



Crystalfontz America, Incorporated

INTELLIGENT LCD MODULE SPECIFICATIONS



Crystalfontz Model Numbers Sold separately or as part of a kit.	CFA735-TFK-KT CFA735-TML-KT CFA735-YYK-KT
Hardware Version	Part 1: CFA735 LCD Module Revision 0v9, January 2012 Part 2: CFA-RS232 Serial Converter Revision 1.0, June 2005
Firmware Version	0v9, January 2012
Data Sheet Version	0v9, January 2012 Preliminary

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

CFA735-xxx-KT is a CFA735-xxx-KR with a CFA-RS232 serial converter board.

HARDWARE PART 1: CFA735-xxx-KR LCD MODULES	
2012/01/05	Hardware Version: 0v9 New module. Using CFA635 emulation, the CFA735 is mechanically and code-compatible with the EOL CFA635 family. See Part Change Notice #10365 for EOL. See Technical Bulletin #10377 for CFA635 to CFA735 migration details.

HARDWARE PART 2: CFA-RS232 SERIAL CONVERTER (mounted on CFA735-xxx-KR module)	
2006/01/05	Current Hardware Revision: v1.0 New hardware.

FIRMWARE	
2012/01/05	Firmware Revision: 0v9 New firmware.

DATA SHEET	
2012/01/05	Data Sheet Revision: 0v9 Preliminary Preliminary 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 © 2012 by Crystalfontz America, Inc., 12412 East Saltese Avenue, Spokane Valley, WA 99216-0357 U.S.A.

Preliminary



CONTENTS

INTRODUCTION.....	9
Difference Between the Two Serial Interfaces - - - - -	9
Explanation of Part Number Codes in This Data Sheet - - - - -	10
Comparison of CFA735 Family and the Previous CFA635 Family - - - - -	10
Additional Module Features - - - - -	11
MECHANICAL SPECIFICATIONS.....	13
Physical Characteristics - - - - -	13
Module Outline Drawing Front and Side Views - - - - -	13
Module Outline Drawing Back View and Pixel Detail - - - - -	15
Panel Mounting Application Cutout Drawing - - - - -	16
Keypad Detail Drawing - - - - -	17
ELECTRICAL SPECIFICATIONS.....	18
System Block Diagram - - - - -	18
LCD Duty and Bias - - - - -	19
Absolute Maximum Ratings - - - - -	19
DC Characteristics - - - - -	19
Input Supply Voltages - - - - -	19
Logic Level GPIO +5 volt Tolerant Pins - - - - -	20
Typical GPIO Current Limits - - - - -	20
Typical Current Consumption - - - - -	21
CFA735-TFK-Kx (Black on White) - - - - -	21
CFA735-TML-Kx (White on Blue) - - - - -	21
CFA735-YYK-Kx (Black on Yellow-Green) - - - - -	22
Supply Current vs. Supply Voltage Over Backlight Range - - - - -	22
Pull-In and Drop-Out Voltage Specifications - - - - -	23
ESD (Electro-Static Discharge) Specifications (By Interface) - - - - -	23
“Full Swing” RS232 Serial Interface - - - - -	23
USB Interface - - - - -	23
fan Tachometer Speed Range - - - - -	24
OPTICAL SPECIFICATIONS.....	24
Viewing Direction - - - - -	24
Optical Characteristics - - - - -	24
CFA735-TFK-Kx - - - - -	24
CFA735-TML-Kx and CFA735-YYK-Kx - - - - -	25
Test Conditions and Definitions for Optical Characteristics - - - - -	25
BACKLIGHT INFORMATION.....	29



CONTENTS, CONTINUED

CONNECTION INFORMATION.....	30
Cables - - - - -	30
Three Connectors on the CFA735 - - - - -	31
USB Connector - - - - -	31
FBSCAB Connector - - - - -	32
H1 Connector for CFA-RS232, "Full Swing" RS232 Serial Interface - - - - -	32
Standard (+5v) Power Supply and Data Communications through USB - - - - -	32
ATX Power Supply Power and Control Connections - - - - -	33
CFA-RS232 Serial Converter for RS232 "Full Swing" - - - - -	34
Connectors on CFA-RS232 Serial Converter - - - - -	35
CFA-RS232 J1 Connector Pin Assignments (Default and Alternate) - - - - -	36
CFA-RS232 J2 Connector Pin Assignments (Includes GPIO Connections) - - - - -	38
Connect Optional FBSCAB - - - - -	39
Connect Optional DOW (Dallas One-Wire) Devices to FBSCAB - - - - -	40
Temperature Sensors - - - - -	40
Other One-Wire Devices - - - - -	40
 HOST COMMUNICATIONS FOR 635 EMULATION	 40
Through USB - - - - -	40
Through "Full Swing" RS232 - - - - -	40
Packet Structure - - - - -	41
About Handshaking - - - - -	42
Report Codes - - - - -	42
0x80: Key Activity - - - - -	42
0x81: Fan Speed Report (FBSCAB required) - - - - -	43
0x82: Temperature Sensor Report (FBSCAB and Temperature Sensor Accessories required) - - - - -	44
Command Codes - - - - -	44
0 (0x00): Ping Command - - - - -	44
1 (0x01): Get Hardware & Firmware Version and Module Information - - - - -	45
2 (0x02): Write User Flash Area - - - - -	45
3 (0x03): Read User Flash Area - - - - -	45
4 (0x04): Store Current State As Boot State - - - - -	45
5 (0x05): Reset Functions - - - - -	46
Reload Boot Settings - - - - -	46
Reset Host - - - - -	46
Power Off Host - - - - -	47
CFA735 Soft Reboot - - - - -	48
CFA735 Soft Reboot and Settings Reset - - - - -	48
Return Packet for all Five Reset Functions - - - - -	48
6 (0x06): Clear LCD Screen - - - - -	49
9 (0x09): Set LCD Special Character Data - - - - -	49



