

Configuring the HD44780 LCD controller / driver which is built onto the range of Hitachi Character Liquid Crystal Display Modules.

The HD44780 gives the user the ability to display alphanumerics, Kana characters, symbols and also the facility to generate custom characters. The interface to a microcontroller/processor can be either 8-bit or 4-bit wide. The HD44780 can drive upto 16 characters, configured as 1 line by 16 characters or 2 lines by 8 characters. However, built into the device is a display data RAM area which can support upto 80 characters maximum, so by attaching 9 off HD44100's (40-channel segment driver) for a 1 line display with a duty factor of 1/8 or 1/11, or 4 off HD44100 for a 2 line display with a duty factor of 1/16 you will be able to drive upto 80 characters.

The functions below have been extracted from a program which is designed to drive any Hitachi Character LCD Module which has the HD44780 controller/driver built-in. This display module is controlled via the Hitachi H8/325 8-bit single chip microcontroller. To enable customers to fully evaluate the H8/325, we have produced a low cost evaluation board, the following code has been written to run on this EV board (LEV8325). The displays control and data lines are controlled via a PIA. (HD63B21)

/* This string of characters is to be displayed on the character display module. */

```
const char char_screen [ ] = {'T','h','i','s',' ','C','h','a','r','a','c','t','e','r',' ','M','o','d','u','l','e',' ','i','s',' ','t','h','e',' ','L','M','O','9','2','L','N',' ','w','h','i','c','h',' ','i','s',' ','a',' ','2',' ','b','y',' ','4',' ','0','D','i','s','p','l','a','y',0x00,0x01,0x02};
```

/* This functon reads back the status of the data bus and is called during the 'Busy Flag' check.*/

```
char read_lcd ( char reg )
{
char temp, read_val;

write_E_port (PIA1_CRA,0);          /* enable DDRA access */
write_E_port (PIA1_DRA,0);          /* port A all inputs */
write_E_port (PIA1_CRA,0x04);       /* enable DRA access */
write_E_port (PIA1_CRB,0x04);
temp = read_E_port (PIA1_DRB);      /* read control signal values */
if ( reg )                          /* if reg is true then access lcd data reg */
    temp = temp | 0x03;             /* set RS and R/W pins */
else
    {
    temp = temp | 0x02;              /* set R/W pin only */
    temp = temp & 0xfe;             /* ensure RS pin = 0 */
    }
write_E_port (PIA1_DRB,temp);        /* output control signals RS and R/W */
temp = temp | 0x04;                 /* set E clock pin */
```

```
write_E_port (PIA1_DRB,temp);          /* output E clock */
for ( read_val = 0; read_val <= 10; read_val ++ )
    { }                                 /* delay to allow data set up from lcd*/
read_val = read_E_port (PIA1_DRA);     /* get data */
temp = temp & 0x0b;                    /* clear E clock pin */
write_E_port (PIA1_DRB,temp);         /* output E clock */
temp = temp & 0xf8;                    /* clear RS and R/W pins */
write_E_port (PIA1_DRB,temp);
return (read_val);
}
```

/* This function is used to send data to the HD44780, to set up the various registers and also to send the character information to the display data RAM. */

```
void write_lcd ( char reg, char write_val )
{
char temp,i;

write_E_port (PIA1_CRA,0);             /* enable DDRA access */
write_E_port (PIA1_DRA,0xff);         /* port A all outputs */
write_E_port (PIA1_CRA,0x04);         /* enable DRA access */
write_E_port (PIA1_DRA, write_val);   /* output the data */
temp = read_E_port (PIA1_DRB);        /* read control signal values */
if ( reg )                             /* if reg is true then access lcd data reg */
    {
temp = temp | 0x01;                    /* set RS pin */
temp = temp & 0xfd;                    /* clear R/W pin */
    }
else
temp = temp & 0xfc;                     /* ensure R/W and RS pins = 0 */
write_E_port (PIA1_DRB,temp);         /* output R/W and RS signals */
temp = temp | 0x04;                    /* set E pin */
write_E_port (PIA1_DRB,temp);         /* output E signal */
for ( i = 0; i <=10; i++ )
    { } /* short delay */
temp = temp & 0xfb;                     /* clear E pin */
write_E_port (PIA1_DRB,temp);         /* output E signal */
temp = temp & 0xf8;                     /* clear RS and R/W pins */
write_E_port (PIA1_DRB,temp);         /* output RS and R/W signals */
}
```

/* This function checks the status of the 'Busy Flag'. If DB7 is High ('1'), this indicates that the HD44780 is busy processing the previous instruction. The controller can only accept the next instruction when DB7 is Low ('0').*/

```
void wait_lcd ( void )
{
while (read_lcd (lcd_control) & 0x80) /* wait for busy flag = '0' */
{ }
}
```

