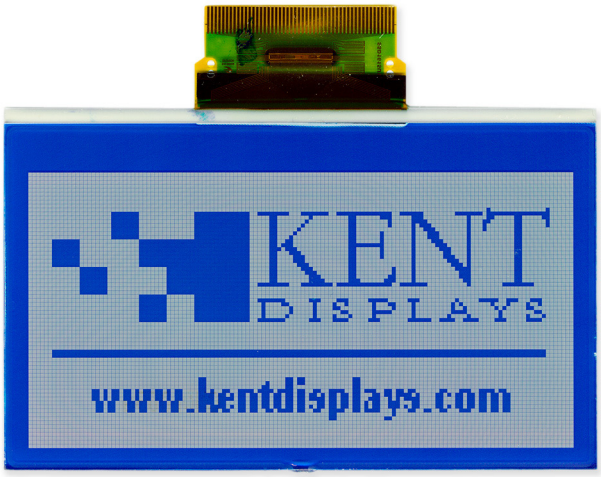
132x64 Cholesteric Display Module with LCD Controller



Product Description

This Graphic Display Module delivers maximum image information content flexibility in an extremely compact form factor. It is one of the most compact and cost efficient bi-stable graphic displays 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-Flex (COF) driver IC includes an LCD controller and a DC/DC charge pump. System integration requires only an external microcontroller for sensing temperature and sending commands to the LCD controller and external capacitors for use by the built-in charge pump.

Product Features

Display Module:

- 132 Columns x 64 Rows
- Approximate size: 3.1x1.9x0.06 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**

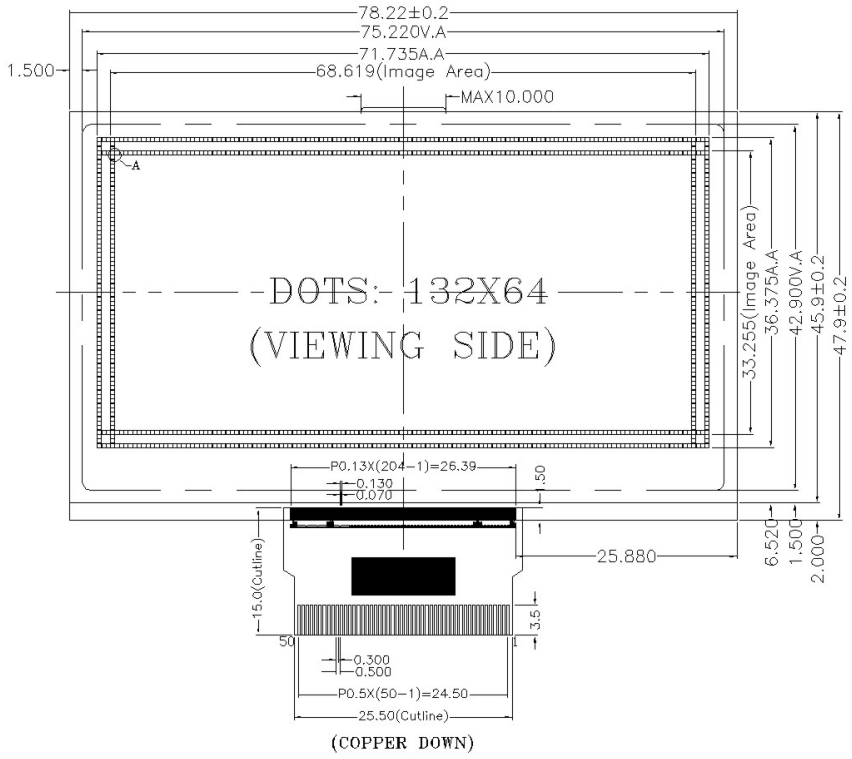
Kent Displays, Inc.
343 Portage Boulevard
Kent, OH 44240 USA

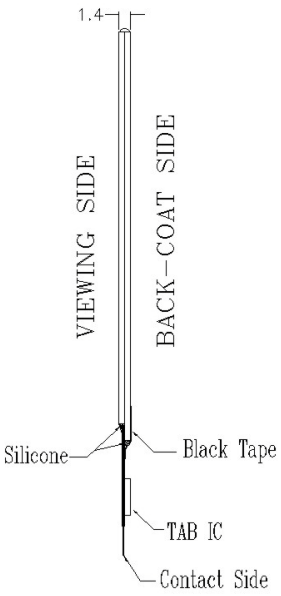
Telephone: 330.673.8784
Fax: 330.673.4408
Email: sales@kentdisplays.com
Website: www.kentdisplays.com

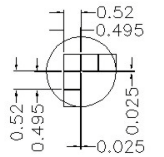
Specification Summary

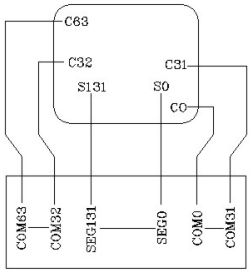
Display Module with COF Controller

General Specifications	
Parameter	**Description**
Display Type	Cholesteric Reflective LCD
Format	132 Columns x 64 Rows
Resolution	49 dots per inch (0.52mm pixel pitch, horizontal & vertical)
Image Area	2.70 in × 1.31 in (68.6mm × 33.3 mm)
Display Module Weight	0.466 oz (13.2 grams)
Operating Temperature Range	0°C to +50°C
Storage Temperature Range	-30°C to +80°C
Full Image Update Rate	1.4 sec. @ 25°C







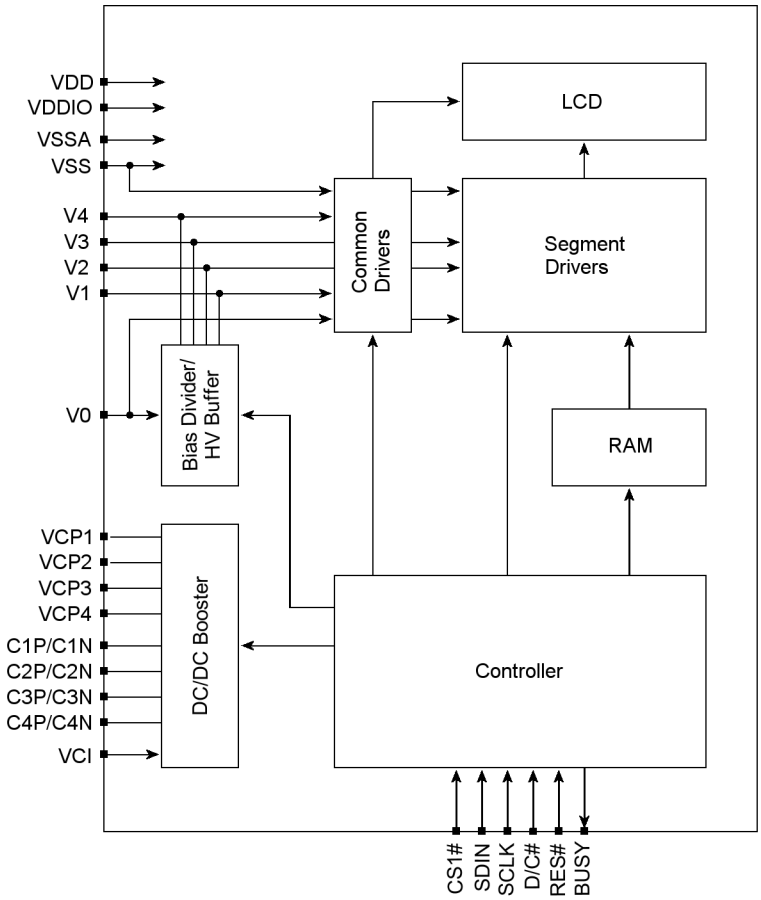


Contents

1 Overview

The 132x64 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.

