Crystalfontz

INTELLIGENT LCD MODULE SPECIFICATIONS

XES635BK-TFK-KU
XES635BK-TML-KU
XES635BK-YYK-KU

Hardware Version: v1.5
Firmware Version: u2.5
Enclosure Version: v4.0

Datasheet Release: 2018-12-12

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

Table of Contents

Table of Figures

1. General Information

Datasheet Revision History

Datasheet Version: **2018-12-12**
Hardware Version: **v1.5**
Firmware Version: **u2.5**
Enclosure Version: v4.0

For information about firmware and hardware revisions, see the Part Change Notifications (PCNs) under "News" in our website's navigation bar. To see the most recent PCN for the CFA635 family at the time of this datasheet release, see PCN #11023.

Previous datasheet Version: **2017-11-23**

For reference, previous datasheets may be downloaded by clicking the "Show Previous Versions of Datasheet" link under the "Datasheets and Files" tab of the product web page.

Product Change Notifications

To check for or subscribe to "Part Change Notices" for this display module, see the Product Notices tab on the product's webpage.

Variations

Slight variations (for example, contrast, color, or intensity) between lots are normal.

Volatility

This display module has volatile memory.

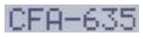
Disclaimer

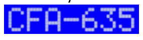
2. Introduction

The CFA635 family of modules has four interface choices:

- CFA635-xxx-KL (logic-level serial / UART)
- CFA635-xxx-KS (CFA-RS232)
- CFA635-xxx-KU (USB)
- XES635BK-xxx-KU (enclosed USB)

This datasheet has information for these interface modules:

- XES635BK -TFK-KU 

(dark letters on a light background; this display can be read in normal office lighting, in dark areas, and in bright sunlight)
- XES635BK -TML-KU 

(light letters on a blue background; this display can be read in normal office lighting and in dark areas)
- XES635BK -YYK-KU

(dark letters on a yellow background; this display can be read in normal office lighting, in dark areas, and in bright sunlight)

Main Features:

- Large, easy-to-read, 20-character x 4-line LCD in a compact overall size.
- Fits nicely in a 1U rack mount case (37 mm overall height).
- May be installed in a standard half-height 51/4 drive bay by using our optional drive bay mounting bracket or our optional SLED bracket. The SLED holds the CFA-635 display module, an optional FBSCAB and has mounting points for a standard 3.5-inch hard disk drive.
- The LCD has a wide viewing angle, with a 12 o'clock preferred viewing direction.
- USB 2.0 full-speed interface.
- Six-button, LED backlit, translucent silicone keypad with screened legend. Fully decoded keypad: any key combination is valid and unique.
- LCD is edge-lit with 8 long-life, high performance, LEDs (4 per side).
- Adjustable contrast. The default contrast value for the module will be acceptable for most applications. If necessary, you can adjust the contrast using command 13 (0x0D): Set LCD Contrast.
- The front of the display has four bicolor (red + green), LED status lights. The LEDs' brightness can be set by the host software that allows for smoothly adjusting the LEDs to produce other colors (for example, yellow, and orange).
- Robust, packet-based protocol with 16-bit CRC ensures error-free communications.
- Nonvolatile memory capability (EEPROM):
  - Customize the "power-on" display settings (backlight brightness, boot screen, LED settings).
  - 16-byte "scratch" register for storing IP address, netmask, system serial number.
- Crystalfontz America, Inc. is ISO 9001:2008 certified.
- A Declaration for Conformity, RoHS, and REACH:SVHC are available under the Datasheets & Files tab on display web pages.

2.1. Module Classification Information

XES	635	BK	-	x	x	x	-	K	U
1	2	3		4	5	6		7	8

1	**Brand**	XES – e**X**ternal **E**nclosure, **S**teel
2	**Model Identifier**	635
3	**Bracket Type**	BK = Black Steel
4	**Backlight Type & Color**	T – LED, white Y – LED, yellow-green
5	**Fluid Type, Image (positive or negative) & LCD Glass Color**	F – FSTN, positive, neutral M – STN, negative blue Y – STN, positive, yellow-green
6	**Polarizer Film Type, Temperature Range & View Angle (O 'Clock)**	K – Transflective, WT, 12:00 L – Transmissive, WT, 12:00
7	**Special Code**	K – Manufacturer's code
8	**Interface**	U – USB interface S – RS232 full-swing interface (uses CFA-RS232 level translator) L – Logic-level inverted serial interface

[1]When you order a CFA635 through our website, you may be offered a choice of configurations (including accessories), to add to your order through our "Customize and Add to Cart" feature.

2.2. Ordering Information

Part Number	Fluid	LCD Glass Color	Image	Polarizer Film	Backlight Color/Type
XES635BK-TFK-KU	FSTN	neutral	positive	transflective	Backlight: white Keypad: white
XES635BK-TML-KU	STN	blue	negative	transmissive	Backlight: white Keypad: blue
XES635BK-YYK-KU	STN	yellow-green	positive	transflective	Backlight: yellow-green Keypad: yellow-green

Modules in the CFA635 family are:

- A USB interface module. Part numbers end in "-KU".
- A serial interface using a CFA-RS232 level translator board. Part numbers end in "-KS". Suitable for embedded controller or host system that has a "real" RS232 serial port (-5v to +5v "full swing" serial interface).
- A serial "logic level, inverted" 0v to +3.3v nominal interface (typical for direct connection to a microcontroller's UART pins). Part numbers end in "-KL".
- An external enclosure with a captive USB "A" cable connection. Please see https://www.crystalfontz.com/family/XES635BK?family=XES635BK.

