Crystalfontz America, Incorporated

EXTERNAL USB INTELLIGENT LCD MODULE SPECIFICATIONS

Crystalfontz Model Number	**XES635BK-YYE-KU**
Hardware Version	**Revision 1.0, March 2008**
Firmware Version	**Revision 1.4, July 2005**
Data Sheet Version	**Revision 1.0, March 2008**
Product Pages	wwww.crystalfontz.com/product/XES635BKYYEKU
Customer Name	
Customer Part Number	

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 LCD MODULE	
2008/03/01	Current hardware version: **v1.0** XES635BK-YYE-KU is a CFA635-YYE-KU module enclosed in a black steel case with a permanently attached USB "A" cable.

FIRMWARE	
2005/07/01	Current firmware version: **v1.4** Command 1: Get Hardware & Firmware Version returns: “CFA635:h1.0,v1.4”

DATA SHEET	
2008/03/01	Current Data Sheet version: **v1.0**: New Data Sheet:

CONTENTS

CONTENTS, CONTINUED

LIST OF FIGURES

MAIN FEATURES

The XES635BK-YYE-KU is a Crystalfontz CFA-635-KU module enclosed in a sturdy steel case. The case is compact – only slightly larger than the bare module. The approximately 9-foot long black low-drop USB "A" cable is permanently attached. The long cable makes it easy to position the module at eye level on a desk or mount it to a wall.

- ❑ 20 characters by 4 lines LCD has a large display area in a compact 146.0 (W) x 39.3 (H) mm x 20.55 mm (D) package (5.75” (W) x 1.55” (H) x 0.81” (D)).
- ❑ USB interface (115200K baud equivalent throughput).
- ❑ Yellow-green edge LED backlit with STN yellow-green positive mode LCD (displays dark characters on yellow-green background).
- ❑ Integrated yellow-green 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.
- ❑ 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 . . .
- ❑ RoHS compliant.

MODULE CLASSIFICATION INFORMATION

XES	635	BK	-	Y	Y	E	-	KU
❶	❷	❸		❹	❺	❻		❼

❶	Family	XES – e**X**ternal **E**nclosure, **S**teel
❷	Model Identifier	635
❸	Finish	BK – black
❹	Backlight Type & Color	Y – LED, yellow-green
❺	Fluid Type, Image (positive or negative), & LCD Glass Color	Y – STN, positive, yellow-green
❻	Polarizer Film Type, Normal (NT) Temperature Range, & View Angle (O ‘Clock)	E – Transflective, NT, 12:00
❼	Special Codes	K – Manufacturer's code U – USB interface

ORDERING INFORMATION

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

MECHANICAL SPECIFICATIONS

PHYSICAL CHARACTERISTICS

ITEM	SIZE
Module Width and Height	146.0 (W) x 39.3 (H) mm x 20.55 mm (D, includes keypad)
Viewing Area	80.95 (W) x 25.5 (H) mm
Active Area	77.95 (W) x 22.35 (H) mm
Character Size	3.2 (W) x 4.85 (H) mm
Character Pitch	3.85 (W) X 5.55 (H) mm
Dot Size	0.60 (W) x 0.65 (H) mm
Dot Pitch	0.65 (W) x 0.70 (H) mm
Keystroke Travel (approximate)	2.4 mm
Weight	338 grams (typical, includes cable)