2 Block Diagram



3 Electrical Interface

3.1 Interconnect

The display module interfaces to the host electronics through a 50-contact COF (chip on flex). Refer to Figure 4 in Section 5.5 for COF dimensions.

3.2 Pin Summary

Number	Name	Type[1]	Description[2]
1	NC	NC	No connection.
2	VFS	NC	No connection.
3	V4	Power	Panel driving voltage. No connection.
4	V3	Power	Panel driving voltage. No connection.
5	V2	Power	Panel driving voltage. No connection.
6	V1	Power	Panel driving voltage. No connection.
7	PS0	Input	Bus interface selection. Tie to VSS.
8	PS1	Input	Bus interface selection. Tie to VSS.
9	PS2	Input	Bus interface selection. Tie to VSS.
10	VDDIO	Supply	Interface logic supply. Tie to VDD.
11	CLS	Input	Internal clock enable. Tie to VDDIO.
12	D/C#	Input	Data/Command control pin. Set to VDD for data input and VSS for command input.
13	CS2	Input	Chip select input. Tie to VDDIO.
14	E(RD#)	Input	Tie to VSS.
15	R/W#(WR#)	Input	Tie to VSS.
16	CS1#	Input	Chip select input. Set to VSS to select the chip.
17	RES#	Input	Reset input. Set to VSS for ≥ 20 µs to initialize chip.
18	CL	NC	No connection.
19	D0(SCLK)	Input	Serial clock input.
20	D1(SDIN)	Input	Serial data input.
21	D2	NC	No connection.
22	D3	Input	Tie to VSS.
23	D4	Input	Tie to VSS.
24	D5	Input	Tie to VSS.
25	D6	Input	Tie to VSS.

[1] Type:
- Supply - provide power to the display module
- Input - logic input
- Output – logic output
- DC/DC – used by internal voltage converter to generate the LCD drive voltage
- Power – provide display drive voltages
- NC – no connection

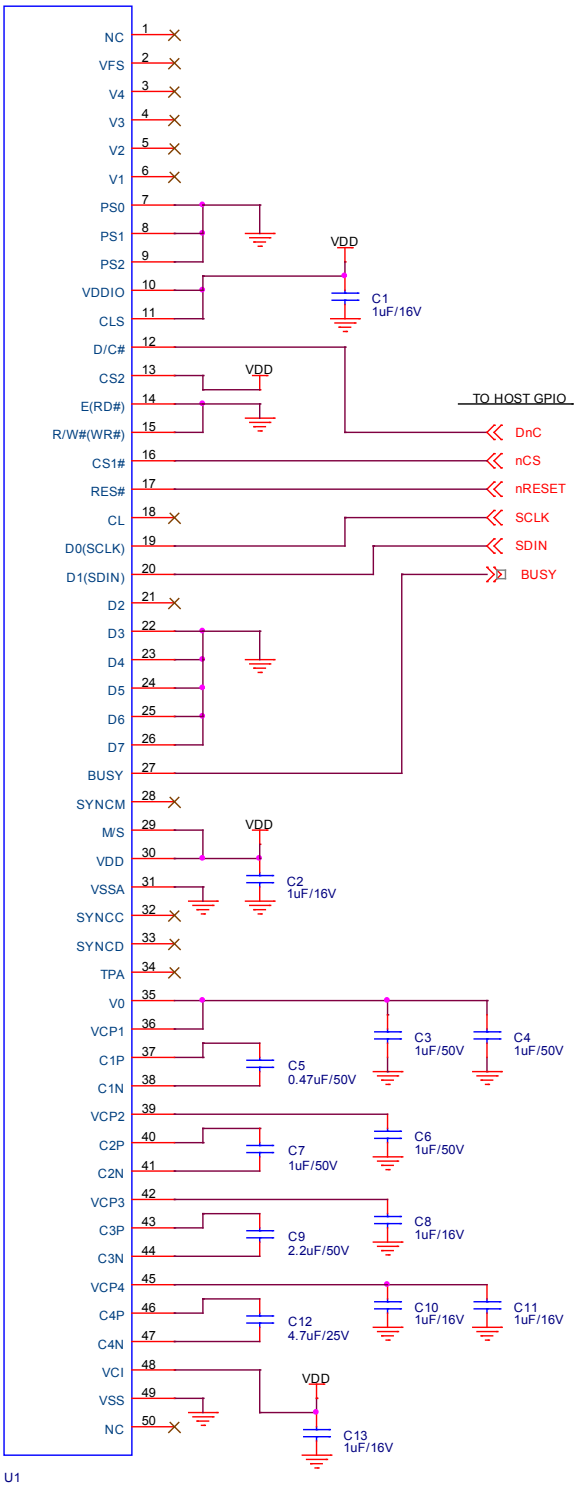
[2] Description:
- Contact sales at Kent Displays for a full driver datasheet that includes alternative MCU interface options.

Number	Name	Type[1]	Description[2]
26	D7	Input	Tie to VSS.
27	BUSY	Output	A logic high indicates driver is busy updating the display.
28	SYNCM	NC	No connection.
29	M/S	Input	Master/Slave select. Tie to VDDIO.
30	VDD	Supply	Power supply pin of the logic block.
31	VSSA	Supply	Tie to VSS.
32	SYNCC	NC	No connection.
33	SYNCD	NC	No connection.
34	TPA	NC	No connection.
35	V0	Power	Panel driving voltage. Tie to VCP1.
36	VCP1	DC/DC	Output voltage. Tie to V0.
37	C1P	DC/DC	Flying capacitor terminal.
38	C1N	DC/DC	Flying capacitor terminal.
39	VCP2	DC/DC	Intermediate output voltage.
40	C2P	DC/DC	Flying capacitor terminal.
41	C2N	DC/DC	Flying capacitor terminal.
42	VCP3	DC/DC	Intermediate output voltage.
43	C3P	DC/DC	Flying capacitor terminal.
44	C3N	DC/DC	Flying capacitor terminal.
45	VCP4	DC/DC	Intermediate output voltage.
46	C4P	DC/DC	Flying capacitor terminal.
47	C4N	DC/DC	Flying capacitor terminal.
48	VCI	Supply	Power supply for DC/DC converter and analog. Tie to VDD.
49	VSS	Supply	Ground.
50	NC	NC	No connection.

[1] Type:
- Supply - provide power to the display module
- Input - logic input
- Output – logic output
- DC/DC – used by internal voltage converter to generate the LCD drive voltage
- Power – provide display drive voltages
- NC – no connection

[2] Description:
- Contact sales at Kent Displays for a full driver datasheet that includes alternative MCU interface options.

3.3 Reference Schematic



*Note: Depending upon the source impedance of the supply, additional capacitance may be required to stabilize VDD when running the internal DC/DC converter.

Figure 1: Reference Schematic

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 commanding the display controller to perform an update.