3. Mechanical Characteristics

3.1. Physical Characteristics

Item	Specification (mm)	Specification (inch, reference)
Overall Width and Height	146.0 (W) x 39.3 (H)	5.59 (W) x 1.55 (H)
Viewing Area 39.3	80.90 (W) x 25.5 (H)	3.19 (W) x 1.00 (H)
Active Area	77.95 (W) x 22.35 (H)	3.07 (W) x 0.88 (H)
5x7 Standard Character Size	3.20 (W) x 4.85 (H)	0.126 (W) x 0.190 (H)
Pixel Size	0.60 (W) x 0.65 (H)	0.024 (W) x 0.026 (H)
Pixel Pitch	0.65 (W) x 0.70 (H)	0.026 (W) x 0.028 (H)
Depth with Keypad	23.6 (D)	0.93 (D)
Keystroke Travel (approximate)	~2.4	~0.1
Weight with Cable (typical)	297 grams	10.48 ounces

3.2. Optical Characteristics

Item	Symbol	Condition	Min	Typ	Max	Direction
Viewing Angle (12 o'clock is the preferred direction for this module)	θ	CR≧2	40°	—	—	above, 12 o'clock
	θ	CR≧2	20°	—	—	below, 6 o'clock
	θ	CR≧2	30°	—	—	right, 3 o'clock
	θ	CR≧2	30°	—	—	left, 9 o'clock
Contrast Ratio	CR	—	—	10	15	—
Response Time	T rise	—	—	80	160	ms
	T fall	—	—	100	200	ms

3.3. LED Backlight Information

The backlights used in the CFA635 are designed for a very long life, but their lifetime is finite. To conserve the LED lifetime and reduce power consumption you can dim or turn off the backlights during periods of inactivity.

4. Electrical Specifications

4.1. System Block Diagram

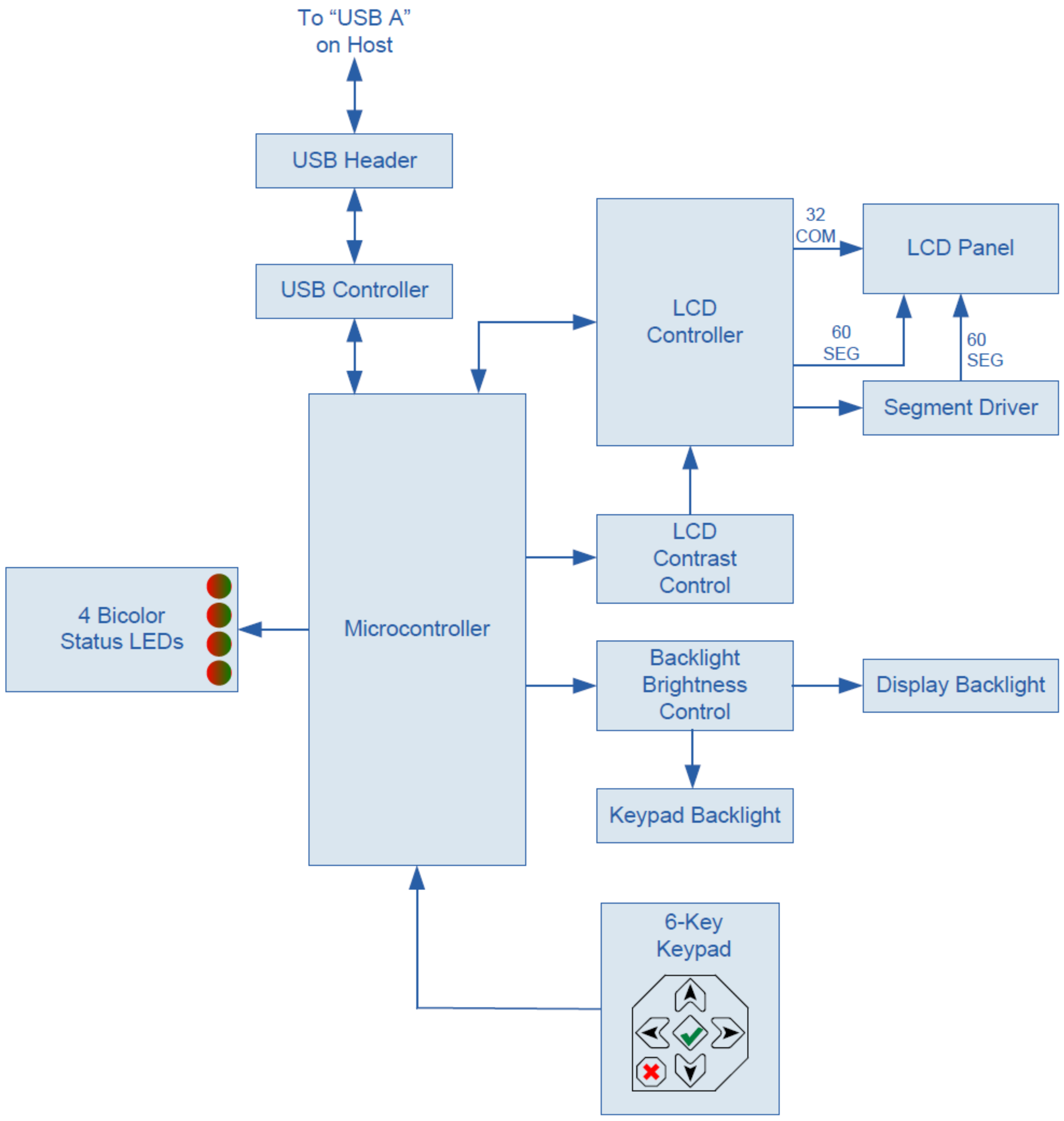


Figure 5. System Block Diagram

4.2. Absolute Maximum Ratings

Absolute Maximum Ratings	Symbol	Minimum	Maximum
Operating Temperature	T_{OP}	-20°C	+70°C
Storage Temperature	T_{ST}	-30°C	+80°C
Humidity Range (Non-condensing)	RH	10%	90%
Supply Voltage for Logic	V_{DD}	-0.3v	+5.5v
Please note that these are stress ratings only. Extended exposure to the absolute maximum ratings listed above may affect device reliability or cause permanent damage. Functional operation of the module at these conditions beyond those listed under DC Characteristics is not implied. Changes in temperature can result in changes in contrast.			