MODULE OUTLINE DRAWING

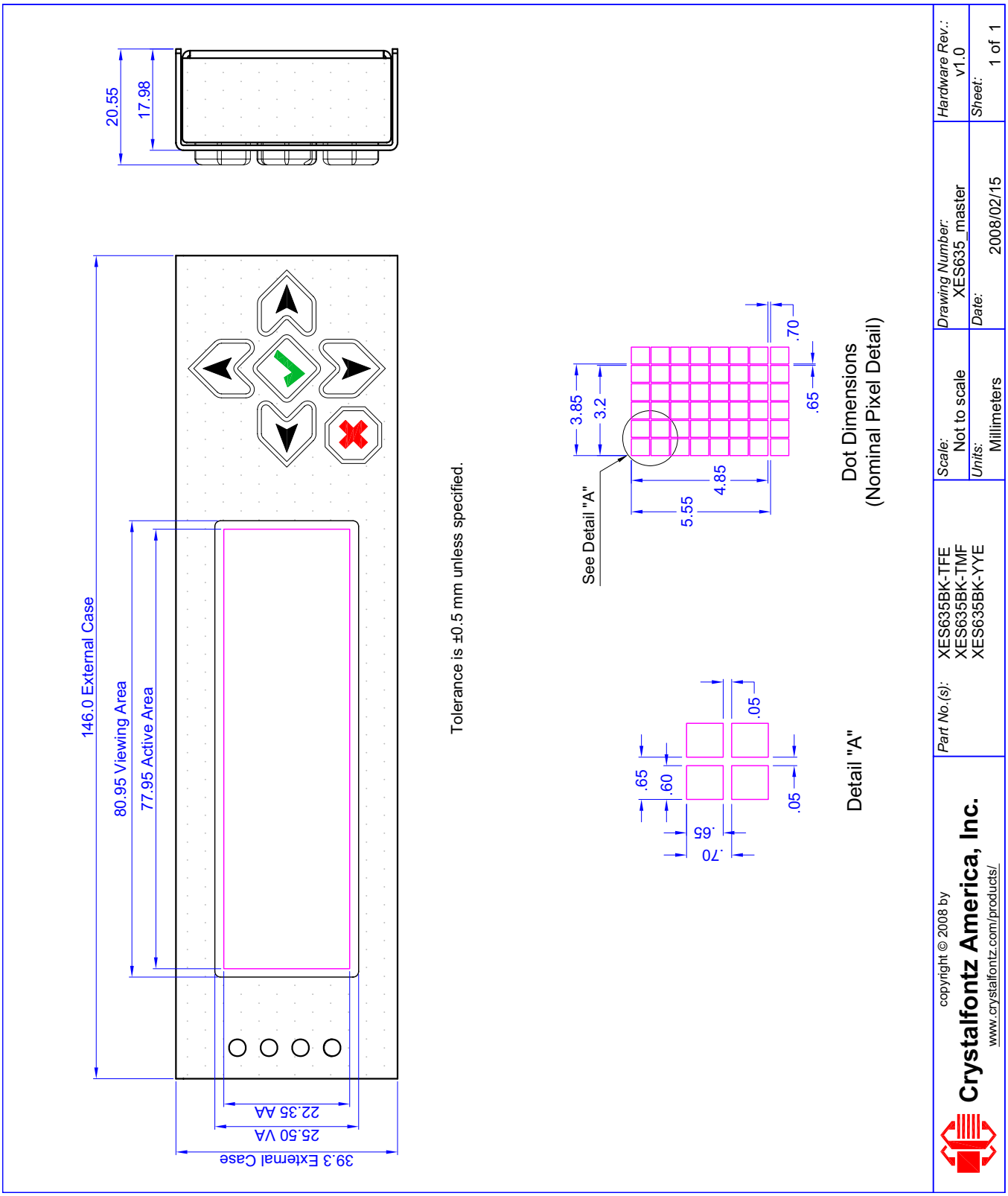


Figure 1. Module Outline Drawing

ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

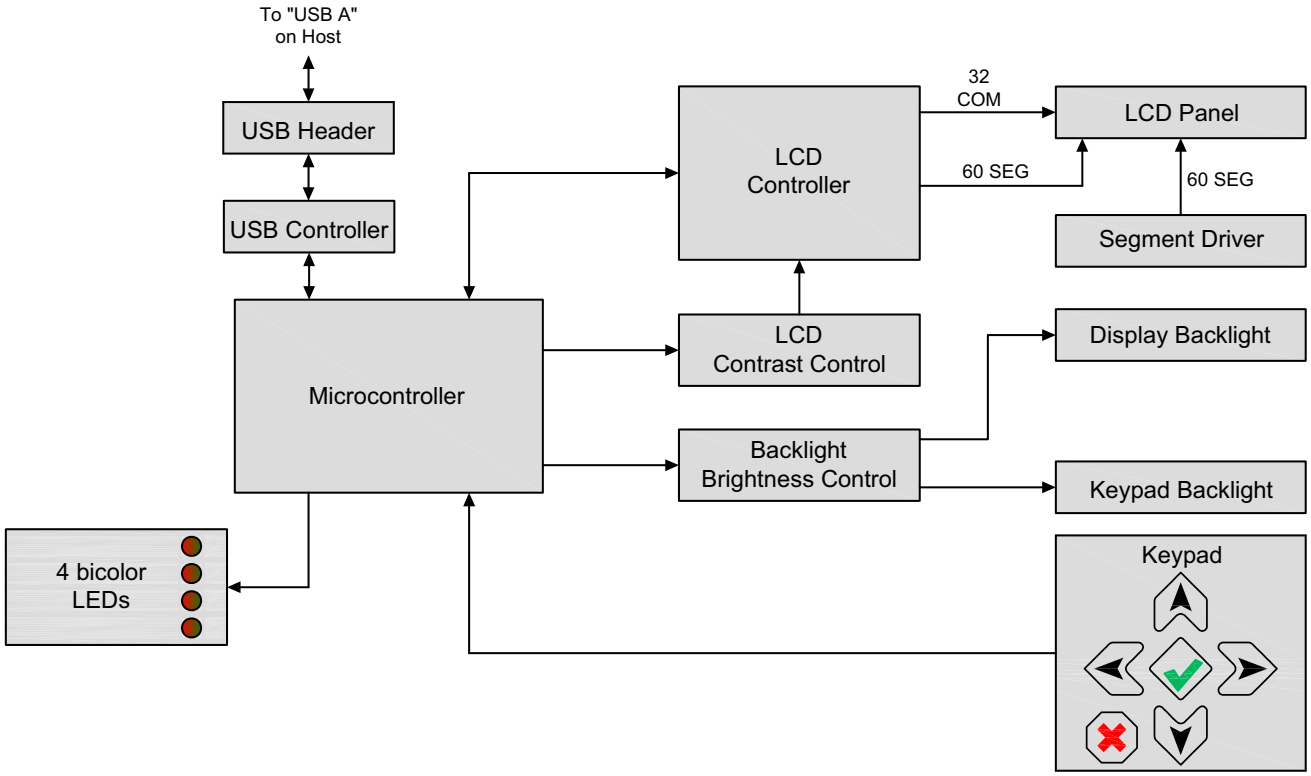


Figure 2. System Block Diagram

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: Prolonged exposure at temperatures outside of this range may cause permanent damage to the module.*			

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	-	108 mA	153 mA
X	-	X	147 mA	175 mA
X	X	X	218 mA	282 mA

BACKLIGHT PWM FREQUENCY

BACKLIGHT PWM FREQUENCY	SPECIFICATION
Backlight PWM Frequency	300 Hz nominal

HOST COMMUNICATIONS

The XES635BK-YYE-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-YYE-KU is connected. This VCP should be opened at 115200 baud, 8 data bits, no parity, 1 stop bit.

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: CALCULATING THE CRC (Pg. 28) 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 three items. 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. 17).

```
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 CFA-633. Please note that the CFA-631 will return codes 13 through 20. (See the CFA-631 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 = 0₁₀
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 = 64₁₀
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 = 1₁₀
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 0x01 = 0x41 = 65₁₀
data_length = 16
data[] = "XES635BK:hX.X,yY.Y"

