128x32 Cholesteric Display Module with LCD Controller

Product Description

This Graphic Display Module delivers maximum image information content flexibility in an extremely compact form factor. It is the most compact and cost efficient bi-stable graphic display available today.

All Kent Displays Cholesteric Liquid Crystal Display (ChLCD) products take advantage of the technology's unique "No Power" attribute without compromising superior optical performance even in direct sunlight. The Chip-on-Glass (COG) driver IC includes an LCD controller and a DC/DC charge pump. System integration requires only an external microcontroller for sending commands and timing signals to the LCD controller, and external capacitors for use by the built-in charge pump.

Product Features

Display Module:

- 128 Columns x 32 Rows
- Approximate size: 1.30x0.61x0.1 Inches
- Available in Multiple Color Schemes
- Low Profile, Compact Design
- Superior Brightness
- Excellent Optical Properties
- Viewing Cone Comparable to Paper
- Indefinite Image Memory ("No Power")

Controller:

- Serial Command and Data Interface
- Integrated DC/DC Charge Pump

Typical Applications

- **Battery Powered Portable Devices**
- **USB Powered Devices**
- **Inventory Tracking Displays**
- **Remote Control Display Applications**

Kent Displays, Inc.
343 Portage Boulevard
Kent, OH 44240 USA

Telephone: 330.673.8784
Fax: 330.673.4408
Email: sales@kentdisplays.com
Website: www.kentdisplays.com

Specification Summary

Display Module with COG Controller

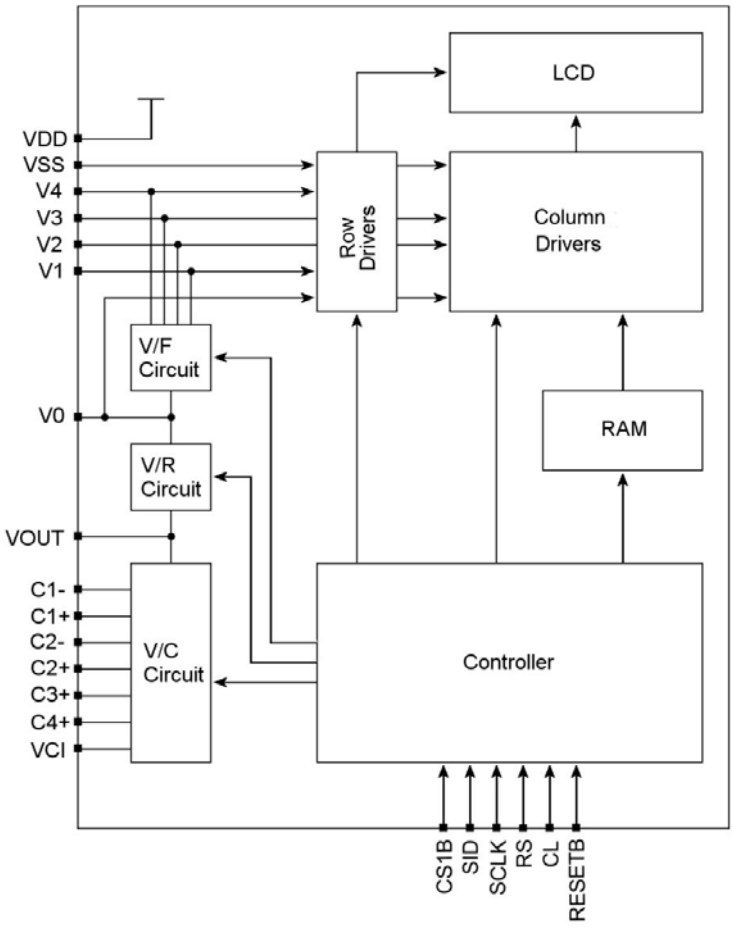
General Specifications	
Parameter	**Description**
Display Type	Cholesteric Reflective LCD
Format	128 Columns x 32 Rows
Resolution	118 dots per inch (0.215mm pixel pitch, horizontal & vertical)
Image Area	1.08 in × 0.27 in (27.5 mm × 6.9 mm)
Display Module Weight	0.07 oz (1.9 grams)
Operating Temperature Range	0°C to +50°C (custom operating temperatures available)
Storage Temperature Range	-10°C to +80°C
Full Image Update Rate	~3.25 seconds (0.5 sec. charge + 2.75 sec. scan) @ 25°C

Contents

1 Overview

The 128x32 ILV (KDI Part #15674) is a passive matrix display module ideally suited for battery/low voltage powered portable devices and display applications that require superior optical performance including wide viewing angle and sunlight readability. The display is a reflective cholesteric liquid crystal display (ChLCD) that takes full advantage of the technology's unique "No Power" image retention attribute. The embedded driver IC contains internal DC/DC conversion circuitry which requires only a small number of external capacitors for generating the LCD drive voltages. Image data and commands are transferred to the display module from the host using a serial interface. The host system controls the image update through a simple sequence of commands and a clock signal.

2 Block Diagram



3 Electrical Interface

3.1 Header

The display module interfaces to the host electronics through a 24-contact flex cable (see Figure 1). Electrical connection may be made by using a 0.5 mm pitch flat flex connector (Hirose part number FH12-24S-0.5SH(55) or equivalent) or by bonding to pads directly on the host PCB.

3.2 Pin Summary

Number	Name	Type[1]	Description
1	TEST0	NC	For factory test
2	V0	Power	LCD driver supply voltage
3	V4	Power	LCD driver supply voltage
4	V3	Power	LCD driver supply voltage
5	V2	Power	LCD driver supply voltage
6	V1	Power	LCD driver supply voltage
7	C2-	DC/DC	Capacitor 2 negative connection pin for voltage converter
8	C2+	DC/DC	Capacitor 2 positive connection pin for voltage converter
9	C1+	DC/DC	Capacitor 1 positive connection pin for voltage converter
10	C1-	DC/DC	Capacitor 1 negative connection pin for voltage converter
11	C3+	DC/DC	Capacitor 3 positive connection pin for voltage converter
12	C4+	DC/DC	Capacitor 2 positive connection pin for voltage converter
13	VOUT	DC/DC	Voltage converter output
14	VSS	Supply	Ground
15	VCI	Supply	Voltage converter supply input
16	VDD	Supply	Power supply
17	SID	Input	Serial input data (DB7)
18	SCLK	Input	Serial input clock (DB6)
19	RS	Input	Register select input pin RS = “H” : Serial data input is display data RS = “L” : Serial data input is control data
20	RESETB	Input	Reset input pin When RESETB is “L”, initialization is executed
21	CS1B	Input	Chip select pin Data /instruction I/O is enabled only when CS1B is “L”
22	CL	Input	Display clock input pin
23	TEST1	NC	For factory test
24	TEST2	NC	For factory test

[1]Notes:
- Supply - provide power to the display module
- Input - logic input from the host controller
- DC/DC – used by internal voltage converter to generate the LCD drive voltage
- Power – provide voltages during display driving (charged by voltage regulator/follower)
- NC – no connection (May be used to debug drive waveforms during development.)