4.3. DC Characteristics

Specifications	Symbol	Minimum	Typical	Maximum
Supply Voltage	V_{DD}	+3.0v	+5.0v (USB)	+5.5v

4.4. Typical Current Consumption

The **XES635BK-xxx-KU** modules are powered by USB and will consume less than the 500mA that is available on a standard USB port.

4.5. GPIO Current Limits

Typical GPIO Current Limits	Specification
Sink	25mA
Source	25mA

4.6. Fan Criteria (Using Optional FBSCAB)

Fan Criteria	Specification
Fan Tachometer Speed Range (assuming two PPR[1])	600 RPM to 3,000,000 RPM
Fan Power Control PWM[2] Frequency	18 Hz nominal
[1]PPR is Pulses Per Revolution, can also written as p/r. [2]PWM is Pulse Width Modulation.	

4.7. USB ESD Characteristics

The D+ and D- pins of the USB connector have IEC 61000-4-2 level 4 compliant ESD Protection:

- 15 kV (air discharge)
- 8 kV (contact discharge)

The remainder of the CFA635 circuitry is industry standard CMOS logic and susceptible to ESD damage. Please use industry standard antistatic precautions as you would for any other static sensitive devices such as expansion cards, motherboards, or integrated circuits. Ground your body, work surfaces, and equipment.

5. Host Communications

5.1. Introduction

XES635BK-xxx-KU communicates with its host using the USB interface through the virtual COM port (VCP) drivers. Using the driver makes it appear to the host software as if there is an additional serial port (the VCP), on the host system when the XES635BK-xxx-KU is connected.

5.2. Packet Structure

All communication between the XES635 and the host takes place in the form of a simple, robust CRC checked packet. The packet format allows for very reliable communications between the XES635 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.

Reconciling packets is recommended rather than using delays when communicating with the module. To reconcile your packets, please ensure that you have received the acknowledgement packet before sending any additional packets.

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 XES635
             01 = normal response from XES635 to host
             10 = normal report from XES635 to host (not in
                  direct response to a command from the host)
             11 = error response from XES635 to host (a packet
                  with valid structure but illegal content
                  was received by the XES635)
```

`<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 typeof packet will have a specified **`<data_length>`** and format for **`<data>`** as well as algorithms for decoding datadetailed 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 A: Demonstration Software and Sample Code 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;
```

Crystalfontz supplies a demonstration and test program, cfTest, that can be used to experiment with and test the XES635’s operation. We also offer 635WinTest, which is a simpler, open-source program. Included in the 635WinTest source is a CRC algorithm and an algorithm that detects and reconciles 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.

5.3. About Handshaking

The nature of XES635's packets makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for a corresponding acknowledge packet from the XES635 before sending the next command packet. The XES635 will respond to all packets within 250 mS. The host software should stop waiting and retry the packet if the XES635 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 XES635 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 XES635 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 baud rate and the reporting configuration of the XES635. For any modern PC 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.

5.4. Report Codes

The XES635 can be configured to report three items. The XES635 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, Key Activity, Fan Speed Report and Temperature Sensor Report.

0x80: Key Activity

If a key is pressed or released, the XES635 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.

Report packet format:

```
type = 0x80
data_length = 1
data[0] = 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
```

0x81: Not Supported

0x82: Not Supported

5.5. Command Codes

Below is a list of valid commands for the XES635. 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 XES635 will return the Ping Command to the host.

Request packet format:

```
type: 0x00 = 0₁₀
data_length = 0 to 16
data[] = up to 16 bytes of arbitrary data
```

Successful return packet format:

```
type: 0x40 | 0x00 = 0x40 = 64₁₀
data_length = (identical to received packet)
data[] = (identical to received packet)
```

1 (0x01): Get Hardware & Firmware Version

The XES635 will return the hardware and firmware version information to the host.

Request packet format:

```
type: 0x01 = 1₁₀
data_length = 0
```

Successful return packet format:

```
type: 0x40 | 0x01 = 0x41 = 65₁₀
data_length = 16
data[] = "XES635:h1.5,u2.1″
```

2 (0x02): Write User Flash Area

The XES635 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.

Request packet format:

```
type: 0x02 = 2₁₀
data_length = 16
data[] = 16 bytes of arbitrary user data to be stored in the XES635's non-
volatile memory
```

Successful return packet format:

```
type: 0x40 | 0x02 = 0x42 = 66₁₀
data_length = 0
```

3 (0x03): Read User Flash Area

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

Request packet format:

```
type: 0x03 = 3₁₀
```

```
data_length = 0
```

Successful return packet format:

```
type: 0x40 | 0x03 = 0x43 = 67₁₀
data_length = 16
data[] = 16 bytes user data recalled from the XES635's non-volatile memory
```

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

The XES635 loads its power-up configuration from nonvolatile memory when power is applied. The XES635 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.
  - Command 31 (0x1F): Send Data to LCD.
- Special character font definitions (Command 9 (0x09): Set LCD Special Character Data).
- Cursor position (Command 11 (0x0B): Set LCD Cursor Position).
- Cursor style (Command 12 (0x0C): Set LCD Cursor Style).
- Contrast setting (Command 13 (0x0D): Set LCD Contrast).
- Backlight setting (Command 14 (0x0E): Set LCD & Keypad Backlight).
- Fan power settings (Command 17 (0x11): Set Fan Power).
- Key press and release masks (Command 23 (0x17): Configure Key Reporting).

