Crystalfontz America, Incorporated

INTELLIGENT SERIAL LCD MODULE SPECIFICATIONS

Crystalfontz Model Number	**CFA635-YYE-KL (Sold separately or as part of a kit.)**
Hardware Version	**Revision v1.1, July 2008**
Firmware Version	**Revision s1.6, May 2010**
Data Sheet Version	**Revision 2.1, October 2010**
Product Pages	http://www.crystalfontz.com/product/CFA635YYEKL.html

Crystalfontz America, Incorporated

12412 East Saltese Avenue
Spokane Valley, WA 99216-0357

Phone: 888-206-9720
Fax: 509-892-1203
Email: techinfo@crystalfontz.com
URL: www.crystalfontz.com

REVISION HISTORY

HARDWARE CFA635 LCD MODULE (CFA635-YYE-KL identical to USB version)	
2010/06/01	Current Hardware Revision: **v1.1 (revision number has not changed)** Changes: ● Effective June 2010, we are transitioning to an improved keypad from 10.5 mm to 12.00 mm high. See PCN 10282. ● Factories have ISO certification.
2009/06/01	Hardware Revision: v1.1 (revision number has not changed) Change: Improved bezel. Changed from nickel plated bezel to stainless steel bezel.
2008/07/01	New Hardware Revision: v1.1 Change since last revision (v1.0): Improved backlight. Display is lit more evenly, contrast can be adjusted to higher, and lifetime is longer.
2005/08/01	Hardware Version: v1.0 Start Public Version Tracking.

FIRMWARE	
2010/05/24	Current Firmware Revision: **s1.6** Changes since last revision (s1.3): ● Improved reset function to make module less sensitive to supply variations. ● Fixed range checking for command 0 (0x00): Ping Command (Pg. 29). ● Command 1 (0x01): Get Hardware & Firmware Version (Pg. 29) returns: "`CFA635:h1.1,s1.6`".
2005/06/01	Start Public Version Tracking Firmware Revision: s1.3 New firmware. (CFA635 v1.3 USB module firmware was modified for this serial module.) Command 1: Get Hardware & Firmware Version returns: "CFA635:h1.0,s1.3".

DATA SHEET	
2010/10/07	Current Data Sheet Revision: **v2.1** Changes since last revision (v2.0): ● Wherever listed or shown, improved explanation of module depth specifications with and without keypad. Module has not changed. ● In Absolute Maximum Ratings (Pg. 20), added important note on specifications. ● Slightly modified specifications in Typical Current Consumption (Pg. 20) to reflect backlight improvement made 2008/07/01. ● In H1 Pin Assignments - Includes GPIO Connections (Pg. 25) figure, corrected from “F4P” to “LCD Tx / Host Rx” and from “F4T” to “LCD Rx/ Host Tx”. Only the USB version of the CFA635 has a fourth fan. ● In APPENDIX A: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC) (Pg. 48), added Algorithm 7: For PIC18F8722 or PIC18F2685 (Pg. 57). ● Wherever needed, made minor modifications in text and illustrations to improve clarity. Includes how Hardware revisions are listed above.

DATA SHEET (Continued)	
2010/08/04	Data Sheet Revision: v2.0 Changes since last revision (v1.0) ● Wherever listed, deleted information on optional SCAB (System Cooling Accessory Board).The SCAB is not supported for this module. It is supported for the series of CFA635-xxx-KU and CFA635-xxx-KS modules. ● Added boxed note “Fine Print” at the bottom of this Revision History. Please read these disclaimers. ● Removed PDF sticky note titled “Changes” from this Revision History. The note read “2008-10-07 Corrected specification of GPIO pull-up/pull-down mode resistance values from “approximately 5Ω” to “approximately 5kΩ”. Note is no longer needed because document has been revised. ● Reorganized content, improved drawings, and edited to improve readability. ● Wherever listed, deleted dash (“-”) from module part numbers to match how they now appear on our website without the dash (“-”). ● Wherever listed, added dash (“-”) to cable part numbers to match how they now appear on our website. ● Wherever needed, added information about which optional Crystalfontz cables to use. ● In Physical Characteristics (Pg. 14), Module Outline Drawing (Pg. 15), Jumper Locations and Functions (Pg. 16), and Keypad Detail Drawing (Pg. 17) drawings, changed keypad dimensions from “10.5 mm” to “12 mm” height and module overall depth from “20.55 mm” to “22.05 mm”. We started this gradual transition June 2010. ● In section Typical Current Consumption (Pg. 22), made slight changes to specifications due to improved backlight. ● Added section ATX Power Supply Power and Control Connections (Pg. 23). ● In Command 4 (0x04): Store Current State As Boot State (Pg. 40), deleted reference to Command 32 which is reserved for CFA631 key legends. ● Deleted “SCAB Required” in the commands listed below. These commands can also be used when the module does not have the optional SCAB and is connected to the host with a Crystalfontz WR-PWR-Y25 cable. - Command 5 (0x05): Reboot CFA635, Reset Host, or Power Off Host (Pg. 41). - Command 13 (0x0D): Set LCD Contrast (Pg. 43), corrected contrast setting from "`(0-255 valid)`" and "`126-255 = very dark`" to "`(0-254 valid)`" and "`126-254 = very dark`". - Command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 51). - Command 29 (0x1D): Enable/Disable and Reset the Watchdog (Pg. 52). ● In command 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 54), - Corrected specification of GPIO pull-up/pull-down mode resistance values from “approximately 5Ω” to “approximately 5kΩ”. - Clarified parameter usage in GPIOs vs GPOs.

DATA SHEET (Continued)	
2010/08/04	● In command 35 (0x23): Read GPIO Pin Levels and Configuration State (Pg. 42), - Corrected from "`data_length: 4`" to "`data_length: 1`". - Index for "`data[0]: index of GPIO to query`" previously listed only 1-4. Added 5-12. - Corrected from "`5-255: reserved`" to "`13-255: reserved`". - Improved LED descriptions for H1 pins. Pins did not change. - Clarified parameter usage in GPIOs versus GPOs. ● Corrected references from J8 and J9 connectors (CFA633 connectors) to H1 and H2 (CFA635 connectors). Connectors were labeled correctly in drawings and have not changed. ● Please read expanded section MODULE RELIABILITY AND LONGEVITY (Pg. 59). ● In APPENDIX C: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC) (Pg. 67), - Added section with hypertext links to our free downloadable code. - Added section Algorithm 2B: “C” Improved Bit Shift Implementation (Pg. 70). This is a simplified algorithm that implements the CRC. - In sample code for Algorithm 1: “C” Table Implementation (Pg. 67) and Algorithm 2: “C” Bit Shift Implementation (Pg. 68), added typedefs for "`ubyte`" and "`word`". - In Algorithm 6: “Perl” Table Implementation (Pg. 75), corrected code from "`my $packet = $type . $length . $data ;`" to "`my $packet = chr(hex $type) .chr(hex $length) .chr(hex $data);` ". ● Reorganized content, improved drawings, and edited to improve readability.
2007/05/15	Data Sheet Revision: v1.0 New Data Sheet.

The Fine Print

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

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

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

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

Copyright © 2010 by Crystalfontz America, Inc., 12412 East Saltese Avenue, Spokane Valley, WA 99216-0357 U.S.A.

CONTENTS

INTRODUCTION - - - - - **10**
MAIN FEATURES - - - - - **10**
ORDERING INFORMATION - - - - - **11**
Module Variant Choices - - - - - 11
Module Classification Information - - - - - 12
Kit Configuration Choices - - - - - 13
MECHANICAL SPECIFICATIONS - - - - - **14**
Physical Characteristics - - - - - 14
Module Outline Drawing - - - - - 15
Jumper Locations and Functions - - - - - 17
Keypad Detail Drawing - - - - - 18
ELECTRICAL SPECIFICATIONS - - - - - **19**
System Block Diagram - - - - - 19
Viewing Direction - - - - - 20
Driving Method - - - - - 20
Absolute Maximum Ratings - - - - - 20
DC Characteristics - - - - - 20
Typical Current Consumption - - - - - 20
ESD (Electro-Static Discharge) Specifications - - - - - 21
Additional Criteria - - - - - 22
CONNECTION INFORMATION - - - - - **22**
ATX Power Supply Power and Control Connections - - - - - 22
CFA635-YYE-KL LCD Module Connectors - - - - - 24
Pin Assignments on LCD Module H1 and H2 Connectors - - - - - 25
H1 Pin Assignments - Includes GPIO Connections - - - - - 25
H2 Pin Assignments - Includes GPO Connections - - - - - 26
Power Connection to Host - - - - - 26
HOST COMMUNICATIONS - - - - - **27**
Packet Structure - - - - - 27
About Handshaking - - - - - 28
Report Codes - - - - - 28
0x80: Key Activity - - - - - 28
0x81: Reserved for Fan Speed Report (SCAB required) - - - - - 29
Command Codes - - - - - 29
0 (0x00): Ping Command - - - - - 29
1 (0x01): Get Hardware & Firmware Version - - - - - 29
2 (0x02): Write User Flash Area - - - - - 29
3 (0x03): Read User Flash Area - - - - - 30
4 (0x04): Store Current State As Boot State - - - - - 30
5 (0x05): Reboot CFA635, Reset Host, or Power Off Host - - - - - 30
6 (0x06): Clear LCD Screen - - - - - 32
9 (0x09): Set LCD Special Character Data - - - - - 32
10 (0x0A): Read 8 Bytes of LCD Memory - - - - - 33
11 (0x0B): Set LCD Cursor Position - - - - - 33

CONTENTS, CONTINUED

12 (0x0C): Set LCD Cursor Style ---- 33
13 (0x0D): Set LCD Contrast ---- 34
14 (0x0E): Set LCD & Keypad Backlight ---- 34
17 (0x11): Reserved for Set Fan Power (SCAB required) ---- 34
22 (0x16): Send Command Directly to the LCD Controller ---- 35
23 (0x17): Configure Key Reporting ---- 35
24 (0x18): Read Keypad, Polled Mode ---- 35
28 (0x1C): Set ATX Power Switch Functionality ---- 37
29 (0x1D): Enable/Disable and Reset the Watchdog ---- 38
30 (0x1E): Read Reporting & Status ---- 39
31 (0x1F): Send Data to LCD ---- 40
33 (0x21): Set Baud Rate ---- 40
34 (0x22): Set or Set and Configure GPIO Pin ---- 40
35 (0x23): Read GPIO Pin Levels and Configuration State ---- 42
CHARACTER GENERATOR ROM (CGROM) ---- 44
MODULE RELIABILITY AND LONGEVITY ---- 45
Module Reliability ---- 45
Module Longevity (EOL / Replacement Policy) ---- 45
CARE AND HANDLING PRECAUTIONS ---- 46
APPENDIX A: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC) ---- 48
Sample Code ---- 48
Algorithms to Calculate the CRC ---- 48
Algorithm 1: “C” Table Implementation ---- 48
Algorithm 2: “C” Bit Shift Implementation ---- 49
Algorithm 2B: “C” Improved Bit Shift Implementation ---- 51
Algorithm 3: “PIC Assembly” Bit Shift Implementation ---- 51
Algorithm 4: “Visual Basic” Table Implementation ---- 53
Algorithm 5: “Java” Table Implementation ---- 54
Algorithm 6: “Perl” Table Implementation ---- 56
Algorithm 7: For PIC18F8722 or PIC18F2685 ---- 57
APPENDIX B: QUALITY ASSURANCE STANDARDS ---- 60

LIST OF FIGURES

Figure 1. Module Outline Drawing (2 pages below) - - -

INTRODUCTION

The CFA635 series was originally conceived as an integrated easy-to-use front panel for 1U internet appliances. Since USB is the standard low-speed interface for these appliances, the CFA635 design was USB. Shortly after its introduction, we received requests for a serial version of CFA635. Many embedded designs could use the integrated LCD, keypad, LEDs, and compact form factor of the CFA635, but these embedded applications rarely had the resources to implement the USB host interface.