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 CS1# line and provides the clock signal (SCLK) used to clock in data. Communication with the display begins with a high-to-low transition on CS1# and ends with a low-to-high transition on CS1#. New data (command or image data) for the display module are placed on SDIN 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 (D7) and ends with the least significant bit (D0). The D/C# signal is sampled with D0. The byte (D7 to D0) is treated as an instruction if D/C# is sampled low and as image data if D/C# is sampled high.

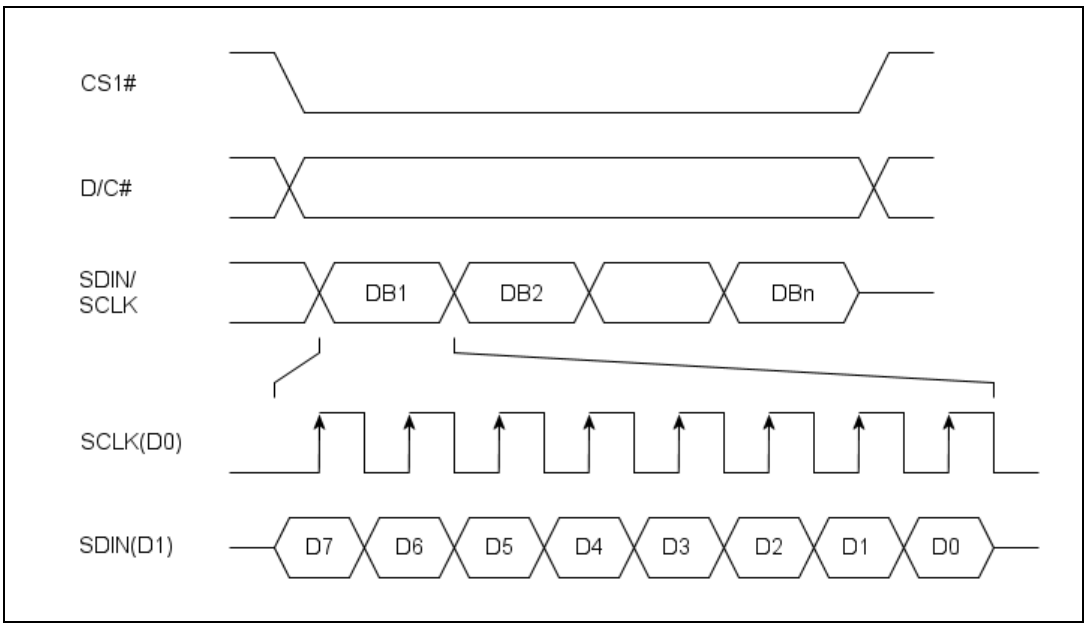


Figure 2: Basic Serial Interface Timing

4.3 Image Data

The graphic display is 132 columns by 64 rows. The 64 rows are organized into 8 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 individual pixels may be mapped to the 1056 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	…	131
0	0	BYTE 0	BYTE 1	BYTE 2	…	BYTE 131
	1					
	2					
	3					
	4					
	5					
	6					
	7					
1	0	BYTE 132	BYTE 133	BYTE 134	…	BYTE 263
	1					
	2					
	3					
	4					
	5					
	6					
	7					
⋮	⋮	⋮	⋮	⋮	⋮	⋮
7	0	BYTE 924	BYTE 925	BYTE 926	…	BYTE 1055
	1					
	2					
	3					
	4					
	5					
	6					
	7					

Figure 3: Image Data to Pixel Mapping

The sample code in Section 6 illustrates how 1056 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	132 Columns x 64 Rows
Resolution	49 dots per inch (0.52mm pixel pitch, horizontal & vertical)
Image Area	2.70 in × 1.31 in (68.6mm × 33.3 mm)
Display Module Weight	0.466 oz (13.2 grams)
Operating Temperature Range	0°C to +50°C
Storage Temperature Range	-30°C to +80°C
Full Image Update Rate	1.4 sec. @ 25°C

5.2 Absolute Maximum Ratings

Parameter	Symbol	Rating	Units
IC Logic Supply	V_{DD}	-0.3 to +3.6	V
Interface Logic Supply	V_{DDIO}	-0.3 to min(V_{DD}+0.5, +3.6)	V
DC/DC Supply	V_{CI}	-0.3 to +3.6	V
Input Voltage	V_{in}	-0.3 to (V_{DDIO} + 0.3)	V
Operating Temperature	T_{OPR}	0 to +50	ºC
Storage Temperature	T_{STR}	-30 to +80	ºC

5.3 Electrical

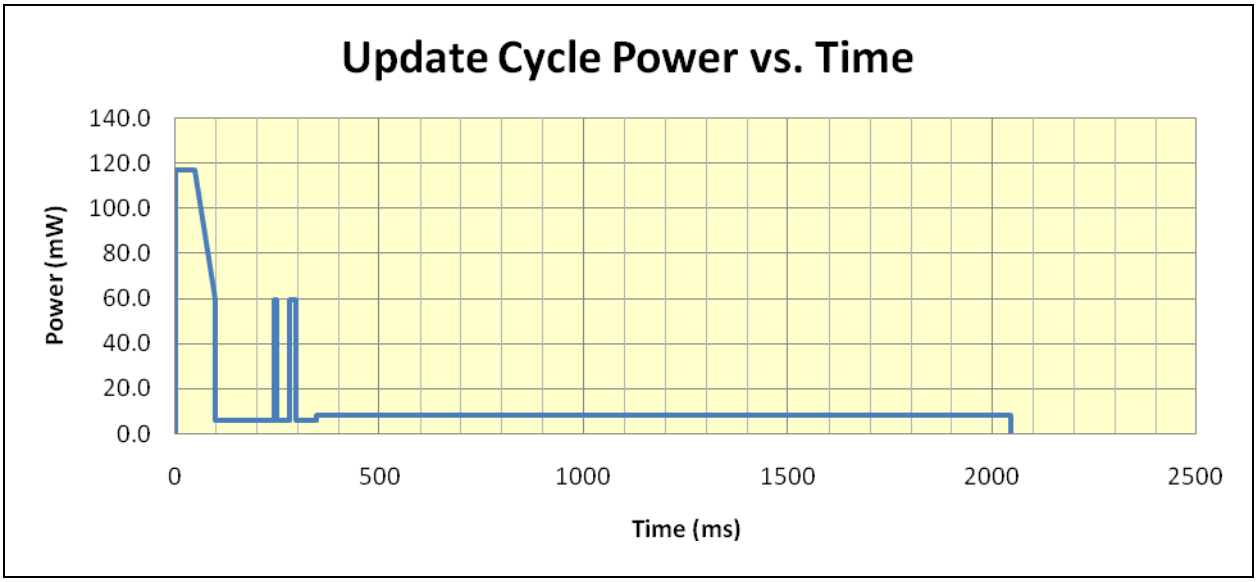
Conditions: T_A = 25 ºC

Parameter		Symbol	Min.	Typ.	Max.	Units
IC Logic Supply		V_{DD}	+2.4	+2.8	+3.5	V
Interface Logic Supply		V_{DDIO}	+1.6	-	V_{DD}	V
DC/DC Supply		V_{CI}	V_{DD}	-	+3.5	V
Input Voltage	High	V_{IH}	0.8 V_{DDIO}	-	V_{DDIO}	V
	Low	V_{IL}	V_{SS}	-	0.2 V_{DDIO}	V
Output Voltage	High	V_{OH}	0.9 V_{DDIO}	-	V_{DDIO}	V
	Low	V_{OL}	V_{SS}	-	0.1 V_{DDIO}	V
Sleep Mode Current[1]		I_{SLP}	-	+1	+15	µA

[1] I_{SLP} is the sum of V_{DD}, V_{DDIO}, and V_{CI} currents measured with V_{DD} = V_{DDIO} = V_{CI} = 2.8V.