3.3 Reference Schematic

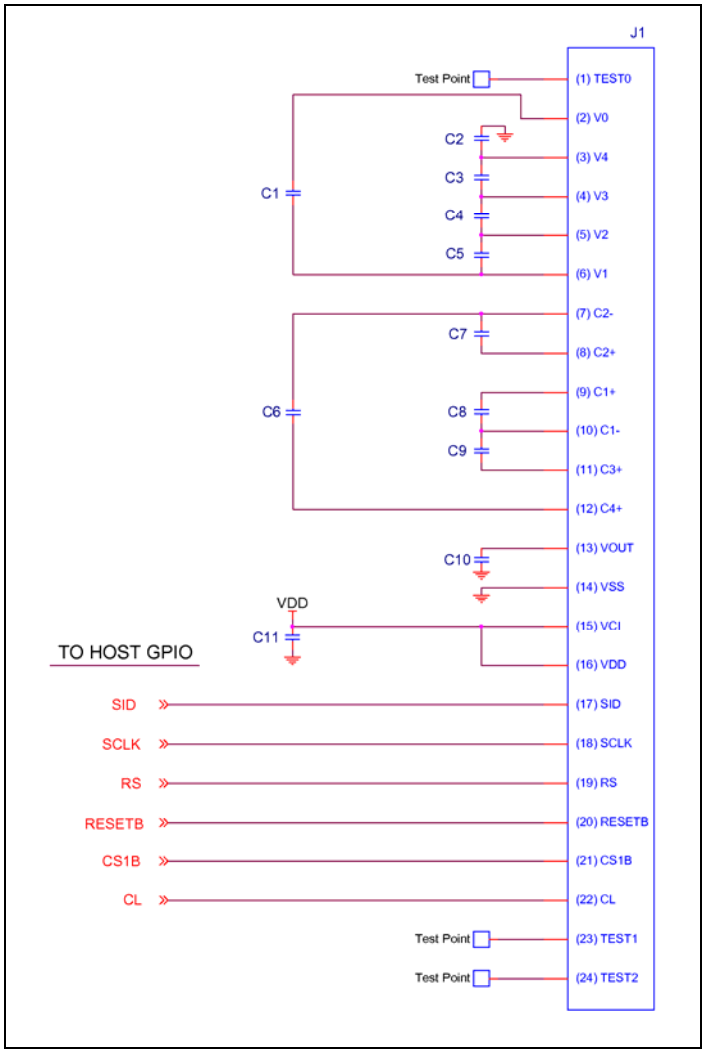


Figure 1: Reference Schematic

Table 1: Reference Component Values

Component	Value	Reference Manufacturer, P/N
C1, C2, C3, C4, C5, C6, C7, C8, C9	2.2µF/16V	Murata, GRM188R61C225KE15D
C10	4.7µF/25V	Panasonic, ECJ-2FB1E475M
C11[1]	0.1µF/16V	Vishay, VJ0402V104ZXJCW1BC
J1	24-pin, 0.5 mm, bottom contact	Hirose, FH12-24S-0.5SH(55)

[1] Depending upon the source impedance of the supply, additional capacitance may be required to stabilize VDD when running the internal DC/DC converter.

4 Operating Principles

4.1 Bistability

The unique bistable property of ChLCD products means that an image placed on the display will remain indefinitely without the need for refreshing. This results in a different paradigm for managing image content from a traditional TN or STN type display. In particular, image data sent to the display controller RAM does not immediately appear on the display screen. The image data only appears on the display screen after executing a defined update sequence. The update sequence involves sending commands and data over the serial interface as well as specialized control of the clock input to the display module.

4.2 Serial Interface

The display module functions similar to an SPI slave device (see Figure 2). The host system (master) selects the display module for communication using the CS1B line and provides the clock signal (SCLK) used to clock in data. Communication with the display begins with a high-to-low transition on CS1B and ends with a low-to-high transition on CS1B. New data (instructions or image data) for the display module are placed on SID on falling edges of SCLK, and the display controller latches the data on the rising edge of SCLK. Transmission of each byte begins with the most significant bit (DB7) and ends with the least significant bit (DB0). The RS signal is sampled with DB0. The byte (DB7 to DB0) is treated as an instruction if RS is sampled low and as image data if RS is sampled high.

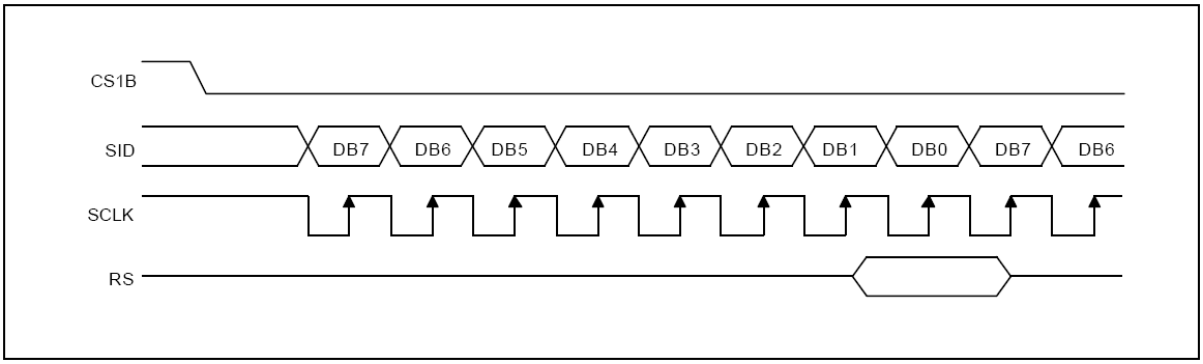


Figure 2: Basic Serial Interface Timing

4.3 Image Data

The graphic display is 128 columns by 32 rows. The 32 rows are organized into 4 pages which are each 8 rows high. A single byte of data encodes the image on the 8 vertical pixels corresponding to a given page and column. Figure 3 illustrates how the pixel data may be mapped to the 512 bytes of image data for the display. Note that bright pixels are encoded as 1's and dark pixels are encoded as 0's.