X.X is the hardware revision, "1.0" for example
yY.Y is the firmware version, "v1.4" 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. 14).
  - Command 31 (0x1F): Send Data to LCD (Pg. 19).
- Special character font definitions (command 9 (0x09): Set LCD Special Character Data (Pg. 14)).
- Cursor position (command 11 (0x0B): Set LCD Cursor Position (Pg. 15)).
- Cursor style (command 12 (0x0C): Set LCD Cursor Style (Pg. 15)).
- Contrast setting (command 13 (0x0D): Set LCD Contrast (Pg. 16)).
- Backlight setting (command 14 (0x0E): Set LCD & Keypad Backlight (Pg. 16)).
- Key press and release masks (command 23 (0x17): Configure Key Reporting (Pg. 17)).
- Baud rate (command 33 (0x21): Set Baud Rate (Pg. 19)).
- The front panel LED/GPO settings (34 (0x22): Set GPO Pin (Pg. 20)).

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

Additional features/actions not supported: Reset Host, or Power Off Host.

This command instructs the XES635BK to simulate a power-on restart of itself.

Rebooting the XES635BK may be useful when testing the boot configuration. It may also be useful to re-enumerate the devices on the 1-Wire bus. 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. 13).

7 (0x07): Deprecated (See command 31 (0x1F): Send Data to LCD (Pg. 19))

8 (0x08): Deprecated (See command 31 (0x1F): Send Data to LCD (Pg. 19))

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. 13).

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. 15).