The *CFA635-YYE-KL* and *CFA635-YYE-KS* address this need by providing serial variants of the CFA635. Both variants of the serial CFA635 use a special version of firmware that brings the two UART pins (Tx & Rx) of the CFA635's microcontroller to one of the CFA635's existing expansion connectors.

The *CFA635-YYE-KL* simply exposes these UART Tx & Rx (inverted, logic level, 0v to 5v nominal) signals on pin 1 and pin 2 of the CFA635's expansion header H1. If your embedded processor is in close physical proximity to the CFA635, you can cable its UART Rx and Tx pins directly to the CFA635-YYE-KL's Tx and Rx pins. No RS-232 level translators are required on either end.

The *CFA635-YYE-KS* is a CFA635-YYE-KL with an RS-232 level translator board (CFA-RS232-01) attached. The CFA635-YYE-KS is the correct choice if your embedded controller or host system has a “real” RS-232 serial port (-10v to +10v "full swing" serial interface, typically through a UART).

> **NOTE**
> The CFA635-YYE-KL has a USB connector but it is disabled and cannot be used.

MAIN FEATURES

- ❑ Large easy-to-read 20 characters x 4 lines LCD has a large display area in a compact size. Fits in a 1U rack mount case (37 mm overall height).
- ❑ Yellow-green edge LED backlit with STN yellow-green positive mode LCD (displays dark characters on yellow-green background).
- ❑ Integrated yellow-green LED backlit translucent silicone keypad.
- ❑ Direct sunlight readable.
- ❑ Adjustable backlight contrast.
- ❑ ATX power supply control functionality allows the keypad buttons on the CFA635-YYE-KL to replace the power and reset switches on your system, simplifying front panel design. Optional 16-pin Crystalfontz WR-PWR-Y25 ATX power switch cable may be used for direct connection to the host’s PC power supply.
- ❑ Four bicolor (red + green) LED indicators. The LEDs' brightness can be set by the host software, which allows smoothly mixing the LEDs to produce other colors (for example, yellow and orange).
- ❑ LCD characters are contiguous in both X and Y directions to allow the host software to display “gapless” bar graphs in horizontal or vertical directions.
- ❑ Unique "Scrolling Marquee" feature continuously scrolls a message across the display without host intervention.
- ❑ Fully decoded keypad: any key combination is valid and unique.
- ❑ Robust packet based communications protocol with 16-bit CRC.
- ❑ Built-in microcontroller.
- ❑ Nonvolatile memory capability (EEPROM):
  - Customize the “power-on” display settings.

- 16-byte “scratch” register for storing IP address, netmask, system serial number . . .

❑ TheCFA635-YYE-KL may be used with the optional Crystalfontz System Cooling Accessory Board (SCAB) for controlling up to three fans and up to 32 temperature sensors. For assistance, please contact Technical Support (888-206-9720 or email techinfo@crystalfontz.com).

❑ RoHS compliant.

❑ Factories have ISO certification.

❑ If you need a CE approved module, please see our XES635 USB series (http://www.crystalfontz.com/product/XES635BK-YYE-KU.html, http://www.crystalfontz.com/product/XES635BK-TMF-KU.html, or http://www.crystalfontz.com/product/XES635BK-TFE-KU.html.The CFA635-YYE-KL does not have CE certification because it is not an end product. The module requires power and communications from another system in order to operate.

ORDERING INFORMATION

MODULE VARIANT CHOICES

PART NUMBER	FLUID	LCD GLASS COLOR	IMAGE	POLARIZER FILM	BACKLIGHTS
CFA635-YYE-KL	STN	yellow-green	positive	transflective	LCD: yellow-green edge LEDs Keypad: yellow-green LEDs
Additional variants available (same form factor and interface).					
CFA635-TMF-KL	STN	blue	negative	transmissive	LCD: white edge LEDs Keypad: blue LEDs
CFA635-TFE-KL	FSTN	neutral	positive	transflective	LCD: white edge LEDs Keypad: white LEDs

The CFA635 series includes:

❑ A USB interface (part numbers end in “-KU”). See www.crystalfontz.com/products/635/.

❑ A serial interface using an RS-232 level translator board (part numbers end in “-KS”). If your embedded controller or host system has a “real” RS-232 serial port (-10v to +10v "full swing" serial interface, typically through a UART), see www.crystalfontz.com/products/635serial/.

❑ An external enclosure with a captive USB “A” cable connection. See www.crystalfontz.com/product/XES635BKYYEKU.

MODULE CLASSIFICATION INFORMATION

CFA 635 - Y Y E - K L
❶ ❷ ❸ ❹ ❺ ❻ ❼

❶	Brand	Crystalfontz America, Inc.
❷	Model Identifier	635
❸	Backlight Type & Color	Y – LED, yellow-green
❹	Fluid Type, Image (positive or negative), & LCD Glass Color	Y – STN, positive, yellow-green
❺	Polarizer Film Type, Normal (NT) Temperature Range, & View Angle (O ‘Clock)	E – Transflective, NT[1], 12:00
❻	Special Code 1	K – Manufacturer's code
❼	Special Code 2	L – Serial, inverted, logic level, 0v to +5v nominal

[1] *Normal Temperature Range is 0°C minimum to +50°C maximum.*

KIT CONFIGURATION CHOICES

Accessories for the CFA635-xxx-KL or CFA635-xxx-KS are:

- ❑ *Bracket*: a 5.25-inch half-height drive bay mounting bracket.
- ❑ *SLED:* a chassis that fits in 5.25-inch half-height drive bay. The SLED can hold a CFA635 module, a SCAB (System Cooling Accessory Board), and a 3.5-inch hard disk drive. (Hard drive is not included.)
- ❑ *Overlay:* an overlay for the front of module with a display window of clear Lexan, a thick hard-coated polycarbonate material. Overlay choices are black brushed anodized aluminum, silver brushed anodized aluminum, beige plastic, and black plastic.

Choose your Crystalfontz cables here: https://www.crystalfontz.com/cart/pricing.php?cat=2.

Kits for CFA635-xxx-KL or CFA635-xxx-KS are available by special order only. (Kits for CFA635-xxx-KU are available on our website.) Use the following part number classification scheme to determine your kit configuration part number. Please send the kit configuration part number and required quantity along with the cable quantities and part numbers *for each kit* that you need in **an email to sales@crystalfontz.com or call (888) 206-9720**. Our logistics team will work with the engineering group to provide you with a DP (Defined Part) number, prices, MOQ (Minimum Order Quantity), and lead time.

Kit Configuration Part Number Classification

D[type]635[overlay]-[variant]-K[interface]

[type]	[overlay]	[variant]	[interface]
B = Bracket **S** = SLED	**AK** = Black Aluminum **AL** = Silver Aluminum **BG** = Beige Plastic **BK** = Black Plastic	**TFE** = dark characters, light background **TMF** = light characters, blue background **YYE** = dark characters, yellow-green background	**S** = RS232 **L** = logic level

Kit Configuration Part Number Example

DB635AL-TMF-KL

[type]	[overlay]	[variant]	[interface]
B = Bracket	**AL** = Silver Aluminum	**TMF** = light characters, blue background	**L** = logic level