/* Before writing any screen data to the display you need to initialise it in terms of number of display lines, cursor, font etc. This function sets up the controller to drive a 2 line display, 5*7 dots, cursor and blink ON, with increment. */

```
void set_up_lcd ( void )
{
char temp;
long i;

/* first set up the PIA */
write_E_port (PIA1_CRB,0x04);          /* allow access to DR */
temp = read_E_port (PIA1_DRB);
temp = temp & 0xf8;                    /* ensure RS, R/W and E are inactive */
write_E_port (PIA1_DRB,temp);
write_E_port (PIA1_CRB,0x00);          /* select access to DDRB */
write_E_port (PIA1_DRB,0x07);          /* RS, R/W and E to outputs */
write_E_port (PIA1_CRB,0x04);          /* allow access to DRB */
write_E_port (PIA1_CRA,0x04);          /* allow access to DRA */
write_E_port (PIA1_DRA,0x00);          /* data bus to 0 */
write_E_port (PIA1_CRA,0x00);          /* allow access to DDRA */
write_E_port (PIA1_DRA,0xff);          /* all port A are outputs */
write_E_port (PIA1_CRA,0x04);          /* allow access to DRA */
for (i = 0; i <= 10000; i++)
    {}

write_lcd ( lcd_control,0x38);          /* initialise display function set, need to
                                        send thrice to ensure correct initialisation */
for (i=0; i <= 10000; i++);            /* delay */
for (i=0; i <= 10000; i++);            /* delay */
write_lcd (lcd_control,0x38);           /* send again! */
for (i=0; i <=10000; i++);             /* delay */
write_lcd ( lcd_control,0x38);         /* and again! */
wait_lcd ( );                          /* now check 'Busy Flag' */
write_lcd (lcd_control,0x01);           /* clear display */
wait_lcd ();
write_lcd ( lcd_control,0x0f);          /* display ON, cursor ON with Blink */
wait_lcd ();
write_lcd (lcd_control,0x02);           /* return to home position */
}

void custom ( void )
{
set_cgram_addr(0x40);                  /* address location for custom chars */
wait_lcd();                             /* check status of busy flag */
write_lcd(lcd_data,0x11),                /* 1st byte of bit pattern for 1st custom char */
write_lcd(lcd_data,0x0f),
write_lcd(lcd_data,0x0f),
write_lcd(lcd_data,0x11),
write_lcd(lcd_data,0x1e),
```

```
write_lcd(lcd_data,0x1e),
write_lcd(lcd_data,0x01),
write_lcd(lcd_data,0x00),

write_lcd(lcd_data,0x00),          /* 1st byte of bit pattern for 2nd custom char */
write_lcd(lcd_data,0x1b),
write_lcd(lcd_data,0x1b),
write_lcd(lcd_data,0x1b),
write_lcd(lcd_data,0x1b),
write_lcd(lcd_data,0x1b),
write_lcd(lcd_data,0x1b),
write_lcd(lcd_data,0x00),

write_lcd(lcd_data,0x1f),          /* 1st byte of bit pattern for 3rd custom char */
write_lcd(lcd_data,0x15),
write_lcd(lcd_data,0x1f),
write_lcd(lcd_data,0x1b),
write_lcd(lcd_data,0x1f),
write_lcd(lcd_data,0x11),
write_lcd(lcd_data,0x1f),
write_lcd(lcd_data,0x00),
}

void main ( void )
{

int n;

set_up_lcd ();                    /* initialise the display */
wait_lcd ( );
n=0;
while ( char_screen[n] ) {
    write_lcd ( lcd_data, char_screen [n++]); /* send the character string to the
                                                LCM*/
}

}
```

When using this document, keep the following in mind,

1. This document may, wholly or partially, be subject to change without notice.

2. All rights are reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without Hitachi's permission.
3. Hitachi will not be held responsible for any damage to the user that may result from accidents or any other reasons during the operation of the user's unit according to this document.
4. Circuitry and other examples described herein are meant only to indicate the characteristics and performance of Hitachi's semiconductor products. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples therein.
5. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi, Ltd.
6. **MEDICAL APPLICATIONS:** Hitachi's products are not authorised for use in **MEDICAL APPLICATIONS** without the written consent of the appropriate officer of Hitachi's sales company. Such use includes, but is not limited to, use in life support systems. Buyers of Hitachi's products are requested to notify the relevant sales office when planning to use the products in **MEDICAL APPLICATIONS**.

Copyright ©Hitachi, Ltd.,1993. All rights reserved