Introduction:

The F-51320 series liquid crystal display (LCD) from Optrex America is an easy to use and versatile LCD technology suitable for a variety of applications. Utilizing chip on glass manufacturing techniques, the 128 wide by 64 high pixel display is ideal for presenting graphics, text, or both in a wide range of configurations. Because of its high level of integration, the F-51320 series LCD minimizes the amount of time required for the engineer to realize his or her design goals. This document contains information that will allow you to quickly realize your design objectives while minimizing the debug and set-up time commonly found with other display technologies.

Note: The F-51320 series display can be configured in a number of different ways depending on the requirements of the application. This includes the power supply sub-system as well as the digital interface. This document will not describe every such combination, but will however, provide design guidance for a specific configuration in sufficient detail to enable the engineer to adapt it to his or her own requirements.[1]

Hardware and Interface:

The F-51320 series LCD uses a flex-circuit connector to interface the power, control, and data lines to your target application. The connector utilizes a surface mount Horz-Bottom contact type and is available from several manufacturers. One manufacturer is Hirose Electronic Co. Ltd. (http://www.hirose.com). You may reference the manufacturer's part number FH12-30S-0.5SH or the Digi-Key (http://www.digikey.com) part number HFJ30CT-ND.

The schematic of the connector with the associated signals are shown below in Figure 1:

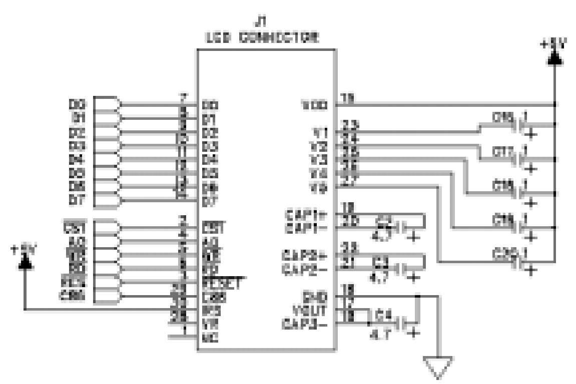

Figure 1: FH12-30S-0.5SH Connector with Signals

[1] For detailed configuration information, reference the *Epson SED1500 Series LCD Driver with RAM Technical Manual*, chapter 8.

Note that the capacitors (C2, C3, C4) are used for the 3X charge pump "step-up" circuit. Using the internal boost capability of the LCD module, these three capacitors and a software command are all that is required to generate the –9 volt rail needed by the LCD. The capacitors (C15, C17, C18, C19, C20) are used to stabilize the output voltages of V1 through V5 respectively. These values can be modified depending on the loading and intensity of the images targeted for the display. In this example, 1µF are sufficient in driving the LCD reliably under heavy load conditions (e.g. horizontal stripes).

With the exception of power and ground, all other signals interface directly with the microcontroller. Signals D0-D7 represent an 8-bit parallel, bi-directional data bus. The remaining output signals under processor control are chip select ($\overline{CS}1$), address select (A0), write ($\overline{WR}$), read ($\overline{RD}$), reset ($\overline{RES}$), and (C86). With the exception of A0 and C86, all of the control signals are active low. Note that C86 controls the type of MPU interface expected by the LCD module. When C86 is pulled high, the LCD module is expecting a "6800 Series MPU Interface" whereas C86 low signals an "8080 Series MPU Interface." This signal would rarely need to be under MPU control and typically can be tied statically, either high or low.

This concludes the hardware interface required for the LCD module. All other configurations are performed under software control.

Software Control:

Since nearly the entire interface to the LCD module is performed under software control, it is critical to configure your microcontroller bus and port pins correctly. Use this checklist and review your configuration:

1. MPU data bus can be configured as inputs or outputs dynamically
2. All control signals are outputs
3. With the exception of A0 and C86, all of the control signals are active low

