Crystalfontz America, Incorporated

EXTERNAL USB INTELLIGENT LCD MODULE SPECIFICATIONS

Crystalfontz Model Number	**XES635BK-TFE-KU**
Hardware Version	**Revision v3.1. June 2010**
Firmware Version	**Revision v1.6, April 2010**
Data Sheet Version	**Revision v2.0, October 2010**
Product Pages	www.crystalfontz.com/product/XES635BKTFEKU.html

Crystalfontz America, Incorporated

12412 East Saltese Avenue
Spokane Valley, WA 99216-0357
Phone: 888-206-9720
Fax: 509-892-1203
Email: techinfo@crystalfontz.com
URL: www.crystalfontz.com

REVISION HISTORY

HARDWARE XES635BK-TFE-KU LCD MODULE XES635BK-TFE-KU v3.1 is a CFA-635-TFE-KU v1.1 module enclosed in a black steel case with a permanently attached USB "A" cable.	
2010/06/24	Current hardware version: **v3.1** Changes since last revision (1.0): By MET Laboratories: ● Product complies with UL60950 CSA C22.2 No. 60950. Product safety certification for Canada, United States, and Europe. ● Product complies to CE standards (European safety, health, and environmental standards). ● Product passed EU ESD immunity (Electrostatic Discharge) and radiated immunity requirements with no anomalies. ● Title 47 of the CFR, Part 15, Subpart B for a Class B Digital Device ● ICES-003 Issue 4, February 7, 2004, Class B. ● Product materials are in compliance with the regulations related to the EU Directive 2006/121/EC for Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). Also, by this date: ● Cable meets requirements for demonstrating compliance with California Proposition 65. ● Factories have ISO certification. ● Improved backlight. Display is lit more evenly, contrast can be adjusted to higher, and lifetime is longer. (As of 2008/07/01). ● Improved ESD (Electrostatic Discharge) protection for USB cable. Also improved cable strain relief. ● We transitioned to an improved keypad from “10.5” millimeters to “12.00” millimeters high.
2008/03/01	Hardware version: v1.0 Start Public Version Tracking.

FIRMWARE	
2010/05/01	Current firmware version: **v1.6** See PCN 10280. Changes since last revision (v1.4): ● Command 1 (0x01): Get Hardware & Firmware Version (Pg. 17) returns: "`CFA635:h1.1,v1.6`". ● Improved reset function to make module less sensitive to supply variations. ● Fixed range checking for command 0 (0x00): Ping Command (Pg. 17).
2005/07/01	Firmware version: v1.4 Command 1: Get Hardware & Firmware Version returns: “CFA635:h1.0,v1.4”

DATA SHEET	
2010/10/10	Current data sheet version: **v2.0** Changes since last revision (1.0): ● Wherever listed, deleted dash (“-”) from module part numbers “CFA-631”, “CFA-633” and “CFA-635” to match how they now appear on our website. ● In Physical Characteristics (Pg. 9) and Module Outline Drawing (Pg. 10), - Changed module overall height from “20.55” millimeters to “22.05” millimeters. Increase is due to improved keypad height from “10.5” millimeters to “12” millimeters. - Added specifications for 5x7 Character Size and 6x8 Matrix. ● In Absolute Maximum Ratings (Pg. 13), added important note about these specifications. ● Slightly modified specifications in Typical Current Consumption (Pg. 31) to reflect backlight improvement made 2008/07/01. ● In command 13 (0x0D): Set LCD Contrast (Pg. 20)), corrected contrast setting from "`(0-255 valid)`" and "`126-255 = very dark`" to "`(0-254 valid)`" and "`126-254 = very dark`". ● In command 33 (0x21): Set Baud Rate (Pg. 24), corrected from "`data_length = 1`" to "`data_length = 0`". ● Please read the revised Product Longevity (EOL / Replacement Policy) (Pg. 27). ● Please read the revised CARE AND HANDLING INFORMATION (Pg. 28) section. ● In APPENDIX B: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC) (Pg. 33): - Added Sample Code (Pg. 33) section with hypertext links to our free downloadable code. - Added typedefs for "`ubyte`" and "`word`" in sample code for Algorithm 1: “C” Table Implementation (Pg. 33) and Algorithm 2: “C” Bit Shift Implementation (Pg. 34). - Added section Algorithm 2B: “C” Improved Bit Shift Implementation (Pg. 36). This is a simplified algorithm that implements the CRC. - In Algorithm 6: “Perl” Table Implementation (Pg. 41), corrected code from "`my $packet = $type . $length . $data ;`" to "`my $packet = chr(hex $type) .chr(hex $length) .chr(hex $data);`". - Added Algorithm 7: For PIC18F8722 or PIC18F2685 (Pg. 42). ● Wherever needed, slightly modified text and illustrations to improve readability.
2008/03/01	Data Sheet version: v1.0: New Data Sheet.

The Fine Print

Certain applications using Crystalfontz America, Inc. products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications"). CRYSTALFONTZ AMERICA, INC. PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. Inclusion of Crystalfontz America, Inc. products in such applications is understood to be fully at the risk of the customer. In order to minimize risks associated with customer applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazard. Please contact us if you have any questions concerning potential risk applications.

Crystalfontz America, Inc. assumes no liability for applications assistance, customer product design, software performance, or infringements of patents or services described herein. Nor does Crystalfontz America, Inc. warrant or represent that any license, either express or implied, is granted under any patent right, copyright, or other intellectual property right of Crystalfontz America, Inc. covering or relating to any combination, machine, or process in which our products or services might be or are used.

The information in this publication is deemed accurate but is not guaranteed.

Company and product names mentioned in this publication are trademarks or registered trademarks of their respective owners.

CONTENTS

CONTENTS, CONTINUED

LIST OF FIGURES

MAIN FEATURES

The XES635BK-TFE-KU is a Crystalfontz CFA635BK-TFE-KU module enclosed in a sturdy steel black case. The case is compact – only slightly larger than the bare module. The approximately 9.5-foot black low-drop USB "A" cable supplies both power and data. The cable is permanently attached. It is easy to position the module at eye level on a work surface or mount it to a wall.

- ❑ Large easy-to-read 20 characters by 4 lines LCD has a large display area in a compact 146.0 (W) x 39.29 (H) mm x 22.05 mm (D) package (5.75” (W) x 1.55” (H) x 0.87” (D)).
- ❑ USB interface (115200K baud equivalent throughput).
- ❑ White edge LED backlit with FSTN positive mode LCD (displays dark characters on light background).
- ❑ Integrated white LED backlit 6-button translucent silicone keypad with screened legend.
- ❑ Direct sunlight readable.
- ❑ Four bicolor (red + green) LED Indicators. The LEDs' brightness can be set by the host software, which allows smoothly mixing the LEDs to produce other colors (for example, yellow and orange).
- ❑ LCD characters are contiguous in both X and Y directions to allow the host software to display “gapless” bar graphs in horizontal or vertical directions.
- ❑ Unique "Scrolling Marquee" feature continuously scrolls a message across the display without host intervention.
- ❑ Fully decoded keypad: any key combination is valid and unique.
- ❑ Robust packet-based communications protocol with 16-bit CRC.
- ❑ Built-in microcontroller.
- ❑ Nonvolatile memory capability (EEPROM):
  - Customize the “power-on” display settings.
  - 16-byte “scratch” register for storing IP address, netmask, system serial number . . .
- ❑ Certifications by MET Laboratories:
  - Product complies with UL60950 CSA C22.2 No. 60950. Product safety certification for Canada, United States, and Europe.
  - Product complies to CE standards (European safety, health, and environmental standards).
  - Product passed EU ESD immunity (Electrostatic Discharge) and radiated immunity requirements with no anomalies.
  - Product complies with Title 47 of the CFR, Part 15, Subpart B for a Class B Digital Device.
  - Product complies with ICES-003 Issue 4, February 7, 20004, Class B.
  - Product materials are in compliance with the regulations related to the EU Directive 2006/121/EC for Registration, Evaluation, Authorization and Restriction of Chemicals (REACH).
- ❑ Also:
  - Meets requirements for demonstrating compliance with California Proposition 65.
  - Factories have ISO certification.
- ❑ RoHS compliant.

ORDERING INFORMATION

MODULE VARIANT CHOICES

PART NUMBER	FLUID	LCD GLASS COLOR	IMAGE	POLARIZER FILM	BACKLIGHTS
XES635BK-TFE-KU	FSTN	neutral	positive	transflective	LCD: white edge LEDs Keypad: white LEDs 
Additional variants available (same form factor, different LCD mode or backlight):					
XES635BK-YYE-KU	STN	yellow-green	positive	transflective	LCD: yellow-green edge LEDs Keypad: yellow -green LEDs 
XES635BK-TMF-KU	STN	blue	negative	transmissive	LCD: white edge LEDs Keypad: blue LEDs 

MODULE CLASSIFICATION INFORMATION

XES 635 BK - T F E - K U
❶ ❷ ❸ ❹ ❺ ❻ ❼ ❽

❶	Family	XES – e**X**ternal **E**nclosure, **S**teel
❷	Model Identifier	635
❸	Finish	BK – black steel
❹	Backlight Type & Color	T – LED, white
❺	Fluid Type, Image (positive or negative), & LCD Glass Color	F – FSTN, positive
❻	Polarizer Film Type, Normal (NT) Temperature Range, & View Angle (O ‘Clock)	E – Transflective, NT, 12:00
❼	Special Code 1	K – Manufacturer's code
❽	Special Code 2	U – USB interface
[1] *Normal Temperature Range is 0°C minimum to +50°C maximum.*		