Page	Bit	Column Address				
		0	1	2	...	127
0	0	BYTE 0	BYTE 1	BYTE 2	...	BYTE 127
	1					
	2					
	3					
	4					
	5					
	6					
	7					
1	0	BYTE 128	BYTE 129	BYTE 130	...	BYTE 255
	1					
	2					
	3					
	4					
	5					
	6					
	7					
2	0	BYTE 256	BYTE 257	BYTE 258	...	BYTE 383
	1					
	2					
	3					
	4					
	5					
	6					
	7					
3	0	BYTE 384	BYTE 385	BYTE 386	...	BYTE 511
	1					
	2					
	3					
	4					
	5					
	6					
	7					

Figure 3: Image Data to Pixel Mapping

The sample code in Section 6 illustrates how the 512 bytes of image data ordered as in Figure 3 may be sent to the driver for display.

5 Specifications

5.1 General

General Specifications	
Parameter	**Description**
Display Type	Cholesteric Reflective LCD
Format	128 Columns x 32 Rows
Resolution	118 dots per inch (0.215mm pixel pitch, horizontal & vertical)
Image Area	1.08 in × 0.27 in (27.5 mm × 6.9 mm)
Display Module Weight	0.07 oz (1.9 grams)
Operating Temperature Range	0°C to +50°C (custom operating temperatures available)
Storage Temperature Range	-10°C to +80°C
Full Image Update Rate	~3.25 seconds (0.5 sec. charge + 2.75 sec. scan) @ 25°C

5.2 Absolute Maximum Ratings

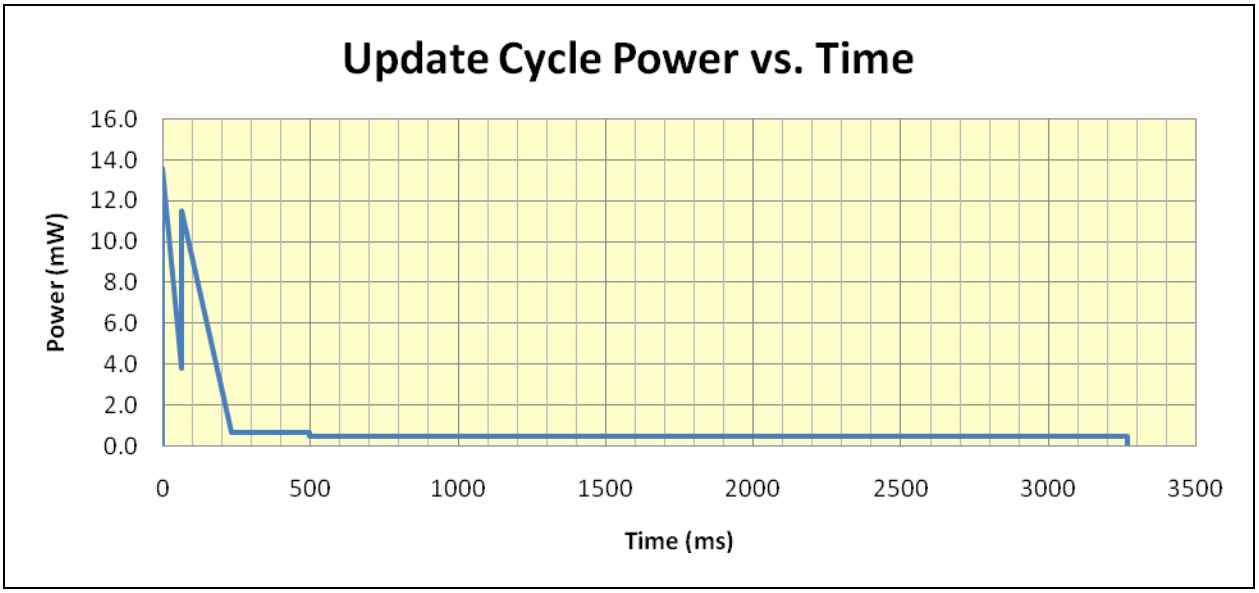
Parameter	Symbol	Rating	Units
Supply Voltage Range	V_{DD}	-0.3 to +3.5	V
Input Voltage Range	V_{in}	-0.3 to (V_{DD} + 0.3)	V
Operating Temperature Range	T_{OPR}	0 to +50	ºC
Storage Temperature Range	T_{STR}	-10 to +80	ºC

5.3 Electrical

Parameter		Symbol	Min.	Typ.	Max.	Units
Operating Voltage		V_{DD}	+3.1	+3.3	+3.4	V
Input Voltage	High	V_{IH}	0.8 V_{DD}	-	V_{DD}	V
	Low	V_{IL}	V_{SS}	-	0.2 V_{DD}	V
Output Voltage	High	V_{OH}	0.8 V_{DD}	-	V_{DD}	V
	Low	V_{OL}	V_{SS}	-	0.2 V_{DD}	V
Sleep Mode Current		I_{DDS}	-	-	2.5	µA

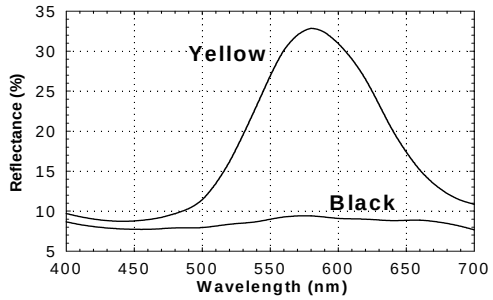
* Specifications are subject to change without prior notice.

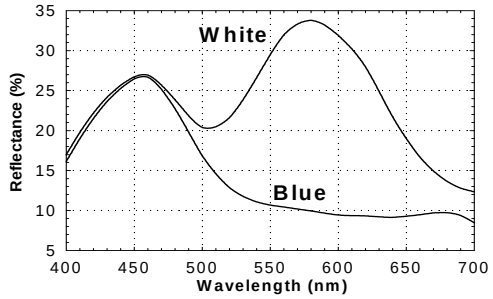
5.4 Power Profile

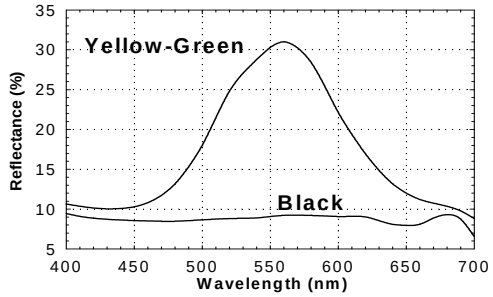


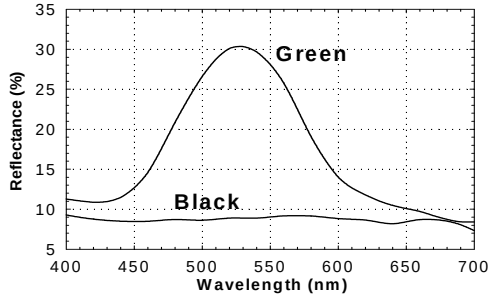
Note: Power measurements based on VDD = 3.3V.