Request packet format:

```
type: 0x04 = 4₁₀
data_length = 0
```

Successful return packet format:

```
type: 0x40 | 0x04 = 0x44 = 68₁₀
data_length = 0
```

If the current state and the boot state do not match after saving, the module will return an error instead of an ACK. In this unlikely error case, the boot state will be undefined.

5 (0x05): Reboot XES635

This command instructs the XES635 to simulate a power-on restart of itself..

Request packet format:

```
    type: 0x05 = 510
    data_length = 3
    data[0] = 8
    data[1] = 18
    data[2] = 99
```

In all of the above cases, Successful return packet format:

```
type: 0x40 | 0x05 = 0x45 = 69₁₀
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.

The LCD contents is one of the items stored by the command 4 (0x04): Store Current State as Boot State.

Request packet format:

```
type: 0x06 = 6₁₀
data_length = 0
```

Successful return packet format:

```
type: 0x40 | 0x06 = 0x46 = 70₁₀
data_length = 0
```

9 (0x09): Set LCD Special Character Data

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

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

Request packet format:

```
type: 0x09 = 9₁₀
data_length = 9
data[0] = index of special character to modify (0-7 valid)
data[1-8] = bitmap of the new font for this character
   data[1]is at the top of the cell.
   data[8]is at the bottom of the cell.
      any value is valid between 0 and 63.
      the msb is at the left of the character cell
      lsb is at the right of the character cell.
```

Successful return packet format:

```
type: 0x40 | 0x09 = 0x49 = 73₁₀
data_length = 0
```

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.

Note: Firmware version prior to v1.9 did not return the address code.

Request packet format:

```
type: 0x0A = 10₁₀
data_length = 1
data[0] = LCD address code of desired data
Valid LCD address codes:
   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
   (an error will be returned if address is outside of these values)
```

Successful return packet format:

```
type: 0x40 | 0x0A = 0x4A = 74₁₀
data_length = 9
data[0] requested address code.
data[1-8] requested 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 XES635'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.

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

Request packet format:

```
type: 0x0B = 11₁₀
data_length = 2
data[0] = column (0-19 valid)
data[1] = row (0-3 valid)
```

Successful return packet format:

```
type: 0x40 | 0x0B = 0x4B = 75₁₀
data_length = 0
```

12 (0x0C): Set LCD Cursor Style

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

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

Note: cursor style 3 behavior is different from previous XES635 versioned firmware v1.6 and earlier.

Request packet format:

```
type: 0x0C = 12₁₀
data_length = 1
data[0] = cursor style (0-4 valid)
   0 = no cursor.
   1 = blinking block cursor.
   2 = static underscore cursor
   3 = blinking underscore cursor
```

Successful return packet format:

```
type: 0x40 | 0x0C = 0x4C = 76₁₀
data_length = 0
```

13 (0x0D): Set LCD Contrast

This command sets the contrast or vertical viewing angle of the display. Initiated by the host, responded to by the XES635.

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

Request packet format:

```
type: 0x0D = 13₁₀
data_length = 1
data[0] = contrast setting (0-254 valid)
   60 = light
   120 = about right
   150 = dark
   151-254 = very dark (may be useful at cold temperatures)
```

Successful return packet format:

```
type = 0x40 | 0x0D = 0x4D = 77₁₀
data_length = 0
```

14 (0x0E): Display & Keypad Backlights

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

If one byte is supplied, both the keypad and LCD backlights are set to that brightness.

Request packet format:

```
type: 0x0E = 14₁₀
data_length = 1
data[0] = keypad and LCD backlight power setting (0-100 valid)
   0 = off
   1-100 = variable brightness
```

If two bytes are supplied, the LCD is set to the brightness of the first byte, the keypad is set to the brightness of the second byte.

Request packet format:

```
type: 0x0E = 14₁₀
data_length = 2
data[0] = LCD backlight power setting (0-100 valid)
   0 = off
   1-100 = variable brightness
data[1] = keypad backlight power setting (0-100 valid)
   0 = off
   1-100 = variable brightness
```

The return packet for both of the above options will be:

```
type: 0x40 | 0x0E = 0x4E = 78₁₀
data_length: 0
```

16 (0x10): Not Supported

18 (0x12): Not Supported

19 (0x13): Not Supported

20 (0x14): Not Supported

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

The controller on the XES635BK-xxx-KU is HD44780 compatible. Generally, you won’t need low-level access to the LCD controller but some arcane functions of the HD44780 are not exposed by the XES635’s command set. This command allows you to access the XES635’s LCD controller directly.

Note: It is possible to corrupt the XES635 display using this command.

Request packet format:

```
type: 0x16 = 22₁₀
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
```

Successful return packet format:

```
type: 0x40 | 0x16 = 0x56 = 86₁₀
data_length = 0
```

23 (0x17): Configure Key Reporting

By default, the XES635 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.

Bitmask options:

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

Request packet format:

```
type: 0x17 = 23₁₀
data_length = 2
data[0] = press mask
data[1] = release mask (0 to 63 valid)
```

Successful return packet format:

```
type: 0x40 | 0x17 = 0x57 = 87₁₀
data_length = 0
```

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the XES635BK-xxx-KU 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. 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.