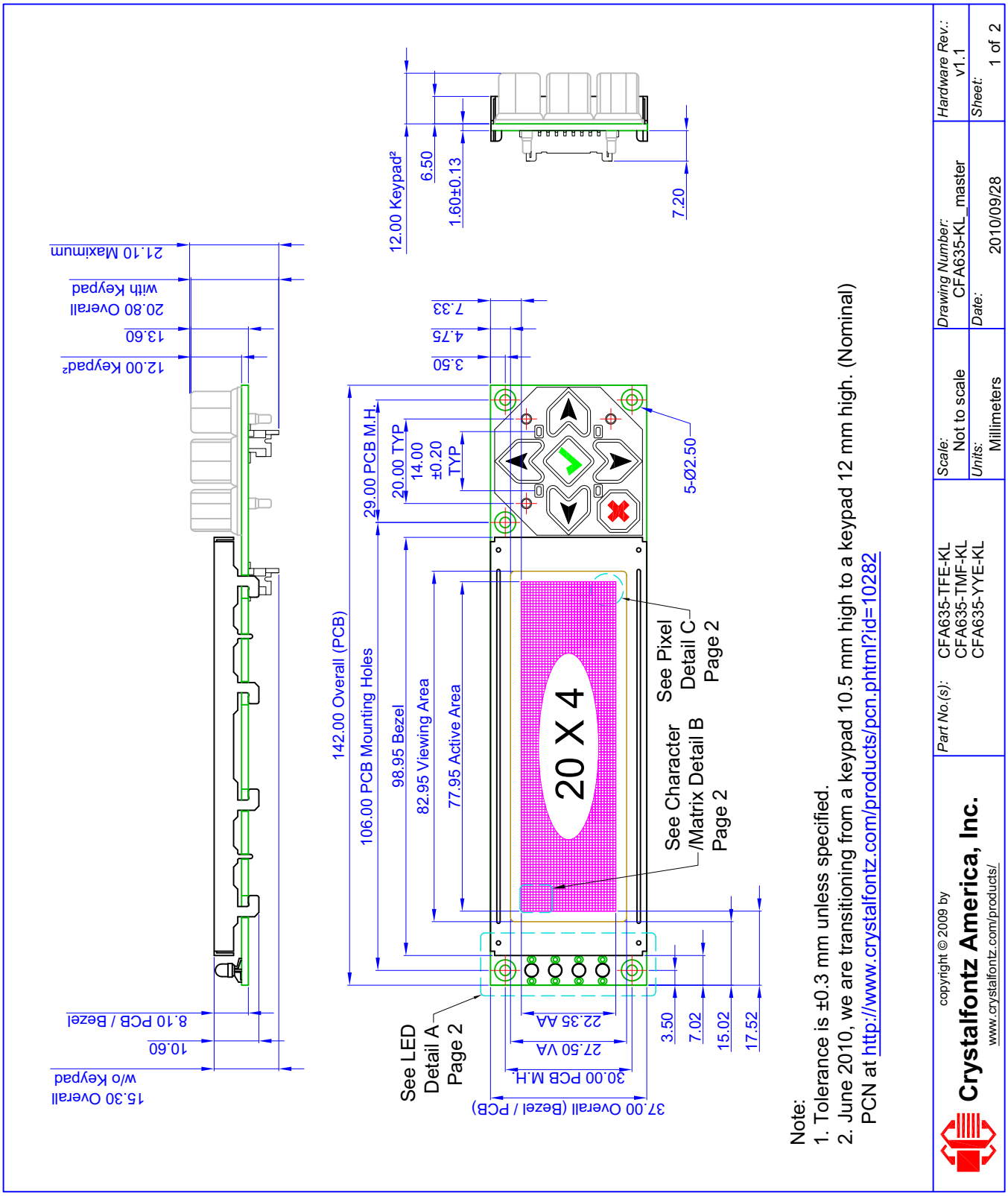
MECHANICAL SPECIFICATIONS

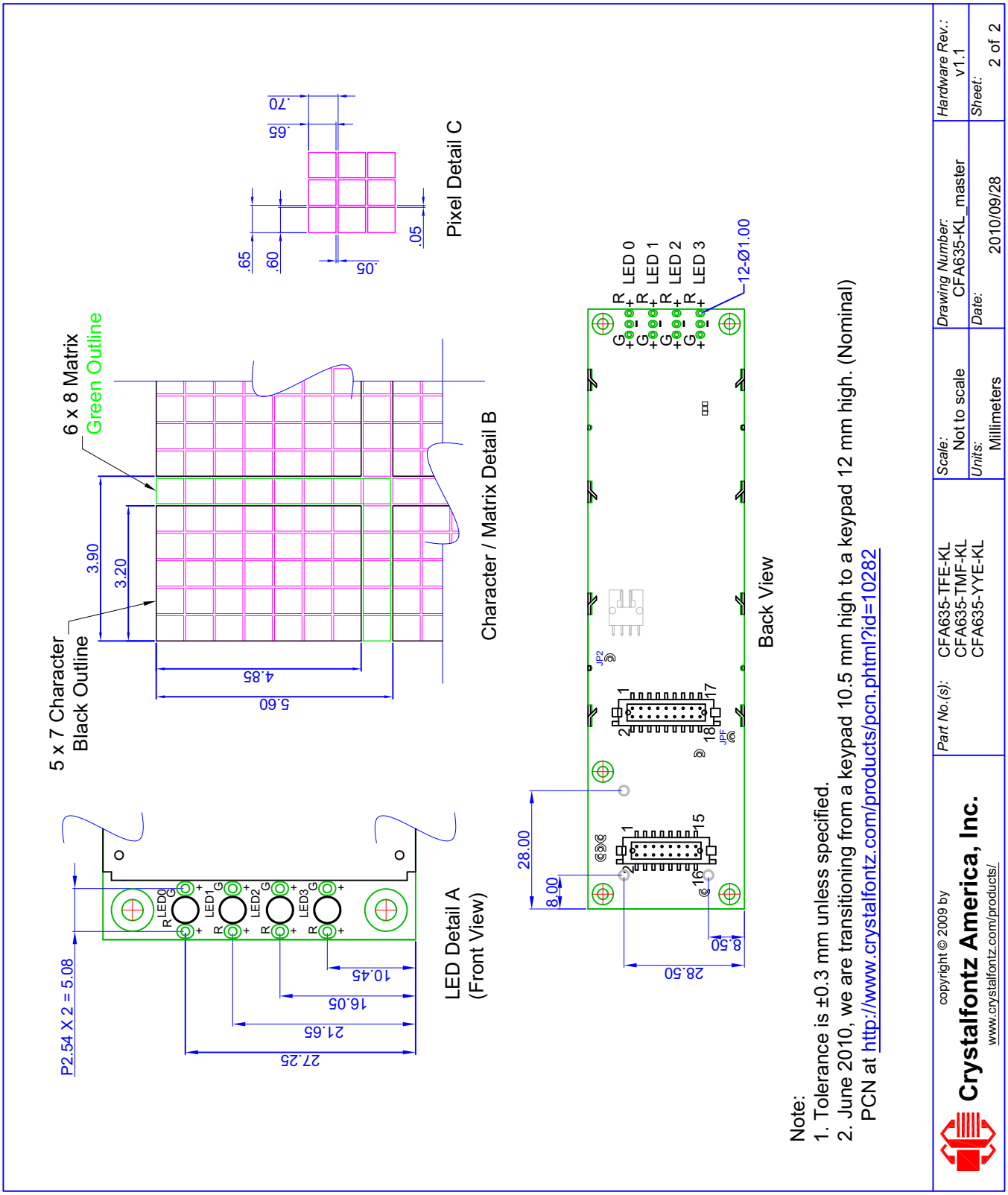
PHYSICAL CHARACTERISTICS

ITEM	SIZE
Module Overall Dimensions	
Width and Height	142.0 (W) mm x 37.0 (H) mm
Module Depth: without Keypad, with Connectors with Keypad, with Connectors	 15.30 mm nominal 20.80 mm nominal 21.10 mm maximum
Viewing Area	82.95 (W) x 27.50 (H) mm
Active Area	77.95 (W) x 22.35 (H) mm
5x7 Standard Character Size	3.20 (W) x 4.85 (H) mm
6x8 Matrix (can be used for special characters or graphics)	3.90 (W) X 5.60 (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	66 grams (typical)

MODULE OUTLINE DRAWING

Figure 1. Module Outline Drawing (2 pages below)




Pixel Detail C
.70
.65
.65
.60
.05
.05
6 x 8 Matrix
Green Outline
5 x 7 Character
Black Outline
3.90
3.20
4.85
5.60
Character / Matrix Detail B
LED Detail A
(Front View)
P2.54 X 2 = 5.08
10.45
16.05
21.65
27.25
LED0
LED1
LED2
LED3
Back View
LED 0
LED 1
LED 2
LED 3
12-Ø1.00
28.00
8.00
8.50
28.50
Note:
1. Tolerance is ±0.3 mm unless specified.
2. June 2010, we are transitioning from a keypad 10.5 mm high to a keypad 12 mm high. (Nominal)
PCN at http://www.crystalfontz.com/products/pcn.phtml?id=10282
copyright © 2009 by
Crystalfontz America, Inc.
www.crystalfontz.com/products/
Part No.(s): CFA635-TFE-KL
CFA635-TMF-KL
CFA635-YYE-KL
Scale: Not to scale
Units: Millimeters
Drawing Number: CFA635-KL_master
Date: 2010/09/28
Hardware Rev.: v1.1
Sheet: 2 of 2

JUMPER LOCATIONS AND FUNCTIONS

This CFA635 Serial LCD module has eight jumpers. Only JPF may be changed. This jumper is normally open. The jumper may be closed by melting a ball of solder across the gap. To reopen the jumper, remove the solder. Solder wick works well for this.