5.5 Optical





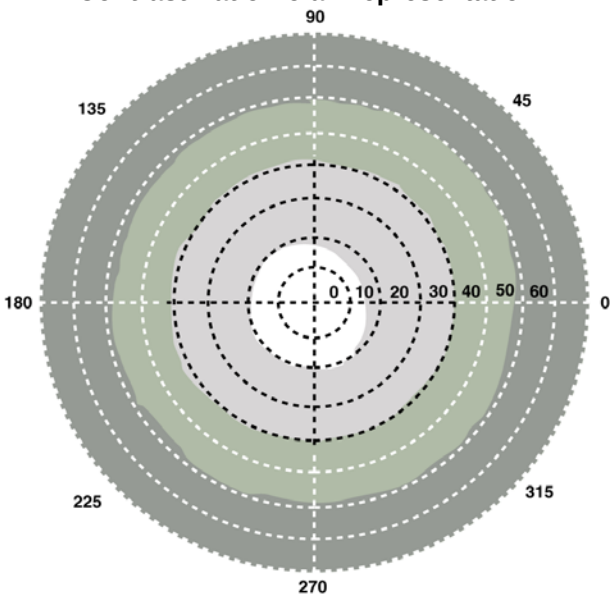




The above reflectance curves are from a single pixel. Actual reflectance will vary depending on display resolution, aperture ratio, and other factors.

The graphs to the left outline the spectral reflectance characteristics for a given display pixel when switched to either of the two possible stable states: reflective planar or transparent focal conic. The top line in each chart outlines the reflective characteristic of the planar state. The bottom line outlines the reflective characteristic of the transparent focal conic state. Graphs for typical color combinations are illustrated.

Contrast Ratio Polar Representation



As illustrated in the polar graph above, all Kent Displays' ChLCD products have a 360-degree viewing cone. When measured normal to the plane of the display, the monochromatic contrast ratio is as high as 25:1 with a peak reflectivity approaching 35% of the incident light. The contrast ratio reduces as the viewing angle approaches the plane of the display but is still excellent at 11:1. Since no polarizers are used, display contrast reduces uniformly in all azimuthal directions when the viewing angle is increased.

5.6 Mechanical

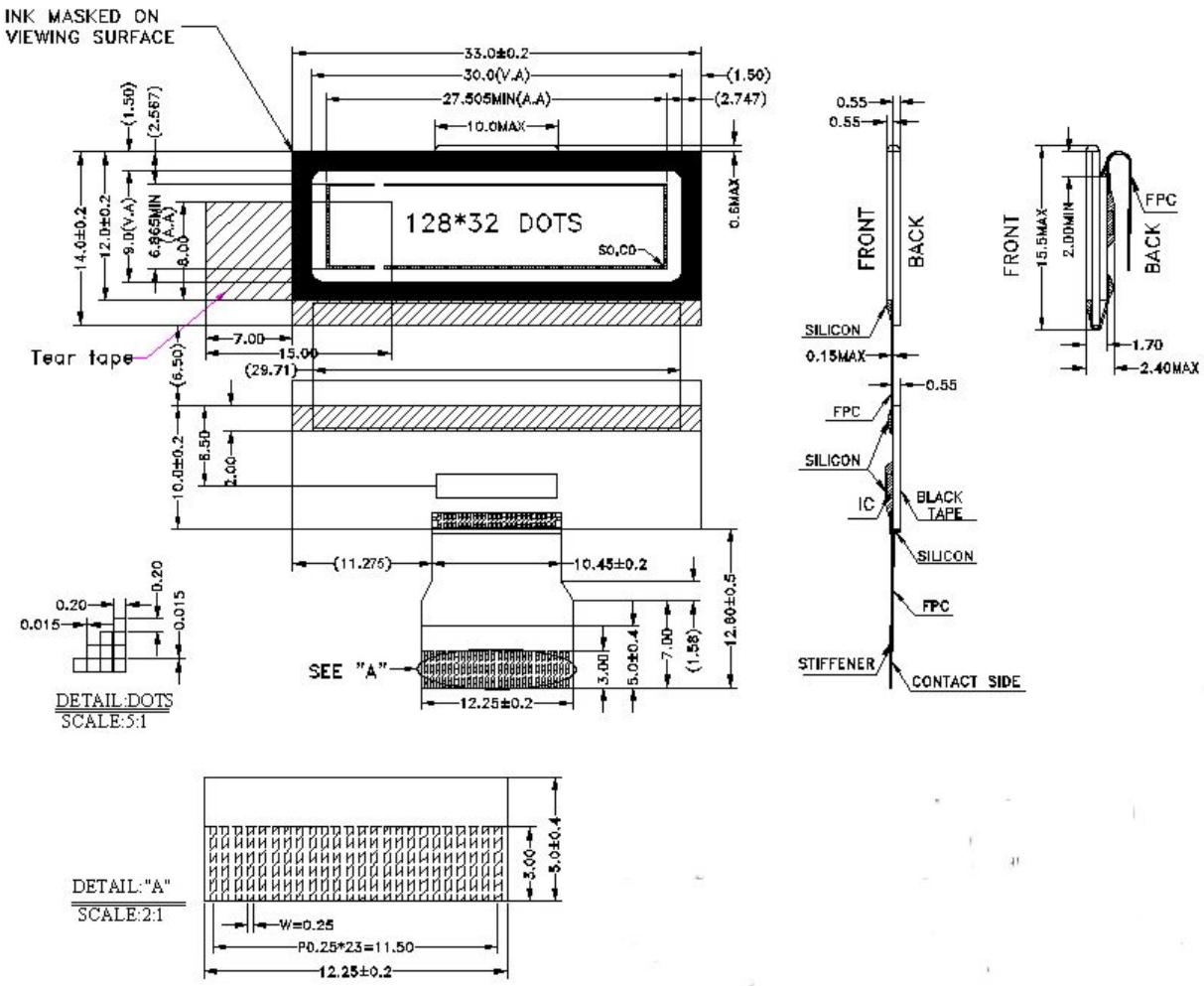


Figure 4: Module Dimensions (mm)

FRONT COVER REQUIREMENTS:

The following front cover requirements are necessary to insure image quality during the life of the display module:

1. Cholesteric Liquid Crystal materials require protection from UV light. A UV blocking material with a minimum 98% cutoff at 380nm and lower spectral components is required.