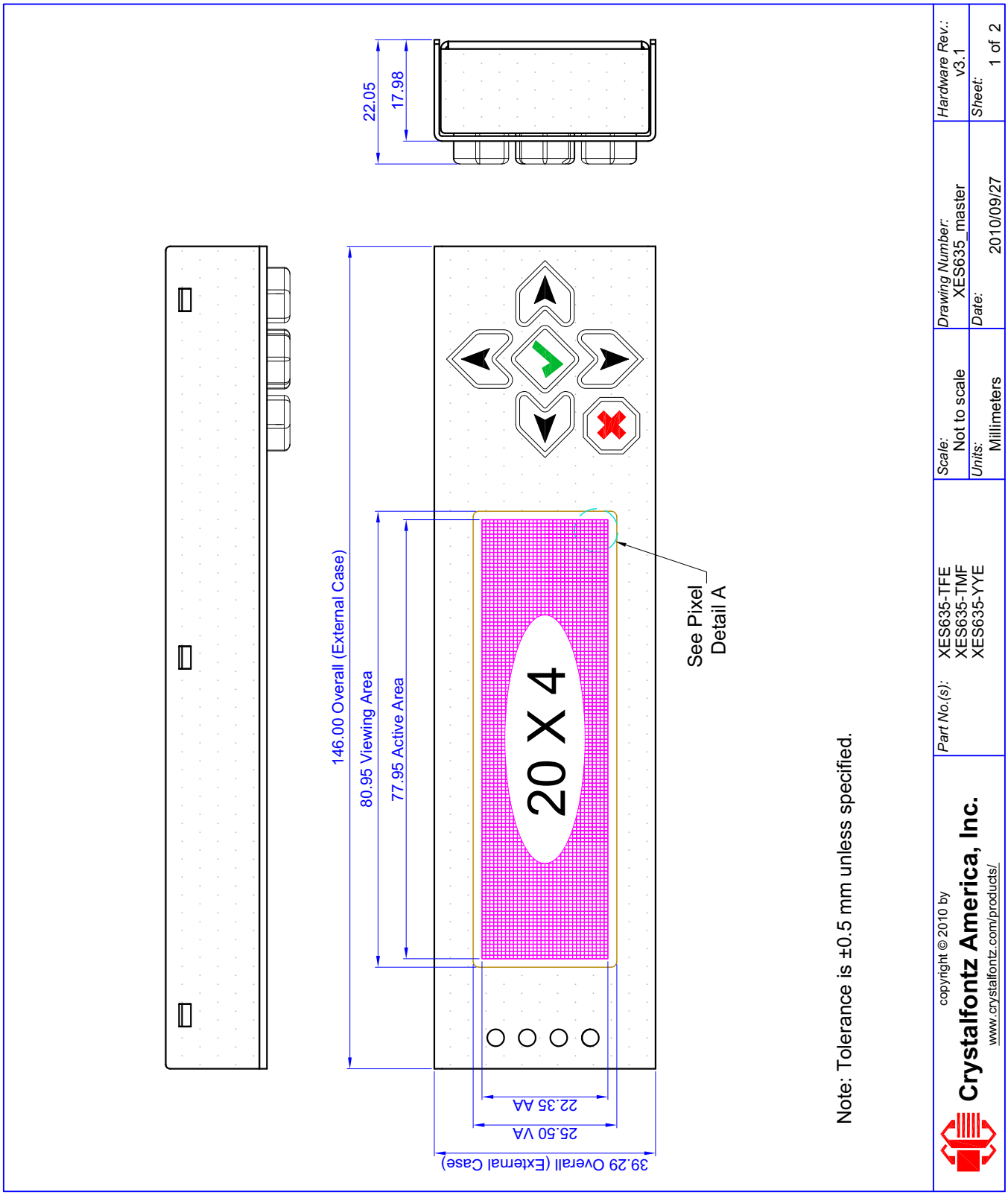
MECHANICAL SPECIFICATIONS

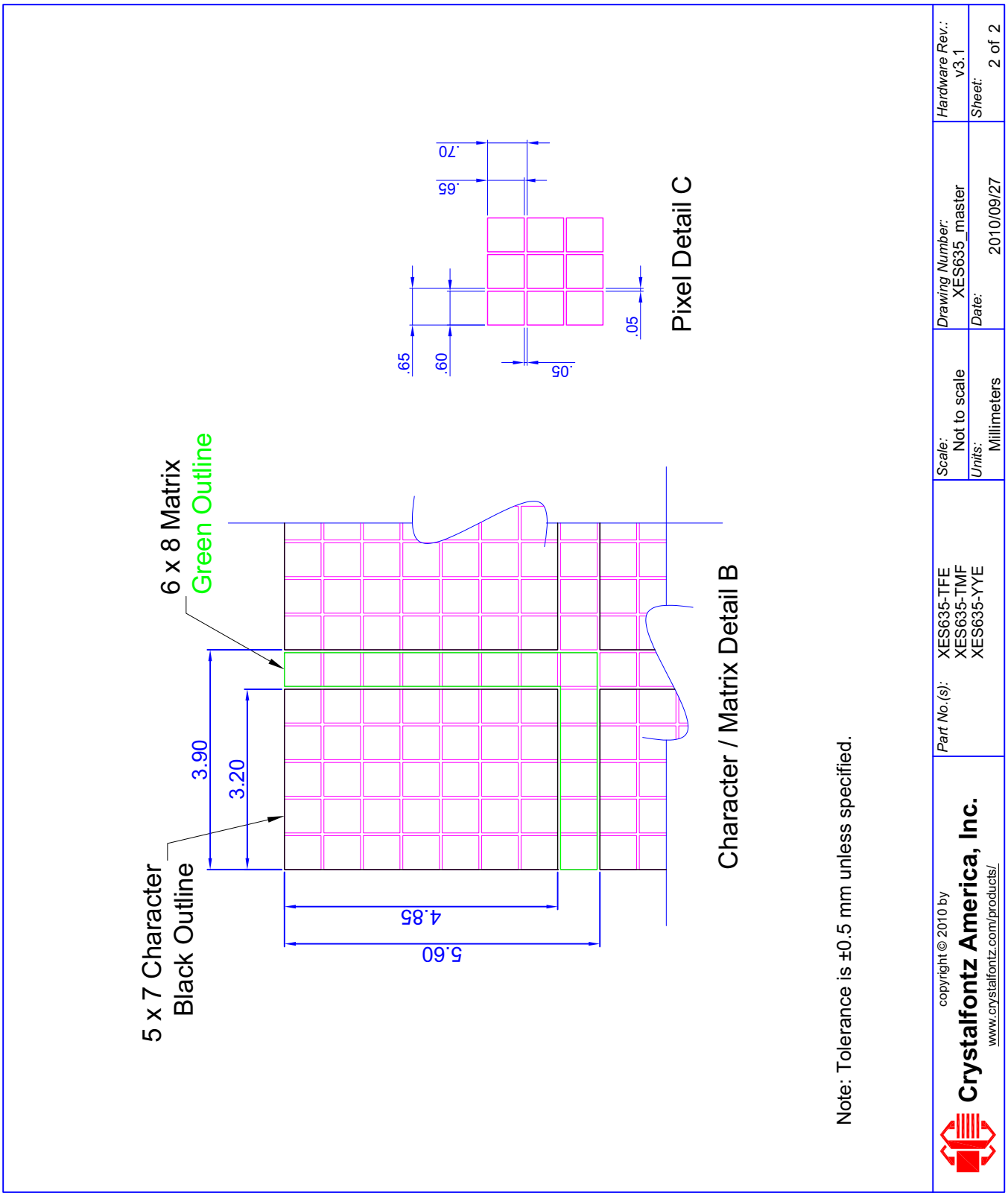
PHYSICAL CHARACTERISTICS

ITEM	SIZE
Module Width and Height	146.0 (W) x 39.29 (H) mm x 22.05 mm (D, includes keypad)
Viewing Area	80.95 (W) x 25.5 (H) mm
Active Area	77.95 (W) x 22.35 (H) mm
5x7 Standard Character Size	3.20 (W) x 4.85 (H) mm
6x8 Matrix (use for special characters or graphics)	3.90 (W) X 5.60 (H) mm
Pixel Size	0.60 (W) x 0.65 (H) mm
Pixel Pitch	0.65 (W) x 0.70 (H) mm
Keystroke Travel (approximate)	2.4 mm
Weight	297 grams (typical, includes cable)

MODULE OUTLINE DRAWING

Figure 1. Module Outline Drawings (two pages below)




Pixel Detail C
.70
.65
.05
.65
.60
.05
Character / Matrix Detail B
6 x 8 Matrix
Green Outline
5 x 7 Character
Black Outline
3.90
3.20
4.85
5.60
Note: Tolerance is ±0.5 mm unless specified.
Part No.(s):
XES635-TFE
XES635-TMF
XES635-YYE
Scale:
Not to scale
Units:
Millimeters
Drawing Number:
XES635_master
Date:
2010/09/27
Hardware Rev.:
v3.1
Sheet:
2 of 2
copyright © 2010 by
Crystalfontz America, Inc.
www.crystalfontz.com/products/

ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

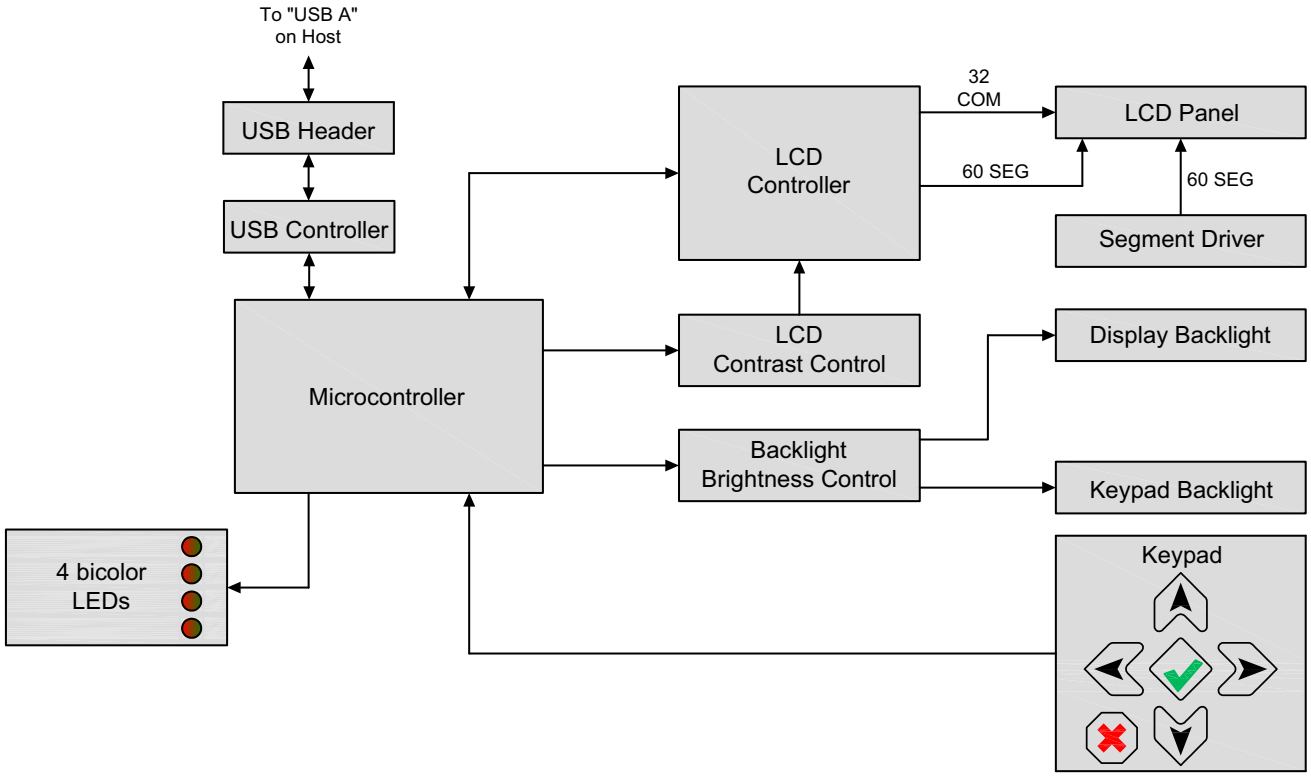


Figure 2. System Block Diagram

VIEWING DIRECTION

Viewing Direction	12 o'clock

DRIVING METHOD

DRIVING METHOD	SPECIFICATION
Duty	1/32
Bias	6.7

ABSOLUTE MAXIMUM RATINGS

ABSOLUTE MAXIMUM RATINGS	SYMBOL	MINIMUM	MAXIMUM
Operating Temperature	T_{OP}	0°C	+50°C
Storage Temperature	T_{ST}	-10°C	+60°C
Supply Voltage for Logic	V_{DD}	0	+5.25v

Note
These are stress ratings only. Functional operation of the module at these or any other conditions beyond those listed under DC Characteristics (Pg. 31) is not implied.

Extended exposure to the absolute maximum ratings listed above may affect device reliability. Stresses beyond those listed above can cause permanent damage.