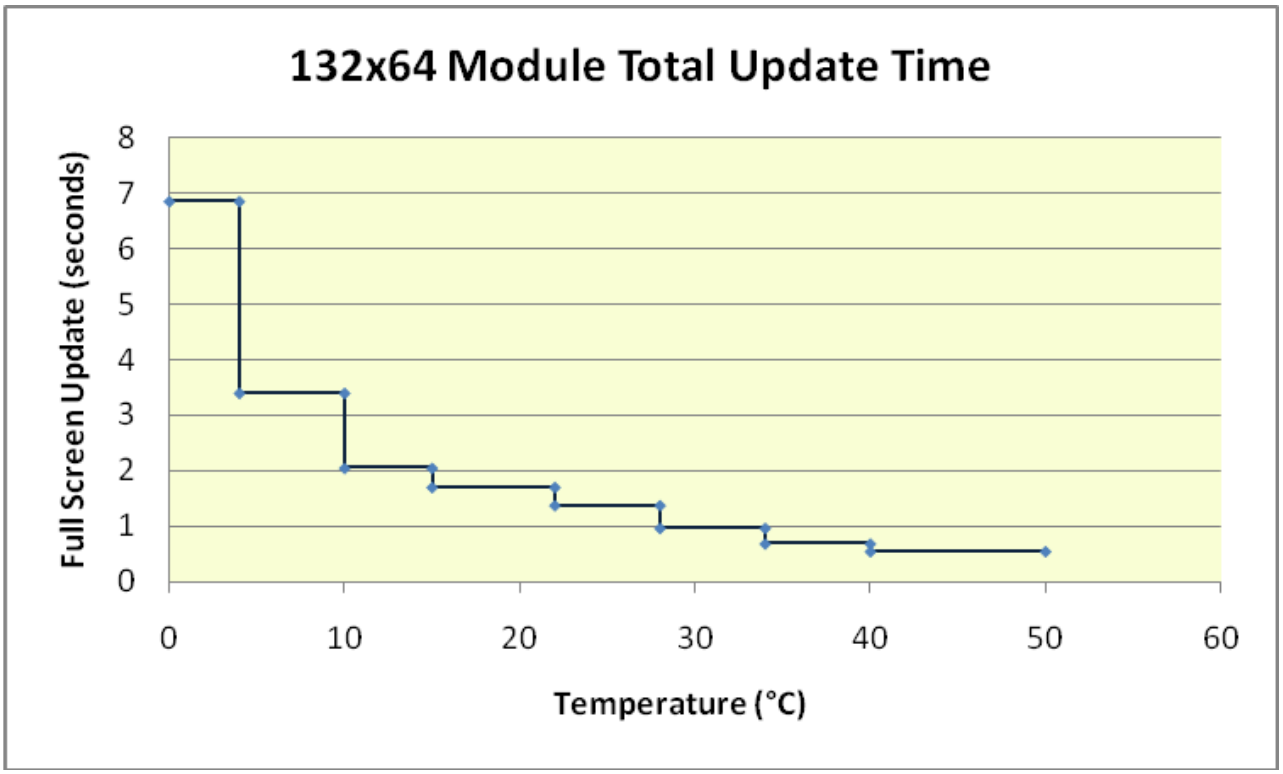
* Specifications are subject to change without prior notice.

5.3.1 Power Profile

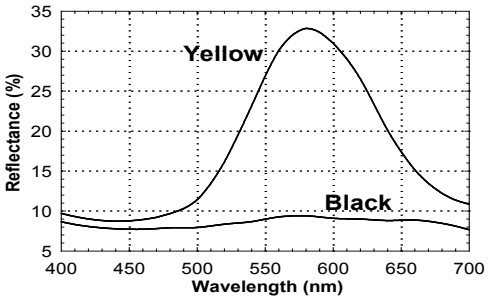


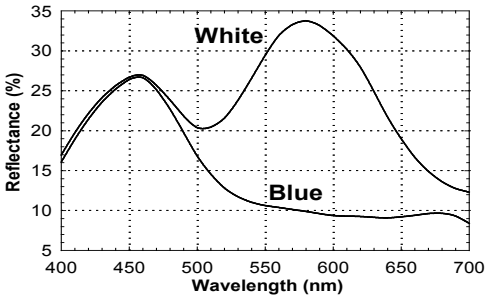
Note: Power measurements based on V_{DD} = 3.3V and T_A = 20ºC.

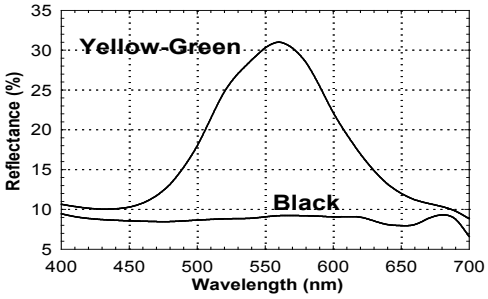
5.3.2 Update Cycle Temperature Performance

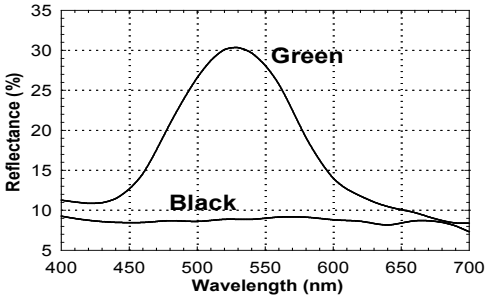


5.4 Optical



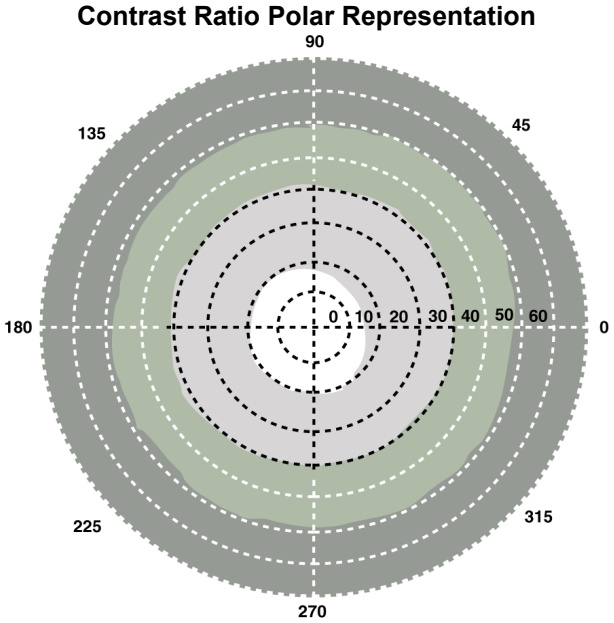






The above reflectance curves are from a single pixel. Actual reflectance will vary depending on display resolution, aperture ratio, and other factors.

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.



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.