10 (0x0A): Read 8 Bytes of LCD Memory	49
11 (0x0B): Set LCD Cursor Position	50
12 (0x0C): Set LCD Cursor Style	50
13 (0x0D): Set LCD Contrast	50
14 (0x0E): Set LCD & Keypad Backlight	51
16 (0x10): Set Up Fan Reporting (FBSCAB required)	51
17 (0x11): Set Fan Power (FBSCABrequired)	52
18 (0x12): Read DOW Device Information (FBSCAB required)	52
19 (0x13): Set Up Temperature Reporting (FBSCAB required)	53
20 (0x14): Arbitrary DOW Transaction (FBSCAB required)	53
23 (0x17): Configure Key Reporting	54
24 (0x18): Read Keypad, Polled Mode	54
25 (0x19): Set Fan Power Fail-Safe (FBSCAB required)	55
26 (0x1A): Set Fan Tachometer Glitch Filter (FBSCAB required)	56
27 (0x1B): Query Fan Power & Fail-Safe Mask (FBSCAB required)	56
28 (0x1C): Set ATX Power Switch Functionality	57
FOUR FUNCTIONS ENABLED BY COMMAND 28	58
Function 1: KEYPAD_RESET	58
Function 2: KEYPAD_POWER_ON	58
Function 3: KEYPAD_POWER_OFF	58
Function 4: LCD_OFF_IF_HOST_IS_OFF	58
29 (0x1D): Enable/Disable and Reset the Watchdog	59
30 (0x1E): Read Reporting & Status	59
31 (0x1F): Send Data to LCD	60
33 (0x21): Set Baud Rate	60
34 (0x22): Set or Set and Configure GPIO Pin	61
35 (0x23): Read GPIO Pin Levels and Configuration State	63

CHARACTER GENERATOR 65

MODULE RELIABILITY AND LONGEVITY 66

Module Reliability	66
Module Longevity (EOL / Replacement Policy)	66

CARE AND HANDLING PRECAUTIONS 67

ESD (Electro-Static Discharge) Specifications	67
“Full Swing” RS232 Serial Interface	67
USB Interface	67

Appendix A: Quality Assurance Standards 70

Appendix B: Demonstration Software and Sample Code 76

Demonstration Software	76
------------------------	----



Algorithms to Calculate the CRC	76
Algorithm 1: "C" Table Implementation	76
Algorithm 2: "C" Bit Shift Implementation	77
Algorithm 2B: "C" Improved Bit Shift Implementation	79
Algorithm 3: "PIC Assembly" Bit Shift Implementation	79
Algorithm 4: "Visual Basic" Table Implementation	81
Algorithm 5: "Java" Table Implementation	82
Algorithm 6: "Perl" Table Implementation	84
Algorithm 7: For PIC18F8722 or PIC18F2685	85

Preliminary



LIST OF FIGURES

Figure 1. Module Outline Drawing Front and Side Views -----	14
Figure 2. Module Outline Drawing Back View and Pixel Detail -----	15
Figure 3. Panel Mounting Application Cutout Drawing -----	16
Figure 4. Keypad Detail Drawing -----	17
Figure 5. System Block Diagram -----	18
Figure 6. Definition of Operation Voltage (V_{OP}) (Negative Image)-----	26
Figure 7. Definition of Operation Voltage (V_{OP}) (Positive Image)-	26
Figure 8. Definition of Response Time (T_r , T_f) (Negative Image)-	27
Figure 9. Definition of Response Time (T_r , T_f) (Positive Image) -	27
Figure 10. Definition of Horizontal and Vertical Viewing Angles ($CR>2$)-	28
Figure 11. Definition of 6:00 O'Clock and 12:00 O'Clock Viewing Angles-	28
Figure 12. Location of the CFA735 Three Connectors -----	31
Figure 13. Standard (+5v) Power Supply and USB Data Communications through USB	32
Figure 14. ATX Power Supply and Control Connections Using CrystalFontz WR-PWR-Y25 Cable	34
Figure 15. Photo of CFA-RS232 mounted on CFA735-xxx-KT -----	34
Figure 16. Top View of CFA-RS232 -----	34
Figure 17. Bottom View of CFA-RS232 -----	35
Figure 18. CFA-RS232 Serial Converter (Side View)-----	35
Figure 19. CFA-RS232 Serial Converter Connectors J1, J2, and J3 -----	36
Figure 20. CFA-RS232 J1 Connector Default RS232 Pin Assignments -----	37
Figure 21. CFA-RS232 J1 Connector Alternate RS232 Pin Assignments-----	38
Figure 22. CFA-RS232 J2 Connector Pin Assignments -----	38
Figure 23. -----	39
Figure 24. CFA735 Module Connected to Optional FBSCAB with WR-EXT-Y37 Cable	39
Figure 25. Character Generator-----	65



**This Data Sheet has information for the three CFA735-xxx-KT variants:
 CFA735-TFK-KT, CFA735-TML-KT, and CFA735-YYK-KT.**

INTRODUCTION

The CFA735 family has three color choices. All variants can use a USB and a serial interface (TTL “logic level” serial” or “full swing” RS232 serial) simultaneously. Modules with “full swing” RS232 serial have a mounted [CFA-RS232 Serial Converter](#) board.

TTL “Logic Level” Serial and USB	CFA735-TFK-KR	CFA735-TML-KR	CFA735-YYK-KR
“Full Swing” RS232 Serial and USB	CFA735-TFK-KT	CFA735-TML-KT	CFA735-YYK-KT

When the information in this Data Sheet applies to all modules, the term “CFA735” is used.

DIFFERENCE BETWEEN THE TWO SERIAL INTERFACES

Both of the two serial interfaces use firmware that bring the two UART pins (Tx & Rx) of the CFA735's microcontroller to the CFA735's H1 connector.

TTL “Logic Level” Serial (Sold as CFA735-xxx-KR)

The CFA735-xxx-KR exposes the UART Tx & Rx (inverted, logic level, 0v to 5v nominal) signals on pin 1 and pin 2 of the CFA735's expansion connector H1. If your embedded processor is close to the CFA735, you can cable its UART Rx and Tx pins directly to the CFA735-xxx-KR's Tx and Rx pins. No RS232 level translators are required on either end.

“Full Swing” Serial (Sold as CFA735-xxx-KT)




The CFA735-xxx-KT is a CFA735-xxx-KR with a mounted RS232 level converter board (CFA-RS232). The CFA735-xxx-KT is the correct choice if your embedded controller or host system has a “real” RS232 serial port (-10v to +10v “full swing” serial interface, typically through a UART).