DC CHARACTERISTICS

DC CHARACTERISTICS	SYMBOL	MINIMUM	TYPICAL	MAXIMUM
Supply Voltage	V_{DD} - V_O	+4.75v	+5.0v	+5.25v

TYPICAL CURRENT CONSUMPTION

ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION	
Logic	LCD and Keypad Backlights	All Indicator LEDs (4 Red + 4 Green)	V_{DD}=4.75V	V_{DD}=5.25V
X	-	-	35 mA	42 mA
X	X	-	129 mA	161 mA
X	-	X	147 mA	175 mA
X	X	X	239 mA	290 mA

BACKLIGHT PWM FREQUENCY

BACKLIGHT PWM FREQUENCY	SPECIFICATION
Backlight PWM Frequency	300 Hz nominal
PWM is *Pulse Width Modulation*. PWM is a way to simulate intermediate levels by switching a level between full on and full off. PWM can be used to control the brightness of LED backlights, relying on the natural averaging done by the human eye, as well as for controlling fan power.	

HOST COMMUNICATIONS

The XES635BK-TFE-KU communicates with its host using the USB interface. The easiest and most common way for the host software to access the USB is through the Crystalfontz virtual COM port (VCP) drivers. A link to VCP drivers download and installation instructions can be found on the Crystalfontz website at USB LCD Drivers. Using these drivers

makes it appear to the host software as if there is an additional serial port (the VCP) on the host system when the XES635BK-TFE-KU is connected. This VCP should be opened at 115200 baud, 8 data bits, no parity, 1 stop bit.

> **NOTE**
> The remaining sections in this Data Sheet use the shorter term “XES635BK” instead of “XES635BK-TFE-KU”.

PACKET STRUCTURE

All communication between the XES635BK and the host takes place in the form of a simple and robust CRC checked packet. The packet format allows for very reliable communications between the XES635BK and the host without the traditional problems that occur in a stream-based serial communication (such as having to send data in inefficient ASCII format, to “escape” certain “control characters”, or losing sync if a character is corrupted, missing, or inserted).

All packets have the following structure:

```
<type><data_length><data><CRC>
```

type is one byte, and identifies the type and function of the packet:

```
TTcc cccc
|||| ||||--Command, response, error or report code 0-63
||---------Type:
             00 = normal command from host to XES635BK
             01 = normal response from XES635BK to host
             10 = normal report from XES635BK to host (not in
                  direct response to a command from the host)
             11 = error response from XES635BK to host (a packet
                  with valid structure but illegal content
                  was received by the XES635BK)
```

data_length specifies the number of bytes that will follow in the data field. The valid range of **data_length** is 0 to 22.

data is the payload of the packet. Each **type** of packet will have a specified **data_length** and format for **data** as well as algorithms for decoding **data** detailed below.

CRC is a standard 16-bit CRC of all the bytes in the packet except the CRC itself. The CRC is sent LSB first. At the port, the CRC immediately follows the last used element of data **[]**. See APPENDIX B: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC) (Pg. 33) for details.

The following C definition may be useful for understanding the packet structure.

```
typedef struct
  {
  unsigned char
    command;
  unsigned char
    data_length;
  unsigned char
    data[MAX_DATA_LENGTH];
  unsigned short
    CRC;
  }COMMAND_PACKET;
```

On our website, Crystalfontz supplies a demonstration and test program, 635_WinTest along with its C source code. Included in the 635_WinTest source is a CRC algorithm and an algorithm that detects packets. The algorithm will

automatically re-synchronize to the next valid packet in the event of any communications errors. Please follow the algorithm in the sample code closely in order to realize the benefits of using the packet communications.

ABOUT HANDSHAKING

The nature of XES635BK's packets makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for a corresponding acknowledge packet from the XES635BK before sending the next command packet. The XES635BK will respond to all packets within 250 mS. The host software should stop waiting and retry the packet if the XES635BK fails to respond within 250 mS. The host software should report an error if a packet is not acknowledged after several retries. This situation indicates a hardware problem — for example, a disconnected cable.

Please note that some operating systems may introduce delays between when the data arrives at the physical port from the XES635BK until it is available to the user program. In this case, the host program may have to increase its timeout window to account for the additional overhead of the operating system.

The XES635BK can be configured to send several types of report packets along with regular acknowledge packets. The host should be able to buffer several incoming packets and must guarantee that it can process and remove packets from its input buffer faster than the packets can arrive given the 115200 equivalent baud rate of the VCP and the reporting configuration of the XES635BK. For any modern PC or microcontroller using reasonably efficient software, this requirement will not pose a challenge.

The report packets are sent asynchronously with respect to the command packets received from the host. The host should not assume that the first packet received after it sends a command is the acknowledge packet for that command. The host should inspect the `type` field of incoming packets and process them accordingly.

REPORT CODES

The XES635BK can be configured to report the items below. The XES635BK sends reports automatically when the data becomes available. Reports are not sent in response to a particular packet received from the host. The three report types are:

0x80: Key Activity

If a key is pressed or released, the XES635BK sends a Key Activity report packet to the host. Key event reporting may be individually enabled or disabled by command 23 (0x17): Configure Key Reporting (Pg. 22).

```
type = 0x80
data_length = 1
data[0] is the type of keyboard activity:
       KEY_UP_PRESS              1
       KEY_DOWN_PRESS            2
       KEY_LEFT_PRESS            3
       KEY_RIGHT_PRESS           4
       KEY_ENTER_PRESS           5
       KEY_EXIT_PRESS            6
       KEY_UP_RELEASE            7
       KEY_DOWN_RELEASE          8
       KEY_LEFT_RELEASE          9
       KEY_RIGHT_RELEASE        10
       KEY_ENTER_RELEASE        11
       KEY_EXIT_RELEASE         12
```

These codes are identical to the codes returned by the by the CFA533, CFA633, and the CFA635. Please note that the CFA631 will return codes 13 through 20. (See the CFA631 Data Sheet on our website for more details.)

0x81: Not Supported (Fan Speed Report)

0x82: Not Supported (Temperature Sensor Report)

COMMAND CODES

Below is a list of valid commands for the XES635BK. Each command packet is answered by either a response packet or an error packet.The low 6 bits of the `type` field of the response or error packet is the same as the low 6 bits of the `type` field of the command packet being acknowledged.

0 (0x00): Ping Command

The XES635BK will return the Ping Command to the host.

```
type = 0x00 = 0(10)
valid data_length is 0 to 16
data[0-(data_length-1)] can be filled with any arbitrary data
```

The return packet is identical to the packet sent, except the type will be 0x40 (normal response, Ping Command):

```
type = 0x40 | 0x00 = 0x40 = 64(10)
data_length = (identical to received packet)
data[0-(data_length-1)] = (identical to received packet)
```

1 (0x01): Get Hardware & Firmware Version

The XES635BK will return the hardware and firmware version information to the host.

```
type = 0x01 = 1(10)
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 0x01 = 0x41 = 65(10)
data_length = 16
data[] = "XES635BK:hX.X,yY.Y"

X.X is the hardware revision, "1.1" for example
yY.Y is the firmware version, "v1.6" for example
```

2 (0x02): Write User Flash Area

The XES635BK reserves 16 bytes of nonvolatile memory for arbitrary use by the host. This memory can be used to store a serial number, IP address, gateway address, netmask, or any other data required. All 16 bytes must be supplied.

```
type = 0x02 = 2(10)
valid data_length is 16
data[] = 16 bytes of arbitrary user data to be stored in
         the XES635BK's non-volatile memory
```

The return packet will be:

```
type = 0x40 | 0x02 = 0x42 = 66(10)
data_length = 0
```

3 (0x03): Read User Flash Area

This command will read the User Flash Area and return the data to the host.

```
type = 0x03 = 3(10)
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 0x03 = 0x43 = 67(10)
data_length = 16
data[] = 16 bytes user data recalled from the XES635BK's non-volatile memory
```

4 (0x04): Store Current State As Boot State

The XES635BK loads its power-up configuration from nonvolatile memory when power is applied. The XES635BK is configured at the factory to display a “welcome screen” when power is applied. This command can be used to customize the welcome screen, as well as the following items:

- Characters shown on LCD, which are affected by:
  - Command 6 (0x06): Clear LCD Screen (Pg. 19).
  - Command 31 (0x1F): Send Data to LCD (Pg. 23).
- Special character font definitions (command 9 (0x09): Set LCD Special Character Data (Pg. 19)).
- Cursor position (command 11 (0x0B): Set LCD Cursor Position (Pg. 20)).
- Cursor style (command 12 (0x0C): Set LCD Cursor Style (Pg. 20)).
- Contrast setting (command 13 (0x0D): Set LCD Contrast (Pg. 20)).
- Backlight setting (command 14 (0x0E): Set LCD & Keypad Backlight (Pg. 21)).
- Key press and release masks (command 23 (0x17): Configure Key Reporting (Pg. 22)).
- Baud rate (command 33 (0x21): Set Baud Rate (Pg. 24)).
- The front panel LED/GPO settings (34 (0x22): Set GPO Pin (Pg. 24)).

To store the current state as the boot state, send the following packet:

```
type = 0x04 = 4(10)
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 0x04 = 0x44 = 68(10)
data_length = 0
```

5 (0x05): Reboot XES635BK Module

This command instructs the XES635BK to simulate a power-on restart of itself. Additional features/actions not supported are Reset Host and Power Off Host.

Rebooting the XES635BK may be useful when testing the boot configuration. To reboot the XES635BK, send the following packet:

```
type = 0x05 = 5(10)
valid data_length is 3
data[0] = 8
data[1] = 18
data[2] = 99
```

The return packet will be:

```
type = 0x40 | 0x05 = 0x45 = 69(10)
data_length = 0
```

6 (0x06): Clear LCD Screen

Sets the contents of the LCD screen DDRAM to ' ' = 0x20 = 32 and moves the cursor to the left-most column of the top line.

```
type = 0x06 = 6(10)
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 0x06 = 0x46 = 70(10)
data_length = 0
```

Clear LCD Screen is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

7 (0x07): Deprecated (See command 31 (0x1F): Send Data to LCD (Pg. 23))

8 (0x08): Deprecated (See command 31 (0x1F): Send Data to LCD (Pg. 23))

9 (0x09): Set LCD Special Character Data

Sets the font definition for one of the special characters (CGRAM).

```
type = 0x09 = 9(10)
valid data_length is 9
data[0]   = index of special character that you would like to modify, 0-7 are valid
data[1-8] = bitmap of the new font for this character
```

`data[1-8]` are the bitmap information for this character. Any value is valid between 0 and 63, the msb is at the left of the character cell of the row, and the lsb is at the right of the character cell.

`data[1]` is at the top of the cell.
`data[8]` is at the bottom of the cell.

Additionally, if you set bit 7 of any of the data bytes, the entire line will blink.

The return packet will be:

```
type = 0x40 | 0x09 = 0x49 = 73(10)
data_length = 0
```

Set LCD Special Character Data is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

10 (0x0A): Read 8 Bytes of LCD Memory

This command will return the contents of the LCD's DDRAM or CGRAM. This command is intended for debugging.

```
type = 0x0A = 10(10)
valid data_length is 1
data[0] = address code of desired data
```

`data[0]` is the address code native to the LCD controller:

```
0x40 ( 64) to 0x7F (127) for CGRAM
0x80 (128) to 0x93 (147) for DDRAM, line 0
0xA0 (160) to 0xB3 (179) for DDRAM, line 1
0xC0 (192) to 0xD3 (211) for DDRAM, line 2
0xE0 (224) to 0xF3 (243) for DDRAM, line 3
```

The return packet will be:

```
type = 0x40 | 0x0A = 0x4A = 74(10)
data_length = 9
```

`data[0]` of the return packet will be the address code.
`data[1-8]` of the return packet will be the data read from the LCD controller's memory.