5.5 Mechanical

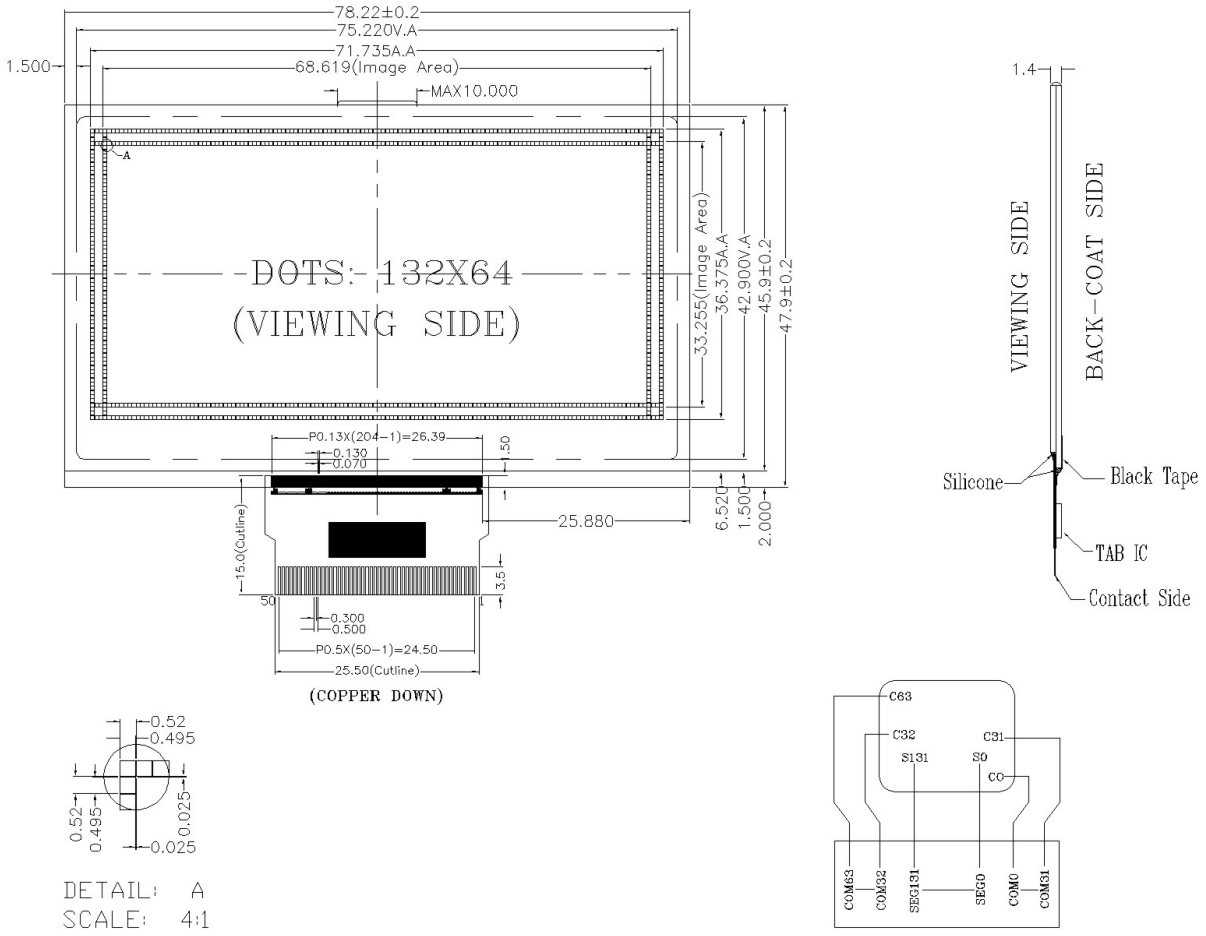


Figure 4: Module Dimensions (mm)

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.

5.6 Timing

5.6.1 Serial Interface

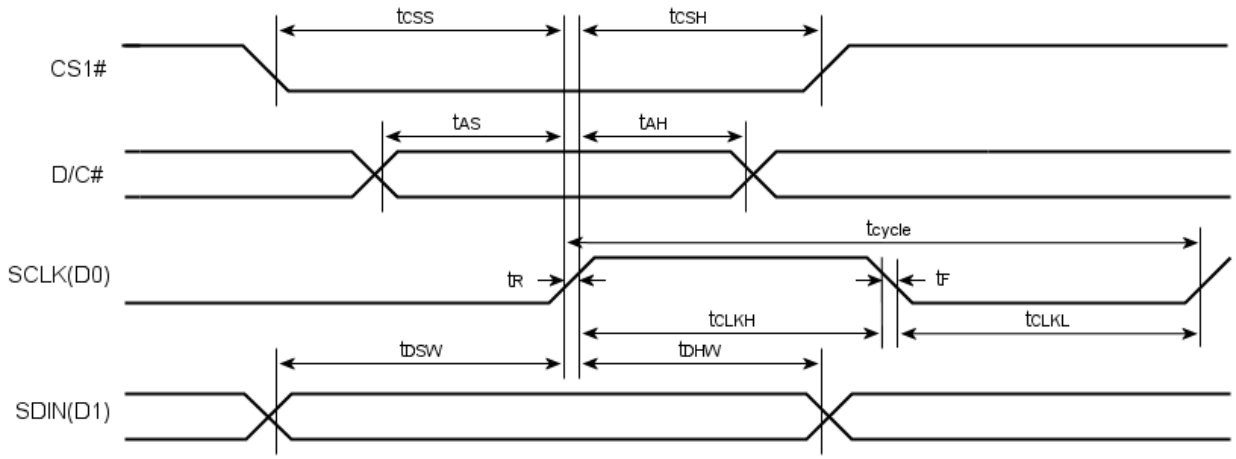


Figure 5: Serial Timing

Table 1: Serial Interface Timing Parameter Values

Item	Signal	Symbol	Min.	Typ.	Max.	Units
Clock Cycle Time	SCLK	tcycle	60	-	-	ns
Clock High Time		tCLKH	30	-	-	
Clock Low Time		tCLKL	30	-	-	
Rise Time		tR	-	-	10	
Fall Time		tF	-	-	10	
Address Setup Time	D/C#	tAS	10	-	-	ns
Address Hold Time		tAH	20	-	-	
Write Data Setup Time	SDIN	tDSW	30	-	-	ns
Write Data Hold Time		tDHW	30	-	-	
Chip Select Setup Time	CS1#	tCSS	30	-	-	ns
Chip Select Hold Time		tCSH	30	-	-	

Conditions: T_A = -35 to 85°C, V_{DD} = V_{CI} = V_{DDIO} = 2.4V to 3.5V

6 Sample Code

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

Note that the entry point for generating the display update is as follows:

```
Display(sampleImage);
```

6.1 User-Defined Code

6.1.1 Type Definitions

The source code uses four data types: int8, int16, uint8, and uint16. These types are for signed and unsigned integers of the indicated number of bits. An example definition (this varies by compiler) is as follows:

```
typedef signed char int8;         // signed 8-bit data type
typedef signed int  int16;        // signed 16-bit data type
typedef unsigned char uint8;      // unsigned 8-bit data type
typedef unsigned int  uint16;     // unsigned 16-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(uint16 ticks);
```

The sample code requires a parameter for the delay function be defined to implement a delay of specific duration. The definition is as follows, where 'x' must be replaced with the appropriate number of timer ticks:

```
#define CHARGE_TICKS      x  // delay timer ticks in 200 milliseconds
```

6.1.3 BUSY Signal Monitoring

A function to monitor the BUSY signal from the driver is declared as follows:

```
void WaitNonBusy(void);
```

The implementation of this function must stall program execution for as long as the BUSY signal from the display controller remains high.