Bitmask options:

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

Request packet format:

```
type: 0x18 = 24₁₀
data_length = 0
```

Successful return packet format:

```
type: 0x40 | 0x18 = 0x58 = 88₁₀
data_length = 3
data[0] = bit mask of keys currently pressed
data[1] = bit mask of keys pressed since the last poll
data[2] = bit mask of keys released since the last poll
```

25 (0x19): Not Supported

26 (0x1A): Not Supported

27 (0x1B): Not Supported

28 (0x1C): Not Supported

29 (0x1D): Not Supported

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. The combination of XES635BK-xxx-KU+FBSCAB+WR-DOW-Y17 temperature sensors is required to report the temperature information The combination of the XES635-xxx-KU+FBSCAB+WR-FAN-X01 cable is required to control fans.

Request packet format:

```
type = 0x1E = 30₁₀
data_length = 0
```

Successful return packet format:

```
type = 0x40 | 0x1E = 0x5E = 94₁₀
data_length = 15
data[ 0] = not relevant
data[ 1] = not relevant
data[ 2] = not relevant
data[ 3] = not relevant
data[ 4] = not relevant
data[ 5] = key presses (as set by command 23)
data[ 6] = key releases (as set by command 23)
data[ 7] = not relevant
data[ 8] = not relevant
data[ 9] = not relevant
data[10] = not relevant
data[11] = not relevant
data[12] = not relevant
data[13] = contrast setting (as set by command 13)
data[14] = backlight setting (as set by command 14)
```

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.

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

Request packet format:

```
type: 0x1F = 31₁₀
data_length = 3 to 22
data[0]: col = x = 0 to 19
data[1]: row = y = 0 to 3
data[2-21]: characters/text to place on the LCD
```

Successful return packet format:

```
type: 0x40 | 0x1F = 0x5F = 95₁₀
data_length = 0
```

33 (0x21): Set Baud Rate (deprecated on USB)

This command has no effect on the XES635BK-xxx-KU module. The module will return an acknowledge for compatibility with older versions of host software.

34 (0x22): Not Supported

35 (0x23): Not Supported

6. 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 Greek letter "β" is in the column labeled “224d” and in the row labeled “2d”.

Add 224 + 2 to get 226. When you send a byte with the value of 226 to the display, the Greek letter "β" will be shown.