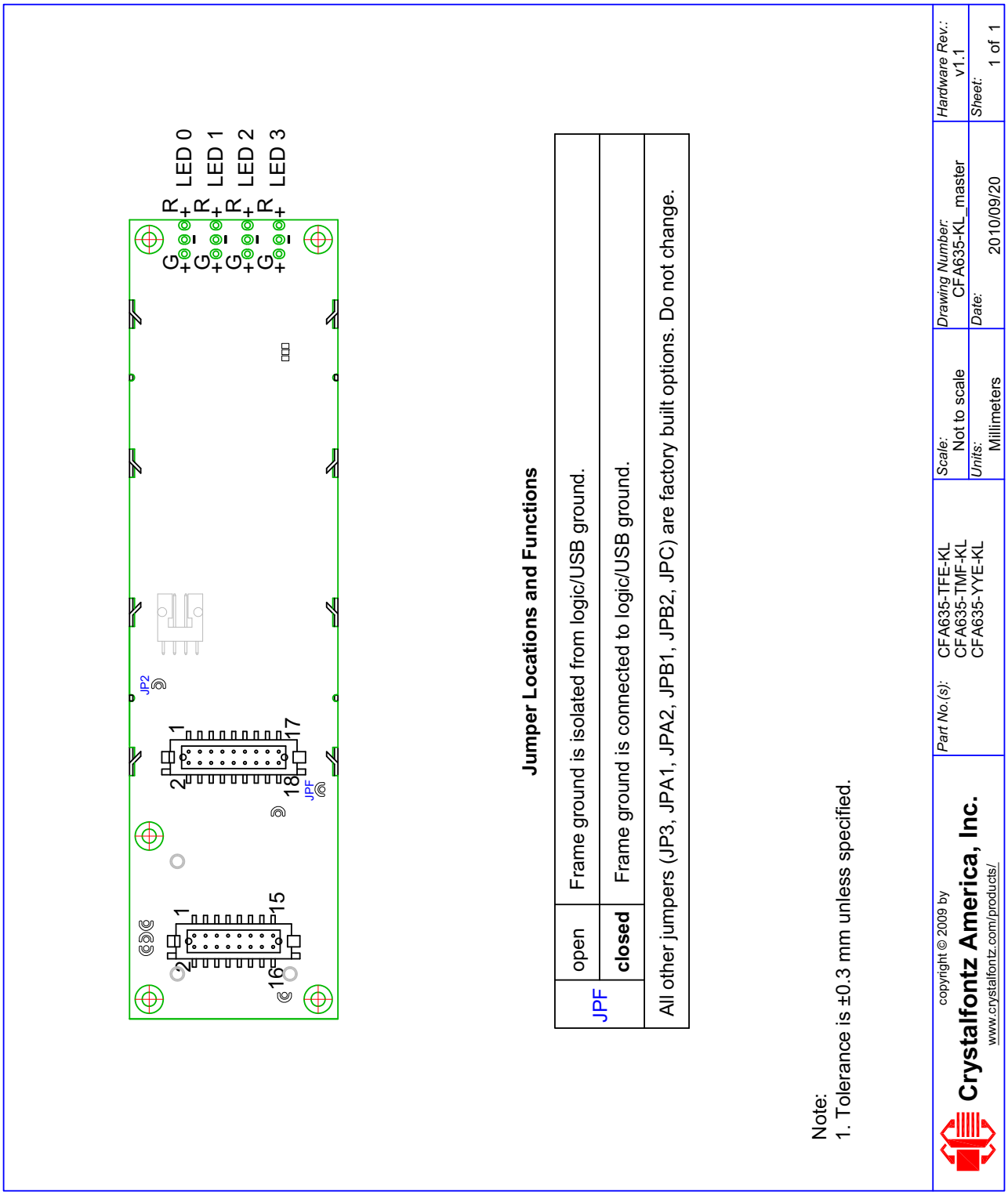


Figure 2. Jumper Locations and Functions

KEYPAD DETAIL DRAWING

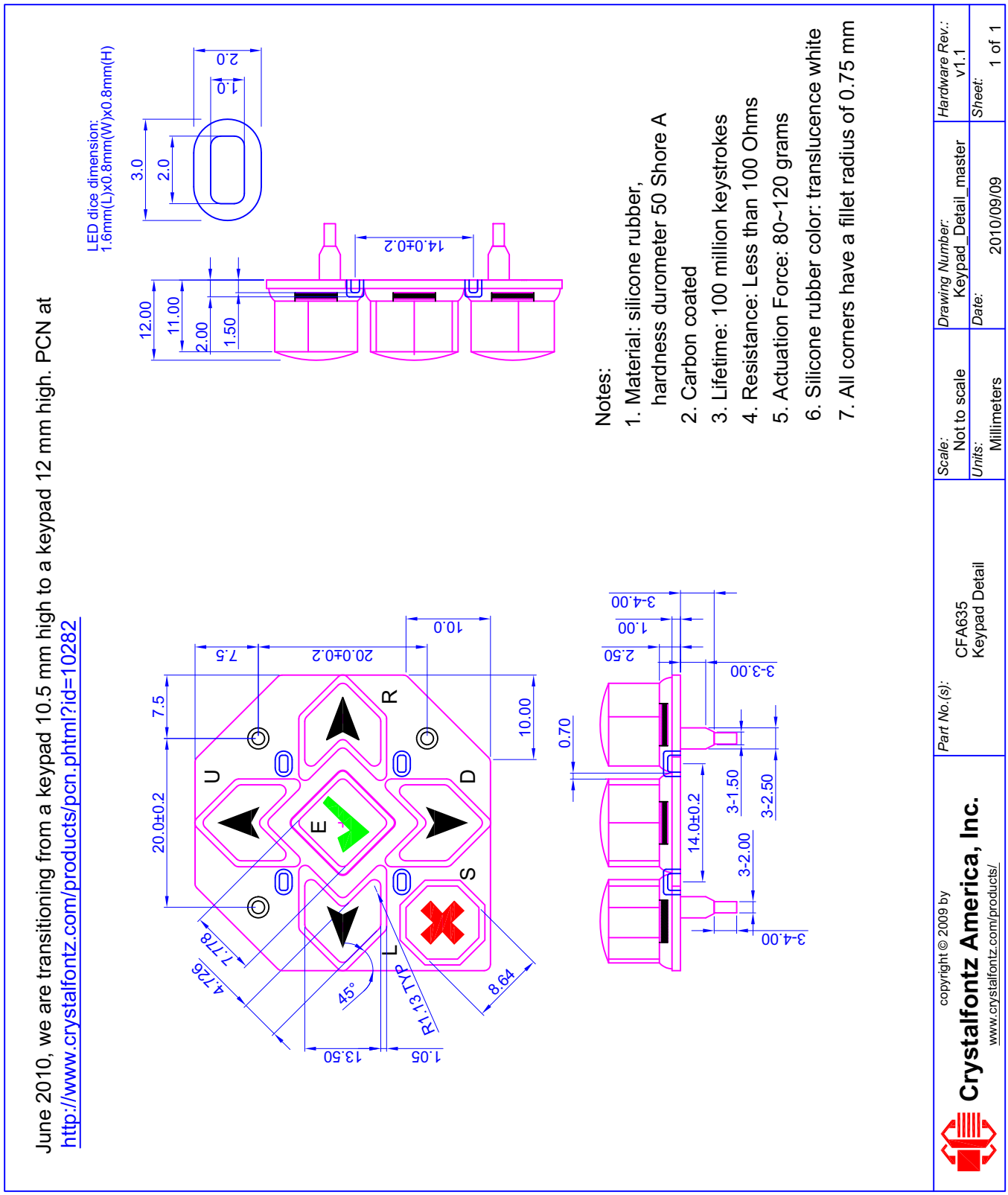


Figure 3. Keypad Detail Drawing

ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

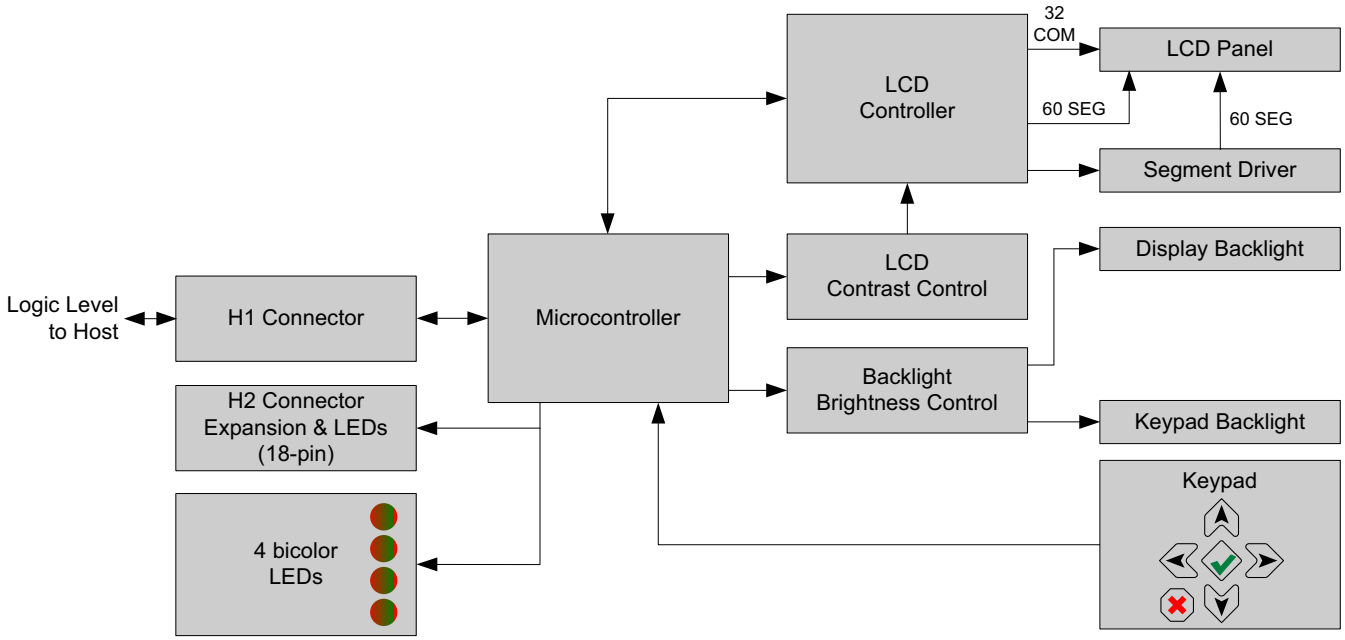


Figure 4. System Block Diagram

VIEWING DIRECTION

Viewing Direction	12 o'clock

DRIVING METHOD

DRIVING METHOD	SPECIFICATION
Duty	1/32
Bias	6.7

ABSOLUTE MAXIMUM RATINGS

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

Note
These are stress ratings only. Functional operation of the module at these or any other conditions beyond those listed under DC Characteristics (Pg. 20) is not implied.

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

DC CHARACTERISTICS

DC CHARACTERISTICS	SYMBOL	MINIMUM	TYPICAL	MAXIMUM
Supply voltage for driving the serial LCD module.	V_{DD} - V_O	+4.75v	+5.0v	+5.25v
Logic Input High Voltage	V_{IH}	+2.1v		V_{DD}
Logic Input Low Voltage	V_{IL}	V_{SS}		+.08v

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	-	-	27 mA	29 mA
X	X	-	150 mA	180 mA
X	-	X	143 mA	166 mA
X	X	X	260 mA	330 mA

GPIO CURRENT LIMITS	SPECIFICATION
Sink	25 mA
Source	10 mA

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS

The circuitry is industry standard CMOS logic and is 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.

ADDITIONAL CRITERIA

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

CONNECTION INFORMATION

ATX POWER SUPPLY POWER AND CONTROL CONNECTIONS

ATX power supply control functionality allows the buttons on the CFA635 to replace the power and reset button on your system, simplifying front panel design.

NOTE

The GPIO pins used for ATX control must not be configured as user GPIO. The GPIO pins must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are factory default but may be changed by the user. Please see command 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 40).

When configuring the CFA635 for ATX functionality, open jumper JP2 in order to ensure correct operation. See Jumper Locations and Functions (Pg. 17) for jumper positions and locations. This configuration for the CFA635 is powered from the PC's VSB signal, the “stand-by” or “always-on” +5v ATX power supply output, on pins 15 and 16 of the H1 connector.

GPIO[1] ATX Host Power Sense
Since the CFA635-YYE-KL must act differently depending on whether the host's power supply is on or off, you must also connect the host's “switched +5v” to GPIO[1]. This GPIO line functions as POWER SENSE. The POWER SENSE pin is configured as an input with a pull-down, 5kΩ nominal.

GPIO[2] ATX Host Power Control
The motherboard's power switch input is connected to GPIO[2]. This GPIO line functions as POWER CONTROL. The POWER CONTROL pin is configured as a high impedance input until the LCD module instructs the host to turn on or off. Then it will change momentarily to low impedance output, driving either low or high depending on the setting of POWER INVERT. See command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 37).

GPIO[3] ATX Host Reset Control
The motherboard's reset switch input is connected to GPIO[3]. This GPIO line functions as RESET. The RESET pin is configured as a high-impedance input until the LCD module wants to RESET the host. Then it will change momentarily to low impedance output, driving either low or high depending on the setting of RESET_INVERT. See command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 37). This connection is also used for the hardware watchdog.

Connection Overview

ATX Power Supply & Control Connections	Pins on CFA635-YYE-KL's Connector H1
V_{SB}, +5v	Pin 16
V_{SB}, Ground	Pin 13
GPIO[1] Host POWER-ON SENSE	Pin 12
GPIO[2] Host POWER CONTROL	Pin 9
GPIO[3] Host RESET CONTROL	Pin 10

Details of Connection using WR-PWR-Y25 Cable
The optional Crystalfontz WR-PWR-Y25 cable simplifies ATX power control connections, allowing all ATX power supply control functionality through the 's H1 connector.

Below is an illustration of how the optional Crystalfontz WR-PWR-Y25 cable connects to the CFA635's connector H1 and your system's motherboard and ATX power supply:

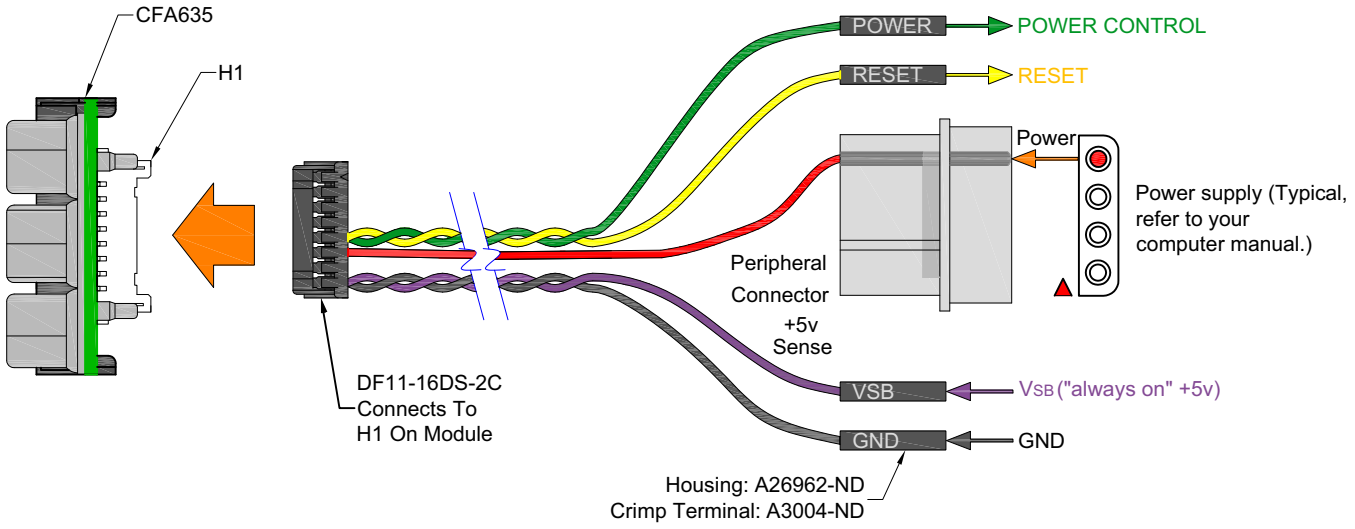


Figure 5. ATX Power Supply and Control Connections using Crystalfontz WR-PWR-Y25 Cable

CFA635-YYE-KL LCD MODULE CONNECTORS

The CFA635 Serial LCD Module has two connectors on its back side, H1 and H2.

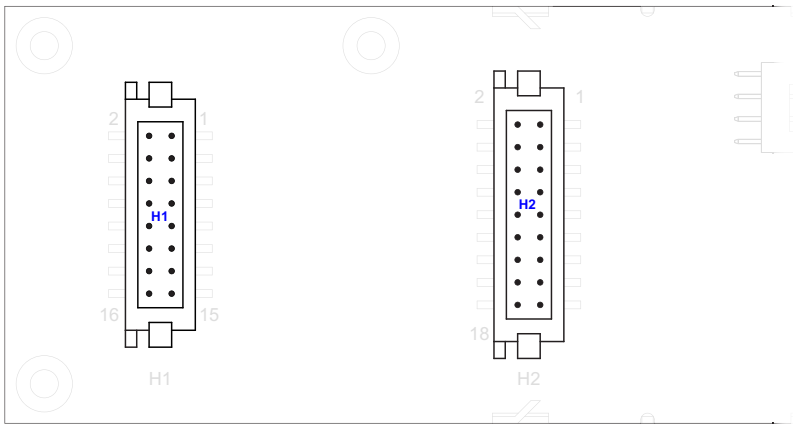


Figure 6. CFA635 Serial LCD Module Connectors H1 and H2

1. The H1 male 16-pin 2 mm connector provides the TTL serial communication and GPIO connections. For pin assignments, see H1 Pin Assignments - Includes GPIO Connections (Pg. 25).
2. The H2 male 18-pin 2 mm connector provides GPO connections, including those that drive the front panel LEDs. For pin assignments, see H1 Pin Assignments - Includes GPIO Connections (Pg. 25).