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 function, PutCharSPI(), must be implemented for sending commands and data. A reference implementation has been provided using general GPIO for the case in which no additional delays are required in order to satisfy the timing constraints of Figure 5.

6.1.5 Interface Signals

The following functions must be implemented in order to set the interface signals to the designated values:

```
void SelectDisplay(void);      // nCS = 'L'
void DeselectDisplay(void);    // nCS = 'H'
void Set_DnC(void);            // DnC = 'H'
void Clear_DnC(void);          // DnC = 'L'
void Set_SDIN(void);           // SDIN = 'H'
void Clear_SDIN(void);         // SDIN = 'L'
void Set_SCLK(void);           // SCLK = 'H'
void Clear_SCLK(void);         // SCLK = 'L'
```

6.1.6 Display Reset

The display controller must be initialized after power is applied by pulsing nRESET to VSS for ≥ 20 µs. The sample code does not include this reset.

6.1.7 Temperature Sensing

The drive parameters of the display are functions of temperature. A function to measure the temperature is declared as follows:

```
int16 GetTemperature(void);
```

The implementation of this function must invoke a hardware resource to measure the temperature in degrees Celcius and return the temperature value as a 16-bit signed integer.

6.2 Standard Code

The following code should require only minimal changes, limited mainly to defines.h, 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

```
// Bitmasks for selecting command byte or data byte.
#define CMD_MASK          0x00
#define DATA_MASK         0x01

// Driver command definitions.
#define SET_COL_ADD_LSN   0x00      // Set Lower Column Address
#define SET_COL_ADD_MSN   0x10      // Set Higher Column Address
#define SET_PAGE_ADD      0xB0      // Set Page Address

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

// User-defined data types
// typedef signed char int8;
// typedef signed int  int16;
// typedef unsigned char uint8;
// typedef unsigned int  uint16;
```

```
// User-defined delay durations
// #define CHARGE_TICKS      x  // delay timer ticks in 200 milliseconds

// User-defined functions.
// extern void delay(uint16);
// extern void WaitNonBusy(void);
// extern void SelectDisplay(void);
// extern void DeselectDisplay(void);
// extern void Set_DnC(void);
// extern void Clear_DnC(void);
// extern void Set_SDIN(void);
// extern void Clear_SDIN(void);
// extern void Set_SCLK(void);
// extern void Clear_SCLK(void);
// extern int16 GetTemperature(void);
////////////////////////////////////////////////////////////////////////////

// Standard functions
extern void Display(const uint8 *);
extern void PutCharSPI(uint8, uint8);
extern void DriveImage(uint8, uint8, uint8);
extern void LoadData(uint16, uint16, const uint8 *);
```

6.2.2 Display.c

```
#include "defines.h"

////////////////////////////////////////////////////////////////////////////
// Function: Display
// Purpose:  Display a fullscreen image
// Inputs:   pImageData - pointer to image data array
// Outputs:  None.
////////////////////////////////////////////////////////////////////////////
void Display(const uint8 *pImageData)
{
  uint16 address = 0x0000;  // Top of display RAM.
  uint16 numBytes;          // Must match bitmap data array size.
  uint8  numRows;           // Must match bitmap height.
  uint8  firstImageRow;     // Screen location for bitmap (first row).
  uint8  firstDataRow;      // Location of data for first update row.

  // Load bitmap to display controller.
  numBytes = 1056;          // = 132*64/8
  LoadData(address, numBytes, pImageData);

  // Update display.
  numRows = 64;             // 64 rows for full screen
  firstImageRow = 0;        // Start update at top of display.
  firstDataRow = 0;         // Get data from top of display RAM.
  DriveImage(firstDataRow, firstImageRow, numRows);
}
```

6.2.3 PutCharSPI.c

```
#include "defines.h"

////////////////////////////////////////////////////////////////////////////
// Function:  PutCharSPI
// Purpose:   Sends a byte to the SPI bus.
// Inputs:    data - byte to send
//            Data_nCmd - command (0) or data (else) attribute of byte to send
// Outputs:   None.
////////////////////////////////////////////////////////////////////////////
void PutCharSPI(uint8 data, uint8 Data_nCmd)
{
  uint8 mask;

  // Assert chip select to display.
  SelectDisplay();

  // Signal byte type to driver.
  if (Data_nCmd)
    Set_DnC();
  else
    Clear_DnC();

  // Bit bang data out MSB first.  Data changes on falling edge and
  // is latched on rising.  Clock is inactive low.
  for (mask = 0x80; mask != 0; mask = mask >> 1)
  {
    if (data & mask)
      Set_SDIN();
    else
      Clear_SDIN();

    Set_SCLK();
    Clear_SCLK();
  }

  // De-assert chip select to display.
  DeselectDisplay();
}
```

6.2.4 LoadData.c

```
#include "defines.h"

////////////////////////////////////////////////////////////////////////////
// Function:  LoadData
// Purpose:   Load a portion of display RAM with image data.
// Inputs:    Index - starting position within image for data (0 - 1055)
//            NumBytes - number of bytes of image data to load
//            pData - pointer to the image data to load
// Outputs:   None.
// Notes:     None.
////////////////////////////////////////////////////////////////////////////
void LoadData(uint16 Index, uint16 NumBytes, const uint8 *pData)
{
```

```
  uint8 colAddr, page;
  uint16 i;

  // Loop through the data to write.
  for (i = Index; i < (Index + NumBytes); i++)
  {
    colAddr = i % 132;

    // Set page and column address as needed.
    if ((colAddr == 0) || (i == Index) )
    {
      page = i/132;
      PutCharSPI( (SET_COL_ADD_MSN | (colAddr / 16)) , CMD_MASK );
      PutCharSPI( (SET_COL_ADD_LSN | (colAddr % 16)) , CMD_MASK );
      PutCharSPI( (SET_PAGE_ADD | page) , CMD_MASK );
    }

    // Write a data byte.
    PutCharSPI( *pData++, DATA_MASK );
  }
}
```