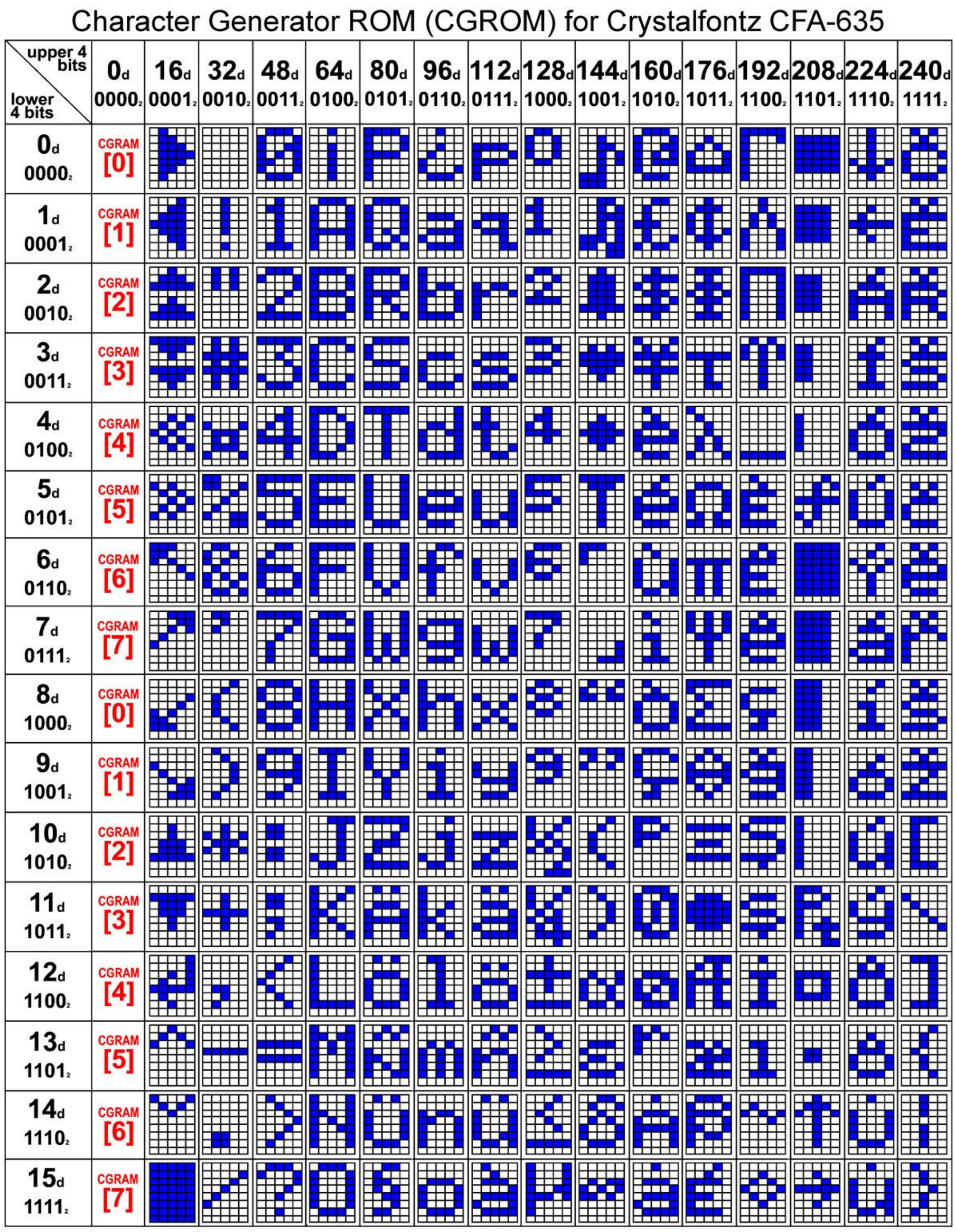

Character Generator ROM (CGROM) for Crystalfontz CFA-635
upper 4 bits
lower 4 bits
0d 0000b
16d 0001b
32d 0010b
48d 0011b
64d 0100b
80d 0101b
96d 0110b
112d 0111b
128d 1000b
144d 1001b
160d 1010b
176d 1011b
192d 1100b
208d 1101b
224d 1110b
240d 1111b
0d 0000b
1d 0001b
2d 0010b
3d 0011b
4d 0100b
5d 0101b
6d 0110b
7d 0111b
8d 1000b
9d 1001b
10d 1010b
11d 1011b
12d 1100b
13d 1101b
14d 1110b
15d 1111b
CGRAM [0]
CGRAM [1]
CGRAM [2]
CGRAM [3]
CGRAM [4]
CGRAM [5]
CGRAM [6]
CGRAM [7]
CGRAM [0]
CGRAM [1]
CGRAM [2]
CGRAM [3]
CGRAM [4]
CGRAM [5]
CGRAM [6]
CGRAM [7]

Figure 10: Character Generator ROM (CGROM)

7. LCD Module Reliability and Longevity

We work to continuously improve our products, including backlights that are brighter and last longer. Slight color variations from module to module and batch to batch are normal.

If you need modules with consistent color, please ask for a custom order.

ITEM	SPECIFICATION	
LCD portion (excluding Keypad and Backlights)	50,000 to 100,000 hours (typical)	
Keypad	1,000,000 keystrokes	
Bicolor status LEDs	50,000 to 100,000 hours	
Yellow-green LED Display and Keypad Backlight (XES635BK-YYK-Kx)	50,000 to 100,000 hours	
White LED Display and Blue LED Keypad Backlights NOTE*: We recommend that the backlight of the white LED backlit modules be dimmed or turned off during periods of inactivity to conserve the white LED backlight lifetime.* Values listed above are approximate and represent typical lifetime.	Power-On Hours	% of Initial Brightness
	<10,000	>70%
	<50,000	>50%

7.1. Module Longevity (EOL / Replacement Policy)

Crystalfontz is committed to making all of our LCD modules available for as long as possible. For each module that 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 datasheet 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 Notices.

8. Care and Handling Precautions

For optimum operation of the XES635-XXX-KU and to prolong its life, please follow the precautions described below.

8.1. ESD (Electrostatic Discharge)

The USB D+ & D- lines have enhanced ESD protection following industry standard practice, please see USB ESD Characteristics.

The remainder of this circuitry is industry standard CMOS logic and susceptible to ESD damage. Please use industry standard antistatic precautions as you would for any other static sensitive devices such as expansion cards, motherboards, or integrated circuits. Ground your body, work surfaces, and equipment.

8.2. Design and Mounting

- Do not remove the module from the case.
- Do not disassemble or modify the module.

8.3. Mechanical Shock, Impact, Torque, or Tension

- Do not expose the XES635 to strong mechanical shock, impact, torque, or tension.
- Do not drop, toss, bend, or twist the XES635.
- Do not place weight or pressure on the XES635.

8.4. LCD Panel Breakage

- If the LCD panel 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 using warm soapy water.

8.5. Cleaning

- The window gasket is made of soft plastic that can easily be scratched or damaged, so use extra care when you clean it.
- Do not clean the window gasket with liquids.
- Do not wipe the window gasket with any type of cloth or swab (for example, Q-tips).
- Use the removable protective film to remove smudges (for example, fingerprints), and any foreign matter. If you no longer have the protective film, use standard transparent office tape (for example, Scotch® brand “Crystal Clear Tape”).
- If the window gasket becomes dusty, carefully blow it off with clean, dry, oil-free compressed air.
- The window gasket will eventually become hazy if you do not use care when cleaning it.
- Contact with moisture may permanently spot or stain the window gasket.