```
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. 13).

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. 13).

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 (0-255 valid)
     0-65 = very light
       66 = light
       95 = about right
      125 = dark
  126-255 = 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. 13).

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. 13).

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. 13).

```
#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. 13).

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. 17). 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), and bit 0x08 will be set if the
            watchdog is active
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. 13).

32 (0x20): Reserved for CFA-631 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. 13) 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 = 1
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. 13).

```
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
   101-255: 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

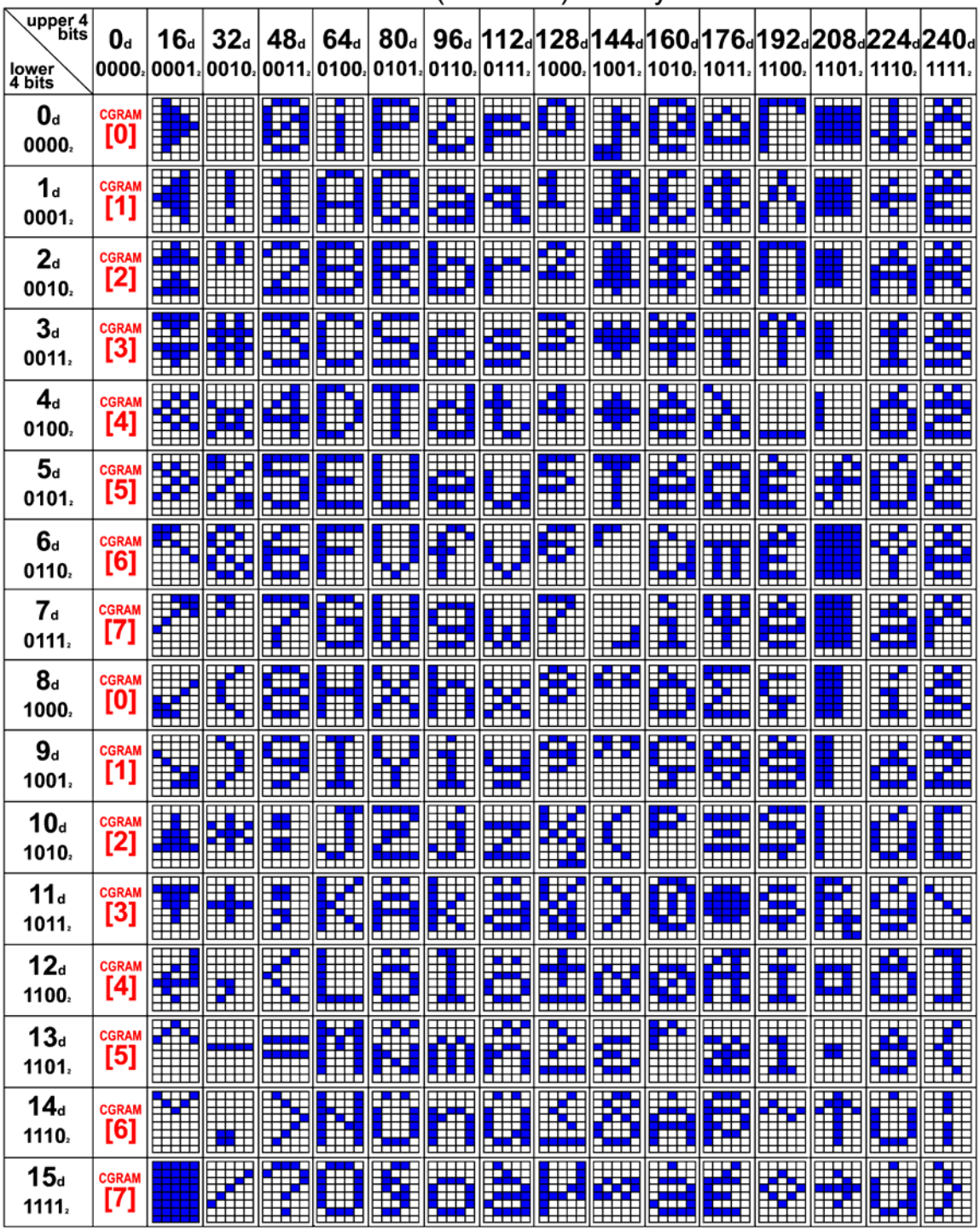


Figure 3. Character Generator ROM (CGROM)

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)
Yellow-green LED Backlights	50,000 to 100,000 hours (typical)

PRODUCT LONGEVITY

Crystalfontz is committed to making all of our LCD modules available for as long as possible. Occasionally, a supplier discontinues a component, or a process used to make the module becomes obsolete, or the process moves to a more modern manufacturing line. In order to continue making the module, we will do our best to find an acceptable replacement part or process which will make the “replacement” fit, form, and function compatible with its predecessor.

Our goal is that the modified design will not change fit, form, or function for your application. In most situations, you should not notice a difference when comparing an older module to a newer module that uses a modified replacement part or process. Sometimes, the change results in a slight variation, perhaps an improvement, over the previous design.

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

- LCD fluid, polarizers, or the LCD manufacturing process. These items may change the appearance of the display, requiring an adjustment.
- 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 V_F).