EXPLANATION OF PART NUMBER CODES IN THIS DATA SHEET

$\frac{\text{CFA}}{\text{①}} \quad \frac{\text{735}}{\text{②}} \quad - \quad \frac{\text{X}}{\text{③}} \quad \frac{\text{X}}{\text{④}} \quad \frac{\text{X}}{\text{⑤}} \quad - \quad \frac{\text{K}}{\text{⑥}} \quad \frac{\text{T}}{\text{⑦}}$

①	Brand	CFA – CrystalFontz America, Inc.
②	Model Identifier	735
③	Backlight Type & Color	T – LED, white Y – LED, yellow-green
④	Fluid Type, Image (positive or negative), & LCD Glass Color	F – FSTN, positive M – STN, negative, blue Y – STN, positive, yellow-green
⑤	Polarizer Film Type, Temperature Range, & View Angle (O’Clock)	K – Transflective, Wide Temperature -20°C to +70°C, 12:00 L – Transmissive, Wide Temperature -20°C to +70°C, 12:00
⑥	Special Code	K – Manufacturer’s code
⑦	Interface Code	T – RS232 “Full Swing” Serial <i>and</i> USB Bidirectional 115200 baud ESD protected RS232 serial interface is provided by the included serial conversion board (named CFA-RS232 v1.0 Serial Converter) when connected with the appropriate cables. (See CFA-RS232 Serial Converter for RS232 “Full Swing” (Pg. 34)). Full-speed USB interface is available simultaneously.

PART NUMBER CONVENTION	 CFA735-TFK-KT	 CFA735-TML-KT	 CFA735-YYK-KT
Fluid	FSTN	STN	STN
LCD Glass Color	neutral	blue	yellow-green
Image	positive	negative	positive
Polarizer Film	transflective	transmissive	transflective
Edge-Lit LED Backlights	LCD: white Keypad: white	LCD: blue Keypad: white	LCD: yellow-green Keypad: yellow-green
<p><i>Notes:</i> Positive Image = Sunlight readable and also readable in dark areas. Negative Image = Not recommended for use in sunlight; may be washed out. LED backlit keypad with six buttons is made of translucent silicon.</p>			

COMPARISON OF CFA735 FAMILY AND THE PREVIOUS CFA635 FAMILY

Using CFA635 emulation, the CFA735 is mechanically and code-compatible with the CFA635 family. An EOL (End of Life) notice was issued for the CFA635. See [Part Change Notice #10365](#). A comparison of the CFA735 family and the previous CFA635 family can be found in the migration document [Technical Bulletin #10377](#).

The CFA635 required different firmware for USB interface. The CFA735 supports serial *and* USB interface with the same firmware. *The CFA735 USB driver is not the same as the CFA635 USB driver. You will need to install the CFA735 USB*



driver. Command structure and communication packet structure are the same. Except for Command 22 (0x16): Send Command Directly to the LCD Controller, the CFA735 firmware emulates the CFA635.

ADDITIONAL MODULE FEATURES

- Large easy-to-read 20 characters x 4 lines in a compact size. Fits nicely in a 1U rack mount case (37.08 mm overall height). Choice of 3 colors. Edge-lit display is backlight with 12 LEDs, 6 per side. Attractive stainless steel bezel.
- Six-button translucent silicon keypad with screened legend is backlit with white LEDs. Fully decoded keypad: any key combination is valid and unique.
- Only a single supply is needed. Wide power supply voltage range ($V_{DD} = +3.3v$ to $+5.5v$) is perfect for embedded systems.
- Adjustable backlight and contrast. Backlight and contrast are fully voltage compensated over the power supply range. No adjustments to the contrast setting or backlight brightness are needed.
- DAC (Digital-to-Analog Converter) controls the constant current LED driver.
- We work to continuously improve our products, including backlights that are brighter and last longer. Slight color variations from module to module and batch to batch are normal. If you need modules with consistent color, please ask for a custom order.
- Four bicolor (red + green) LED status lights. Using constant current LED driver, the LEDs' brightness can be set by the host software, which allows smoothly adjusting the LEDs to produce other colors (for example, yellow and orange).
- The CFA735 is powered by an ST-Micro STM32F103 series 32-bit ARM-based microcontroller and Sitronix ST7529 driver/controller.
- The microSD card slot may be used for future firmware updates. Firmware updates are announced through our PCN (Part Change Notices). To ensure that the appropriate people in your organization receive notices, please ask them to subscribe at www.crystalfontz.com/news/pcn.php. **Leave the supplied dummy microSD card in the slot.** If you remove the dummy card and the slot is empty when the module is powered on, you can irreparably damage the module.
- Robust packet based communications protocol with 16-bit CRC.
- Nonvolatile memory (flash) capability:
 - Customize the "power-on" display settings.
 - 16-byte "scratch" register for storing IP, netmask, system serial number, etc.
- Optional ATX functionality allows the keypad buttons to replace the power and reset switches on your system, simplifying front panel design. The 16-pin Crystalfontz [WR-PWR-Y25](#) ATX power switch cable may be used for direct connection to the host's power supply.
- Hardware watchdog can reset host on host software failure.
- Firmware support for the optional Crystalfontz [FBSCAB](#) (FB System Cooling Accessory Board). For more information, see [Connect Optional FBSCAB \(Pg. 39\)](#). The combination of the CFA735 with the optional FBSCAB (CFA735+FBSCAB) allows:
 - Control up to four fans with RPM monitoring. Fail-safe fan power settings allows host to safely control four fans based on temperature.
 - You can add DOW (Dallas One-Wire) devices including the Crystalfontz [WR-DOW-Y17](#) that has a DS18B20 temperature sensor. Monitor temperatures up to 16 channels at up to 0.5°C absolute accuracy.
 - RS232 to DOW bridge functionality allows control of other One-Wire compatible devices (ADC, voltage monitoring, current monitoring, RTC, GPIO, counters, identification/encryption). Additional hardware required.
- RoHS compliant.
- Factories have ISO certification.
- Product materials are in compliance with the regulations related to the EU Directive 2006/121/EC for Registration, Evaluation, Authorization and Restriction of Chemicals (REACH).



ACCESSORIES

These CFA735-xxx-KT variants are also sold as a part of a kit that includes three cables:

Three kit choices.	Three serial cables are in each kit.
CFA735-TFK-KT1	1. WR-232-Y08 : RS232 DB9 female to 0.1" 2x5 female, 27" ribbon
CFA735-TML-KT1	cable for a PC's 9-pin serial port.
CFA735-YYK-KT1	2. WR-232-Y22 : 0.1" 2x5 (female) to 0.1" 2x5 (female) x2 standard/
	alternate pinout. 10-pin to 10-pin for a 10-pin header.
	3. WR-PWR-Y24 : PC power supply to 16-pin connector. Connect
	power directly from the power supply.

In addition to cables, we also sell these accessories for the CFA735-xxx-KT:

- 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 CFA735 module, an [FBSCAB](#) (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 thick hard-coated polycarbonate. Overlay choices are black brushed anodized aluminum, silver brushed anodized aluminum, beige plastic, and black plastic.