2. The finished product design should incorporate a transparent cover such as acrylic, polycarbonate, etc., to protect the viewing area of the display. Place the protective cover as close to the display module as possible. The protective cover should be of sufficient thickness to resist flexing, or if flexed should not touch the surface of the display.

Acrylite® OP-3 P-99 (matte finish) and Acrylite® OP-3 (without matte finish) are examples of protective cover materials that also provide the required UV blocking.

Adding an anti-glare and/or anti-reflective surface film or finish to the viewing side of the protective cover may improve the optical performance in certain display applications and lighting conditions.

5.7 Timing

5.7.1 Serial Interface

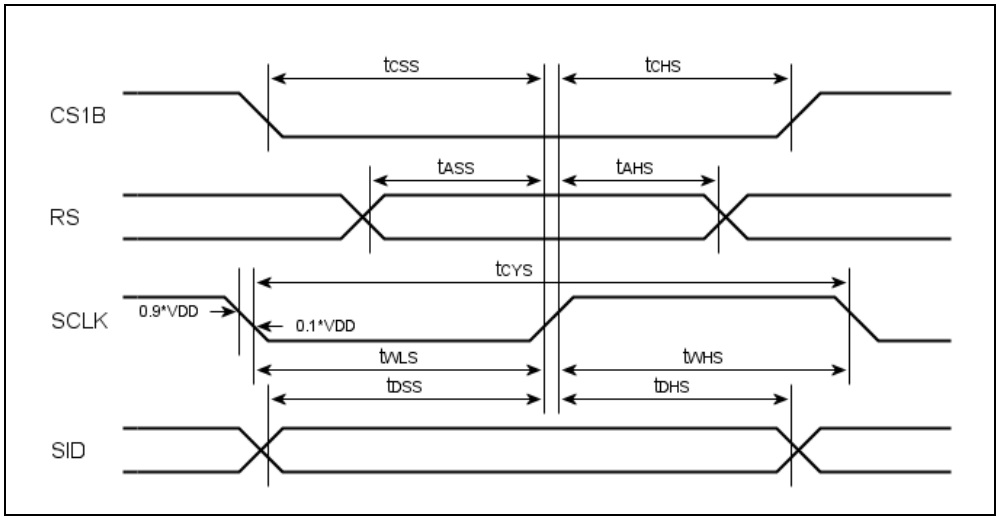


Figure 5: Serial Timing

Table 2: Serial Interface Timing Parameter Values

Item	Signal	Symbol	Min.	Typ.	Max.	Units
Serial clock cycle	SCLK	tCYS	250	-	-	ns
SCLK high pulse width		tWHS	100	-	-	
SCLK low pulse width		tWLS	100	-	-	
Address setup time	RS	tASS	150	-	-	ns
Address hold time		tAHS	150	-	-	
Data setup time	SID	tDSS	100	-	-	ns
Data hold time		tDHS	100	-	-	
CS1B setup time	CS1B	tCSS	150	-	-	ns
CS1B hold time		tCHS	150	-	-	

5.7.2 Reset

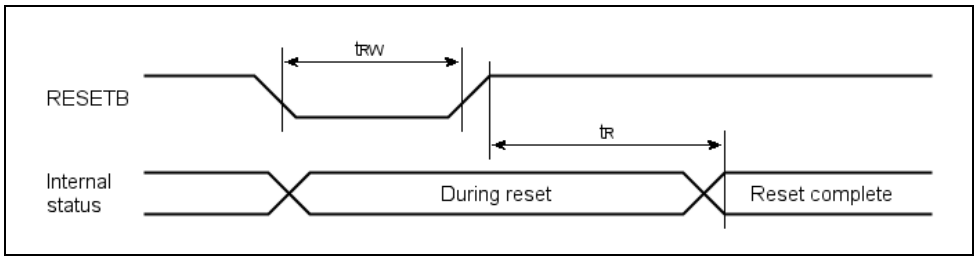


Figure 6: Reset Timing

Table 3: Reset Timing Parameter Values

Item	Signal	Symbol	Min.	Typ.	Max.	Units
Reset low pulse width	RESETB	t_{RW}	1.0	-	-	µs
Reset time	-	t_R	-	-	1.0	µs

6 Sample Code

Source code is provided in "C" for implementing a display update. A small number of user-defined functions and parameters specific to the host platform must be implemented independently.

6.1 User-Defined Code

6.1.1 Type Definitions

The source code uses three data types: uint8, uint16, and uint32. These types are for unsigned integers of the indicated number of bits. An example definition (this varies by compiler) is as follows:

```
#define uint8 unsigned char      // unsigned 8-bit data type
#define uint16 unsigned short    // unsigned 16-bit data type
#define uint32 unsigned long     // unsigned 32-bit data type.
```

6.1.2 Timing

A function for implementing variable length delays is also required. An example function declaration, assuming the delay duration is specified as a number of ticks of a timer resource, is as follows:

```
void delay(uint32 ticks);
```

The sample code requires that four parameters for the delay function be defined to implement delays of specific required durations. Their definitions are as follows, where 'x' must be replaced with the appropriate number of timer ticks in each case:

```
#define PRIME_TICKS       x  // delay timer ticks in 8 milliseconds
#define CHARGE_TICKS      x  // delay timer ticks in 212 microseconds
#define RECHARGE_TICKS    x  // delay timer ticks in 100 microseconds
#define DEFAULT_PW_TICKS  x  // delay timer ticks for 20C PW (see LookupParams())
```

6.1.3 Clock Signal Control

The sample code uses the `ClearCL()` and `SetCL()` functions to set the state of the CL signal (see Figure 1) to low and high, respectively. The `ClockDisplay()` function similarly is used to first clear the CL signal (low) and then set the CL signal (high). These three functions must either be implemented or the code lines replaced to control the CL signal accordingly.

6.1.4 Command and Data Transmission

Commands and data are transmitted serially to the display according to the serial protocol of Figure 2 and subject to the timing constraints of Figure 5. The source code requires that two functions be implemented, one for sending commands and the other for sending data. The functions have the following declarations:

```
void SendCommand(uint8 cmd);
void SendData(const uint8 *pData, uint8 NumBytes);
```

The `SendCommand()` function transmits a single byte command (the `cmd` parameter) to the display. The `SendData()` function transmits data to the display. The `pData` parameter points to an array of data bytes to

be written and the `NumBytes` parameter tells the number of bytes to be written. Note that the RS signal (see Figure 1) is used to distinguish between commands and data.