8.6. Operation

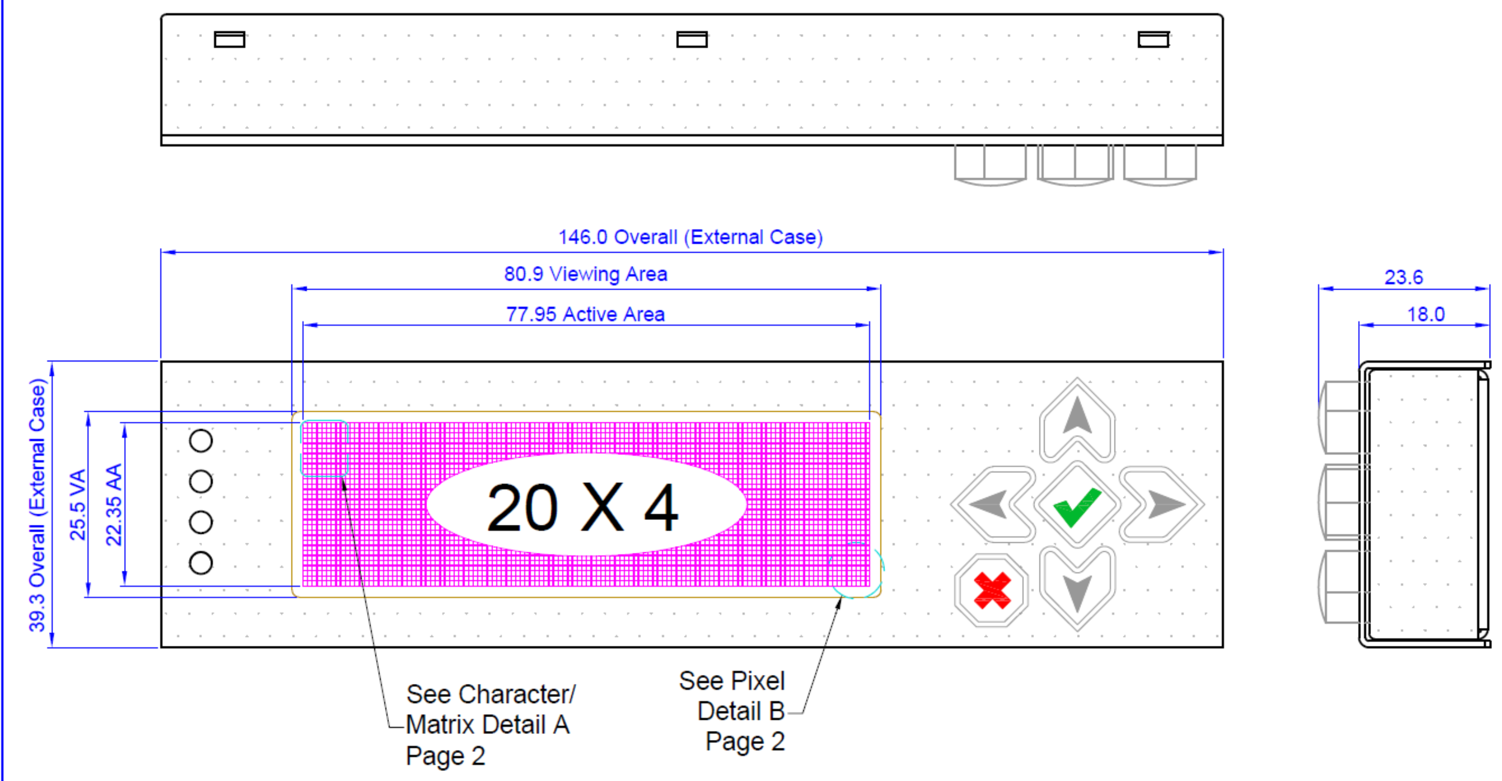
- Protect the XES635 from ESD and power supply transients.
- Observe the operating temperature limitations: a minimum of -20°C to a maximum of +70°C with minimal fluctuation. Operation outside of these limits may shorten life and/or harm display.
- At lower temperatures of this range, response time is delayed.
- At higher temperatures of this range, display becomes dark. (You may need to adjust the contrast.)
- Operate away from dust, moisture, and direct sunlight.
- 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 backlight lifetime.

8.7. Storage and Recycling

- Store in an ESD-approved container away from dust, moisture, and direct sunlight.
- Observe the storage temperature limitations: -30°C minimum, +80°C maximum with minimal fluctuation. Rapid temperature changes can cause moisture to form, resulting in permanent damage.
- Do not allow weight to be placed on the XES635 while in storage.
- Please recycle your outdated Crystalfontz modules at an approved facility.

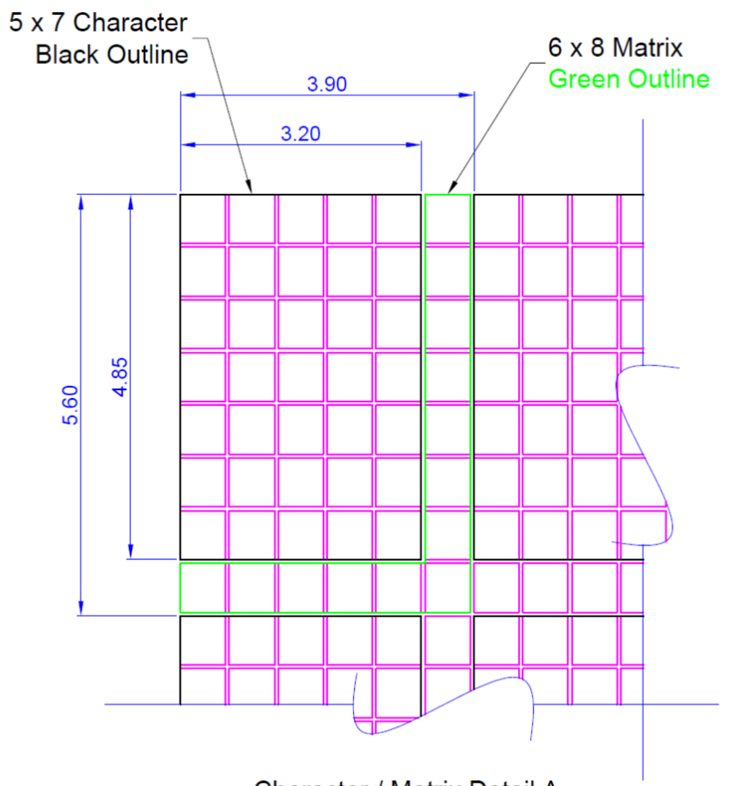
9. Mechanical Drawings

9.1. Module Outline Drawings

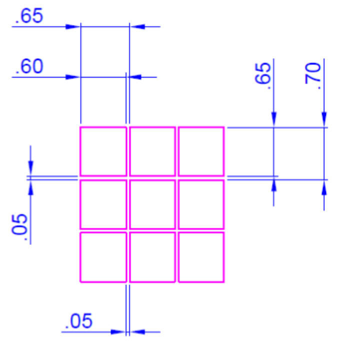

146.0 Overall (External Case)
80.9 Viewing Area
77.95 Active Area
23.6
18.0
39.3 Overall (External Case)
25.5 VA
22.35 AA
20 X 4
See Character/ Matrix Detail A Page 2
See Pixel Detail B Page 2

Tolerance is ±0.5 mm unless specified.