11 (0x0B): Set LCD Cursor Position

This command allows the cursor to be placed at the desired location on the XES635BK's LCD screen. If you want the cursor to be visible, you may also need to send a command 12 (0x0C): Set LCD Cursor Style (Pg. 20).

```
type = 0x0B = 11(10)
valid data_length is 2
data[0] = column (0-19 valid)
data[1] = row (0-3 valid)
```

The return packet will be:

```
type = 0x40 | 0x0B = 0x4B = 75(10)
data_length = 0
```

Set LCD Cursor Position is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

12 (0x0C): Set LCD Cursor Style

This command allows you to select among four hardware generated cursor options.

```
type = 0x0C = 12(10)
valid data_length is 1
data[0] = cursor style (0-4 valid)
       0 = no cursor
       1 = blinking block cursor
       2 = underscore cursor
       3 = blinking block plus underscore
       4 = inverting, blinking block
```

The return packet will be:

```
type = 0x40 | 0x0C = 0x4C = 76(10)
data_length = 0
```

Set LCD Cursor Style is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

13 (0x0D): Set LCD Contrast

This command sets the contrast or vertical viewing angle of the display.

```
type = 0x0D = 13(10)
valid data_length is 1
data[0] = contrast setting (0-254 valid)
     0-65 = very light
       66 = light
       95 = about right
      125 = dark
  126-254 = very dark
```

The return packet will be:

```
type = 0x40 | 0x0D = 0x4D = 77(10)
data_length = 0
```

Set LCD Contrast is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

14 (0x0E): Set LCD & Keypad Backlight

This command sets the brightness of the LCD and keypad backlights.

```
type = 0x0E = 14(10)
valid data_length is 1
data[0] = backlight power setting (0-100 valid)
        0 = off
    1-99 = variable brightness
     100 = on
```

The return packet will be:

```
type = 0x40 | 0x0E = 0x4E = 78(10)
data_length = 0
```

Set LCD & Keypad Backlight is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

15 (0x0F): Deprecated

16 (0x10): Not Supported (Set Up Fan Reporting)

17 (0x11): Not Supported (Set Fan Power)

18 (0x12): Not Supported (Read DOW Device Information)

19 (0x13): Not Supported (Set Up Temperature Reporting)

20 (0x14): Not Supported (Arbitrary DOW Transaction)

21 (0x15): Deprecated

22 (0x16): Send Command Directly to the LCD Controller

The LCD controller on the XES635BK is S6A0073 compatible. Generally you won't need low-level access to the LCD controller but some arcane functions of the S6A0073 are not exposed by the XES635BK's command set. This command allows you to access the XES635BK's LCD controller directly. Note: It is possible to corrupt the XES635BK display using this command.

```
type = 0x16 = 22(10)
data_length = 2
data[0]: location code
      0 = "Data" register
      1 = "Control" register, RE=0
      2 = "Control" register, RE=1
data[1]: data to write to the selected register
```

The return packet will be:

```
type = 0x40 | 0x16 = 0x56 = 86(10)
data_length = 0
```

23 (0x17): Configure Key Reporting

By default, the XES635BK reports any key event to the host. This command allows the key events to be enabled or disabled on an individual basis. The key events set to report are one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

```
#define KP_UP     0x01
#define KP_ENTER  0x02
#define KP_CANCEL 0x04
#define KP_LEFT   0x08
#define KP_RIGHT  0x10
#define KP_DOWN   0x20

type = 0x17 = 23(10)
data_length = 2
data[0]: press mask
data[1]: release mask
```

The return packet will be:

```
type = 0x40 | 0x17 = 0x57 = 87(10)
data_length = 0
```

Configure Key Reporting is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the XES635BK for key activity. This command allows the host to detect which keys are currently pressed, which keys have been pressed since the last poll, and which keys have been released since the last poll.

This command is independent of the key reporting masks set by command 23 (0x17): Configure Key Reporting (Pg. 22). All keys are always visible to this command. Typically both masks of command 23 would be set to "0" if the host is reading the keypad in polled mode.

```
#define KP_UP     0x01
#define KP_ENTER  0x02
#define KP_CANCEL 0x04
#define KP_LEFT   0x08
#define KP_RIGHT  0x10
#define KP_DOWN   0x20

type = 0x18 = 24(10)
data_length = 0
```

The return packet will be:

```
type = 0x40 | 0x18 = 0x58 = 88(10)
data_length = 3
data[0] = bit mask showing the keys currently pressed
data[1] = bit mask showing the keys that have been pressed since
          the last poll
data[2] = bit mask showing the keys that have been released since
          the last poll
```

25 (0x19): Not Supported (Set Fan Power Fail-Safe)

26 (0x1A): Not Supported (Set Fan Tachometer Glitch Filter)

27 (0x1B): Not Supported (Query Fan Power & Fail-Safe Mask)

28 (0x1C): Not Supported (Set ATX Power Switch Functionality)

29 (0x1D): Not Supported (Enable/Disable and Reset the Watchdog)

30 (0x1E): Read Reporting & Status

This command can be used to verify the current items configured to report to the host, as well as some other miscellaneous status information.

```
type = 0x1E = 30(10)
data_length = 0
```

The return packet will be:

```
type = 0x40 | 0x1E = 0x5E = 94(10)
data_length = 15
data[0]   = Not Supported  (fan 1-4 reporting status as set by command 16)
data[1]   = Not Supported (temperatures 1-8 reporting status as set by command 19)
data[2]   = Not Supported (temperatures 9-15 reporting status as set by command 19)
data[3]   = Not Supported (temperatures 16-23 reporting status as set by command 19)
data[4]   = Not Supported (temperatures 24-32 reporting status as set by command 19)
data[5]   = key presses (as set by command 23)
data[6]   = key releases (as set by command 23)
data[7]   = Not Supported (ATX Power Switch Functionality as set by command 28)
data[8]   = Not Supported (current watchdog counter as set by command 29)
data[9]   = Not Supported (fan RPM glitch delay[0] as set by command 26)
data[10]  = Not Supported (fan RPM glitch delay[1] as set by command 26)
data[11]  = Not Supported (fan RPM glitch delay[2] as set by command 26)
data[12]  = Not Supported (fan RPM glitch delay[3] as set by command 26)
data[13]  = contrast setting (as set by command 13)
data[14]  = backlight setting (as set by command 14)
```

Please Note: Previous and future firmware versions may return fewer or additional bytes.

31 (0x1F): Send Data to LCD

This command allows data to be placed at any position on the LCD.

```
type = 0x1F = 31(10)
data_length = 3 to 22
data[0]: col = x = 0 to 19
data[1]: row = y = 0 to 3
data[2-21]: text to place on the LCD, variable from 1 to 20 characters
```

The return packet will be:

```
type = 0x40 | 0x1F = 0x5F = 95(10)
data_length = 0
```

Send Data to LCD is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

32 (0x20): Reserved for CFA631 Key Legends

33 (0x21): Set Baud Rate

This command will change the XES635BK's baud rate. The XES635BK will send the acknowledge packet for this command and change its baud rate to the new value. The host should send the baud rate command, wait for a positive acknowledge from the XES635BK at the old baud rate, and then switch itself to the new baud rate. The baud rate must be saved by the command 4 (0x04): Store Current State As Boot State (Pg. 18) if you want the XES635BK to power up at the new baud rate.

The factory default baud rate is 115200.

```
type = 0x21 = 33(10)
data_length = 0
data[1]: 0 = 19200 baud
         1 = 115200 baud
```

The return packet will be:

```
type = 0x40 | 0x21 = 0x61 = 97(10)
data_length = 0
```

34 (0x22): Set GPO Pin

The XES635BK four bicolor LEDs at the left of the LCD. These LEDs are controlled by the GPO (general purpose output) pins on the module.

The GPO can output constant high or low signals or a variable duty cycle 100 Hz PWM signal.

The GPO configuration is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 18).

```
type: 0x22 = 34(10)
data_length:
  2 bytes

data[0]: index of GPIO/GPO to modify
       0  = GPIO[0] = (reserved)
       1  = GPIO[1] = (reserved)
       2  = GPIO[2] = (reserved)
       3  = GPIO[3] = (reserved)
       4  = GPIO[4] = (reserved)
       5  = GPO[5]  = LED 3 (bottom) green die
       6  = GPO[6]  = LED 3 (bottom) red die
       7  = GPO[7]  = LED 2          green die
       8  = GPO[8]  = LED 2          red die
       9  = GPO[9]  = LED 1          green die
       10 = GPO[10] = LED 1          red die
       11 = GPO[11] = LED 0 (top)    green die
       12 = GPO[12] = LED 0 (top)    red die

  13-255: reserved

  Please note: Future versions of this command on future
  hardware models may accept additional values for data[0],
  which would control the state of future additional GPIO
  pins

data[1] = Pin output state:
        0 = Output set to low
     1-99 = Output duty cycle percentage (100 Hz nominal)
      100 = Output set to high
  101-254 = invalid
```

The return packet will be:

```
type = 0x40 | 0x22 = 0x62 = 98(10)
data_length = 0
```

35 (0x23): Not Supported (Read GPIO Pin Levels and Configuration State)

CHARACTER GENERATOR ROM (CGROM)

To find the code for a given character, add the two numbers that are shown in bold for its row and column. For example, the superscript "9" is in the column labeled "128d" and in the row labeled "9d". So you would add 128 + 9 to get 137. When you send a byte with the value of 137 to the display, then a superscript "9" will be shown.

Character Generator ROM (CGROM) for Crystalfontz CFA-635

upper 4 bits / lower 4 bits	**0**d 0000₂	**16**d 0001₂	**32**d 0010₂	**48**d 0011₂	**64**d 0100₂	**80**d 0101₂	**96**d 0110₂	**112**d 0111₂	**128**d 1000₂	**144**d 1001₂	**160**d 1010₂	**176**d 1011₂	**192**d 1100₂	**208**d 1101₂	**224**d 1110₂	**240**d 1111₂
0d 0000₂	CGRAM [0]															
1d 0001₂	CGRAM [1]															
2d 0010₂	CGRAM [2]															
3d 0011₂	CGRAM [3]															
4d 0100₂	CGRAM [4]															
5d 0101₂	CGRAM [5]															
6d 0110₂	CGRAM [6]															
7d 0111₂	CGRAM [7]															
8d 1000₂	CGRAM [0]															
9d 1001₂	CGRAM [1]															
10d 1010₂	CGRAM [2]															
11d 1011₂	CGRAM [3]															
12d 1100₂	CGRAM [4]															
13d 1101₂	CGRAM [5]															
14d 1110₂	CGRAM [6]															
15d 1111₂	CGRAM [7]															

Figure 3. Character Generator ROM (CGROM)

MODULE RELIABILITY AND LONGEVITY

PRODUCT RELIABILITY

ITEM	SPECIFICATION	
LCD portion (excluding Keypad, Indicator LEDs, and Backlights)	50,000 to 100,000 hours (typical)	
Keypad	1,000,000 keystrokes	
Bicolor LED Indicators	50,000 to 100,000 hours (typical)	
White LED Display and White LED Keypad Backlights *Note: We recommend that the backlight of white LED backlit modules be dimmed or turned off during periods of inactivity to conserve the white LED backlight lifetime.*	*Power-On Hours*	*% of Initial Brightness (New Module)*
	<10,000 hours	>90%
	<50,000 hours	>50%

PRODUCT LONGEVITY (EOL / REPLACEMENT POLICY)

Crystalfontz is committed to making all of our LCD modules available for as long as possible. For each module we introduce, we intend to offer it indefinitely. We do not preplan a module's obsolescence. The majority of modules we have introduced are still available.