6.1.5 Display Reset and Low Power Mode

A function must be implemented to reset the display according to the timing in Figure 6. This function should be called once the system is initially powered. This function is declared as follows:

```
void ResetDisplay();
```

The sample code automatically puts the display into low power mode after an update. However, this mode is not automatically entered by the controller following a hard reset. The following command should be sent after a reset to put the display into low power mode:

```
SendCommand( (ENTIRE_DISP_ON_OFF | EON_ENTIRE) );
```

6.2 Standard Code

The following code should require only minimal changes, limited mainly to defines.h and LookupParams.c, once the user-defined code is created. Note that the Display() function is the entry point for performing display updates.

6.2.1 defines.h

```
// Driver command definitions.

// Display ON/OFF
#define DISPLAY_ON_OFF      0xAE
#define DON_ON              0x01
#define DON_OFF             0x00

// Set page address
#define SET_PAGE_ADDRESS    0xB0

// Set column address MSB/LSB
#define SET_COL_ADD_MSB     0x10
#define SET_COL_ADD_LSB     0x00

// Column to segment output map
#define ADC_SELECT          0xA0
#define ADC_NORMAL          0x00
#define ADC_REVERSE         0x01

// Entire display ON/OFF
#define ENTIRE_DISP_ON_OFF  0xA4
#define EON_NORMAL          0x00
#define EON_ENTIRE          0x01

// LCD bias select
#define LCD_BIAS_SELECT     0xA2
#define BIAS_0              0x00
#define BIAS_1              0x01

// Power control
#define POWER_CONTROL       0x28
```

```
#define PWR_VC              0x04
#define PWR_VR              0x02
#define PWR_VF              0x01

// Regulator resistor select
#define REG_RESISTOR_SEL    0x20
#define REG_R2              0x04
#define REG_R1              0x02
#define REG_R0              0x01

// Set reference voltage mode
#define SET_REF_VOLT_MODE   0x81

// Set reference voltage register
#define SET_REF_VOLT_REG    0x00

// Set static indicator mode
#define SET_STATIC_IND_MODE 0xAC
#define SM_OFF              0x00
#define SM_ON               0x01

// Set static indicator register
#define SET_STATIC_IND_REG  0x00
#define SI_S1               0x02
#define SI_S0               0x01


////////////////////////////////////////////////////////////////////////////
// TODO:  Create user-defined data types, delay durations, and functions and
//        uncomment.

// User-defined data types
// #define uint8 unsigned char
// #define uint16 unsigned short
// #define uint32 unsigned long

// User-defined delay durations
// #define PRIME_TICKS       x   // delay timer ticks in 8 milliseconds
// #define CHARGE_TICKS      x   // delay timer ticks in 212 microseconds
// #define RECHARGE_TICKS    x   // delay timer ticks in 100 microseconds
// #define DEFAULT_PW_TICKS  x   // delay timer ticks for 20C PW
//                                 (seeLookupParams())

// User-defined functions
// extern void delay(uint32);
// extern void ClearCL();
// extern void SetCL();
// extern void ClockDisplay();
// extern void ResetDisplay();
// extern void SendCommand(uint8);
// extern void SendData(const uint8 *, uint8);
////////////////////////////////////////////////////////////////////////////

// Standard functions
extern void ChargeAC(uint16, uint32);
extern void Display(const uint8 *);
extern void DriveImage(uint8, uint8, uint32, uint32);
extern void LoadImage(const uint8 *);
extern void LookupParams(uint8 *, uint8 *, uint32 *, uint32 *);
```

6.2.2 Display.c

```
#include "defines.h"

////////////////////////////////////////////////////////////////////////////
// Function:  Display
// Purpose:   Updates the display with the provided data.
// Inputs:    *pImage – pointer to 512 bytes of bitmap data
// Outputs:   None.
// Notes:     Doesn't return until update is complete.
////////////////////////////////////////////////////////////////////////////
void Display(const uint8 *pImage)
{
  uint8 scans, pulses;
  uint32 driveTicks, pauseTicks;

  LookupParams( &scans, &pulses, &driveTicks, &pauseTicks );

  ResetDisplay();
  SendCommand( (ADC_SELECT | ADC_REVERSE) ); // remap columns

  LoadImage( pImage );
  DriveImage( scans, pulses, driveTicks, pauseTicks );
}
```

6.2.3 LoadImage.c

```
#include "defines.h"

////////////////////////////////////////////////////////////////////////////
// Function:  LoadImage
// Purpose:   Load the display RAM with a fullscreen bitmap.
// Inputs:    pData - pointer to 512 bytes of image data
// Notes:     None.
////////////////////////////////////////////////////////////////////////////
void LoadImage(const uint8 *pData)
{
  uint8 page;

  // Load each of the four 128-byte pages.
  for (page = 0; page < 4; page++)
  {
    // Set display page address;
    SendCommand( (SET_PAGE_ADDRESS | page) );

    // Set display column address.
    SendCommand( (SET_COL_ADD_MSB | 0x00) );  // Or in address high nibble.
    SendCommand( (SET_COL_ADD_LSB | 0x00) );  // Or in address low nibble.

    // Send page of data to display.
    SendData(pData, 128);

    // Advance data pointer to next page.
    pData += 128;
  }
}
```

6.2.4 ChargeAC.c

```
#include "defines.h"

////////////////////////////////////////////////////////////////////////////
// Function:  ChargeAC
// Purpose:   Charge capacitors (usually with 250 Hz waveform on glass).
// Inputs:    frames - Number of display frames to charge over.
//            chargeTicks - Timer ticks between transitions in charge cycles.
// Outputs:   None.
// Notes:     There are 4 charge cycles per frame.
////////////////////////////////////////////////////////////////////////////
void ChargeAC(uint16 frames, uint32 chargeTicks)
{
  uint16 i;
  uint8 j;

  for (i = 0; i < frames; i++)
  {
    for (j = 0; j < 4; j++)
    {
      ClearCL();
      delay(chargeTicks);
      SetCL();
      delay(chargeTicks);
    }
    for (j = 4; j < 33; j++)
    {
      ClearCL();
      SetCL();
    }
  }
}
```