Once you are satisfied that the MPU interface is correct, the most important software capability is to be able to reliably communicate with the LCD module. Several useful routines written in C are provided throughout this document. An example of a low-level display write and display read commands are found on the following page:

```
/*****************************************************************************
** Write_Display ********************************************** MKO 13JUN01 ****
*******************************************************************************
This is the one of the main low level routines used to write either commands
or data to the SED1565 LCD driver.  A flag must be passed to the routine to
indicate whether it is being used to pass a command or data to the controller.


Usage Examples:
      Write_Display(CMD,RESET_DISPLAY);
      Write_Display(DATA,i);

******************************************************************************/

void Write_Display(Uint8 command, Uint8 data)

      /* Configure the bus for output.  Ensure the control lines transfer
         ownership of the bus from the display to the microcontroller. */
      DISPLAY_CONTROL = BUS_RELEASE;
      DIRECTION_8 = OUTPUT_BUS;

      /* Load the control signals and data for a particular command. */
      if(command == CMD)
         DISPLAY_A0 = LO;
      else
         DISPLAY_A0 = HI;

      CS1 = LO;
      DISPLAY_WRITE = LO;

          /* Place the data on the bus. */
      DATA_8 = data;

      DISPLAY_WRITE = HI;
      CS1 = HI;

      /* Latch the data into the display.  The minimum latch time is 60 nsec
         for a display write.  Since the maximum bus speed for a HC908 device
         is 8 MHz, the minimum instruction time is 125 nsec.  Therefore, back to
         back instructions can be used. */

      /* After the microcontroller is finished, it will always release the bus
         to the display. */

      DIRECTION_8 = INPUT_BUS;
      DISPLAY_CONTROL = BUS_RELEASE;
};

/*--------------------------------------------------------------------------*/



/*****************************************************************************
** Read_Display *********************************************** MKO 13JUN01 ****
*******************************************************************************
This is the one of the main low level routines used to read data and the status
byte from the SED1565 LCD controller.  A flag must be passed to the routine to
indicate whether it is being used to pass a command or data to the controller.


Usage Examples:
      status_read_byte = Read_Display(CMD);
      display_data = Read_Display(DATA);

******************************************************************************/

Uint8 Read_Display(Uint8 command)
{
      Uint8 return_byte;

      DISPLAY_CONTROL = BUS_RELEASE;

      /* Configure the bus for input. */
      DIRECTION_8 = INPUT_BUS;

      /* Load the control signals and data for a particular command. */

      if(command == CMD)
         DISPLAY_A0 = LO;
      else
         DISPLAY_A0 = HI;

      /* Latch the data into the display.  The minimum latch time is 120 nsec
         for a display write.  Since the maximum bus speed for a HC908 device
         is 8 MHz, the minimum instruction time is 125 nsec.  Therefore, back to
         back instructions can be used. */

      CS1 = LO;
      DISPLAY_READ = LO;

      /* Grab the data from the bus. */
      return_byte = DATA_8;
      DISPLAY_READ = HI;
      CS1 = HI;

      DISPLAY_CONTROL = BUS_RELEASE;

      return(return_byte);
};

/*--------------------------------------------------------------------------*/
```

Using only these basic commands, you are able to initialize, configure, and control the LCD module[2].

In C, it can be useful to describe all of the specific command codes using the **#define** descriptor. This allows you to refer to the commands in your program without having to recall the specific binary code associated with each command. An example of these declarations are shown below and on the following page:

```
/*-----------------------------------------------------------------------------*/

/* **** External Hardware Definitions **** */

#define NUMBER_OF_COLUMNS                    128
#define NUMBER_OF_ROWS                       64

#define NUMBER_OF_PAGES                      8

#define FIRST_PAGE                           0
#define FIRST_COLUMN                         0

#define LAST_PAGE                            7
#define LAST_COLUMN                          127


/* **** Global Function Prototypes **** */

extern void Write_Display(Uint8 command, Uint8 data);
extern Uint8 Read_Display(Uint8 command);


/*-----------------------------------------------------------------------------*/


/* These are the flags that are passed to Write_Display() or Read_Display()
   routines. */
#define CMD                                  0x01
#define DATA                                 0xFF

/*-----------------------------------------------------------------------------*/

/* The individual bit declarations of the control lines. */
#define CS1_BIT                              0x80
#define A0_BIT                               0x40
#define WR_BIT                               0x20
#define RD_BIT                               0x10
#define RES_BIT                              0x08
#define C86_BIT                              0x04

/*-----------------------------------------------------------------------------*/

/* The BUS_RELEASE define is used to the control bus output so that all control
   signals are pulled high.  This ensures that when the microcontroller gains
   control of the bus, it does not change the direction of the bus to outputs
   while the display is driving the bus. */
#define BUS_RELEASE                          0xF8

/*-----------------------------------------------------------------------------*/
```