We recognize that discontinuing a module may cause problems for some customers. However, rapidly changing technologies, component availability, or low customer order levels may force us to discontinue (“End of Life”, EOL) a module. For example, we must occasionally discontinue a module when a supplier discontinues a component or a manufacturing process becomes obsolete. When we discontinue a module, we will do our best to find an acceptable replacement module with the same fit, form, and function.

In most situations, you will not notice a difference when comparing a “fit, form, and function” replacement module to the discontinued module it replaces. However, sometimes a change in component or process for the replacement module results in a slight variation, perhaps an improvement, over the previous design.

Although the replacement module is still within the stated Data Sheet specifications and tolerances of the discontinued module, changes may require modification to your circuit and/or firmware. Possible changes include:

- *Backlight LEDs.* Brightness may be affected (perhaps the new LEDs have better efficiency) or the current they draw may change (new LEDs may have a different VF).
- *Controller.* A new controller may require minor changes in your code.
- *Component tolerances.* Module components have manufacturing tolerances. In extreme cases, the tolerance stack can change the visual or operating characteristics.

Please understand that we avoid changing a module whenever possible; we only discontinue a module if we have no other option. We post Part Change Notices (PCN) on the product's website page as soon as possible. If interested, you can subscribe to future part change notifications.

CARE AND HANDLING INFORMATION

For optimum operation of the XES635BK-TFE-KU and to prolong its life, please follow the precautions below.

OPERATION AND STORAGE

- The case is not waterproof. To reduce the risk of fire, electric shock, or product damage, do not expose to drips or splashes.
- Do not use or store in a very dusty or high humidity area. Dust and moisture can enter inside the case and damage the components.
- Do not install near any heat sources such as radiators, heat registers, stoves, or other appliances that produce heat. Do not expose to direct sunlight for long periods of time.
- Storage temperature limitations: from -10°C minimum to +60°C maximum with minimal fluctuations. Rapid temperature changes can cause moisture to form inside the case, resulting in permanent damage.
- Operating temperature limitations: from 0°C minimum to a maximum of 50°C with minimal fluctuation. Operation outside of these limits may shorten life and/or harm display.
  - At lower temperatures of this range, response time is delayed.
  - At higher temperatures of this range, display becomes dark. (You may need to adjust the contrast.)
- Adjust backlight brightness so the display is readable but not too bright. Dim or turn off the backlight during periods of inactivity to conserve the white LED backlight lifetime.
- Do not try to disassemble or modify the XES635BK-TFE-KU.
- Do not expose the module to strong mechanical shock, impact, torque, or tension.
- Do not drop, toss, bend, or twist the module.
- To avoid damage to the cable, do not tightly twist, pinch, or pull hard.
- Do not place weight or pressure on the XES635BK-TFE-KU.
- The XES635BK-TFE-KU is shipped with a protective plastic film over the display window. For best view, please remove the film.
- The case window is made out of plastic. It is “scratch resistant” polycarbonate but can be scratched or damaged by abuse.
- If you must discard the XES635BK-TFE-KU, please recycle at an approved facility.

IF LCD PANEL BREAKS

- If the XES635BK-TFE-KU is severely damaged and the LCD panel behind the polycarbonate breaks, be careful to not get the liquid crystal fluid in your mouth or eyes.
- If the liquid crystal fluid touches your skin, clothes, or work surface, wash it off immediately with soap and plenty of water.

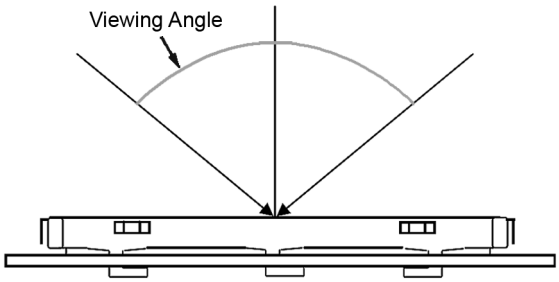
HOW TO CLEAN

- Do not spray any liquid directly on the module. The case is not waterproof.
- Clean the XES635BK-TFE-KU with a soft cloth slightly moistened with clear liquid glass cleaner (for example, Windex) and then wipe dry. Do not use chemical cleaners or solvents.
- The case window is polycarbonate. The polycarbonate may be scratched or damaged. Damage will be especially obvious on a "negative" module (a module that appear dark when power is “off”). Be extra gentle when you clean the case window.

APPENDIX A: QUALITY ASSURANCE STANDARDS

INSPECTION CONDITIONS

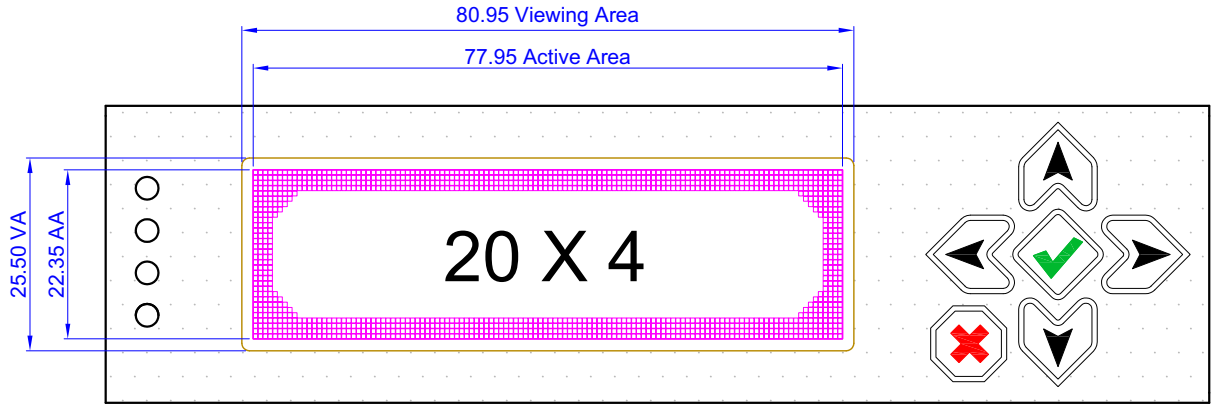
- Environment
  - Temperature: 25±5°C
  - Humidity: 30~85% RH
- For visual inspection of active display area
  - Source lighting: two 20 watt or one 40 watt fluorescent light
  - Display adjusted for best contrast
  - Viewing distance: 30±5 cm (about 12 inches)
  - Viewing angle: inspect at 45° angle of normal line right and left, top and bottom



COLOR DEFINITIONS

We try to describe the appearance of our LCD modules as accurately as possible. For the photos, we adjust the backlight (if any) and contrast for optimal appearance. Actual display appearance may vary due to (1) different operating conditions, (2) small variations of component tolerances, (3) inaccuracies of our camera, (4) color interpretation of the photos on your monitor, and/or (5) personal differences in the perception of color.

DEFINITION OF ACTIVE AREA AND VIEWABLE AREA



ACCEPTANCE SAMPLING

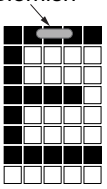
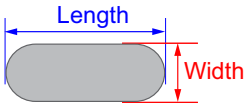
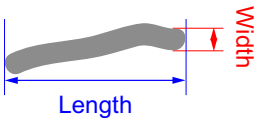
DEFECT TYPE	AQL*
Major	≤.65%
Minor	<1.0%
* Acceptable Quality Level: maximum allowable error rate or variation from standard	

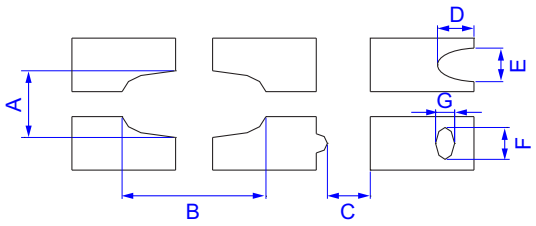
DEFECTS CLASSIFICATION

Defects are defined as:

- A *major defect* is a defect that substantially reduces usability of unit for its intended purpose.
- A *minor defect*: is a defect that is unlikely to reduce usability for its intended purpose.

ACCEPTANCE STANDARDS

#	DEFECT TYPE	CRITERIA			MAJOR / MINOR
1	Electrical defects	1. No display, display malfunctions, or shorted segments. 2. Current consumption exceeds specifications.			Major
2	Viewing area defect	Viewing area does not meet specifications. (See Inspection Conditions (Pg. 29)).			Major
3	Contrast adjustment defect	Contrast adjustment fails or malfunctions.			Major
4	Blemishes or foreign matter on display segments	Blemish 	*Defect Size*	*Acceptable Qty*	Minor
			≤0.3 mm	3	
			≤2 defects within 10 mm of each other		
			a≤1/4W		
5	Blemishes or foreign matter outside of display segments	Defect Size = (Width + Length)/2 	*Defect Size*	*Acceptable Qty*	Minor
			≤0.15 mm	Ignore	
			0.15 to 0.20 mm	3	
			0.20 to 0.25 mm	2	
			0.25 to 0.30 mm	1	
6	Dark lines or scratches in display area 	*Defect Width*	*Defect Length*	*Acceptable Qty*	Minor
		≤0.03 mm	≤3.0 mm	3	
		0.03 to 0.05	≤2.0 mm	2	
		0.05 to 0.08	≤2.0 mm	1	
		0.08 to 0.10	≤3.0 mm	0	
		≥0.10	>3.0 mm	0	

<table>
<tr><th>#</th><th>DEFECT TYPE</th><th colspan="3">CRITERIA</th><th>MAJOR / MINOR</th></tr>
<tr><td rowspan="5">7</td><td rowspan="5">Bubbles between polarizer film and glass</td><td colspan="2">Defect Size</td><td>Acceptable Qty</td><td rowspan="5">Minor</td></tr>
<tr><td colspan="2">≤0.2 mm</td><td>Ignore</td></tr>
<tr><td colspan="2">0.20 to 0.40 mm</td><td>3</td></tr>
<tr><td colspan="2">0.40 to 0.60 mm</td><td>2</td></tr>
<tr><td colspan="2">≥0.60 mm</td><td>0</td></tr>
<tr><td rowspan="6">8</td><td rowspan="6">Display pattern defect</td><td colspan="3"> 
</td><td></td></tr>
<tr><td>Dot Size</td><td colspan="2">Acceptable Qty</td><td>Minor</td></tr>
<tr><td>((A+B)/2)≤0.2 mm</td><td colspan="2" rowspan="4">≤3 total defects
≤2 pinholes per digit</td><td rowspan="4"></td></tr>
<tr><td>C>0 mm</td></tr>
<tr><td>((D+E)/2)≤0.25 mm</td></tr>
<tr><td>((F+G)/2)≤0.25 mm</td></tr>
<tr><td>9</td><td>Backlight defects</td><td colspan="3">1. Light fails or flickers.
2. Color and luminance do not correspond to specifications.
3. Exceeds standards for display’s blemishes or foreign matter (see test 5, page 31), and dark lines or scratches (see test 6, page 31).
*Minor if display functions correctly. Major if the display fails.</td><td>Minor</td></tr>
</table>

APPENDIX B: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC)

SAMPLE CODE

Free downloadable code on our website:

- Windows compatible test/demonstration program and source.
  http://www.crystalfontz.com/product/635WinTest.html
- Linux compatible command-line demonstration program with C source code. 8K.
  http://www.crystalfontz.com/product/linux_cli_examples.html
- Supported by CrystalControl freeware.
  http://www.crystalfontz.com/product/CrystalControl2.html
- Windows USB driver and installation instructions.
  http://www.crystalfontz.com/product/USB_LCD_Driver.html

ALGORITHMS TO CALCULATE THE CRC

Below are seven sample algorithms that will calculate the CRC of a CFA635 packet. Some of the algorithms were contributed by forum members and originally written for the CFA631. The CRC used in the CFA635 is the same one that is used in IrDA,which came from PPP, which seems to be related to a CCITT (ref: Network Working Group Request for Comments: 1171) standard. At that point, the trail was getting a bit cold and diverged into several referenced articles and papers, dating back to 1983.

