_send_cmd(SET_EVR_2 | VOLUME_CONTROL_VAL);
    dt1;

    lcd_send_cmd(SET_BIAS);
    dt1;
    lcd_send_cmd(POWER_CONTROL);
    dt1;

    lcd_send_cmd(CLEAR_POWER_SAVE);
    dt1;
    lcd_send_cmd(DISPLAY_INVERT | DISPLAY_INVERT_VAL);
    dt1;

    lcd_send_cmd(DISPLAY_ON);
    dt1;
}

///************** lcd set invert *********************************
void lcd_set_invert(unsigned short val)
{
  lcd_send_cmd(DISPLAY_INVERT | (DISPLAY_INVERT_VAL & val));
}
```

```
/***********************************************************************
****
***************************** s6b0755.c
***********************************
******************* commands for s6b0755 glcd
****************************
**** LeRoy Olson ************************** Feb 26 2012 ******************
************************************************************************
**/

#include "s6b0755.h"
#include "fontdata6x8.h"

void lcd_send_cmd(unsigned short mdata);
void lcd_init(void);
void lcd_set_invert(unsigned short val);
void lcd_send_data(unsigned short mdata);

#define MAX_COLS              96
#define MAX_ROWS               65
#define MAX_PAGES              MAX_ROWS/8

#define SCAN_DIR              0b00000001   //0 for normal 1 to flip upside down
#define BLACK_ON_WHITE         0b00000000   //0000 000x 0 white text on black 1
black text on white

/*****************************************************************
*************** DISPLAY COMMANDS START HERE *********************/

//***************** display carage return *******************
void display_cr(void)
{
 if(SCAN_DIR==0B00000000)
 {
  lcd_send_cmd(SET_COL_ADDR_HI | 0X00);
   lcd_send_cmd(SET_COL_ADDR_LO | 0X00);
 }
 else
 {
  lcd_send_cmd(SET_COL_ADDR_HI | 0X02);
   lcd_send_cmd(SET_COL_ADDR_LO | 0X00);
 }
}

//***** display Home **************************
void display_home(void)
```

```
{
  display_cr();
   lcd_send_cmd(SET_PAGE_ADDR | 0X00);
}

//************* display set line *************
void display_set_line(unsigned short line)
{
  lcd_send_cmd(SET_PAGE_ADDR | (line & 0X0F));
}

//************ display set column **********
void display_set_col(unsigned short col)
{
 if(SCAN_DIR== 0X00)
 {
    lcd_send_cmd(SET_COL_ADDR_HI | (col>>4 & 0B00000111));
     lcd_send_cmd(SET_COL_ADDR_LO | (col & 0B00001111));
 }
 else
 {
    col = col+32;
     lcd_send_cmd(SET_COL_ADDR_HI | (col>>4 & 0B00000111));
      lcd_send_cmd(SET_COL_ADDR_LO | (col & 0B00001111));
 }
}

//************display initialize *****************
void display_init(void)
{
  lcd_init();
  display_home();
  if(SCAN_DIR==0)
  {
      lcd_send_cmd(SHL_SELECT | 0B00000000);
       lcd_send_cmd(ADC_SELECT | 0B00000000);
  }
  else
  {
       lcd_send_cmd(SHL_SELECT | 0B00001000);
        lcd_send_cmd(ADC_SELECT | 0B00000001);
  }
  if(BLACK_ON_WHITE)
  {
    lcd_set_invert(BLACK_ON_WHITE);
  }
}

//*************** display text *****************
void display_text(code char *addr)
{
 code const char *charptr;
 unsigned short line =0;
 unsigned short temp =0;

 display_cr();

 while (*addr != 0&& line<MAX_COLS)
 {
  charptr=((((unsigned int)*addr)-32)*6)+ charmap; // can use glcd font creator
  for(temp=0;temp<6;temp++)
  {
        lcd_send_data(*charptr++);
        line++;
  }
  *addr++;
 }
 while (line<MAX_COLS)
 {
   lcd_send_data(0X00);
   line++;
 }
}