6.2.5 DriveImage.c

```
#include "defines.h"

////////////////////////////////////////////////////////////////////////////
// Function:  DriveImage
// Purpose:   Updates the display with the image data in the driver.
// Inputs:    numScans - number of image scans
//            numPulses - number of drive pulses per scan on a row
//            driveTicks - number of clock ticks in a drive pulse
//            pauseTicks - number of clock ticks to pause between pulses
// Outputs:   None.
// Notes:     Voltage regulator used for ~13.5V drive voltage.
//            Doesn't return until image update is complete.
//            Either numScans or numPulses should be even number >= 2.
//            Internal Supply - drive capacitors recharge after every row.
////////////////////////////////////////////////////////////////////////////
void DriveImage(uint8 numScans, uint8 numPulses, uint32 driveTicks,
                uint32 pauseTicks)
{
  uint16 i;
  uint8 scan, row, pulse;
  const uint8 RechargeCycles = 1;  // Controls amount of capacitor recharging.
```

```
// No update if pulse ticks not valid.
if (driveTicks == 0)
  return;

// Release power save (if on).
SendCommand( (ENTIRE_DISP_ON_OFF | EON_NORMAL) );
SendCommand( (SET_STATIC_IND_MODE | SM_ON) );
SendCommand( (SET_STATIC_IND_REG | 0x00) );

// Configure drive voltage levels.
SendCommand( (LCD_BIAS_SELECT | BIAS_0) );        // LCD Bias = 1/6
SendCommand( (REG_RESISTOR_SEL | (REG_R2 + REG_R1 + REG_R0)) );
SendCommand( (SET_REF_VOLT_MODE) );
SendCommand( (SET_REF_VOLT_REG | (0x3f & 63)) );

// Turn on voltage converter to charge up V0, V1, V2, V3, and V4 capacitors.
SendCommand( (POWER_CONTROL | (PWR_VC)) );

// Low frequency clock ticks 'prime' the charge pump.
ChargeAC(1, PRIME_TICKS);     // 1x4 charge cycles.
ChargeAC(32, CHARGE_TICKS);   // 32x4 charge cycles (250 Hz nonselect).

// Enable regulator and give 4x4 charge cycles (250 Hz nonselect)
SendCommand( (POWER_CONTROL | (PWR_VC + PWR_VR)) );
ChargeAC(4, CHARGE_TICKS);

// Enable follower and give 190x4 charge cycles (250 Hz nonselect).
SendCommand( (POWER_CONTROL | (PWR_VC + PWR_VR + PWR_VF)) );
ChargeAC(190, CHARGE_TICKS);

// Disable converter and follower.
SendCommand( (POWER_CONTROL | PWR_VR) );

// Turn on display.
SendCommand( (DISPLAY_ON_OFF | DON_ON) );

// Perform update.
// Capacitors recharge after each row by rapidly clocking to the frame
// end, disabling display, charging caps, enabling display, and rapidly
// clocking to next display row.
for (scan = 0; scan < numScans; scan++)
{
  // Loop through all rows to drive.
  for (row = 0; row < 33; row++)
  {
    // Loop through all but last drive pulse for current row/scan.
    for (pulse = 0; pulse < (numPulses - 1); pulse++)
    {
      // clock down to drive row.
      for (i = 0; i < row; i++)
        ClockDisplay();

      // Let display drive.
      delay(driveTicks);

      // Clock to end of frame.
      for (i = row; i < 33; i++)
        ClockDisplay();
```

```
        // Stir up display (drive row sees nonselect at this time).
        delay(pauseTicks);
      }

      // Drive final pulse (fewer post-drive clocks and no stir time)

      // clock down to drive row.
      for (i = 0; i < row; i++)
        ClockDisplay();

      // Let display drive.
      delay(driveTicks);

      // Clock to end of frame.
      for (i = row; i < 32; i++)
        ClockDisplay();

      // Charge up DC/DC.
      SendCommand( (DISPLAY_ON_OFF | DON_OFF) );
      SendCommand( (POWER_CONTROL | (PWR_VC + PWR_VR + PWR_VF)) );
      for (i = 0; i < (RechargeCycles*66 + 1); i++)
      {
        ClearCL();
        delay(RECHARGE_TICKS);
        SetCL();
        delay(RECHARGE_TICKS);
      }
      SendCommand( (POWER_CONTROL | PWR_VR) );
      SendCommand( (DISPLAY_ON_OFF | DON_ON) );
    }
  }

  // Disable voltage converter, regulator, and follower.
  SendCommand( (POWER_CONTROL | 0x00) );

  // Turn off display.
  SendCommand( (DISPLAY_ON_OFF | DON_OFF) );

  // Enter power save.
  SendCommand( (ENTIRE_DISP_ON_OFF | EON_ENTIRE) );
}
```

6.2.6 LookupParams.c

```
#include "defines.h"

////////////////////////////////////////////////////////////////////////////
// Function:  LookupParams
// Purpose:   Updates the display with the image data in the driver.
// Inputs:    pNumScans – ptr. to number of image scans
//            pNumPulses – ptr. to number of drive pulses per scan on a row
//            pDriveTicks – ptr. to number of ticks in a drive pulse
//            pPauseTicks – ptr. to number of ticks to pause between pulses
// Outputs:   None.
////////////////////////////////////////////////////////////////////////////
void LookupParams(uint8 *pNumScans, uint8 *pNumPulses, uint32 *pDriveTicks,
```

```
                  uint32 *pPauseTicks)
{
   // Constant values.
   *pNumScans = 2;
   *pNumPulses = 1;
   *pPauseTicks = 0;

   // Default pulsewidth prior to implementing temperature compensation.
   *pDriveTicks = DEFAULT_PW_TICKS;

   // TODO:  Measure temperature and set *pDriveTicks to the proper value for
   //        the Delay() function to implement the temperature-dependent
   //        delay from the table below.
   //
   //        Interpolation may be used or the pulsewidth may be set to the
   //        nearest table entry lower than the current temperature.  For
   //        instance, use the 20 C entry if the temperature is 23.5 C.
   //
   //        Set *pDriveTicks to 0 if the temperature is outside of the table.
   //
   //
   //        Temp (deg. C)    PW (microseconds)
   //        -------------    -----------------
   //             50                 7000
   //             45                 9000
   //             40                12000
   //             35                15000
   //             30                20000
   //             25                27000
   //             20                36000
   //             15                49000
   //             10                66000
   //              5                97000
   //              0               155000
}
```