PIN ASSIGNMENTS ON LCD MODULE H1 AND H2 CONNECTORS

H1 Pin Assignments - Includes GPIO Connections

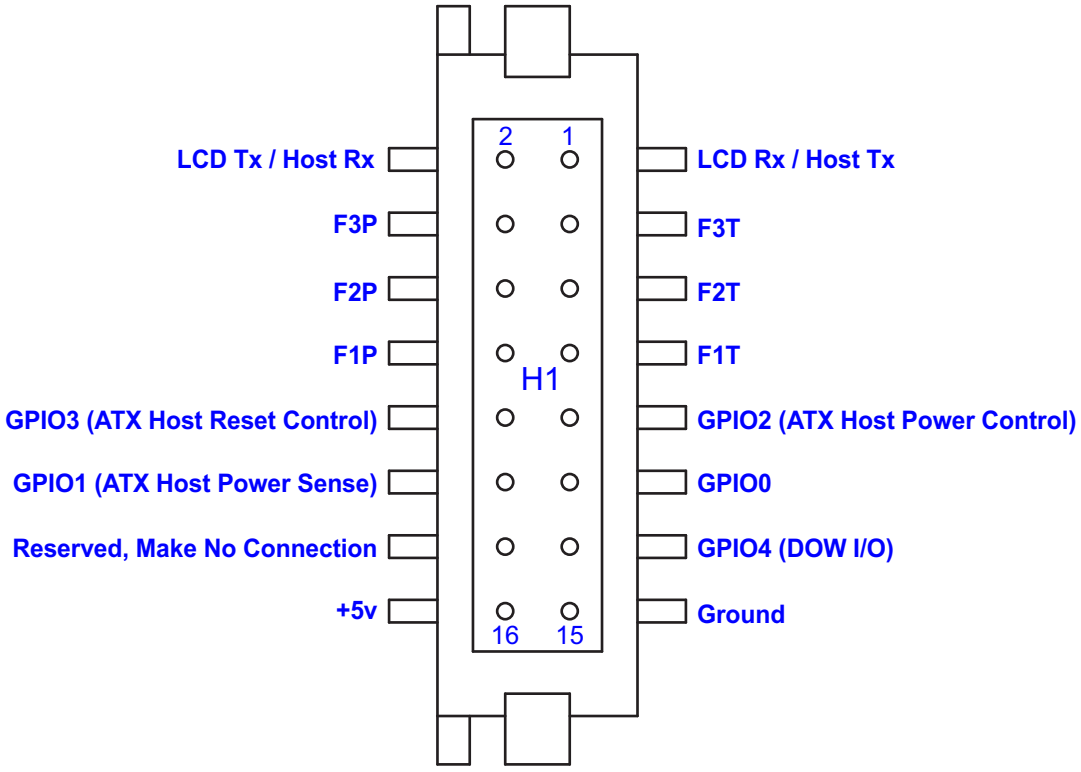


Figure 7. Pin Assignments on CFA635 Serial LCD Module H1 Connector (Includes GPIOs)

The following parts may be used to make a mating cable for H1:

- 16-position housing: Hirose DF11-16DS-2C / Digi-Key H2025-ND.
- Crimping contact (tape & reel): Hirose DF11-2428SCF / Digi-Key H1504TR-ND.
- Crimping contact (loose): Hirose DF11-2428SC / Digi-Key H1504-ND.
- Pre-terminated interconnect wire: Hirose / Digi-Key H3BBT-10112-B4-ND is typical.

H2 Pin Assignments - Includes GPO Connections

The CFA635-YYE-KL has eight GPIO/GPO connections available on header H2. By factory default, these GPOs drive the front panel LEDs. By removing the LEDs, these GPOs could be used for other purposes.

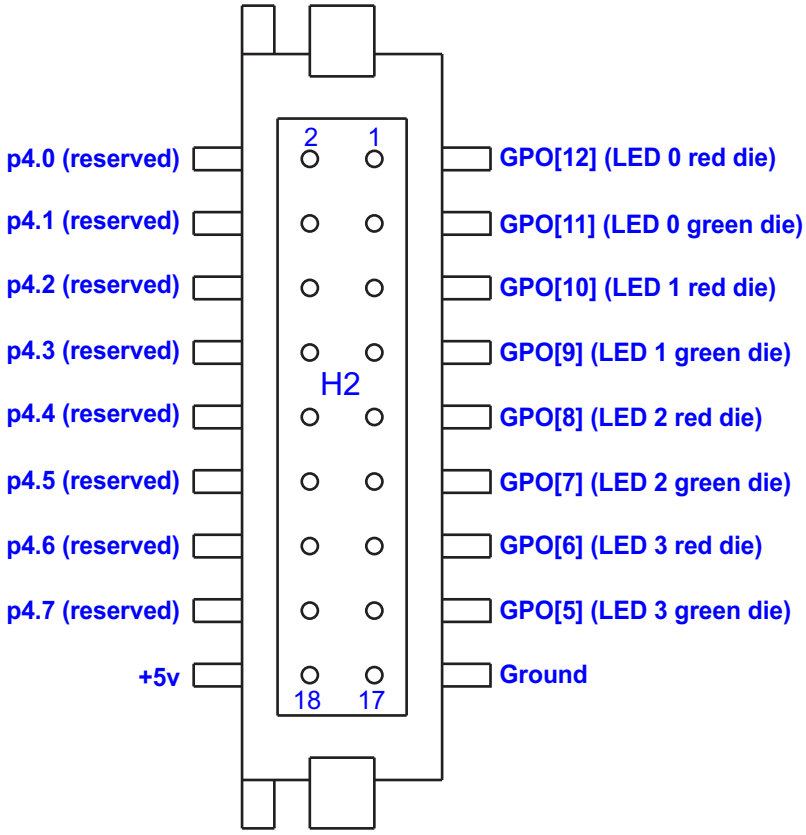


Figure 8. Pin Assignments on CFA635 Serial LCD Module H2 Connector (Includes GPIOs)

Please see the commands 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 40), and 35 (0x23): Read GPIO Pin Levels and Configuration State (Pg. 42) below for details on how to control the GPIOs.

The following parts may be used to make a mating cable for H2:

- 18-position housing: Hirose DF11-18DS-2C / Digi-Key H2026-ND.
- Terminal (tape & reel): Hirose DF11-2428SCF / Digi-Key H1504TR-ND.
- Terminal (loose): Hirose DF11-2428SC / Digi-Key H1504-ND.
- Pre-terminated interconnect wire: Hirose / Digi-Key H3BBT-10112-B4-ND is typical.

POWER CONNECTION TO HOST

The Crystalfontz 16-pin WR-EXT-Y15 (16 inches) or the 16-pin WR-EXT-Y19 (3.5 inches) cable may be useful for connecting the CFA635-YYE-KL to your host's controller board.The CFA635-YYE-KL may be used with the optional Crystalfontz System Cooling Accessory Board (SCAB) for controlling up to three fans and up to 32 temperature sensors. For assistance, please contact Technical Support (888-206-9720 or email techinfo@crystalfontz.com).

HOST COMMUNICATIONS

NOTE

Because there is no difference in communications and commands for *serial* and *USB* variants of the CFA635, the remaining sections in this Data Sheet use the shorter term “CFA635” instead of “CFA635-YYE-KL”.

The CFA635 communicates with its host using the RS-232 interface. The host’s RS-232 communications port should be opened at 115200 baud, 8 data bits, no parity, 1 stop bit.

PACKET STRUCTURE

All communication between the CFA635 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 CFA635 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 CFA635
             01 = normal response from CFA635 to host
             10 = normal report from CFA635 to host (not in
                  direct response to a command from the host)
             11 = error response from CFA635 to host (a packet
                  with valid structure but illegal content
                  was received by the CFA635)
```

`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 A: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC) (Pg. 48) 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 CFA635's packets makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for a corresponding acknowledge packet from the CFA635 before sending the next command packet. The CFA635 will respond to all packets within 250 mS. The host software should stop waiting and retry the packet if the CFA635 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 CFA635 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 CFA635 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 CFA635. 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 CFA635 can be configured to report three items. The CFA635 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 (1) 0x80: Key Activity, (2) 0x81: Reserved for Fan Speed Report (SCAB required), and (3) 0x82: Reserved for Temperature Sensor Report (SCAB required). Details are below.

0x80: Key Activity

If a key is pressed or released, the CFA635 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. 35).

```
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 CFA633. Please note that the CFA631 will return codes 13 through 20. (See the CFA631 Data Sheet on our website for more details.)

0x81: Reserved for Fan Speed Report (SCAB required)

0x82: Reserved for Temperature Sensor Report (SCAB required)

COMMAND CODES

Below is a list of valid commands for the CFA635. 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 CFA635 will return the Ping Command to the host.

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

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

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

1 (0x01): Get Hardware & Firmware Version

The CFA635 will return the hardware and firmware version information to the host.

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

The return packet will be:

```
type = 0x40 | 0x01 = 0x41 = 65(10)
data_length = 16
data[] = "CFA635:hX.X,yY.Y"

X.X is the hardware revision, "1.1" for example
yY.Y is the firmware version, "s1.6" for example
```

2 (0x02): Write User Flash Area

The CFA635 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 CFA635'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 CFA635's non-volatile memory
```

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

The CFA635 loads its power-up configuration from nonvolatile memory when power is applied. The CFA635 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. 32).
  - Command 31 (0x1F): Send Data to LCD (Pg. 40).
- Special character font definitions (command 9 (0x09): Set LCD Special Character Data (Pg. 32)).
- Cursor position (command 11 (0x0B): Set LCD Cursor Position (Pg. 33)).
- Cursor style (command 12 (0x0C): Set LCD Cursor Style (Pg. 33)).
- Contrast setting (command 13 (0x0D): Set LCD Contrast (Pg. 34)).
- Backlight setting (command 14 (0x0E): Set LCD & Keypad Backlight (Pg. 34)).
- Key press and release masks (command 23 (0x17): Configure Key Reporting (Pg. 35)).
- ATX function enable and pulse length settings (command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 37)).
- Baud rate (command 33 (0x21): Set Baud Rate (Pg. 40)).
- GPIO settings (command 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 40)).
- The front panel LED/GPO settings (command 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 40)).

You cannot store the fan or temperature reporting, or the fan fail-safe or host watchdog. The host software should enable these items once the system is initialized and it is ready to receive the data.

```
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 CFA635, Reset Host, or Power Off Host

This command instructs the CFA635+WR-PWR-Y25 cable to simulate a power-on restart of itself, reset the host, or turn the host's power off. The ability to reset the host may be useful to allow certain host operating system configuration

changes to complete. The ability to turn the host's power off under software control may be useful in systems that do not have ACPI compatible BIOS.

> **NOTE**
> The GPIO pins used for ATX control must not be configured as user GPIO, and must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are factory default, but may be changed by the user. Please see command 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 40).