- Controller. A new controller may require you to make 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 change whenever possible; we only change a part or process if we have no other option to keep the module available. If we cannot find a way to keep a module in production, we will be forced to discontinue the module (“End of Life,” EOL) and offer a substitute of a similar existing or new module. If you must be notified that a change / EOL is to occur, please contact Crystalfontz Technical Support. Technical Support will generate a semi-custom part number that ensures you will be notified if any changes have occurred since your last order.

CARE AND HANDLING PRECAUTIONS

For optimum operation of the and to prolong the module's life, please follow the precautions described below.

ESD (ELECTRO-STATIC DISCHARGE)

The circuitry is industry standard CMOS logic and susceptible to ESD damage. Please use industry standard antistatic precautions as you would for any other PCB such as expansion cards or motherboards. Ground your body, work surfaces, and equipment.

DESIGN AND MOUNTING

- The case window is made out of plastic. It is "scratch resistant" polycarbonate but still can be scratched or damaged by abuse.
- Do not disassemble or modify the module.

AVOID SHOCK, IMPACT, TORQUE, AND TENSION

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

IF LCD PANEL BREAKS

- 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 soap and plenty of water.

CLEANING

The case window is plastic. The plastic may be scratched or damaged. Damage will be especially obvious on a "negative" module (a module that appear dark when power is "off"). Be very careful when you clean the case window..

- You may use a soft cloth moistened in standard glass cleaner (for example, Windex).
- Use standard transparent office tape or masking tape to remove smudges (for example, fingerprints) and any foreign matter.

OPERATION

- Observe the 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.)
- Do not expose to heavy dust, constant moisture, or direct sunlight.
-

STORAGE 

- Store in an ESD-approved container away from dust, moisture, and direct sunlight.

- Observe the storage temperature limitations: from -10°C minimum to +60°C maximum with minimal fluctuations. Rapid temperature changes can cause moisture to form, resulting in permanent damage.
- Do not allow weight to be placed on the modules while they are in storage.