6.2.7 SampleImage.c

```
#include "defines.h"

// A pointer to this array may be passed to the Display() function for initial
// debugging.  The data is otherwise not required.
const uint8 sampleImage[512] =
{
  // chlcd_tech.bmp
 0xFF,0xFF,0xFF,0xFF,0x07,0x03,0xF9,0xFD,0xFD,0xF9,0xFB,0xFF,0x01,0x01,0xCF,0xE7,
 0xE7,0x0F,0x1F,0xFF,0x01,0x01,0xFF,0xFF,0xFF,0xFF,0xFF,0x07,0x03,0xF9,0xFD,0xFD,
 0xF9,0xFB,0xFF,0x01,0x01,0xFD,0xF9,0x03,0x07,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFD,0xFD,0x01,0x01,0xFD,0xFD,0xFF,0x0F,0x07,0xB7,
 0xB7,0x87,0x8F,0xFF,0x0F,0x07,0xF7,0xF7,0xE7,0xEF,0xFF,0x01,0x01,0xE7,0xF7,0x07,
 0x0F,0xFF,0x07,0x07,0xF7,0xF7,0x07,0x0F,0xFF,0x0F,0x07,0xF7,0xF7,0x07,0x0F,0xFF,
 0xFD,0xFD,0x01,0x01,0xFF,0xFF,0xFF,0x0F,0x07,0xF7,0xF7,0x07,0x0F,0xFF,0x8F,0x07,
 0x77,0x77,0x07,0x0F,0xFF,0x87,0x07,0x7F,0x7F,0x07,0x07,0xFF,0xFF,0xFF,0xFF,0xFF,
 0x07,0x07,0x07,0x07,0x07,0x06,0x04,0x05,0x05,0x04,0x06,0xC7,0xC4,0xC4,0xC7,0x07,
 0x07,0x04,0x04,0x07,0x04,0x04,0x05,0xC5,0xC5,0xC5,0xC7,0xC7,0xC6,0xC4,0xC5,0x05,
 0x04,0x06,0x07,0xE4,0xE4,0x05,0x04,0xE6,0xE7,0x07,0x07,0x07,0x07,0x07,0x07,0x07,
 0x07,0x07,0x07,0x07,0x07,0x07,0x07,0x07,0x07,0x04,0xE4,0xE7,0x07,0x07,0x06,0x04,0x05,
```

```
 0x05,0x05,0x07,0x07,0x06,0x04,0xE5,0xE5,0x64,0xE6,0x87,0x04,0x04,0x07,0x07,0x64,
 0x64,0x07,0x04,0x04,0x07,0x07,0x04,0x04,0x07,0x06,0x04,0x05,0x05,0x04,0x06,0x07,
 0x05,0x65,0x64,0xE4,0xE5,0x05,0x07,0x06,0x04,0x05,0x05,0x04,0x06,0x07,0x07,0x05,
 0x05,0x05,0x04,0x06,0x07,0x07,0x05,0x05,0x05,0x04,0x06,0x07,0x07,0x07,0x07,0x07,
 0x00,0x00,0x00,0x3C,0x3C,0x3C,0x3C,0xC0,0xC0,0xC0,0xC0,0x03,0x03,0x03,0x03,0x3C,
 0x3C,0x3C,0x3C,0xFC,0xFC,0xFC,0xFC,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,
 0x00,0x00,0x00,0xFF,0xFF,0x60,0xFE,0x9F,0x01,0x00,0xF8,0xFE,0x86,0x86,0xFE,0xF8,
 0x00,0xFE,0xFE,0x06,0x06,0xFE,0xF8,0x00,0x06,0xFF,0xFF,0x06,0x06,0x06,0x00,0x00,
 0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFF,0x00,0x01,0xFF,0xFE,0x00,0x06,0x06,0xFE,
 0xFE,0x00,0x00,0x00,0x78,0xFE,0x86,0x86,0x86,0x06,0x00,0xFE,0xFE,0x06,0x06,0xFE,
 0xF8,0x00,0x00,0xFF,0xFF,0x00,0x00,0x00,0x00,0x86,0x86,0x86,0xFE,0xF8,0x00,0xFE,
 0xFE,0x00,0x00,0xFE,0xFE,0x00,0x78,0xFE,0x86,0x86,0x86,0x06,0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x03,0x03,0x03,0x00,0x00,0x00,0x00,0x3C,
 0x3C,0x3C,0x3C,0x03,0x03,0x03,0x03,0x3F,0x3F,0x3F,0x3F,0x3F,0x3F,0x3F,0x3F,0x00,
 0x00,0x00,0x00,0x7F,0x7F,0x00,0x07,0x7F,0x78,0x00,0x1F,0x7F,0x61,0x61,0x61,0x01,
 0x00,0x7F,0x7F,0x00,0x00,0x7F,0x7F,0x00,0x00,0x1F,0x7F,0x60,0x60,0x60,0x00,0x00,
 0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0x7F,0x60,0x78,0x1F,0x07,0x00,0x60,0x60,0x7F,
 0x7F,0x60,0x60,0x00,0x60,0x61,0x61,0x61,0x7F,0x1E,0x00,0x7F,0x7F,0x06,0x06,0x07,
 0x01,0x60,0x60,0x7F,0x7F,0x60,0x60,0x00,0x1E,0x7F,0x61,0x61,0x7F,0x7F,0x00,0x01,
 0x67,0x66,0x66,0x7F,0x1F,0x00,0x60,0x61,0x61,0x61,0x7F,0x1E,0x00,0x00,0x00,0x00
};
```

7 Ordering Information

Displays may be ordered as either standalone modules or as part of a demonstration kit.

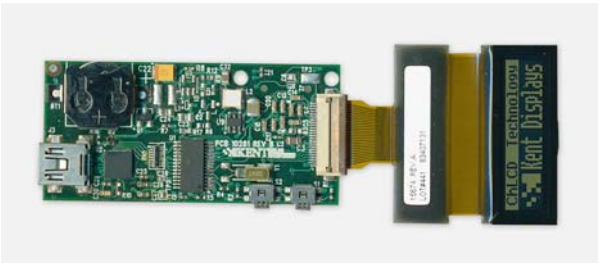
7.1 Display Module

The 128x32 ILV Display Module requires connection to an external PCB in order to display images.

128x32 ILV Display Module	
Part #	**Description**
015674221	Module, 128x32 ILV Yellow/Black

7.2 Demonstration Kit

The 128x32 ILV Demonstration Kit contains everything required to quickly evaluate the features and capabilities of the 128x32 ILV display module. The demonstration kit contains a display module, demonstration controller PCB, Mini USB cable, coin cell battery (CR1220), and software CD. The controller PCB may be used to cycle forward and reverse through preprogrammed images on the display by pressing the two buttons located on the edge of the PCB. Custom images may be downloaded from a PC using the included software and Mini USB cable.



Note: PCB size is not representative of an optimized circuit design.

Demonstration Kit	
Part #	**Description**
090035221	Demo. Kit, 128x32 ILV Yellow/Black

Contact Kent Displays at sales@kentdisplays.com for additional color options, custom configurations, pricing, and additional information.

Products and technologies of Kent Displays, Inc. are protected by the US Patents: 5,493,430, 5,570,216, 5,636,044, 5,644,330, 5,251,048, 5,384,067, 5,437,811, 5,453,863, 5,668,614, 5,691,796, 5,695,682, 5,748,277, 5,766,694, 5,847,798 and numerous other patent applications by Kent Display Systems, Inc., Kent Displays, Inc. and Kent State University pending in the U.S. and in foreign patent filings include: PCT, Canada, China, Europe, Israel, Japan, Korea, and Taiwan among others.