[2] The entire command set is described in detail in the *Epson SED1500 Series LCD Driver with RAM Technical Manual*, chapter 8, section 7.

```
/*-----------------------------------------------------------------------------*/

/* The following definitions are the command codes that are passed to the
   display via the data bus. */
#define DISPLAY_ON                              0xAF
#define DISPLAY_OFF                             0xAE

#define START_LINE_SET                          0x40

#define PAGE_ADDRESS_SET                        0xB0

/* The Column Address is a two byte operation that writes the most significant
   bits of the address to D3 - D0 and then writes the least significant bits to
   D3- D0. Since the Column Address auto increments after each write, direct
   access is infrequent. */

#define COLUMN_ADDRESS_HIGH                     0x10
#define COLUMN_ADDRESS_LOW                      0x00

#define ADC_SELECT_NORMAL                       0xA0
#define ADC_SELECT_REVERSE                      0xA1

#define DISPLAY_NORMAL                          0xA6
#define DISPLAY_REVERSE                         0xA7

#define ALL_POINTS_ON                           0xA5

#define LCD_BIAS_1_9                            0xA2
#define LCD_BIAS_1_7                            0xA3

#define READ_MODIFY_WRITE                       0xE0

#define END                                     0xEE

#define RESET_DISPLAY                           0xE2

#define COMMON_OUTPUT_NORMAL                    0xC0
#define COMMON_OUTPUT_REVERSE                   0xC8

/* The power control set value is obtained by OR'ing the values together to
   create the appropriate data value.  For example:

      data = (POWER_CONTROL_SET | BOOSTER_CIRCUIT |
              VOLTAGE_REGULATOR | VOLTAGE_FOLLOWER);

   Only the bits that are desired need be OR'ed in because the initial value
   of POWER_CONTROL_SET sets them to zero. */

#define POWER_CONTROL_SET                       0x28
#define BOOSTER_CIRCUIT                         0x04
#define VOLTAGE_REGULATOR                       0x02
#define VOLTAGE_FOLLOWER                        0x01

/* The initial value of the V5_RESISTOR_RATIO sets the Rb/Ra ratio to the
   smallest setting.  The valid range of the ratio is:

      0x20 <= V5_RESISTOR_RATIO <= 0x27 */

#define V5_RESISTOR_RATIO                       0x26

/* When the electronic volume command is input, the electronic volume register
   set command becomes enabled. Once the electronic volume mode has been set,
   no other command except for the electronic volume register command can be
   used. Once the electronic volume register set command has been used to set
   data into the register, then the electronic volume mode is released. */

#define ELECTRONIC_VOLUME_SET                   0x81
#define ELECTRONIC_VOLUME_INIT                  0x20

/*-----------------------------------------------------------------------------*/
```

Now using the display write routine and the control value declarations, you can easily initialize the display module.

```
/*******************************************************************************
** Init_Display ********************************************** MKO 13JUN01 ****
********************************************************************************
This performs all of the necessary initialization of the SED-1565 controller
inside the Optrex display.


Usage Examples:
      Init_Display();

*******************************************************************************/

void Init_Display(void)
{

      DISPLAY_RESET = LO;                     /* Hold the display in reset. */
      DISPLAY_RESET = LO;                     /* Multiple instructions to guarantee timing. */


      DISPLAY_RESET = HI;                     /* Release the display in reset. */
      DISPLAY_RESET = HI;                     /* Multiple instructions to guarantee timing. */

      /* When the RES input comes to the "L" level, the display is initialized to a
         default state.  That default is as follows:
```

```
        1) Display OFF
        2) Normal display
        3) ADC select: Normal (ADC command D0 = 0)
        4) Power control register: (D2, D1, D0) = (0, 0, 0)
        5) Serial interface internal register data clear
        6) LCD power supply bias rate: 1/9 bias
        7) All-indicator lamps-on OFF (All-indicator lamps ON/OFF command D0 = 0)
        8) Power saving clear
        9) V5 voltage regulator internal resistors Ra and Rb separation
       10) Output conditions of SEG and COM terminals SEG : V2/V3, COM : V1/V4
       11) Read modify write OFF
       12) Static indicator OFF.  Static indicator register : (D1, D2) = (0, 0)
       13) Display start line set to first line
       14) Column address set to Address 0
       15) Page address set to Page 0
       16) Common output status normal
       17) V5 voltage regulator internal resistor ratio set mode clear
       18) Electronic volume register set mode clear.  Electronic volume register:
           (D5, D4, D3, D2, D1 ,D0) = (1, 0, 0, 0, 0, 0)
       19) Test mode clear

       The initialization sequence of the SED-1565 should perform the following
       functions within 5 msec after the release of reset:

          1. LCD Bias Set
          2. ADC Set
          3. Common Output
          4. Built in Resistance set
          5. Electronic Volume set
          6. Power Control */
/*-------------------------------------------------------------------------*/

        Write_Display(CMD,RESET_DISPLAY);

        Write_Display(CMD,LCD_BIAS_1_9);

        Write_Display(CMD,ADC_SELECT_REVERSE);

        Write_Display(CMD,COMMON_OUTPUT_NORMAL);

        Write_Display(CMD,V5_RESISTOR_RATIO);

        Write_Display(CMD,ELECTRONIC_VOLUME_SET);
        Write_Display(CMD,ELECTRONIC_VOLUME_INIT);

        /* Since we are using the all features of the power control register, we
           must set all of the flags within the power control set command.  The
           individual bits are OR'ed together resulting in the correct control command. */
        Write_Display(CMD,(POWER_CONTROL_SET | VOLTAGE_REGULATOR |
                      VOLTAGE_FOLLOWER | BOOSTER_CIRCUIT));

        Write_Display(CMD,DISPLAY_ON);
};

/*-------------------------------------------------------------------------*/
```