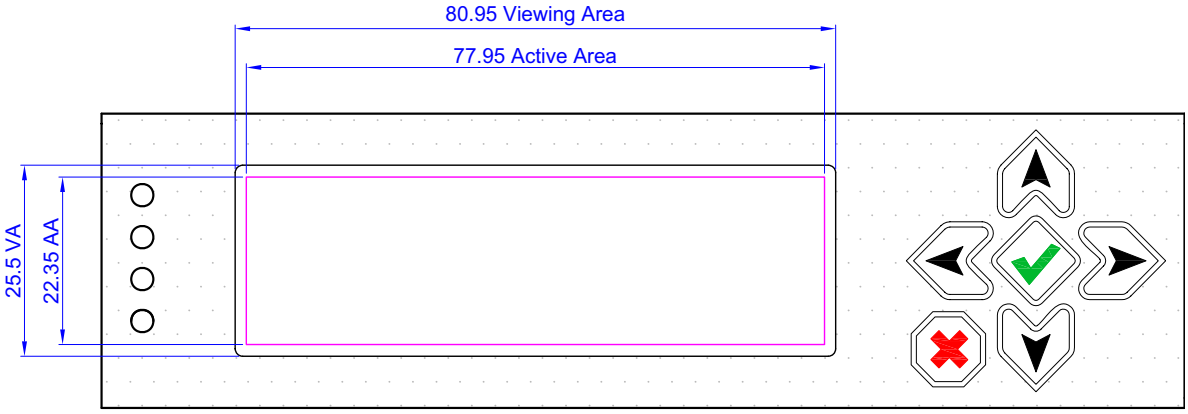
APPENDIX A: QUALITY ASSURANCE STANDARDS

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.



ACCEPTANCE SAMPLING

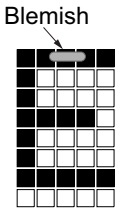
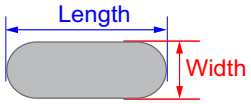
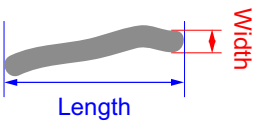
DEFECT TYPE	AQL*
Major	≤.65%
Minor	<1.0%
* Acceptable Quality Level: maximum allowable error rate or variation from standard	

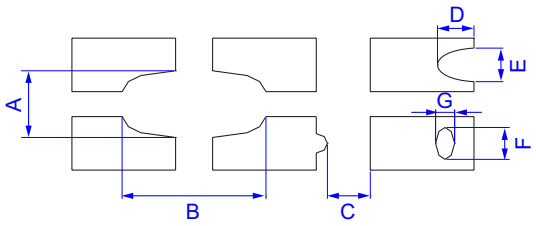
DEFECTS CLASSIFICATION

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. 25).			Major
3	Contrast adjustment defect	Contrast adjustment fails or malfunctions.			Major
4	Blemishes or foreign matter on display segments		*Defect Size*	*Acceptable Qty*	Minor
			≤0.3 mm	3	
			≤2 defects within 10 mm of each other		
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
a≤1/4W</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. (Major)
2. Color and luminance do not correspond to specifications. (Major)
3. Exceeds standards for display's blemishes or foreign matter (see test 5, page 26), and dark lines or scratches (see test 6, page 26). (Minor)</td><td>See list ←</td></tr>
<tr><td>10</td><td>PCB defects</td><td colspan="3">1. Oxidation or contamination on connectors.*
2. Wrong parts, missing parts, or parts not in specification.*
3. Jumpers set incorrectly. (Minor)
4. Solder (if any) on bezel, LED pad, zebra pad, or screw hole pad is not smooth. (Minor)
*Minor if display functions correctly. Major if the display fails.</td><td>See list ←</td></tr>
<tr><td>11</td><td>Soldering defects</td><td colspan="3">1. Unmelted solder paste.
2. Cold solder joints, missing solder connections, or oxidation.*
3. Solder bridges causing short circuits.*
4. Residue or solder balls.
5. Solder flux is black or brown.
*Minor if display functions correctly. Major if the display fails.</td><td>Minor</td></tr>
</table>

APPENDIX B: CALCULATING THE CRC

Below are five sample algorithms that will calculate the CRC of a XES635 packet. Some of the algorithms were contributed by forum members and originally written for the CFA-631 or CFA-633. The CRC used in the XES635 is the same one that is used in IrDA, which came from PPP, which to at least some extent 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.
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.

```
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 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.

```
'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/products/635/635_WinTest.zip
'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);
```