Copyright © 2017 by Crystalfontz America, Inc. www.crystalfontz.com/products/	*Part No.(s):* XES635BK-TFE-KU XES635BK-TMF-KU XES635BK-YYE-KU	*Scale:* Not to scale	*Drawing Number:* XES635BK_master
		Units: Millimeters	*Date:* 2017-05-03 / *Sheet:* 1 of 2



Character / Matrix Detail A

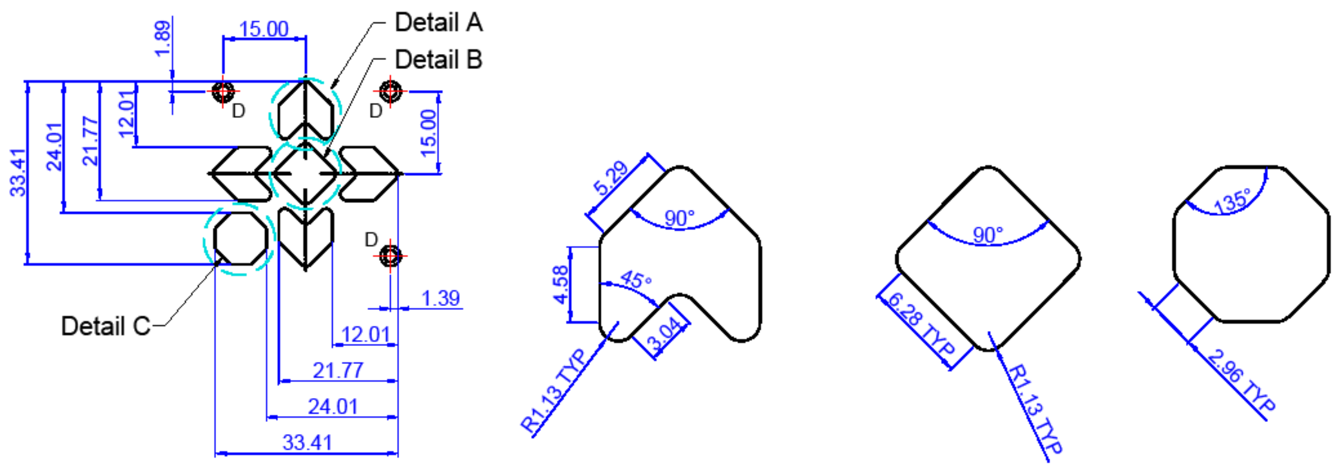


Pixel Detail B

Tolerance is ±0.5 mm unless specified.

Copyright © 2017 by Crystalfontz America, Inc. www.crystalfontz.com/products/	Part No.(s): XES635BK-TFE-KU XES635BK-TMF-KU XES635BK-YYE-KU	Scale: Not to scale	Drawing Number: XES635BK_master
		Units: Millimeters	Date: 2017-05-03 — Sheet: 2 of 2

9.2. Panel Mounting Application Cutout Drawing



Keypad Cutout Detail

Detail A

Detail B

Detail C

Typical mounting hardware at locations "D" (5 places):

- PEM FH-256-8
- Bivar Inc. 9913-5 mm spacer
- 2-56 "small profile" hex nut
- Use appropriate screen printed overlay to cover display bezel and mounting hardware, and to protect LCD from scratching.

Copyright © 2017 by Crystalfontz America, Inc. www.crystalfontz.com/products/	*Part No.(s):* 6-Button Keypad Panel Mounting Application Cutout	*Scale:* Not to scale	*Drawing Number:* 6-Button_Panel_Cutout_Master
		Units: Millimeters	*Date:* 2017-03-27 — *Sheet:* 1 of 1

10. Appendix A: Example Software and Sample Source Code

10.1. Example Software

- Crystalfontz cfTest (Windows compatible test/demonstration software): https://www.crystalfontz.com/product/cftest
- Crystalfontz WinTest (Windows compatible example program and source): https://www.crystalfontz.com/product/635wintest
- Linux compatible command-line demonstration and source: https://www.crystalfontz.com/product/linuxexamplecode
- Crystalfontz CrystalControl2 (Windows compatible LCD display software): https://www.crystalfontz.com/product/CrystalControl2.html
- LCDProc (Linux compatible open-source LCD display software):

http://lcdproc.org/

10.2. Algorithms to Calculate the CRC

Below are eight sample algorithms that will calculate the CRC of a XES635 packet. Some of the algorithms were contributed by forum members and originally written for CFA631 and XES635. The CRC used in the XES635 is the same as that used in IrDA, which came from PPP, which seems to be related to a CCITT standard (ref: Network Working Group Request for Comments: 1171). 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/IrLAP
11_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 = 0xFFFF;
  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;
   ubyte data;
   int bitcount;

   //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(bitcount = 0; bitcount <= 7; bitcount++)
      {
         //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(bitcount = 0; bitcount <= 15; bitcount++)
   {
      //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 XES635 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 XES635 packet) data. This routine was adapted from the C table implementation. The complete project can be found in our forums.

```
'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 633_WinTest:
'http://www.crystalfontz.com/products/633/633_WinTest.zip
'Full zip of the project is available in our forum:
'https://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.
Sub Initialize_CRC_Lookup_Table()
  crcLookupTable(0).Lo = &H0
  crcLookupTable(0).Hi = &H0
  . . .
  'For purposes of brevity in this Datasheet, I have removed 251 entries of this
  table, the 'full source is available in our forum:
  'https://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 by customer Virgil Stamps of ATOM Instrument Corporation for our XES635 module.

```
; CRC Algorithm for CrystalFontz XES635 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 the byte 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           ; XOR poly mask with CRC register
     xorwf          dsplyCRC,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
                              ; XES635
     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
;=====================================================================
```