The polynomial used is $X^{16} + X^{12} + X^5 + X^0$ (0x8408)
The result is bit-wise inverted before being returned.

Algorithm 1: “C” Table Implementation

This algorithm is typically used on the host computer, where code space is not an issue.

```
//This code is from the IRDA LAP documentation, which appears to
//have been copied from PPP:
//
// http://irda.affiniscape.com/associations/2494/files/Specifications/
IrLAP11_Plus_Errata.zip
//
//I doubt that there are any worries about the legality of this code,
//searching for the first line of the table below, it appears that
//the code is already included in the linux 2.6 kernel "Driver for
//ST5481 USB ISDN modem". This is an "industry standard" algorithm
//and I do not think there are ANY issues with it at all.
typedef unsigned char ubyte;
typedef unsigned short word;
word get_crc(ubyte *bufptr,word len)
  {
  //CRC lookup table to avoid bit-shifting loops.
  static const word crcLookupTable[256] =
    {0x00000,0x01189,0x02312,0x0329B,0x04624,0x057AD,0x06536,0x074BF,
     0x08C48,0x09DC1,0x0AF5A,0x0BED3,0x0CA6C,0x0DBE5,0x0E97E,0x0F8F7,
     0x01081,0x00108,0x03393,0x0221A,0x056A5,0x0472C,0x075B7,0x0643E,
     0x09CC9,0x08D40,0x0BFDB,0x0AE52,0x0DAED,0x0CB64,0x0F9FF,0x0E876,
     0x02102,0x0308B,0x00210,0x01399,0x06726,0x076AF,0x04434,0x055BD,
     0x0AD4A,0x0BCC3,0x08E58,0x09FD1,0x0EB6E,0x0FAE7,0x0C87C,0x0D9F5,
     0x03183,0x0200A,0x01291,0x00318,0x077A7,0x0662E,0x054B5,0x0453C,
     0x0BDCB,0x0AC42,0x09ED9,0x08F50,0x0FBEF,0x0EA66,0x0D8FD,0x0C974,
```

```
    0x04204,0x0538D,0x06116,0x0709F,0x00420,0x015A9,0x02732,0x036BB,
    0x0CE4C,0x0DFC5,0x0ED5E,0x0FCD7,0x08868,0x099E1,0x0AB7A,0x0BAF3,
    0x05285,0x0430C,0x07197,0x0601E,0x014A1,0x00528,0x037B3,0x0263A,
    0x0DECD,0x0CF44,0x0FDDF,0x0EC56,0x098E9,0x08960,0x0BBFB,0x0AA72,
    0x06306,0x0728F,0x04014,0x0519D,0x02522,0x034AB,0x00630,0x017B9,
    0x0EF4E,0x0FEC7,0x0CC5C,0x0DDD5,0x0A96A,0x0B8E3,0x08A78,0x09BF1,
    0x07387,0x0620E,0x05095,0x0411C,0x035A3,0x0242A,0x016B1,0x00738,
    0x0FFCF,0x0EE46,0x0DCDD,0x0CD54,0x0B9EB,0x0A862,0x09AF9,0x08B70,
    0x08408,0x09581,0x0A71A,0x0B693,0x0C22C,0x0D3A5,0x0E13E,0x0F0B7,
    0x00840,0x019C9,0x02B52,0x03ADB,0x04E64,0x05FED,0x06D76,0x07CFF,
    0x09489,0x08500,0x0B79B,0x0A612,0x0D2AD,0x0C324,0x0F1BF,0x0E036,
    0x018C1,0x00948,0x03BD3,0x02A5A,0x05EE5,0x04F6C,0x07DF7,0x06C7E,
    0x0A50A,0x0B483,0x08618,0x09791,0x0E32E,0x0F2A7,0x0C03C,0x0D1B5,
    0x02942,0x038CB,0x00A50,0x01BD9,0x06F66,0x07EEF,0x04C74,0x05DFD,
    0x0B58B,0x0A402,0x09699,0x08710,0x0F3AF,0x0E226,0x0D0BD,0x0C134,
    0x039C3,0x0284A,0x01AD1,0x00B58,0x07FE7,0x06E6E,0x05CF5,0x04D7C,
    0x0C60C,0x0D785,0x0E51E,0x0F497,0x08028,0x091A1,0x0A33A,0x0B2B3,
    0x04A44,0x05BCD,0x06956,0x078DF,0x00C60,0x01DE9,0x02F72,0x03EFB,
    0x0D68D,0x0C704,0x0F59F,0x0E416,0x090A9,0x08120,0x0B3BB,0x0A232,
    0x05AC5,0x04B4C,0x079D7,0x0685E,0x01CE1,0x00D68,0x03FF3,0x02E7A,
    0x0E70E,0x0F687,0x0C41C,0x0D595,0x0A12A,0x0B0A3,0x08238,0x093B1,
    0x06B46,0x07ACF,0x04854,0x059DD,0x02D62,0x03CEB,0x00E70,0x01FF9,
    0x0F78F,0x0E606,0x0D49D,0x0C514,0x0B1AB,0x0A022,0x092B9,0x08330,
    0x07BC7,0x06A4E,0x058D5,0x0495C,0x03DE3,0x02C6A,0x01EF1,0x00F78};

  register word
    newCrc;
  newCrc=0xFFFF;
  //This algorithm is based on the IrDA LAP example.
  while(len--)
    newCrc = (newCrc >> 8) ^ crcLookupTable[(newCrc ^ *bufptr++) & 0xff];

  //Make this crc match the one's complement that is sent in the packet.
  return(~newCrc);
  }
```

Algorithm 2: "C" Bit Shift Implementation

This algorithm was mainly written to avoid any possible legal issues about the source of the routine (at the request of the LCDproc group). This routine was "clean" coded from the definition of the CRC. It is ostensibly smaller than the table driven approach but will take longer to execute. This routine is offered under the GPL.

```
typedef unsigned char ubyte;
typedef unsigned short word;
word get_crc(ubyte *bufptr,word len)
  {
  register unsigned int
    newCRC;
  //Put the current byte in here.
  ubyte
    data;
  int
    bit_count;
  //This seed makes the output of this shift based algorithm match
  //the table based algorithm. The center 16 bits of the 32-bit
  //"newCRC" are used for the CRC. The MSb of the lower byte is used
  //to see what bit was shifted out of the center 16 bit CRC
  //accumulator ("carry flag analog");
  newCRC=0x00F32100;
  while(len--)
    {
    //Get the next byte in the stream.
    data=*bufptr++;
    //Push this byte's bits through a software
    //implementation of a hardware shift & xor.
    for(bit_count=0;bit_count<=7;bit_count++)
```

```
    {
    //Shift the CRC accumulator
    newCRC>>=1;

    //The new MSB of the CRC accumulator comes
    //from the LSB of the current data byte.
    if(data&0x01)
      newCRC|=0x00800000;

    //If the low bit of the current CRC accumulator was set
    //before the shift, then we need to XOR the accumulator
    //with the polynomial (center 16 bits of 0x00840800)
    if(newCRC&0x00000080)
      newCRC^=0x00840800;
    //Shift the data byte to put the next bit of the stream
    //into position 0.
    data>>=1;
    }
  }

//All the data has been done. Do 16 more bits of 0 data.
for(bit_count=0;bit_count<=15;bit_count++)
  {
  //Shift the CRC accumulator
  newCRC>>=1;

  //If the low bit of the current CRC accumulator was set
  //before the shift we need to XOR the accumulator with
  //0x00840800.
  if(newCRC&0x00000080)
    newCRC^=0x00840800;
  }
//Return the center 16 bits, making this CRC match the one's
//complement that is sent in the packet.
return((~newCRC)>>8);
}
```

Algorithm 2B: "C" Improved Bit Shift Implementation

This is a simplified algorithm that implements the CRC.

```
unsigned short get_crc(unsigned char count,unsigned char *ptr)
  {
  unsigned short
    crc;   //Calculated CRC
  unsigned char
    i;     //Loop count, bits in byte
  unsigned char
    data;  //Current byte being shifted

  crc = 0xFFFF; // Preset to all 1's, prevent loss of leading zeros

  while(count--)
    {
    data = *ptr++;
    i = 8;
    do
      {
      if((crc ^ data) & 0x01)
        {
        crc >>= 1;
        crc ^= 0x8408;
        }
      else
        crc >>= 1;
      data >>= 1;
      } while(--i != 0);
    }
  return (~crc);
```