These accessories will be available in different kit configurations here: https://www.crystalfontz.com/products/select_kit.html.

Preliminary



MECHANICAL SPECIFICATIONS

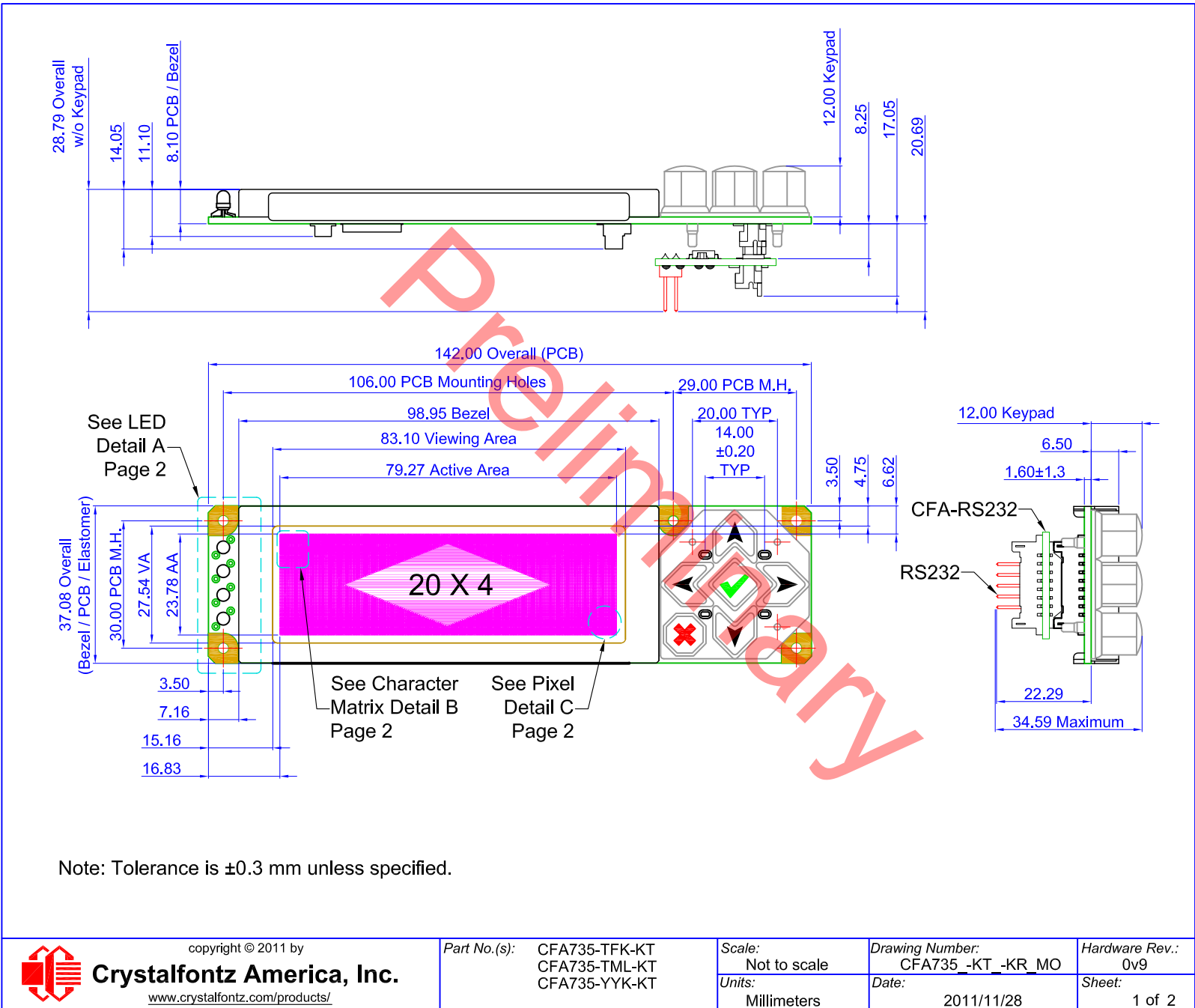
PHYSICAL CHARACTERISTICS

PHYSICAL CHARACTERISTIC	SIZE
Module Overall Dimensions	
Width and Height	142.0 (W) mm x 37.08 (H) mm
Module Depth (Thickness includes CFA-RS232 mounted to back of module): without Keypad, with Connectors with Keypad, with Connectors	28.79 mm nominal 34.29 mm nominal 34.59 mm maximum
Viewing Area	83.10 (W) x 27.54 (H) mm <i>In CFA635 emulation:</i> 82.95 (W) x 27.50 (H) mm
Active Area	79.27 (W) x 23.78 (H) mm <i>In CFA635 emulation:</i> 77.95 (W) x 22.35 (H) mm
5x7 Standard Character	3.225 (W) x 4.875 (H) mm
6x8 Character Matrix	3.90 (W) x 5.60 (H) mm
Pixel Size	0.300 (W) x 0.325 (H) mm
Pixel Pitch	0.325 (W) x 0.350 (H) mm
Keystroke Travel (approximate)	2.4 mm
Weight	62 grams (typical)