Rebooting the CFA635 may be useful when testing the boot configuration. It may also be useful to re-enumerate the devices on the 1-Wire bus (CFA635+WR-PWR-Y25 cable). To reboot the CFA635, send the following packet:

```
type = 0x05 = 5_10
valid data_length is 3
data[0] = 8
data[1] = 18
data[2] = 99
```

> **NOTE**
> The CFA635 will return the acknowledge packet immediately, then reboot itself. The module will not respond to new command packets for up to 3 seconds..

To reset the host (CFA635+WR-PWR-Y25 cable), assuming the host's reset line is connected to GPIO[3] as described in command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 37), send the following packet:

```
type = 0x05 = 5_10
valid data_length is 3
data[0] = 12
data[1] = 28
data[2] = 97
```

> **NOTE**
> The CFA635 will return the acknowledge packet immediately, then reset the host. After resetting the host (~1.5 seconds), the module will reboot itself. The module will not respond to new command packets for up to 3 seconds (~4.5 seconds overall) after its reboot.

To turn the host's power off (CFA635+WR-PWR-Y25 cable), assuming the host's power control line is connected to GPIO[2] as described in command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 37), send the following packet:

```
type = 0x05 = 5_10
valid data_length is 3
data[0] = 3
data[1] = 11
data[2] = 95
```

NOTE

The CFA635 will return the acknowledge packet immediately, then power cycle the host. The power cycle length is dependent on the length of the power pulse (command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 37)). After power cycling the host, the module will reboot itself. The module will not respond to new command packets for up to 3 seconds after its reboot.

In any of the above cases, 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. 30).

7 (0x07): Deprecated (See command 31 (0x1F): Send Data to LCD (Pg. 40))

8 (0x08): Deprecated (See command 31 (0x1F): Send Data to LCD (Pg. 40))

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. 30).

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 CFA635'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. 33).

```
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. 30).

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. 30).

13 (0x0D): Set LCD Contrast

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

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

The return packet will be:

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

Set LCD Contrast is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 30).

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. 30).

15 (0x0F): Deprecated

16 (0x10): Reserved for Set Up Fan Reporting (SCAB required)

17 (0x11): Reserved for Set Fan Power (SCAB required)

18 (0x12): Reserved for Read DOW Device Information (SCAB required)

19 (0x13): Reserved for Set Up Temperature Reporting (SCAB required)

20 (0x14): Reserved for Arbitrary DOW Transaction (SCAB required)

21 (0x15): Deprecated

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

The LCD controller on the CFA635 is Samsung a S6A0073. Generally you won't need low-level access to the LCD controller but some arcane functions of the S6A0073 are not exposed by the CFA635's command set. This command allows you to access the CFA635's LCD controller directly. Note: It is possible to corrupt the CFA635 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 CFA635 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. 30).

```
#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. 30).

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the CFA635 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. 35). 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): Reserved for Set Fan Power Fail-Safe (SCAB required)

26 (0x1A): Reserved for Set Fan Tachometer Glitch Filter (SCAB required)

27 (0x1B): Reserved for Query Fan Power & Fail-Safe Mask (SCAB required)

28 (0x1C): Set ATX Power Switch Functionality

The combination of the CFA635+WR-PWR-Y25 cable can be used to replace the function of the power and reset switches in a standard ATX-compatible system. The ATX Power Switch Functionality is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 30).

NOTE ON COMMAND 28: SET ATX POWER SWITCH FUNCTIONALITY

The GPIO pins used for ATX control must not be configured as user GPIO. The pins must be configured to their default drive mode in order for the ATX functions to work correctly.

These settings are factory default but may be changed by the user. Please see command 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 40). These settings must be saved as the boot state.

To ensure that GPIO[1] will operate correctly as ATX SENSE, user GPIO[1] must be configured as:

```
DDD = "011: 1=Resistive Pull Up, 0=Fast, Strong Drive Down".
F = "0: Port unused for user GPIO."
```

This configuration can be assured by sending the following command:

```
command = 34
length = 3
data[0] = 1
data[1] = 0
data[2] = 3
```

To ensure that GPIO[2] will operate correctly as ATX POWER, user GPIO[2] must be configured as:

```
DDD = "010: Hi-Z, use for input".
F = "0: Port unused for user GPIO."
```

This configuration can be assured by sending the following command:

```
command = 34
length = 3
data[0] = 2
data[1] = 0
data[2] = 2
```

To ensure that GPIO[3] will operate correctly as ATX RESET, user GPIO[3] must be configured as:

```
DDD = "010: Hi-Z, use for input".
F = "0: Port unused for user GPIO."
```

This configuration can be assured by sending the following command:

```
command = 34
length = 3
data[0] = 3
data[1] = 0
data[2] = 2
```

These settings must be saved as the boot state.

The RESET (GPIO[3]) and POWER CONTROL (GPIO[2]) lines on the CFA635+WR-PWR-Y25 cable are normally high-impedance. Electrically, they appear to be disconnected or floating. When the CFA635+WR-PWR-Y25 cable asserts the RESET or POWER CONTROL lines, they are momentarily driven high or low (as determined by the RESET_INVERT and POWER_INVERT bits, detailed below). To end the power or reset pulse, the CFA635+WR-PWR-Y25 cable changes the lines back to high-impedance.

FOUR FUNCTIONS ENABLED BY COMMAND 28

Function 1: KEYPAD_RESET

If POWER-ON SENSE (GPIO[1]) is high, holding the green check key for 4 seconds will pulse RESET (GPIO[3]) pin for 1 second. During the 1-second pulse, the CFA635 will show "RESET", and then the CFA635 will reset itself, showing its boot state as if it had just powered on. Once the pulse has finished, the CFA635 will not respond to any commands until after it has reset the host and itself.

Function 2: KEYPAD_POWER_ON

If POWER-ON SENSE (GPIO[1]) is low, pressing the green check key for 0.25 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified by in data[1] or the default of 1 second. During this time the CFA635 will show "POWER ON", then the CFA635 will reset itself.

Function 3: KEYPAD_POWER_OFF

If POWER-ON SENSE (GPIO[1]) is high, holding the red X key for 4 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified by in data[1] or the default of 1 second. If the user continues to hold the power key down, then the CFA635 will continue to drive the line for a maximum of 5 additional seconds. During this time the CFA635 will show "POWER OFF".

Function 4: LCD_OFF_IF_HOST_IS_OFF

If LCD_OFF_IF_HOST_IS_OFF is set, the CFA635 will blank its screen and turn off its backlight to simulate its power being off any time POWER-ON SENSE (GPIO[1]) is low. The CFA635 will still be active (since it is powered by V_{SB}), monitoring the keypad for a power-on keystroke. If +12v remains active (which would not be expected, since the host is “off”), the fans will remain on at their previous settings. Once POWER-ON SENSE (GPIO[1]) goes high, the CFA635 will reboot as if power had just been applied to it.

```
#define RESET_INVERT             0x02 //Reset pin drives high instead of low
#define POWER_INVERT             0x04 //Power pin drives high instead of low
#define LCD_OFF_IF_HOST_IS_OFF 0x10
#define KEYPAD_RESET           0x20
#define KEYPAD_POWER_ON        0x40
#define KEYPAD_POWER_OFF       0x80

type = 0x1C = 28(10)
data_length = 1 or 2
data[0]: bit mask of enabled functions
data[1]: (optional) length of power on & off pulses in 1/32 second
       1 = 1/32 sec
       2 = 1/16 sec
      16 = 1/2 sec
     255 = 8 sec
```

The return packet will be:

```
type = 0x40 | 0x1C = 0x5C = 92(10)
data_length = 0
```

29 (0x1D): Enable/Disable and Reset the Watchdog

Some high-availability systems use hardware watchdog timers to ensure that a software or hardware failure does not result in an extended system outage. Once the host system has booted, a system monitor program is started. The

system monitor program would enable the watchdog timer on the CFA635+WR-PWR-Y25 cable. If the system monitor program fails to reset the watchdog timer, the CFA635+WR-PWR-Y25 cable will reset the host system.

NOTE

The GPIO pins used for ATX control must not be configured as user GPIO. They must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are factory default, but may be changed by the user. Please see the note under command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 37) or command 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 40).

```
type = 0x1D = 29(10)
data_length = 1
data[0] = enable/timeout

If timeout is 0, the watchdog is disabled.

If timeout is 1-255, then this command must be issued again within timeout seconds to
avoid a watchdog reset.

To turn the watchdog off once it has been enabled, simply set timeout to 0.

If the command is not re-issued within timeout seconds, then the CFA635+WR-PWR-Y25 cable
will reset the host (see command 28 for details). Since the watchdog is off by default
when the it powers up, the CFA635+WR-PWR-Y25 cable will not issue another host reset until
the host has once again enabled the watchdog.
```

The return packet will be:

```
type = 0x40 | 0x1D = 0x5D = 93(10)
data_length = 0
```

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 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]  = ATX Power Switch Functionality (as set by command 28)
data[8]  = 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. 30).

32 (0x20): Reserved for CFA631 Key Legends

33 (0x21): Set Baud Rate

This command will change the CFA635's baud rate. The CFA635 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 CFA635 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. 30) if you want the CFA635 to power up at the new baud rate.

The factory default baud rate is 115200.

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

The return packet will be:

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

34 (0x22): Set or Set and Configure GPIO Pin

The CFA635 has five pins for user-definable general purpose input / output (GPIO). These pins are shared with the DOW and ATX functions. Be careful when you configure the GPIO if you want to use the ATX or DOW at the same time.

The architecture of the CFA635 allows great flexibility in the configuration of the GPIO pins. They can be set as input or output. They can output constant high or low signals or a variable duty cycle 100 Hz PWM signal.

In output mode using the PWM (and a suitable current limiting resistor), an LED may be turned on or off and even dimmed under host software control. With suitable external circuitry, the GPIOs can also be used to drive external logic or power transistors.

The CFA635 continuously polls the GPIOs as inputs at 32 Hz. The present level can be queried by the host software at a lower rate. The CFA635 also keeps track of whether there were rising or falling edges since the last host query (subject to the resolution of the 32 Hz sampling). This means that the host is not forced to poll quickly in order to detect short events. The algorithm used by the CFA635 to read the inputs is inherently "bounce-free".

The GPIOs also have "pull-up" and "pull-down" modes. These modes can be useful when using the GPIO as an input connected to a switch since no external pull-up or pull-down resistor is needed. For instance, the GPIO can be set to pull

up. Then when a switch connected between the GPIO and ground is open, reading the GPIO will return a "1". When the switch is closed, the input will return a "0".