//************ user put ** User defined stream destination ******
void display_user_putc(int cc)
{
 code const char *charptr;
 unsigned short line=0;
 unsigned short temp=0;

 if (cc != 0 && line<MAX_COLS)
 {
  charptr=((((unsigned int)cc)-32)*6)+ charmap;
  for (temp=0;temp<6;temp++)
  {
   lcd_send_data(*charptr++);
   line++;
  }
 }
}
```

```
//*********** display clear ******************
void display_clear(void)
{
  unsigned short i,j;
  display_home();
  for (i=0;i<MAX_PAGES;i++)
  {
   display_set_line(i);
   display_cr();
   for (j=0;j<MAX_COLS;j++)
   {
    lcd_send_data(0X00);
   }
  }
  display_home();
}


/*  Written by LeRoy Olson  Jan 18 2011
 * Header Name: s6b0755.h
 * Description: Values for operating S6B0755 GLCD Driver in G15623 GLCD
 * Arguments: None
 * Returns:  None
 * NOTES: Header with all the glcd comand data
 *     S6B0755 defines
 */

#ifndef S6B0755_H_
#define S6B0755_H_

#define SOFT_RESET             0b11100010   //Initialize internal functions
#define SET_PAGE_ADDR          0b10110000   //0000xxxx page address

#define SET_COL_ADDR_HI        0b00010000   //00010xxx
#define SET_COL_ADDR_LO        0b00000000   //0000xxxx

#define SHL_SELECT             0b11000000   //COM bidrectionaol 1100 000X 0 normal,
1 rev scan direction.
#define ADC_SELECT             0b10100000   //Bi directional 0000 000X seg 0 norm, 1
rev.

#define SET_DUTY_1             0b01001000
#define SET_DUTY_2             0b01000000

#define SET_BIAS             0b01010001   //01010xxx = bias, Should be 001 for 1/5 bias
#define SET_OSC_ON             0b10101011   // start internal oscilator
```

```
#define DC_STEP_UP              0b01100100   //011001xx select 3-5 boosting
#define REG_RESISTOR            0b00100000   //00100000 (center)
#define REG_RESISTOR_VAL        0b00000001   // 0010 0xxx (center) range 0x00 to
0x03 B0000 0001 is best

#define SET_EVR_1               0b10000001   // s byte instruction
#define SET_EVR_2               0b00011100   //00xxxxxx=valus for center

#define POWER_CONTROL           0b00101111   // All circuits on
#define SET_POWER_SAVE          0b10101000   //1010100x power save level
0=standby, 1=sleep
#define CLEAR_POWER_SAVE        0b11100001

#define DISPLAY_ON              0b10101111    //1010 111x 1=display on 0=display off
#define DISPLAY_OFF             0b10101110

#define DISPLAY_INVERT          0b10100110   //1010011x, 0= normal,1= invert video
#define DISPLAY_INVERT_VAL      0b00000001   //range 0000 0000 to 0000 0001

#define ALL_BLACK               0b10100101   //1010010x 1 forces entire display on
0=normal
#define VOLUME_CONTROL_VAL      0b00011100   //range 00XX XXXX to 0011
1111

#define dt1                 Delay_us(60)// works with 60 us delay
#define dt2                 Delay_us(1) //works with 1 ss delay
#define dt3                 Delay_ms(600) //Set delay for switches on stk500 board

#define LCD_DDR_SET          DDRA=0B00101111 //Pins 0 to 3 and 5 as output
#define LCD_PORT_SET         PORTA=0B11010000//Set pins 0 to 3 and 5 low

#define LCD_RS              PORTA1_bit   //PORTA1_bit
#define LCD_SCK             PORTA0_bit   //PORTA0_bit
#define LCD_RESETB          PORTA3_bit   //PORTA3_bit
#define LCD_SID             PORTA2_bit   //PORTA2_bit
#define LCD_CS1B            PORTA5_bit   //PORTA5_bit

#endif /* S6B0755_H_ */
```