Module Outline Drawing Front and Side Views

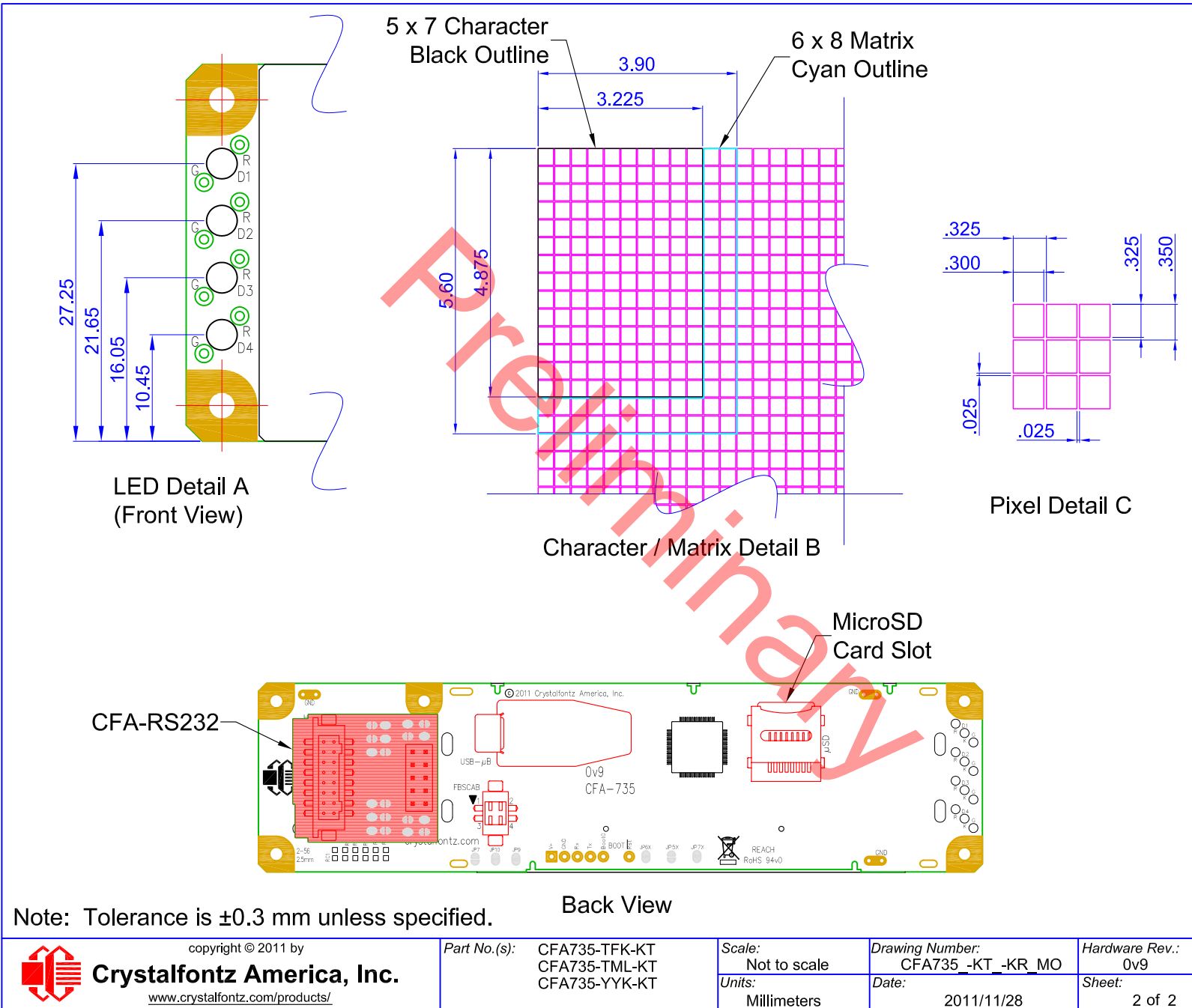
Figure 1. Module Outline Drawing Front and Side Views