Algorithm 3: "PIC Assembly" Bit Shift Implementation

This routine was graciously donated by one of our customers.

```
;=====================================================================
; Crystalfontz CFA635 PIC CRC Calculation Example
;
; This example calculates the CRC for the hard coded example provided
; in the documentation.
;
; It uses "This is a test. " as input and calculates the proper CRC
; of 0x93FA.
;=====================================================================
#include "p16f877.inc"
;=====================================================================
; CRC16 equates and storage
;---------------------------------------------------------------------
accuml      equ     40h         ; BYTE - CRC result register high byte
accumh      equ     41h         ; BYTE - CRC result register high low byte
datareg     equ     42h         ; BYTE - data register for shift
j           equ     43h         ; BYTE - bit counter for CRC 16 routine
Zero        equ     44h         ; BYTE - storage for string memory read
index       equ     45h         ; BYTE - index for string memory read
savchr      equ     46h         ; BYTE - temp storage for CRC routine
```

```
;
seedlo      equ     021h        ; initial seed for CRC reg lo byte
seedhi      equ     0F3h        ; initial seed for CRC reg hi byte
;
polyL       equ     008h        ; polynomial low byte
polyH       equ     084h        ; polynomial high byte
;=====================================================================
;  CRC Test Program
;---------------------------------------------------------------------
        org         0           ; reset vector = 0000H
;
        clrf        PCLATH      ; ensure upper bits of PC are cleared
        clrf        STATUS      ; ensure page bits are cleared
        goto        main        ; jump to start of program
;
; ISR Vector
;
        org         4           ; start of ISR
        goto        $           ; jump to ISR when coded
;
        org         20          ; start of main program
main
        movlw       seedhi      ; setup intial CRC seed value.
        movwf       accumh      ; This must be done prior to
        movlw       seedlo      ; sending string to CRC routine.
        movwf       accuml      ;
        clrf        index       ; clear string read variables
;
main1
        movlw       HIGH InputStr   ; point to LCD test string
        movwf       PCLATH      ; latch into PCL
        movfw       index       ; get index
        call        InputStr    ; get character
        movwf       Zero        ; setup for terminator test
        movf        Zero,f      ; see if terminator
        btfsc       STATUS,Z    ; skip if not terminator
         goto         main2        ; else terminator reached, jump out of loop
        call        CRC16       ; calculate new crc
        call        SENDUART    ; send data to LCD
        incf        index,f     ; bump index
        goto        main1       ; loop
;
main2
        movlw       00h         ; shift accumulator 16 more bits.
        call        CRC16       ; This must be done after sending
        movlw       00h         ; string to CRC routine.
        call        CRC16       ;
;
        comf        accumh,f    ; invert result
        comf        accuml,f    ;
;
        movfw       accuml      ; get CRC low byte
        call        SENDUART    ; send to LCD
        movfw       accumh      ; get CRC hi byte
        call        SENDUART    ; send to LCD
;
stop    goto        stop           ; word result of 0x93FA is in accumh/accuml
;=====================================================================
; calculate CRC of input byte
;---------------------------------------------------------------------
CRC16
        movwf       savchr      ; save the input character
        movwf       datareg     ; load data register
        movlw       .8          ; setup number of bits to test
        movwf       j           ; save to incrementor
_loop
        clrc                    ; clear carry for CRC register shift
         rrf        datareg,f   ; perform shift of data into CRC register
```

```
        rrf         accumh,f    ;
        rrf         accuml,f    ;
        btfss       STATUS,C    ; skip jump if if carry
        goto        _notset     ; otherwise goto next bit
        movlw       polyL       ; XOR poly mask with CRC register
        xorwf       accuml,F    ;
        movlw       polyH       ;
        xorwf       accumh,F    ;
_notset
        decfsz      j,F         ; decrement bit counter
        goto        _loop       ; loop if not complete
        movfw       savchr      ; restore the input character
        return                  ; return to calling routine
;=====================================================================
; USER SUPPLIED Serial port transmit routine
;---------------------------------------------------------------------
SENDUART
        return                  ; put serial xmit routine here
;=====================================================================
; test string storage
;---------------------------------------------------------------------
        org     0100h
;
InputStr
        addwf   PCL,f
        dt      7h,10h,"This is a test. ",0
;
;=====================================================================
        end
```

Algorithm 4: “Visual Basic” Table Implementation

Visual BASIC has its own challenges as a language (such as initializing static arrays), and it is also challenging to use Visual BASIC to work with “binary” (arbitrary length character data possibly containing nulls—such as the “data” portion of the CFA635 packet) data. This routine was adapted from the C table implementation. The complete project can be found in our forums.

```
'This program is brutally blunt. Just like VB. No apologies.
'Written by Crystalfontz America, Inc. 2004 http://www.crystalfontz.com
'Free code, not copyright copyleft or anything else.
'Some visual basic concepts taken from:
'http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=21434&lngWId=1
'most of the algorithm is from functions in 635_WinTest:
'http://www.crystalfontz.com/product/635WinTest.html
'Full zip of the project is available in our forum:
'http://www.crystalfontz.com/forum/showthread.php?postid=9921#post9921

Private Type WORD
   Lo As Byte
   Hi As Byte
End Type

Private Type PACKET_STRUCT
   command As Byte
   data_length As Byte
   data(22) As Byte
   crc As WORD
End Type

Dim crcLookupTable(256) As WORD

Private Sub MSComm_OnComm()
'Leave this here
End Sub
```

```
'My understanding of visual basic is very limited--however it appears that there is no way
'to initialize an array of structures. Nice language. Fast processors, lots of memory, big
'disks, and we fill them up with this . . this . . this . . STUFF.
Sub Initialize_CRC_Lookup_Table()
  crcLookupTable(0).Lo = &H0
  crcLookupTable(0).Hi = &H0
  . . .
'For purposes of brevity in this data sheet, I have removed 251 entries of this table, the
'full source is available in our forum:
'http://www.crystalfontz.com/forum/showthread.php?postid=9921#post9921
  . . .
  crcLookupTable(255).Lo = &H78
  crcLookupTable(255).Hi = &HF
End Sub

'This function returns the CRC of the array at data for length positions
Private Function Get_Crc(ByRef data() As Byte, ByVal length As Integer) As WORD
  Dim Index As Integer
  Dim Table_Index As Integer
  Dim newCrc As WORD
  newCrc.Lo = &HFF
  newCrc.Hi = &HFF
  For Index = 0 To length - 1
    'exclusive-or the input byte with the low-order byte of the CRC register
    'to get an index into crcLookupTable
    Table_Index = newCrc.Lo Xor data(Index)
    'shift the CRC register eight bits to the right
    newCrc.Lo = newCrc.Hi
    newCrc.Hi = 0
    ' exclusive-or the CRC register with the contents of Table at Table_Index
    newCrc.Lo = newCrc.Lo Xor crcLookupTable(Table_Index).Lo
    newCrc.Hi = newCrc.Hi Xor crcLookupTable(Table_Index).Hi
  Next Index
  'Invert & return newCrc
  Get_Crc.Lo = newCrc.Lo Xor &HFF
  Get_Crc.Hi = newCrc.Hi Xor &HFF
End Function

Private Sub Send_Packet(ByRef packet As PACKET_STRUCT)
  Dim Index As Integer
  'Need to put the whole packet into a linear array
  'since you can't do type overrides. VB, gotta love it.
  Dim linear_array(26) As Byte
  linear_array(0) = packet.command
  linear_array(1) = packet.data_length
  For Index = 0 To packet.data_length - 1
    linear_array(Index + 2) = packet.data(Index)
  Next Index
  packet.crc = Get_Crc(linear_array, packet.data_length + 2)
  'Might as well move the CRC into the linear array too
  linear_array(packet.data_length + 2) = packet.crc.Lo
  linear_array(packet.data_length + 3) = packet.crc.Hi
  'Now a simple loop can dump it out the port.
  For Index = 0 To packet.data_length + 3
    MSComm.Output = Chr(linear_array(Index))
  Next Index
End Sub
```

Algorithm 5: "Java" Table Implementation

This code was posted in our forum by user "norm" as a working example of a Java CRC calculation.

```
public class CRC16 extends Object
  {
  public static void main(String[] args)
    {
    byte[] data = new byte[2];
```

```
  // hw - fw
  data[0] = 0x01;
  data[1] = 0x00;
  System.out.println("hw -fw req");
  System.out.println(Integer.toHexString(compute(data)));

  // ping
  data[0] = 0x00;
  data[1] = 0x00;
  System.out.println("ping");
  System.out.println(Integer.toHexString(compute(data)));

  // reboot
  data[0] = 0x05;
  data[1] = 0x00;
  System.out.println("reboot");
  System.out.println(Integer.toHexString(compute(data)));

  // clear lcd
  data[0] = 0x06;
  data[1] = 0x00;
  System.out.println("clear lcd");
  System.out.println(Integer.toHexString(compute(data)));

  // set line 1
  data = new byte[18];
  data[0] = 0x07;
  data[1] = 0x10;
  String text = "Test Test Test  ";
  byte[] textByte = text.getBytes();
  for (int i=0; i < text.length(); i++) data[i+2] = textByte[i];
  System.out.println("text 1");
  System.out.println(Integer.toHexString(compute(data)));
  }
private CRC16()
  {
  }
private static final int[] crcLookupTable =
  {
  0x00000,0x01189,0x02312,0x0329B,0x04624,0x057AD,0x06536,0x074BF,
  0x08C48,0x09DC1,0x0AF5A,0x0BED3,0x0CA6C,0x0DBE5,0x0E97E,0x0F8F7,
  0x01081,0x00108,0x03393,0x0221A,0x056A5,0x0472C,0x075B7,0x0643E,
  0x09CC9,0x08D40,0x0BFDB,0x0AE52,0x0DAED,0x0CB64,0x0F9FF,0x0E876,
  0x02102,0x0308B,0x00210,0x01399,0x06726,0x076AF,0x04434,0x055BD,
  0x0AD4A,0x0BCC3,0x08E58,0x09FD1,0x0EB6E,0x0FAE7,0x0C87C,0x0D9F5,
  0x03183,0x0200A,0x01291,0x00318,0x077A7,0x0662E,0x054B5,0x0453C,
  0x0BDCB,0x0AC42,0x09ED9,0x08F50,0x0FBEF,0x0EA66,0x0D8FD,0x0C974,
  0x04204,0x0538D,0x06116,0x0709F,0x00420,0x015A9,0x02732,0x036BB,
  0x0CE4C,0x0DFC5,0x0ED5E,0x0FCD7,0x08868,0x099E1,0x0AB7A,0x0BAF3,
  0x05285,0x0430C,0x07197,0x0601E,0x014A1,0x00528,0x037B3,0x0263A,
  0x0DECD,0x0CF44,0x0FDDF,0x0EC56,0x098E9,0x08960,0x0BBFB,0x0AA72,
  0x06306,0x0728F,0x04014,0x0519D,0x02522,0x034AB,0x00630,0x017B9,
  0x0EF4E,0x0FEC7,0x0CC5C,0x0DDD5,0x0A96A,0x0B8E3,0x08A78,0x09BF1,
  0x07387,0x0620E,0x05095,0x0411C,0x035A3,0x0242A,0x016B1,0x00738,
  0x0FFCF,0x0EE46,0x0DCDD,0x0CD54,0x0B9EB,0x0A862,0x09AF9,0x08B70,
  0x08408,0x09581,0x0A71A,0x0B693,0x0C22C,0x0D3A5,0x0E13E,0x0F0B7,
  0x00840,0x019C9,0x02B52,0x03ADB,0x04E64,0x05FED,0x06D76,0x07CFF,
  0x09489,0x08500,0x0B79B,0x0A612,0x0D2AD,0x0C324,0x0F1BF,0x0E036,
  0x018C1,0x00948,0x03BD3,0x02A5A,0x05EE5,0x04F6C,0x07DF7,0x06C7E,
  0x0A50A,0x0B483,0x08618,0x09791,0x0E32E,0x0F2A7,0x0C03C,0x0D1B5,
  0x02942,0x038CB,0x00A50,0x01BD9,0x06F66,0x07EEF,0x04C74,0x05DFD,
  0x0B58B,0x0A402,0x09699,0x08710,0x0F3AF,0x0E226,0x0D0BD,0x0C134,
  0x039C3,0x0284A,0x01AD1,0x00B58,0x07FE7,0x06E6E,0x05CF5,0x04D7C,
```

```
    0x0C60C,0x0D785,0x0E51E,0x0F497,0x08028,0x091A1,0x0A33A,0x0B2B3,
    0x04A44,0x05BCD,0x06956,0x078DF,0x00C60,0x01DE9,0x02F72,0x03EFB,
    0x0D68D,0x0C704,0x0F59F,0x0E416,0x090A9,0x08120,0x0B3BB,0x0A232,
    0x05AC5,0x04B4C,0x079D7,0x0685E,0x01CE1,0x00D68,0x03FF3,0x02E7A,
    0x0E70E,0x0F687,0x0C41C,0x0D595,0x0A12A,0x0B0A3,0x08238,0x093B1,
    0x06B46,0x07ACF,0x04854,0x059DD,0x02D62,0x03CEB,0x00E70,0x01FF9,
    0x0F78F,0x0E606,0x0D49D,0x0C514,0x0B1AB,0x0A022,0x092B9,0x08330,
    0x07BC7,0x06A4E,0x058D5,0x0495C,0x03DE3,0x02C6A,0x01EF1,0x00F78
    };
  public static int compute(byte[] data)
    {
    int newCrc = 0x0FFFF;
    for (int i = 0; i < data.length; i++ )
      {
      int lookup = crcLookupTable[(newCrc ^ data[i]) & 0xFF];
      newCrc = (newCrc >> 8) ^ lookup;
      }
    return(~newCrc);
    }
  }
```