Pull-up/pull-down resistance values are approximately 5kΩ. Do not exceed current of 25 mA per GPIO.
The GPIO configuration is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 30).

```
type: 0x22 = 34(10)
data_length:
  2 bytes to change value only
  3 bytes to change value and configure function and drive mode

data[0]: index of GPIO/GPO to modify
       0  = GPIO[0] = H1, Pin 11
       1  = GPIO[1] = H1, Pin 12 (default is ATX Host Power Sense)
       2  = GPIO[2] = H1, Pin 9  (default is ATX Host Power Control)
       3  = GPIO[3] = H1, Pin 10 (default is ATX Host Reset Control)
       4  = GPIO[4] = H1, Pin 13 (default is DOW I/O--has 1 KΩ hardware pull-up on SCAB)
       5  = GPO[ 5] = H2, Pin 15 = LED 3 (bottom) green die
       6  = GPO[ 6] = H2, Pin 13 = LED 3 (bottom) red die
       7  = GPO[ 7] = H2, Pin 11 = LED 2 green die
       8  = GPO[ 8] = H2, Pin  9 = LED 2 red die
       9  = GPO[ 9] = H2, Pin  7 = LED 1 green die
       10 = GPO[10] = H2, Pin  5 = LED 1 red die
       11 = GPO[11] = H2, Pin  3 = LED 0 (top) green die
       12 = GPO[12] = H2, Pin  1 = 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 (actual behavior depends on drive mode):
        0 = Output set to low
     1-99 = Output duty cycle percentage (100 Hz nominal)
      100 = Output set to high
  101-254 = invalid

data[2] = Pin function select and drive mode (optional, 0-15 valid)

  ---- FDDD
  |||| ||||-- DDD = Drive Mode (based on output state of 1 or 0)
  |||| |||       ========================================================
  |||| |||       000: 1=Fast, Strong Drive Up, 0=Resistive Pull Down
  |||| |||       001: 1=Fast, Strong Drive Up, 0=Fast, Strong Drive Down
  |||| |||       010: Hi-Z, use for input
  |||| |||       011: 1=Resistive Pull Up,     0=Fast, Strong Drive Down
  |||| |||       100: 1=Slow, Strong Drive Up, 0=Hi-Z
  |||| |||       101: 1=Slow, Strong Drive Up, 0=Slow, Strong Drive Down
  |||| |||       110: reserved, do not use
  |||| |||       111: 1=Hi-Z,                  0=Slow, Strong Drive Down
  |||| |||
  |||| ||------ F = Function (only valid for GPIOs, index of 0-4)
  ||||  ||       Only meaningful for GPIOs (index 0-4). GPOs (index of 5-12) will ignore.
  ||||  ||       ========================================================
  ||||  ||       0: Port unused for GPIO. It will take on the default
  ||||  ||          function such as ATX, DOW or unused. The user is
  ||||  ||          responsible for setting the drive to the correct
  ||||  ||          value in order for the default function to work
  ||||  ||          correctly.
  ||||  ||       1: Port used for GPIO under user control. The user is
  ||||  ||          responsible for setting the drive to the correct
  ||||  ||          value in order for the desired GPIO mode to work
  ||||  ||          correctly.
  ||||  |-------- reserved, must be 0
```

The return packet will be:

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

35 (0x23): Read GPIO Pin Levels and Configuration State

Please see command 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 40) for details on the GPIO architecture.

```
type: 0x23 = 35(10)
data_length: 1
data[0]: index of GPIO to query
      0 = GPIO[0] = H1, Pin 11
      1 = GPIO[1] = H1, Pin 12 (default is ATX Host Power Sense)
      2 = GPIO[2] = H1, Pin  9 (default is ATX Host Power Control)
      3 = GPIO[3] = H1, Pin 10 (default is ATX Host Reset Control)
      4 = GPIO[4] = H1, Pin 13 (default is DOW I/O--always has 1 KΩ hardware pull-up on
          SCAB.)
      5 = GPO[ 5] = H2, Pin 15 = LED 3 (bottom) green die
      6 = GPO[ 6] = H2, Pin 13 = LED 3 (bottom) red die
      7 = GPO[ 7] = H2, Pin 11 = LED 2 green die
      8 = GPO[ 8] = H2, Pin  9 = LED 2 red die
      9 = GPO[ 9] = H2, Pin  7 = LED 1 green die
     10 = GPO[10] = H2, Pin  5 = LED 1 red die
     11 = GPO[11] = H2, Pin  3 = LED 0 (top) green die
     12 = GPO[12] = H2, Pin  1 = 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 return the status of future additional GPIO
  pins
```

The return packet will be:

```
type = 0x40 | 0x23 = 0x63 = 99(10)
data_length = 4
```

```
data[0] = index of GPIO to read
data[1] = Pin state & changes since last poll
          Only useful for GPIOs (index 0-4). GPOs (index of 5-12) will return 0.
  ---- -RFS Enable Reporting of this Fan's Tach Input
  |||| |||-- S = state at the last reading
  |||| ||--- F = at least one falling edge has
  ||||  ||       been detected since the last poll
  ||||  |---- R = at least one rising edge has
  ||||  |       been detected since the last poll
  ||||  ----- reserved

    (This reading is the actual pin state, which may
     or may not agree with the pin setting, depending
     on drive mode and the load presented by external
     circuitry. The pins are polled at approximately
     32 Hz asynchronously with respect to this command.
     Transients that happen between polls will not be
     detected.)
data[2] = Requested Pin level/PWM level
  0-100: Output duty cycle percentage
    (This value is the requested PWM duty cycle. The
     actual pin may or may not be toggling in agreement
     with this value, depending on the drive mode and
     the load presented by external circuitry)
data[3] = Pin function select and drive mode
  ---- FDDD
  |||| |||-- DDD = Drive Mode
  ||||  |       ========================================================
  ||||  |       000: 1=Fast, Strong Drive Up, 0=Resistive Pull Down
  ||||  |       001: 1=Fast, Strong Drive Up, 0=Fast, Strong Drive Down
  ||||  |       010: Hi-Z, use for input
  ||||  |       011: 1=Resistive Pull Up,     0=Fast, Strong Drive Down
  ||||  |       100: 1=Slow, Strong Drive Up, 0=Hi-Z
  ||||  |       101: 1=Slow, Strong Drive Up, 0=Slow, Strong Drive Down
  ||||  |       110: reserved
  ||||  |       111: 1=Hi-Z,                 0=Slow, Strong Drive Down
  ||||  |
  ||||  |----- F = Function
  ||||         Only meaningful for GPIOs (index 0-4). GPOs (index of 5-12) will return 0
  ||||         ========================================================
  ||||         0: Port unused for GPIO. It will take on the default
  ||||            function such as ATX, DOW or unused. The user is
  ||||            responsible for setting the drive to the correct
  ||||            value in order for the default function to work
  ||||            correctly.
  ||||         1: Port used for GPIO under user control. The user is
  ||||            responsible for setting the drive to the correct
  ||||            value in order for the desired GPIO mode to work
  ||||            correctly.
  ||||------- reserved, will return 0
```

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”. 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

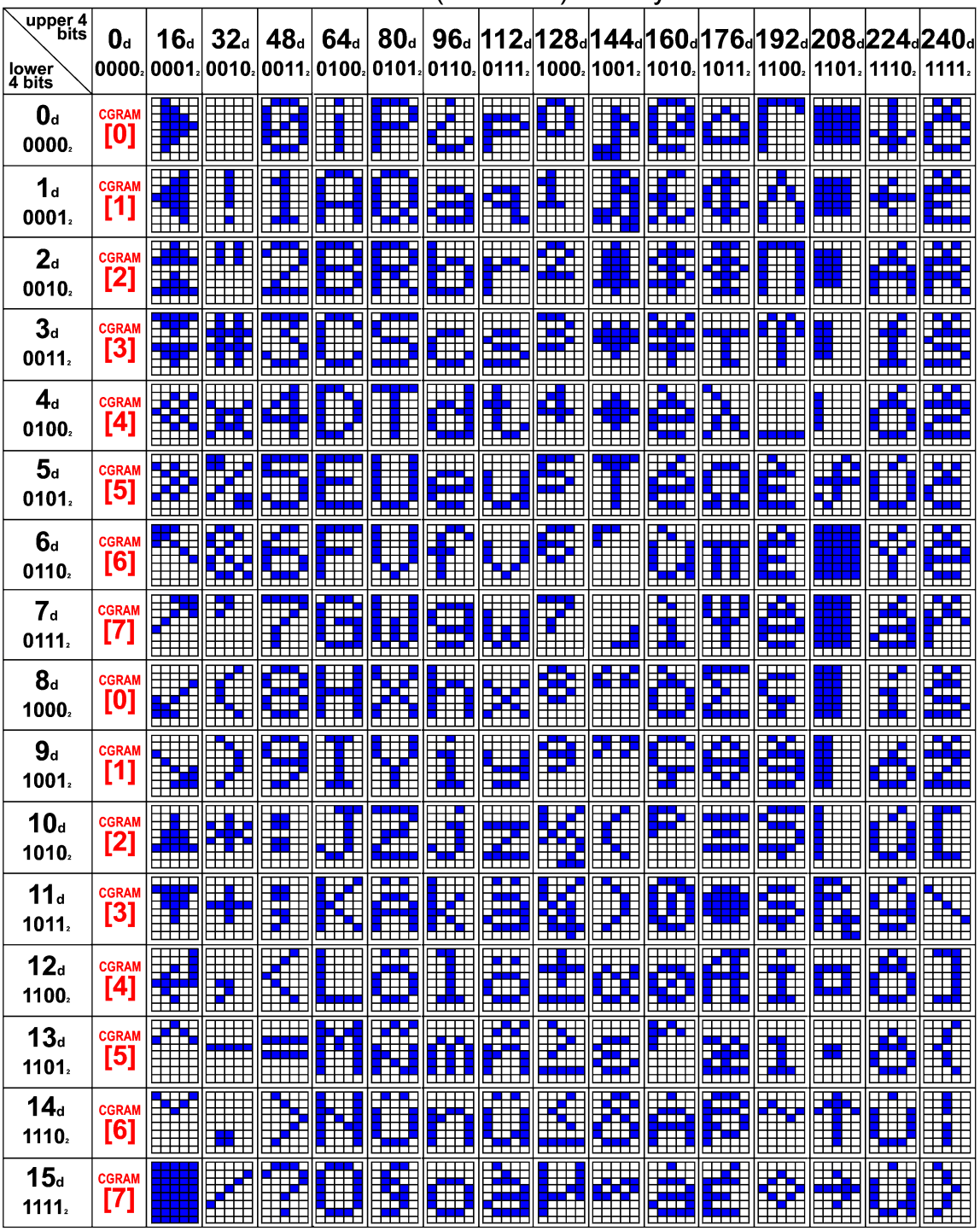


Figure 9. Character Generator ROM (CGROM)

MODULE RELIABILITY AND LONGEVITY

MODULE 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)

MODULE LONGEVITY (EOL / REPLACEMENT POLICY)

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.

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

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

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

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

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

CARE AND HANDLING PRECAUTIONS

For optimum operation of the CFA635-YYE-KL and to prolong its life, please follow the precautions described below.

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS

The circuitry is industry standard CMOS logic and is 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.

DESIGN AND MOUNTING

- The exposed surface of the LCD “glass” is actually a polarizer laminated on top of the glass. To protect the polarizer from damage, the CFA635-YYE-KL ships with a protective film over the polarizer. Please peel off the protective film slowly. Peeling off the protective film abruptly may generate static electricity.
- The polarizer is made out of soft plastic and is easily scratched or damaged. When handling the module, avoid touching the polarizer. Finger oils are difficult to remove.
- To protect the soft plastic polarizer from damage, place a transparent plate (for example, acrylic, polycarbonate, or glass) in front of the CFA635-YYE-KL, leaving a small gap between the plate and the display surface. We recommend GE HP-92 Lexan, which is readily available and works well.
- Do not disassemble or modify the module.
- Do not modify the eight tabs of the metal bezel or make connections to them.
- Solder only to the I/O terminals. Use care when removing solder so you do not damage the PCB.
- Do not reverse polarity to the power supply connections. Reversing polarity will immediately ruin the module.

AVOID SHOCK, IMPACT, TORQUE, OR TENSION

- Do not expose the module to strong mechanical shock, impact, torque, or 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 module is severely damaged and 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.

HOW TO CLEAN

- The polarizer (laminated to the glass) is soft plastic. The soft plastic is easily 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 polarizer.
- Do not clean the polarizer with liquids. Do not wipe the polarizer 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 polarizer is dusty, you may carefully blow it off with clean, dry, oil-free compressed air.

- CFA635-YYE-KL *without Crystalfontz overlay:* The exposed surface of the LCD “glass” is actually the front polarizer laminated to the glass. The polarizer is made out of a fairly soft plastic and is easily scratched or damaged. The polarizer will eventually become hazy if you do not take great care when cleaning it. Long contact with moisture (from condensation or cleaning) may permanently spot or stain the polarizer.