6.2.5 DriveImage.c

```
#include "defines.h"

// Temperature compensation data element definition.
typedef struct
{
  int8 temp;
  uint8 clearPW;
  uint8 drivePW;
  uint8 voltage;
} TC_Table_Element;

// Temperature compensation data table.
const TC_Table_Element tcTable[] =
    {
      { 40 /* 40C */, 0x11 /*  30 ms */, 0x08 /*   8 ms */, 0x44 /* 25.0V */},
      { 34 /* 34C */, 0x11 /*  40 ms */, 0x09 /*  10 ms */, 0x44 /* 25.0V */},
      { 28 /* 28C */, 0x13 /*  60 ms */, 0x0b /*  14 ms */, 0x44 /* 25.0V */},
      { 22 /* 22C */, 0x14 /*  80 ms */, 0x0d /*  20 ms */, 0x44 /* 25.0V */},
      { 16 /* 16C */, 0x14 /*  80 ms */, 0x0e /*  25 ms */, 0x44 /* 25.0V */},
      { 10 /* 10C */, 0x15 /* 100 ms */, 0x0f /*  30 ms */, 0x44 /* 25.0V */},
      {  4 /*  4C */, 0x16 /* 150 ms */, 0x12 /*  50 ms */, 0x44 /* 25.0V */},
      { -5 /* -5C */, 0x19 /* 350 ms */, 0x15 /* 100 ms */, 0x40 /* 24.0V */},
    };

#define TEMP_PTS   8    // Number of entries in tcTable.
#define TEMP_MAX  50    // Maximum display update temperature.

////////////////////////////////////////////////////////////////////////////
// Function: DriveImage
// Purpose:  Updates display with image data in driver RAM using
//           temperature compensation table drive parameters.
// Inputs:   firstDataRow - location of data for first update row
//           firstImageRow - first physical display row to update
```

```
//           numRows - number of rows to update
// Outputs:  none.
////////////////////////////////////////////////////////////////////////////
void DriveImage(uint8 firstDataRow, uint8 firstImageRow, uint8 numRows)
{
  uint8 offset, startLineCmd, i;

  uint8 clearPW, drivePW, voltage;
  int16 temperature;

  // Read the temperature in degrees C.
  temperature = GetTemperature();

  // Update display if the temperature is within limits.
  if ((temperature >= tcTable[TEMP_PTS-1].temp) && (temperature <= TEMP_MAX))
  {
    // Determine drive parameters.
    for (i = 0; i < TEMP_PTS; i++)
    {
      if ( temperature >= tcTable[i].temp ) // Within range of table entry?
      {
        // Read parameters from table.
        clearPW = tcTable[i].clearPW;
        drivePW = tcTable[i].drivePW;
        voltage = tcTable[i].voltage;
        break;
      }
    }

    // Compute image location parameters.
    offset = 0x40 - firstImageRow;
    startLineCmd = 0x40 + firstDataRow;

    // Initialization code.
    PutCharSPI( 0xA3, CMD_MASK );  // Enable bandgap and other analog control.
    PutCharSPI( 0x18, CMD_MASK );
    PutCharSPI( 0xF6, CMD_MASK );  // Enable oscillator.
    PutCharSPI( 0x40, CMD_MASK );
    PutCharSPI( 0xAE, CMD_MASK );  // Set auto charge pump threshold value.
    PutCharSPI( 0x00, CMD_MASK );

    // Set drive parameters.
    PutCharSPI(    0x80, CMD_MASK );
    PutCharSPI(    0x00, CMD_MASK );
    PutCharSPI( clearPW, CMD_MASK );  // VA Clear Duration
    PutCharSPI( drivePW, CMD_MASK );  // VA Idle  Duration
    PutCharSPI(    0x00, CMD_MASK );  // AA Clear Duration
    PutCharSPI(    0x00, CMD_MASK );  // AA Idle  Duration
    PutCharSPI( drivePW, CMD_MASK );  // Drive    Duration
    PutCharSPI( voltage, CMD_MASK );  // Clear  Voltage
    PutCharSPI( voltage, CMD_MASK );  // Drive  Voltage

    // Set dummy waveform for supply initialization.
    PutCharSPI( 0x93, CMD_MASK );
    PutCharSPI( 0x00, CMD_MASK );  // Skip VA Clear.
    PutCharSPI( 0x94, CMD_MASK );
    PutCharSPI( 0x00, CMD_MASK );  // Skip VA Idle.
    PutCharSPI( 0x95, CMD_MASK );
    PutCharSPI( 0x00, CMD_MASK );  // Skip AA Clear.
```

```
    PutCharSPI( 0x96, CMD_MASK );
    PutCharSPI( 0x00, CMD_MASK );  // Skip AA Idle.
    PutCharSPI( 0x97, CMD_MASK );
    PutCharSPI( 0x00, CMD_MASK );  // Skip Drive.

    // Dummy update results in future supply initialization to clear voltage.
    PutCharSPI( 0x31, CMD_MASK );  // Dummy update.
    WaitNonBusy();

    // More init. code.
    PutCharSPI( 0xA3, CMD_MASK );  // Enable other analog control.
    PutCharSPI( 0x1A, CMD_MASK );
    PutCharSPI( 0xE9, CMD_MASK );  // Enable bias driven.
    PutCharSPI( 0x84, CMD_MASK );
    PutCharSPI( 0x2F, CMD_MASK );  // Enable booster and high voltage buffer.

    // Delay for HVC to initialize.
    delay(CHARGE_TICKS);

    // Set up update (VA planar erase and drive scan).
    PutCharSPI( 0x93, CMD_MASK );
    PutCharSPI( 0x01, CMD_MASK );  // VA Clear Repeats
    PutCharSPI( 0x94, CMD_MASK );
    PutCharSPI( 0x01, CMD_MASK );  // VA Idle  Repeats
    PutCharSPI( 0x95, CMD_MASK );
    PutCharSPI( 0x00, CMD_MASK );  // AA Clear Repeats
    PutCharSPI( 0x96, CMD_MASK );
    PutCharSPI( 0x00, CMD_MASK );  // AA Idle  Repeats
    PutCharSPI( 0x97, CMD_MASK );
    PutCharSPI( 0x01, CMD_MASK );  // Drive    Repeats

    PutCharSPI( 0x32, CMD_MASK );
    PutCharSPI( 0x32, CMD_MASK );  // drive scheme (ON data)

    // Configure update area.
    PutCharSPI( 0xa8, CMD_MASK);
    PutCharSPI( numRows, CMD_MASK);       // set MUX ratio
    PutCharSPI( 0xd3, CMD_MASK);
    PutCharSPI( offset, CMD_MASK);        // set physical location on screen
    PutCharSPI( startLineCmd, CMD_MASK);  // controls what data is used

    // Drive image content.
    PutCharSPI( 0x31, CMD_MASK);
    WaitNonBusy();

    // Put display to sleep.
    PutCharSPI( 0x2A, CMD_MASK);
    PutCharSPI( 0xE9, CMD_MASK);
    PutCharSPI( 0x04, CMD_MASK);
    PutCharSPI( 0xF6, CMD_MASK);
    PutCharSPI( 0x00, CMD_MASK);
    PutCharSPI( 0xA3, CMD_MASK);
    PutCharSPI( 0x00, CMD_MASK);
    PutCharSPI( 0xAB, CMD_MASK);
    PutCharSPI( 0x00, CMD_MASK);
  }
}
```