Algorithm 6: “Perl” Table Implementation

This code was translated from the C version by one of our customers.

```
#!/usr/bin/perl

use strict;

my @CRC_LOOKUP =
   (0x00000,0x01189,0x02312,0x0329B,0x04624,0x057AD,0x06536,0x074BF,
    0x08C48,0x09DC1,0x0AF5A,0x0BED3,0x0CA6C,0x0DBE5,0x0E97E,0x0F8F7,
    0x01081,0x00108,0x03393,0x0221A,0x056A5,0x0472C,0x075B7,0x0643E,
    0x09CC9,0x08D40,0x0BFDB,0x0AE52,0x0DAED,0x0CB64,0x0F9FF,0x0E876,
    0x02102,0x0308B,0x00210,0x01399,0x06726,0x076AF,0x04434,0x055BD,
    0x0AD4A,0x0BCC3,0x08E58,0x09FD1,0x0EB6E,0x0FAE7,0x0C87C,0x0D9F5,
    0x03183,0x0200A,0x01291,0x00318,0x077A7,0x0662E,0x054B5,0x0453C,
    0x0BDCB,0x0AC42,0x09ED9,0x08F50,0x0FBEF,0x0EA66,0x0D8FD,0x0C974,
    0x04204,0x0538D,0x06116,0x0709F,0x00420,0x015A9,0x02732,0x036BB,
    0x0CE4C,0x0DFC5,0x0ED5E,0x0FCD7,0x08868,0x099E1,0x0AB7A,0x0BAF3,
    0x05285,0x0430C,0x07197,0x0601E,0x014A1,0x00528,0x037B3,0x0263A,
    0x0DECD,0x0CF44,0x0FDDF,0x0EC56,0x098E9,0x08960,0x0BBFB,0x0AA72,
    0x06306,0x0728F,0x04014,0x0519D,0x02522,0x034AB,0x00630,0x017B9,
    0x0EF4E,0x0FEC7,0x0CC5C,0x0DDD5,0x0A96A,0x0B8E3,0x08A78,0x09BF1,
    0x07387,0x0620E,0x05095,0x0411C,0x035A3,0x0242A,0x016B1,0x00738,
    0x0FFCF,0x0EE46,0x0DCDD,0x0CD54,0x0B9EB,0x0A862,0x09AF9,0x08B70,
    0x08408,0x09581,0x0A71A,0x0B693,0x0C22C,0x0D3A5,0x0E13E,0x0F0B7,
    0x00840,0x019C9,0x02B52,0x03ADB,0x04E64,0x05FED,0x06D76,0x07CFF,
    0x09489,0x08500,0x0B79B,0x0A612,0x0D2AD,0x0C324,0x0F1BF,0x0E036,
    0x018C1,0x00948,0x03BD3,0x02A5A,0x05EE5,0x04F6C,0x07DF7,0x06C7E,
    0x0A50A,0x0B483,0x08618,0x09791,0x0E32E,0x0F2A7,0x0C03C,0x0D1B5,
    0x02942,0x038CB,0x00A50,0x01BD9,0x06F66,0x07EEF,0x04C74,0x05DFD,
    0x0B58B,0x0A402,0x09699,0x08710,0x0F3AF,0x0E226,0x0D0BD,0x0C134,
    0x039C3,0x0284A,0x01AD1,0x00B58,0x07FE7,0x06E6E,0x05CF5,0x04D7C,
    0x0C60C,0x0D785,0x0E51E,0x0F497,0x08028,0x091A1,0x0A33A,0x0B2B3,
    0x04A44,0x05BCD,0x06956,0x078DF,0x00C60,0x01DE9,0x02F72,0x03EFB,
    0x0D68D,0x0C704,0x0F59F,0x0E416,0x090A9,0x08120,0x0B3BB,0x0A232,
    0x05AC5,0x04B4C,0x079D7,0x0685E,0x01CE1,0x00D68,0x03FF3,0x02E7A,
    0x0E70E,0x0F687,0x0C41C,0x0D595,0x0A12A,0x0B0A3,0x08238,0x093B1,
    0x06B46,0x07ACF,0x04854,0x059DD,0x02D62,0x03CEB,0x00E70,0x01FF9,
    0x0F78F,0x0E606,0x0D49D,0x0C514,0x0B1AB,0x0A022,0x092B9,0x08330,
    0x07BC7,0x06A4E,0x058D5,0x0495C,0x03DE3,0x02C6A,0x01EF1,0x00F78);

#  our test packet read from an enter key press over the serial line:
#   type = 80       (key press)
#   data_length = 1     (1 byte of data)
#   data = 5
```

```
my $type = '80';
my $length = '01';
my $data = '05';

my $packet = chr(hex $type) .chr(hex $length) .chr(hex $data);

my $valid_crc = '5584' ;

print "A CRC of Packet ($packet) Should Equal ($valid_crc)\n";

my $crc = 0xFFFF ;

printf("%x\n", $crc);

foreach my $char (split //, $packet)
  {
  # newCrc = (newCrc >> 8) ^ crcLookupTable[(newCrc ^ *bufptr++) & 0xff];
  # & is bitwise AND
  # ^ is bitwise XOR
  # >> bitwise shift right
  $crc = ($crc >> 8) ^ $CRC_LOOKUP[($crc ^ ord($char) ) & 0xFF] ;
  # print out the running crc at each byte
  printf("%x\n", $crc);
  }

# get the complement
$crc = ~$crc ;
$crc = ($crc & 0xFFFF) ;

# print out the crc in hex
printf("%x\n", $crc);
```

Algorithm 7: For PIC18F8722 or PIC18F2685

This code was written for the CFA635 by customer Virgil Stamps of ATOM Instrument Corporation.

```
; CRC Algorithm for CrystalFontz CFA-635 display (DB535)
; This code written for PIC18F8722 or PIC18F2685
;
; Your main focus here should be the ComputeCRC2 and
; CRC16_ routines
;
;=====================================================================
ComputeCRC2:
        movlb   RAM8
        movwf   dsplyLPCNT      ;w has the byte count
nxt1_dsply:
        movf    POSTINC1,w
        call    CRC16_
        decfsz  dsplyLPCNT
        goto    nxt1_dsply
        movlw   .0              ; shift accumulator 16 more bits
        call    CRC16_
        movlw   .0
        call    CRC16_
        comf    dsplyCRC,F      ; invert result
        comf    dsplyCRC+1,F
        return
;=====================================================================
CRC16_ movwf:
        dsplyCRCData            ; w has byte to crc
        movlw   .8
        movwf   dsplyCRCCount
```

```
_cloop:
        bcf     STATUS,C        ; clear carry for CRC register shift
        rrcf    dsplyCRCData,f  ; perform shift of data into CRC
                                ;register
        rrcf    dsplyCRC,F
        rrcf    dsplyCRC+1,F
        btfss   STATUS,C        ; skip jump if carry
        goto    _notset         ; otherwise goto next bit
        movlw   0x84
        xorwf   dsplyCRC,F
        movlw   0x08            ; XOR poly mask with CRC register
        xorwf   dsplyCRC+1,F
_notset:
        decfsz  dsplyCRCCount,F ; decrement bit counter
        bra _cloop              ; loop if not complete
        return
;=================================================================
; example to clear screen
dsplyFSR1_TEMP  equ     0x83A   ; 16-bit save for FSR1 for display
                                ; message handler
dsplyCRC        equ     0x83C   ; 16-bit CRC (H/L)
dsplyLPCNT      equ     0x83E   ; 8-bit save for display message
                                ; length - CRC
dsplyCRCData    equ     0x83F   ; 8-bit CRC data for display use
dsplyCRCCount   equ     0x840   ; 8-bit CRC count for display use
SendCount       equ     0x841   ; 8-bit byte count for sending to
                                ; display
RXBUF2          equ     0x8C0   ; 32-byte receive buffer for
                                ; Display
TXBUF2          equ     0x8E0   ; 32-byte transmit buffer for
                                ; Display
;-----------------------------------------------------------------
ClearScreen:
        movlb   RAM8
        movlw   .0
        movwf   SendCount
        movlw   0xF3
        movwf   dsplyCRC        ; seed ho for CRC calculation
        movlw   0x21
        movwf   dsplyCRC+1      ; seen lo for CRC calculation
        call    ClaimFSR1
        movlw   0x06
        movwf   TXBUF2
        LFSR    FSR1,TXBUF2
        movf    SendCount,w
        movwf   TXBUF2+1        ; message data length
        call    BMD1
        goto    SendMsg
;=================================================================
; send message via interrupt routine. The code is made complex due
; to the limited FSR registers and extended memory space used
;
; example of sending a string to column 0, row 0
;-----------------------------------------------------------------
SignOnL1:
        call    ClaimFSR1
        lfsr    FSR1,TXBUF2+4   ; set data string position
        SHOW    C0R0,BusName    ; move string to TXBUF2
        movlw   .2              ;
        addwf   SendCount       ;
        movff   SendCount,TXBUF2+1
                                ; insert message data length
        call    BuildMsgDSPLY
        call    SendMsg
        return
;=================================================================
; BuildMsgDSPLY used to send a string to LCD
;-----------------------------------------------------------------
```

```
BuildMsgDSPLY:
        movlw   0xF3
        movwf   dsplyCRC        ; seed hi for CRC calculation
        movlw   0x21
        movwf   dsplyCRC+1      ; seed lo for CRC calculation
        LFSR    FSR1,TXBUF2     ; point at transmit buffer
        movlw   0x1F            ; command to send data to LCD
        movwf   TXBUF2          ; insert command byte from us to
                                ; CFA-635
        BMD1    movlw .2
        ddwf    SendCount,w     ; + overhead
        call    ComputeCRC2     ; compute CRC of transmit message
        movf    dsplyCRC+1,w
        movwf   POSTINC1        ; append CRC byte
        movf    dsplyCRC,w
        movwf   POSTINC1        ; append CRC byte
        return
;=====================================================================
SendMsg:
        call    ReleaseFSR1
        LFSR    FSR0,TXBUF2
        movff   FSR0H,irptFSR0
        movff   FSR0L,irptFSR0+1
                                ; save interrupt use of FSR0
        movff   SendCount,TXBUSY2
        bsf     PIE2,TX2IE
                                ; set transmit interrupt enable
                                ; (bit 4)
        return
;=====================================================================
; macro to move string to transmit buffer
SHOW macro src, stringname
        call    src
        MOVLF   upper stringname, TBLPTRU
        MOVLF   high stringname, TBLPTRH
        MOVLF   low stringname, TBLPTRL
        call    MOVE_STR
        endm
;=====================================================================
MOVE_STR:
        tblrd   *+
        movf    TABLAT,w
        bz      ms1b
        movwf   POSTINC1
        incf    SendCount
        goto    MOVE_STR
ms1b:
        return
;=====================================================================
```