OPERATION

- Your circuit should be designed to protect the module from ESD and power supply transients.
- Observe the operating temperature limitations: a minimum of 0°C to a maximum of 50°C noncondensing 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.

STORAGE AND RECYCLING

- Store in an ESD-approved container away from dust, moisture, and direct sunlight.
- Observe the storage temperature limitations: a minimum of -10°C minimum to +60°C non-condensing 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.
- Please recycle your outdated Crystalfontz modules at an approved facility.

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

SAMPLE CODE

We encourage you to use the free sample code listed below. Please leave the original copyrights in the code.

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

ALGORITHMS TO CALCULATE THE CRC

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

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

Algorithm 1: “C” Table Implementation

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

```
//This code is from the IRDA LAP documentation, which appears to
//have been copied from PPP:
//
// http://irda.affiniscape.com/associations/2494/files/Specifications/
IrLAP11_Plus_Errata.zip
//
//I doubt that there are any worries about the legality of this code,
//searching for the first line of the table below, it appears that
//the code is already included in the linux 2.6 kernel "Driver for
//ST5481 USB ISDN modem". This is an "industry standard" algorithm
//and I do not think there are ANY issues with it at all.
typedef unsigned char ubyte;
typedef unsigned short word;
word get_crc(ubyte *bufptr,word len)
  {
  //CRC lookup table to avoid bit-shifting loops.
  static const word crcLookupTable[256] =
    {0x00000,0x01189,0x02312,0x0329B,0x04624,0x057AD,0x06536,0x074BF,
     0x08C48,0x09DC1,0x0AF5A,0x0BED3,0x0CA6C,0x0DBE5,0x0E97E,0x0F8F7,
     0x01081,0x00108,0x03393,0x0221A,0x056A5,0x0472C,0x075B7,0x0643E,
     0x09CC9,0x08D40,0x0BFDB,0x0AE52,0x0DAED,0x0CB64,0x0F9FF,0x0E876,
     0x02102,0x0308B,0x00210,0x01399,0x06726,0x076AF,0x04434,0x055BD,
     0x0AD4A,0x0BCC3,0x08E58,0x09FD1,0x0EB6E,0x0FAE7,0x0C87C,0x0D9F5,
     0x03183,0x0200A,0x01291,0x00318,0x077A7,0x0662E,0x054B5,0x0453C,
     0x0BDCB,0x0AC42,0x09ED9,0x08F50,0x0FBEF,0x0EA66,0x0D8FD,0x0C974,
     0x04204,0x0538D,0x06116,0x0709F,0x00420,0x015A9,0x02732,0x036BB,
     0x0CE4C,0x0DFC5,0x0ED5E,0x0FCD7,0x08868,0x099E1,0x0AB7A,0x0BAF3,
     0x05285,0x0430C,0x07197,0x0601E,0x014A1,0x00528,0x037B3,0x0263A,
```

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

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

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

Algorithm 2: "C" Bit Shift Implementation

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

```
typedef unsigned char ubyte;
typedef unsigned short word;
word get_crc(ubyte *bufptr,word len)
  {
  register unsigned int
    newCRC;
  //Put the current byte in here.
  ubyte
    data;
  int
    bit_count;
  //This seed makes the output of this shift based algorithm match
  //the table based algorithm. The center 16 bits of the 32-bit
  //"newCRC" are used for the CRC. The MSb of the lower byte is used
  //to see what bit was shifted out of the center 16 bit CRC
  //accumulator ("carry flag analog");
  newCRC=0x00F32100;
  while(len--)
    {
    //Get the next byte in the stream.
    data=*bufptr++;
    //Push this byte's bits through a software
    //implementation of a hardware shift & xor.
    for(bit_count=0;bit_count<=7;bit_count++)
      {
      //Shift the CRC accumulator
      newCRC>>=1;
```

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

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

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

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

Algorithm 2B: "C" Improved Bit Shift Implementation

This is a simplified algorithm that implements the CRC.

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

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

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

Algorithm 3: "PIC Assembly" Bit Shift Implementation

This routine was graciously donated by one of our customers.

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

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

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

Algorithm 4: “Visual Basic” Table Implementation

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

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

Private Type WORD
   Lo As Byte
   Hi As Byte
End Type

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

Dim crcLookupTable(256) As WORD

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

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

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

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

Algorithm 5: "Java" Table Implementation

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

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

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

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

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

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

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

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

Algorithm 6: “Perl” Table Implementation

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

```
#!/usr/bin/perl

use strict;

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

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

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

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

my $valid_crc = '5584' ;

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

my $crc = 0xFFFF ;

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

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

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

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

Algorithm 7: For PIC18F8722 or PIC18F2685

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

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

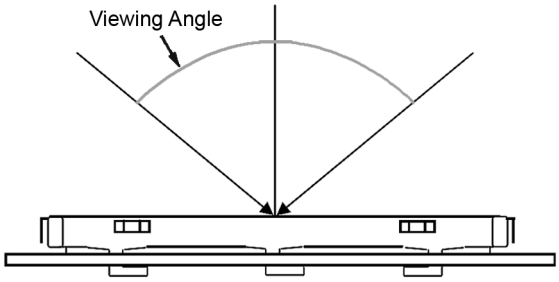
```
        bcf     STATUS,C        ; clear carry for CRC register shift
        rrcf    dsplyCRCData,f  ; perform shift of data into CRC
                                ;register
        rrcf    dsplyCRC,F
        rrcf    dsplyCRC+1,F
        btfss   STATUS,C        ; skip jump if carry
        goto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

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

APPENDIX B: QUALITY ASSURANCE STANDARDS

INSPECTION CONDITIONS

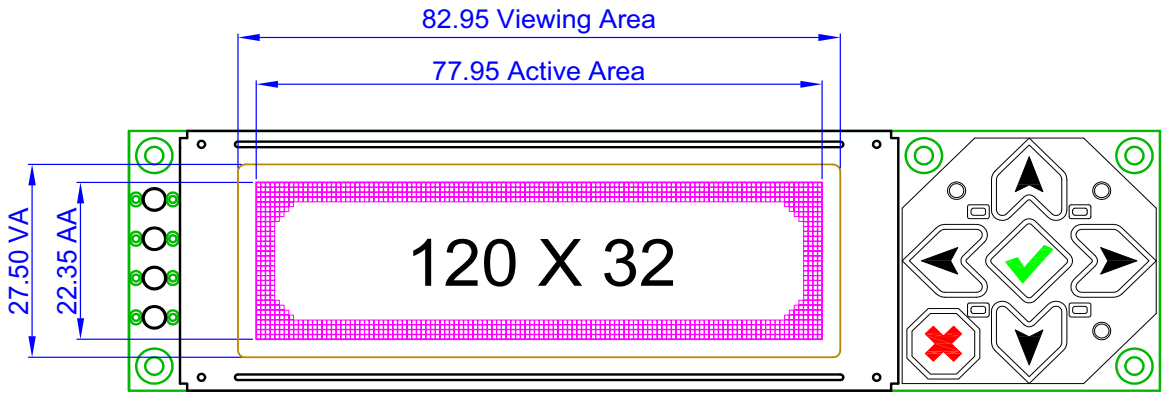
- Environment
  - Temperature: 25±5°C
  - Humidity: 30~85% RH
- For visual inspection of active display area
  - Source lighting: two 20 watt or one 40 watt fluorescent light
  - Display adjusted for best contrast
  - Viewing distance: 30±5 cm (about 12 inches)
  - Viewing angle: inspect at 45° angle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



COLOR DEFINITIONS

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

DEFINITION OF ACTIVE AREA AND VIEWING AREA



ACCEPTANCE SAMPLING

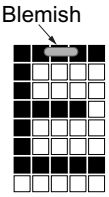
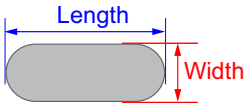
DEFECT TYPE	AQL*
Major	≤.65%
Minor	<1.0%
* Acceptable Quality Level: maximum allowable error rate or variation from standard	

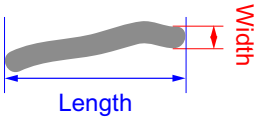
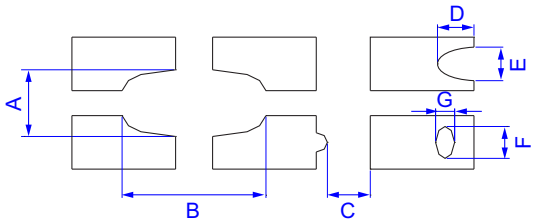
DEFECTS CLASSIFICATION

Defects are defined as:

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

ACCEPTANCE STANDARDS

#	DEFECT TYPE	CRITERIA			MAJOR / MINOR
1	Electrical defects	1. No display, display malfunctions, or shorted segments. 2. Current consumption exceeds specifications.			Major
2	Viewing area defect	Viewing area does not meet specifications. (See Inspection Conditions (Pg. 60)).			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	

<table>
<tr><th>#</th><th>DEFECT TYPE</th><th colspan="3">CRITERIA</th><th>MAJOR / MINOR</th></tr>
<tr><td rowspan="6">6</td><td rowspan="6">Dark lines or scratches in display area
 
</td><td>Defect Width</td><td>Defect Length</td><td>Acceptable Qty</td><td rowspan="6">Minor</td></tr>
<tr><td>≤0.03 mm</td><td>≤3.0 mm</td><td>3</td></tr>
<tr><td>0.03 to 0.05</td><td>≤2.0 mm</td><td>2</td></tr>
<tr><td>0.05 to 0.08</td><td>≤2.0 mm</td><td>1</td></tr>
<tr><td>0.08 to 0.10</td><td>≤3.0 mm</td><td>0</td></tr>
<tr><td>≥0.10</td><td>>3.0 mm</td><td>0</td></tr>
<tr><td rowspan="5">7</td><td rowspan="5">Bubbles between polarizer film and glass</td><td colspan="2">Defect Size</td><td>Acceptable Qty</td><td rowspan="5">Minor</td></tr>
<tr><td colspan="2">≤0.2 mm</td><td>Ignore</td></tr>
<tr><td colspan="2">0.20 to 0.40 mm</td><td>3</td></tr>
<tr><td colspan="2">0.40 to 0.60 mm</td><td>2</td></tr>
<tr><td colspan="2">≥0.60 mm</td><td>0</td></tr>
<tr><td rowspan="6">8</td><td rowspan="6">Display pattern defect</td><td colspan="3"> 
</td><td></td></tr>
<tr><td>Dot Size</td><td colspan="2">Acceptable Qty</td><td>Minor</td></tr>
<tr><td>((A+B)/2)≤0.2 mm</td><td colspan="2" rowspan="4">≤3 total defects
≤2 pinholes per digit</td><td rowspan="4"></td></tr>
<tr><td>C>0 mm</td></tr>
<tr><td>((D+E)/2)≤0.25 mm</td></tr>
<tr><td>((F+G)/2)≤0.25 mm</td></tr>
<tr><td>9</td><td>Backlight defects</td><td colspan="3">1. Oxidation or contamination on connectors.*
2. Wrong parts, missing parts, or parts not in specification.*
3. Jumpers set incorrectly.
4. Solder (if any) on bezel, LED pad, zebra pad, or screw hole pad is not smooth.
*Minor if display functions correctly. Major if the display fails.</td><td>Minor</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.
4. Solder (if any) on bezel, LED pad, zebra pad, or screw hole pad is not smooth.
*Minor if display functions correctly. Major if the display fails.</td><td>Minor</td></tr>
</table>

#	DEFECT TYPE	CRITERIA	MAJOR / MINOR
11	Soldering defects	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.*	Minor