6.2.6 SampleImage.c

```
#include "defines.h"

// A pointer to this array may be passed to the Display() function for initial
// debugging.  The array encodes the same image pictured on the displays in the
datasheet.  The data is otherwise not required.
const uint8 sampleImage[1056] = {
   0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
   0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
   0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0xFF, 0xFF, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
   0x7F, 0x7F, 0x7F, 0xFF, 0xFF, 0xFF, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
   0x7F, 0xFF, 0xFF, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
   0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0xFF, 0xFF, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
   0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x7F, 0x7F,
   0x7F, 0x7F, 0x7F, 0x7F, 0xFF, 0x3F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
   0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x3F, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0,
   0xC0, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
   0x3F, 0x3F, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00,
   0xFF, 0x7F, 0x3F, 0x1F, 0xEF, 0xF7, 0xFB, 0xFD, 0xFE, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x7F, 0x7F, 0x7F, 0x7F,
   0x7F, 0x7F, 0x0E, 0xFE, 0xF8, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0xFC,
   0xF8, 0xF1, 0xC3, 0x87, 0x0F, 0x3F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0x00, 0xFF, 0xFF, 0xFF, 0xF1, 0xFC, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xF8, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0x1F, 0x1F,
   0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0x00, 0x00, 0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00,
   0xFE, 0xFF, 0xFE, 0xFC, 0xF0, 0xE1, 0xC3, 0x8F, 0x1F, 0x3F, 0x7F, 0xFF,
   0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xF8, 0xFF, 0x7F, 0x1F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0xFF,
   0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xF8, 0xF0, 0xE3, 0xC7, 0x0F, 0x1F,
   0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0xF0,
   0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0xF0, 0xF0, 0xF0, 0xF0,
   0xF0, 0xF0, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
   0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xDD, 0x1D, 0xDD, 0xDC, 0xDC,
   0xDD, 0x9D, 0xBD, 0x7F, 0xFF, 0xFF, 0xFD, 0xFD, 0xDC, 0x1C, 0xDC, 0xFC,
   0xFD, 0xFD, 0x3F, 0x5D, 0xDD, 0x9D, 0xFC, 0xFC, 0xFC, 0xFD, 0xFD, 0xDD,
   0x1D, 0xDD, 0xDD, 0xDC, 0xBC, 0x3E, 0xFF, 0xFD, 0xFD, 0xDD, 0x1C, 0xDD,
   0xFD, 0xFD, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0xDF, 0xBF, 0x7E,
   0xFC, 0xFF, 0xFF, 0xDF, 0xDF, 0xDF, 0x9F, 0x5F, 0xFF, 0x7D, 0x9D, 0xDD,
   0xDC, 0xFC, 0xFD, 0x3D, 0x5D, 0xDF, 0x9F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF, 0xFF, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8,
   0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xFF, 0xFF, 0xFB, 0xF8, 0xFB, 0xFB, 0xFB,
```

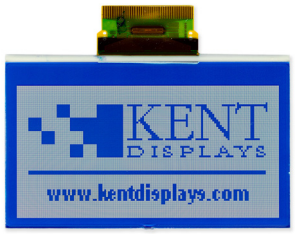
```
    0xFB, 0xFB, 0xFD, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFB, 0xF8, 0xFB, 0xFF,
    0xFF, 0xFF, 0xF9, 0xFB, 0xFA, 0xFA, 0xFD, 0xFF, 0xFF, 0xFF, 0xFF, 0xFB,
    0xF8, 0xFA, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFB, 0xF8, 0xFB,
    0xFB, 0xFB, 0xFB, 0xFF, 0xFF, 0xFB, 0xFB, 0xF9, 0xFA, 0xFE, 0xFE, 0xFA,
    0xF9, 0xFB, 0xFB, 0xFF, 0xFF, 0xFF, 0xFF, 0xFB, 0xF8, 0xFB, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xF9, 0xFB, 0xFA, 0xFA, 0xFD, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3,
    0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xF3, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0x8F, 0x0F, 0x7F, 0x8F, 0x7F, 0x0F, 0x8F, 0xFF, 0x8F, 0x0F, 0x7F,
    0x8F, 0x7F, 0x0F, 0x8F, 0xFF, 0x8F, 0x0F, 0x7F, 0x8F, 0x7F, 0x0F, 0x8F,
    0xFF, 0xFF, 0xFF, 0xFF, 0x01, 0x01, 0x1F, 0x0F, 0xFF, 0x1F, 0x0F, 0xAF,
    0x9F, 0xFF, 0x0F, 0x0F, 0xEF, 0x0F, 0x1F, 0xEF, 0x07, 0x03, 0xEF, 0x1F,
    0x0F, 0xEF, 0x01, 0x01, 0xFF, 0x09, 0x09, 0xFF, 0x9F, 0x8F, 0x2F, 0x1F,
    0xFF, 0x0F, 0x0F, 0xEF, 0x0F, 0x1F, 0xFF, 0x01, 0x01, 0xFF, 0x6F, 0xAF,
    0x0F, 0x1F, 0xFF, 0xCF, 0x0F, 0x7F, 0x0F, 0xCF, 0x9F, 0x8F, 0x2F, 0x1F,
    0xFF, 0xFF, 0xFF, 0xFF, 0x1F, 0x0F, 0xEF, 0xDF, 0xFF, 0x1F, 0x0F, 0xEF,
    0x0F, 0x1F, 0xFF, 0x0F, 0x0F, 0xEF, 0x0F, 0x1F, 0xEF, 0x0F, 0x1F, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFC, 0xFC, 0xFF, 0xFC, 0xFC, 0xFF, 0xFF, 0xFF, 0xFC, 0xFC,
    0xFF, 0xFC, 0xFC, 0xFF, 0xFF, 0xFF, 0xFC, 0xFC, 0xFF, 0xFC, 0xFC, 0xFF,
    0xFF, 0xFD, 0xFD, 0xFF, 0xFC, 0xFC, 0xFF, 0xFC, 0xFC, 0xFE, 0xFC, 0xFD,
    0xFE, 0xFF, 0xFC, 0xFC, 0xFF, 0xFC, 0xFC, 0xFF, 0xFE, 0xFC, 0xFD, 0xFE,
    0xFC, 0xFD, 0xFC, 0xFC, 0xFF, 0xFC, 0xFC, 0xFF, 0xFE, 0xFD, 0xFC, 0xFE,
    0xFF, 0xF0, 0xF0, 0xFD, 0xFC, 0xFE, 0xFF, 0xFC, 0xFC, 0xFF, 0xFC, 0xFD,
    0xFC, 0xFC, 0xFF, 0xF7, 0xF6, 0xF8, 0xFE, 0xFF, 0xFE, 0xFD, 0xFC, 0xFE,
    0xFF, 0xFD, 0xFD, 0xFF, 0xFE, 0xFC, 0xFD, 0xFE, 0xFF, 0xFE, 0xFC, 0xFD,
    0xFC, 0xFE, 0xFF, 0xFC, 0xFC, 0xFF, 0xFC, 0xFC, 0xFF, 0xFC, 0xFC, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
};
```

7 Ordering Information

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

7.1 Display Module

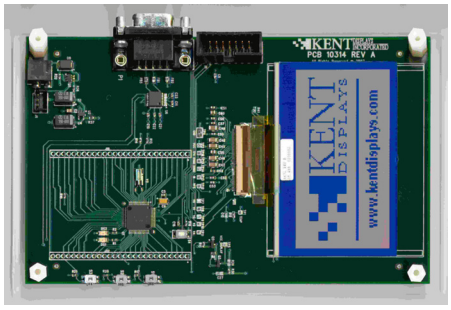
The 132x64 Display Module requires connection to an external PCB in order to display images.



Display Module	
Part #	**Description**
01568301208	Module, 132x64x3.0 White/Blue

7.2 Development Kit

The 132x64 Development Kit contains everything required to quickly evaluate the features and capabilities of the 132x64 display module. The development kit contains a display module with embedded controller, development module, AAA cell holder assembly, batteries, A/C power supply, PC to development module serial communication cable, JTAG programming device, software CD with source code samples, TI Code Composer Essentials development environment, user interface software, and related documentation.



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

Development Kit	
Part #	**Description**
09003801208	Development Kit, 132x64x3.0 White/Blue

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

REV	REVISIONS	DATE	BY
A	Initial Release.	2/18/2008	DWM
B	Modified image update rate and included update time vs. temperature chart. Added assembly drawing to specification summary page.	5/9/2008	AKB
C	Removed references to built-in temperature sensing/compensation that are no longer supported by the driver. Updated sample code to include host system sensing of temperature.	4/9/2009	DWM