Module Outline Drawing Back View and Pixel Detail

Figure 2. Module Outline Drawing Back View and Pixel Detail



copyright © 2011 by
Crystalfontz America, Inc.
www.crystalfontz.com/products/

Part No.(s): CFA735-TFK-KT
 CFA735-TML-KT
 CFA735-YYK-KT

Scale:
 Not to scale
 Units:
 Millimeters

Drawing Number:
 CFA735 _KT _KR_MO
 Date:
 2011/11/28

Hardware Rev.:
 0v9
 Sheet:
 2 of 2



PANEL MOUNTING APPLICATION CUTOUT DRAWING

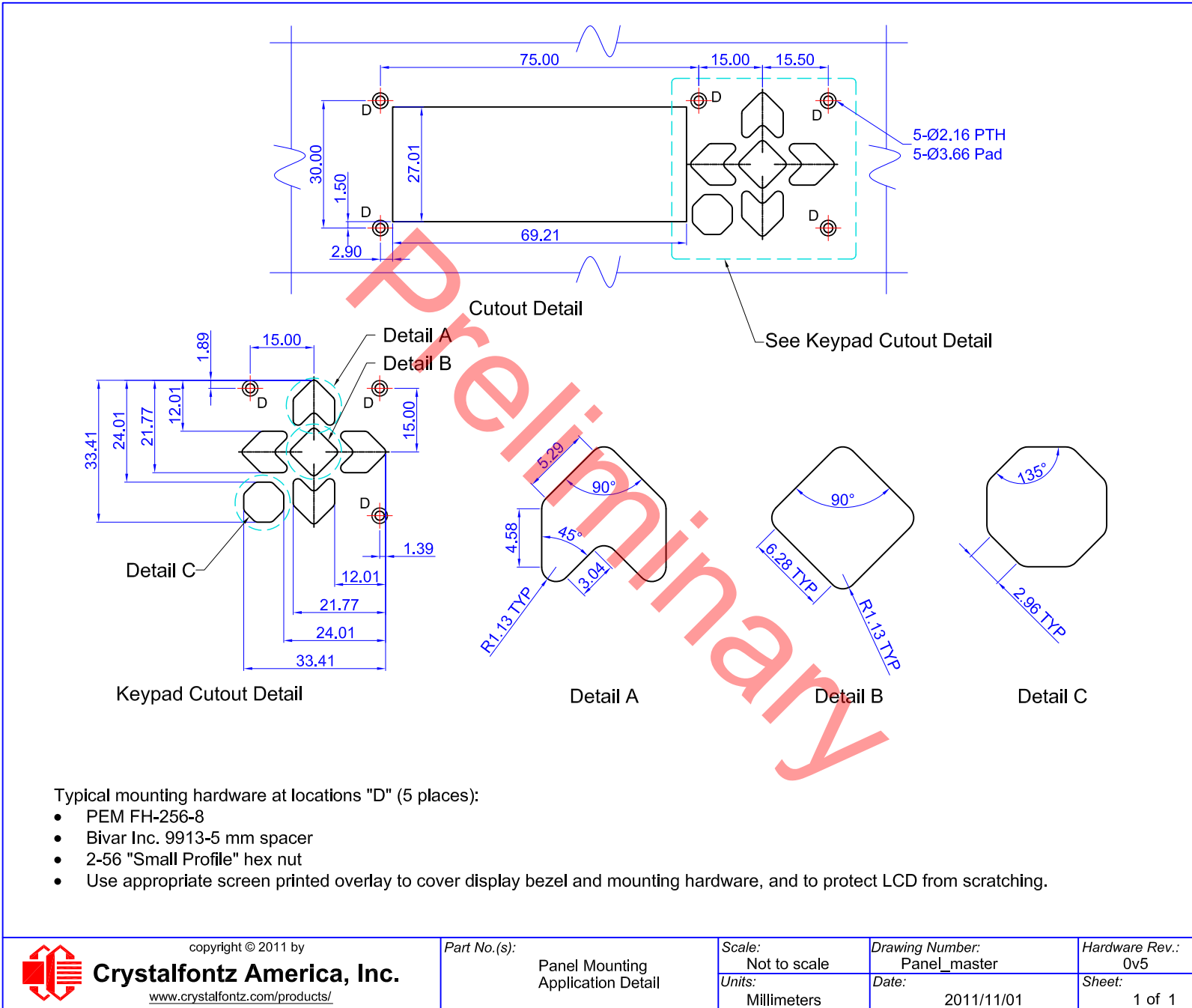


Figure 3. Panel Mounting Application Cutout Drawing



KEYPAD DETAIL DRAWING

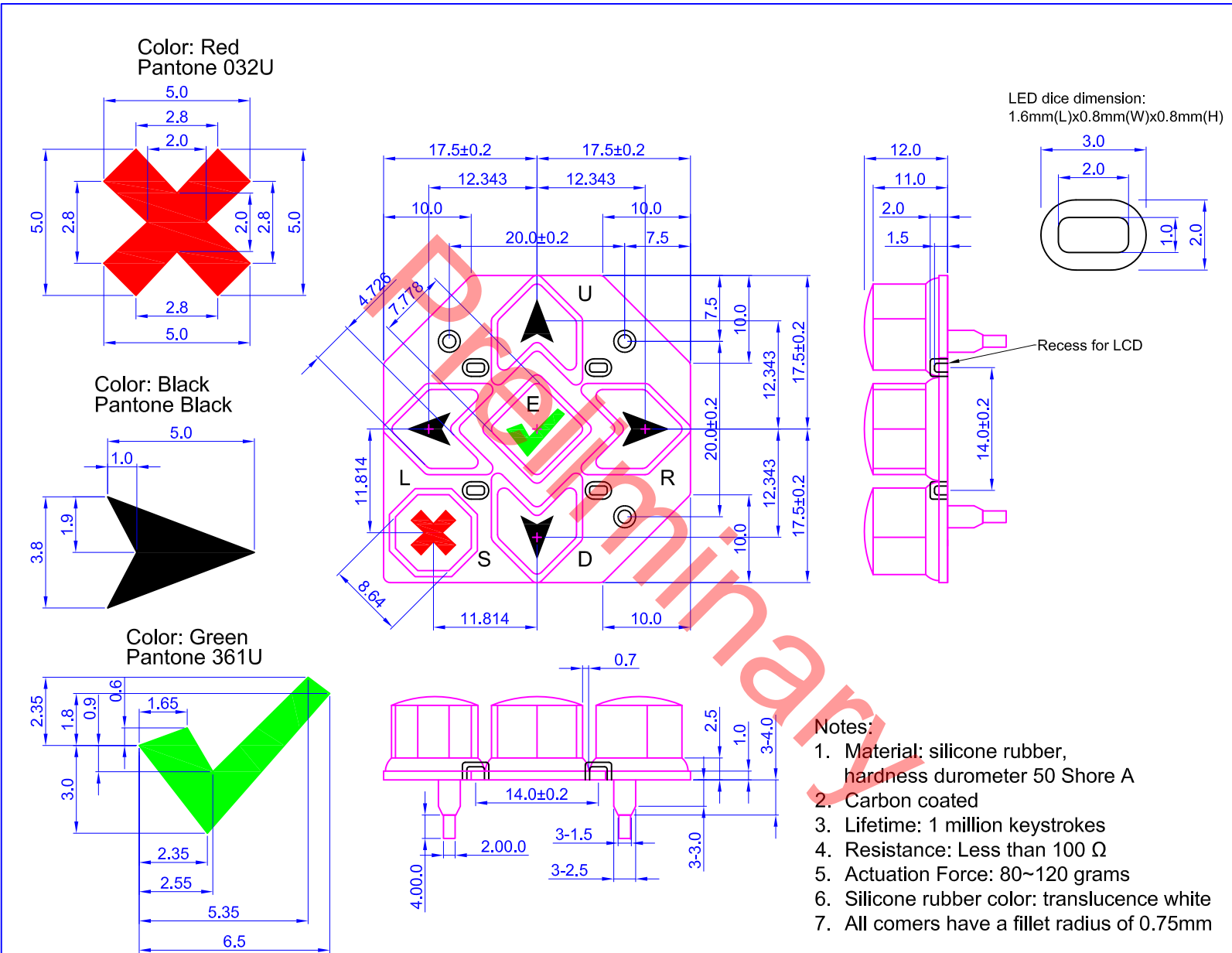



Figure 4. Keypad Detail Drawing

 copyright © 2011 by Crystalfontz America, Inc. www.crystalfontz.com/products/	Part No.(s): CFA735 Family Keypad Detail	Scale: Not to scale	Drawing Number: CFA735_Keypad_Detail	Hardware Rev.: 0v9
		Units: Millimeters	Date: 2011/11/18	Sheet: 1 of 1



ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

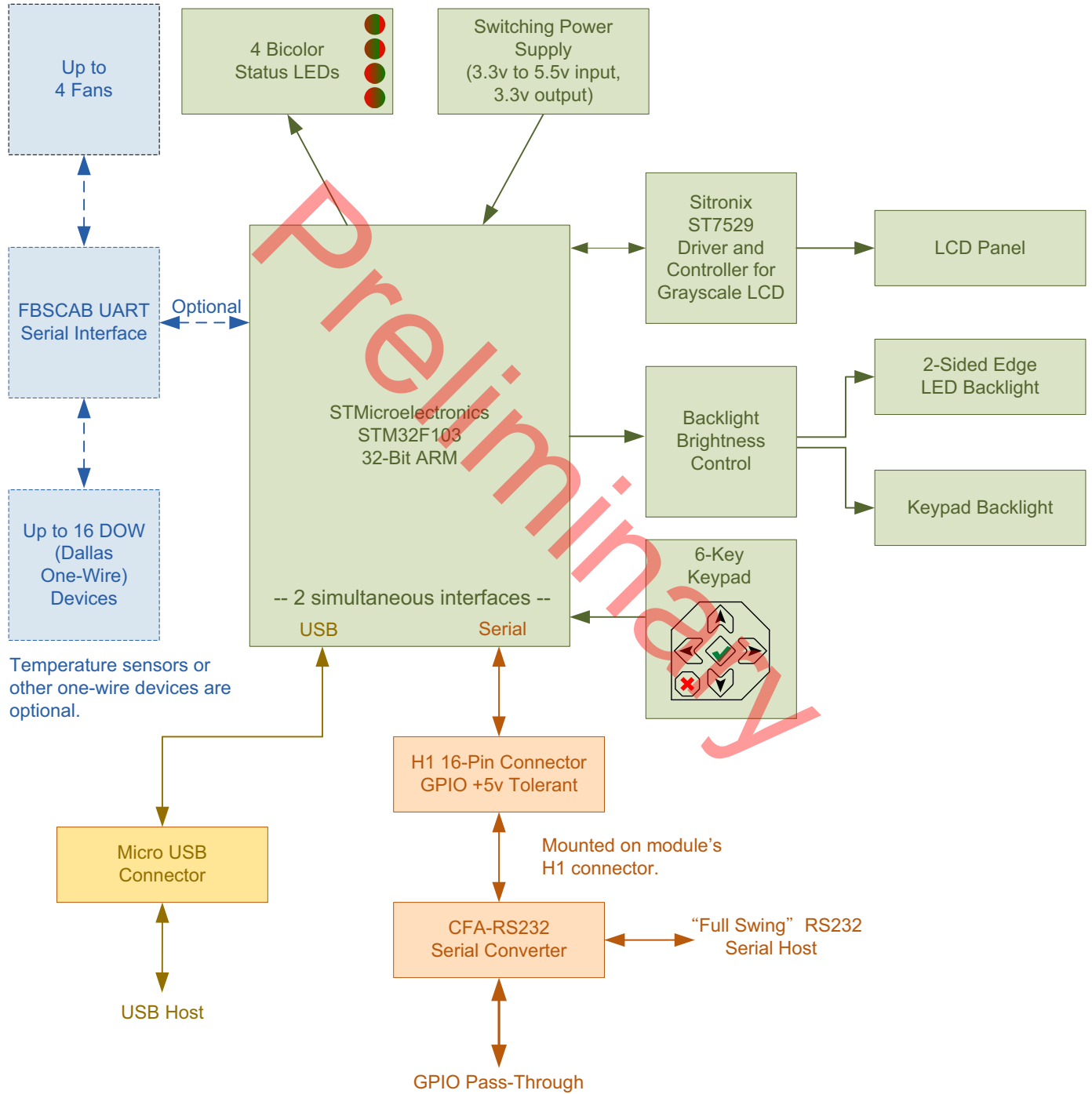


Figure 5. System Block Diagram



LCD DUTY AND BIAS

DRIVING METHOD	SPECIFICATION
Duty	1/32
Bias	6.7

ABSOLUTE MAXIMUM RATINGS

All variants (all colors)

ABSOLUTE MAXIMUM RATINGS	SYMBOL	MINIMUM	MAXIMUM
Operating Temperature	T _{OP}	-20°C	+70°C
Storage Temperature	T _{ST}	-30°C	+80°C
Humidity Range (Noncondensing)	RH	10%	90%
Supply Voltage for Logic	V _{DD}	0v	+5.5v
RS232 Input Pin	V _{RX}	-25v	+25v
RS232 Output Pin	V _{TX}	-13v	+13v
<p><i>Notes:</i> These are stress ratings only. Functional operation of the module at these or any other conditions beyond those listed under DC Characteristics (Pg. 19) is not implied.</p> <p>Extended exposure to the absolute maximum ratings listed above may affect device reliability. Stresses beyond those listed above can cause permanent damage.</p>			

DC CHARACTERISTICS

Input Supply Voltages

- Supply voltage for module: +3.3v minimum to +5.5v maximum. *Do not exceed +5.5v maximum.*
- Internal processor and logic supply voltage: +3.3v. (Developed from module supply voltage with buck/boost converter.)



LOGIC LEVEL GPIO +5 VOLT TOLERANT PINS

	DC CHARACTERISTICS	SYMBOL	MINIMUM	MAXIMUM
CONTROLLER AND BOARD	Input High Voltage	V_{IH}	$0.42*(V_{DD}-2\text{ v})+1\text{v}$ If $V_{DD} = +3.3\text{v}$ = +1.55v	+5.5v
	Input Low Voltage	V_{IL}	-0.3v	$0.32*(V_{DD}-2\text{v})+0.75\text{v}$ If $V_{DD} = +3.3\text{v}$ = +1.17v
	Output High Voltage	V_{OH}	+2.4v	+3.3v
	Output Low Voltage	V_{OL}	+0.4v	+1.3v

TYPICAL GPIO CURRENT LIMITS

These are the typical GPIO current limits when sinking or sourcing all five GPIO pins simultaneously. If you need more information about GPIO current limits, please call Technical Support (888) 206-9720 or (509) 892-1200.

GPIO CURRENT LIMITS	SPECIFICATION
Sink	8 mA
Source	8 mA



TYPICAL CURRENT CONSUMPTION

Variables that affect current consumption include the choice of color, interface type, brightness of backlights, brightness of the four status lights, power supply voltage, and whether an [FBSCAB](#) is attached to the module.



CFA735-TFK-Kx (Black on White)

All interfaces: TTL "Logic Level" Serial, "Full swing" RS232 Serial, and USB

ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION		
Logic	LCD and Keypad Backlights at 100%	All Status LEDs 4 Red + 4 Green at 100%	V _{DD} = +5.0v	V _{DD} = +4.75v	V _{DD} = +3.3v
X	X	X	258 mA	268 mA	369 mA
X	X	-	146 mA	156 mA	234 mA
X	-	X	152 mA	128 mA	186 mA
X	-	-	38 mA	40 mA	58 mA



CFA735-TML-Kx (White on Blue)

All interfaces: TTL "Logic Level" Serial, "Full swing" RS232 Serial, and USB

ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION		
Logic	LCD and Keypad Backlights at 100%	All Status LEDs 4 Red + 4 Green at 100%	V _{DD} = +5.0v	V _{DD} = +4.75v	V _{DD} = +3.3v
X	X	X	240 mA	250 mA	368 mA
X	X	-	142 mA	152 mA	232 mA
X	-	X	152 mA	128 mA	186 mA
X	-	-	38 mA	40 mA	58 mA