Once you have successfully initialized the unit, you may then pass your display data to the LCD module. You can use the checkerboard generator code below to test your display:

```
/*******************************************************************************
** Write_Checkerboard ***************************************** MKO 14JUN01 ****
********************************************************************************
This generates a checkerboard test pattern across the entire display.  The size
of the square is passed to the routine.  The routine requires that the checker
size be an integer multiple of a single page (i.e. 1, 2, 4, or 8 pixels)


Usage Examples:
      Write_Checkerboard(4);

*******************************************************************************/
void Write_Checkerboard(Uint8 width)
{

Uint8 i,j,page,pattern;

/* Determine the pattern for the appropriately sized checkerboard.  If the
      desired width is not a legitimate value of 1, 2, 4, or 8, it will be sized
      to 8 by default. */
      switch(width)
      {
         case 1:
            pattern = 0x55;
            break;

         case 2:
            pattern = 0x33;
            break;

         case 4:
            pattern = 0x0F;
            break;

         default:
            width = 8;
            pattern = 0x00;
      };

      /* Initialize the column address to zero and allow the auto increment to
         move the column address after each write. */
```

```
      for(page=0;page<NUMBER_OF_PAGES;page++)
      {
         Write_Display(CMD,START_LINE_SET);
         Write_Display(CMD,PAGE_ADDRESS_SET + page);
         Write_Display(CMD,COLUMN_ADDRESS_HIGH);
         Write_Display(CMD,COLUMN_ADDRESS_LOW);

         /* If the width is eight, then complement the pattern every other
            page.  This is a special case since the pattern for a single square
            occupies the entire page. */
         if(width == 8)
            pattern = ~pattern;

         for(i=0;i<NUMBER_OF_COLUMNS/width;i++)
         {
            for(j=0;j<width;j++)
               Write_Display(DATA,pattern);

            pattern = ~pattern;
         };
       };

      /* Turn on the display to see the new checkerboard. */
      Write_Display(CMD,DISPLAY_ON);
 };
/*------------------------------------------------------------------------------*/
```

Now you have all of the necessary tools needed to test your display. Your main routine could look something like this:

```
/*------------------------------------------------------------------------------*/
void main(void)
{
Uint8 size[4] = {1,2,4,8};
static Uint8 x;


      Init_Ports();
      Init_Display();

      while(1)
      {
         Write_Checkerboard(size[x]);
         Delay_Ms(250);
         x++;
         if(x>3)
            x=0;
      };
/*------------------------------------------------------------------------------*/
```

As expected, this should produce a checkerboard pattern with the size of the squares changing between 1, 2, 4, and 8 pixels every quarter second.

Special Considerations:

One of the more difficult aspects of working with an LCD controller is contrast adjustment. The implications of inappropriate contrast are obvious. Even if you are communicating properly with the module, you cannot see the effect of the data you have transmitted. The F-51320 series LCD module provides 9-bits of software contrast adjustment. This is further subdivided into a 3-bit rough adjustment called “V5 Voltage Regulator Internal Resistor Ratio Set” and a 6-bit fine adjustment called “Electronic Volume”. The values provided in the initialization code should work in most cases but it is possible that your particular design requires a different combination of resistor ratio and electronic volume. The code on the following page will allow you to sweep all resistor ratios and volume settings in software. As the routine is running, you can monitor the display and determine exactly which settings are appropriate for your application.

```
/*------------------------------------------------------------------------*/
#define RESISTOR_RATIO_START                  0x20

void Scan_Contrast(void)
{
Uint8 i,j;

      for(i=0;i<8;i++)
      {
         Write_Display(CMD,RESISTOR_RATIO_START+i);

         for(j=0;j<64;j++)
         {
            Write_Display(CMD,ELECTRONIC_VOLUME_SET);
            Write_Display(CMD,j);
            printf("The resistor ratio is %d and the e-volume is %d.\r\n",i,j);
            Delay_Ms(1000);
         };
      };
};
/*------------------------------------------------------------------------*/
```

Conclusion:

F-51320 series LCD module can add significant capability to your product while requiring very little engineering investment. By following the guidelines in this document, you can minimize the time needed for interfacing the display to your target and focus your efforts on your specific application.