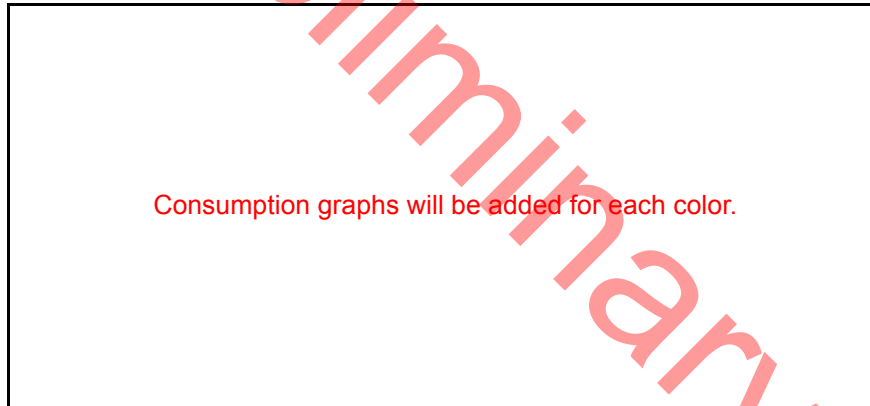


CFA735-YYK-Kx (Black on Yellow-Green)

All interfaces: TTL "Logic Level" Serial, "Full swing" RS232 Serial, and USB

Logic	ITEMS ENABLED		TYPICAL CURRENT CONSUMPTION		
	LCD and Keypad Backlights at 100%	All Status LEDs 4 Red + 4 Green at 100%	V _{DD} = +5.0v	V _{DD} = +4.75v	V _{DD} = +3.3v
X	X	X	276 mA	292 mA	375 mA
X	X	-	164 mA	170 mA	244 mA
X	-	X	152 mA	128 mA	186 mA
X	-	-	38 mA	40 mA	58 mA

SUPPLY CURRENT VS. SUPPLY VOLTAGE OVER BACKLIGHT RANGE





Pull-In and Drop-Out Voltage Specifications

SUPPLY VOLTAGE	MINIMUM	MAXIMUM
Power Supply Voltage (V_{DD})	+3.3v	+5.5v
Pull-in Voltage		+3.2v
Drop-out Voltage		+3.0v

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS (BY INTERFACE)

“Full Swing” RS232 Serial Interface

Tx and Rx pins of connector RS232 only:
+15 kV Human Body Model
+15 kV IEC1000-4-2 Air Discharge
+8 kV IEC1000-4-2 Contact Discharge

The remainder of 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.

USB Interface

D+ and D- pins of USB connector only: Electrostatic Discharge Voltage ($I < 1 \mu A$): +/- 2000 v.

The remainder of 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.



FAN TACHOMETER SPEED RANGE

ADDITIONAL ELECTRICAL CRITERIA	SPECIFICATION
Fan Tachometer Speed Range (assuming two PPR)	600 RPM to 3,000,000 RPM
<i>PPR is Pulses Per Revolution, also written as p/r.</i>	

OPTICAL SPECIFICATIONS

VIEWING DIRECTION

VIEWING DIRECTION
12 O'Clock

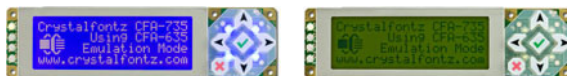
OPTICAL CHARACTERISTICS

CFA735-TFK-Kx



ITEM	SYMBOL	CONDITION	MINIMUM	TYPICAL	MAXIMUM
Viewing Angle (12 o'clock)	Deg $\theta = 90^\circ$	CR \geq 2		35	
	Deg $\theta = 270^\circ$			60	
	Deg $\theta = 0^\circ$			45	
	Deg $\theta = 180^\circ$			45	
Contrast Ratio ¹	CR			3.8	5.0
LCD Response Time ²	T rise	Ta = 25°C		180 ms	
	T fall	Ta = 25°C		200 ms	

¹Contrast Ratio = (brightness with pixels light)/(brightness with pixels dark).
²Response Time: The amount of time it takes a liquid crystal cell to go from active to inactive or back again.



CFA735-TML-Kx and CFA735-YYK-Kx

ITEM	SYMBOL	CONDITION	MINIMUM	TYPICAL	MAXIMUM
Viewing Angle (12 o'clock)	Deg $\theta = 90^\circ$	CR \geq 2		30	
	Deg $\theta = 270^\circ$			40	
	Deg $\theta = 0^\circ$			30	
	Deg $\theta = 180^\circ$			30	
Contrast Ratio ¹	CR			3.8	5.0
LCD Response Time ²	T rise	Ta = 25°C		180 ms	
	T fall	Ta = 25°C		200 ms	

¹Contrast Ratio = (brightness with pixels light)/(brightness with pixels dark).
²Response Time: The amount of time it takes a liquid crystal cell to go from active to inactive or back again.

TEST CONDITIONS AND DEFINITIONS FOR OPTICAL CHARACTERISTICS

We work to continuously improve our products, including backlights that are brighter and last longer. Slight color variations from module to module and batch to batch are normal. If you need modules with consistent color, please ask for a custom order.

- Viewing Angle
 - Vertical (V) θ : 0°
 - Horizontal (H) ϕ : 0°
- Frame Frequency: 78 Hz
- Driving Waveform: 1/16 Duty, 1/13 Bias
- Ambient Temperature (Ta): 25°C

Definition of Operation Voltage (V_{OP})



CFA735-TML-Kx 

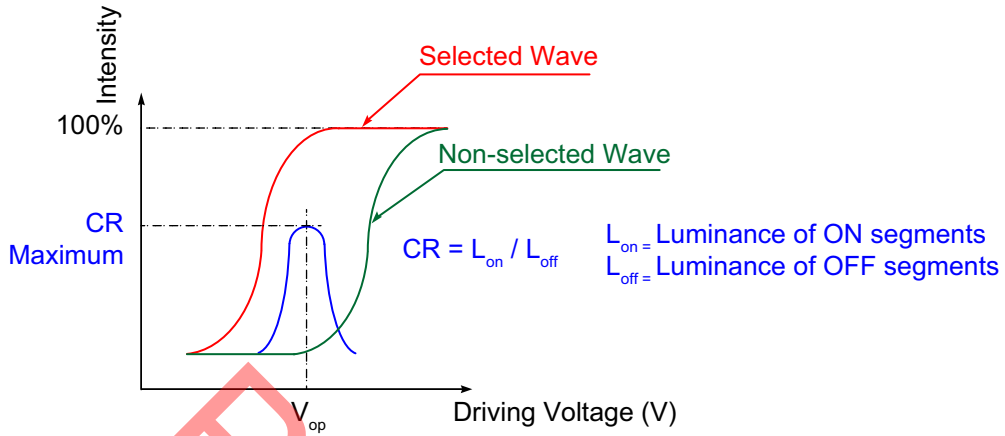




Figure 6. Definition of Operation Voltage (V_{OP}) (Negative Image)

CFA735-TFK-Kx  and CFA735-YYK-Kx 

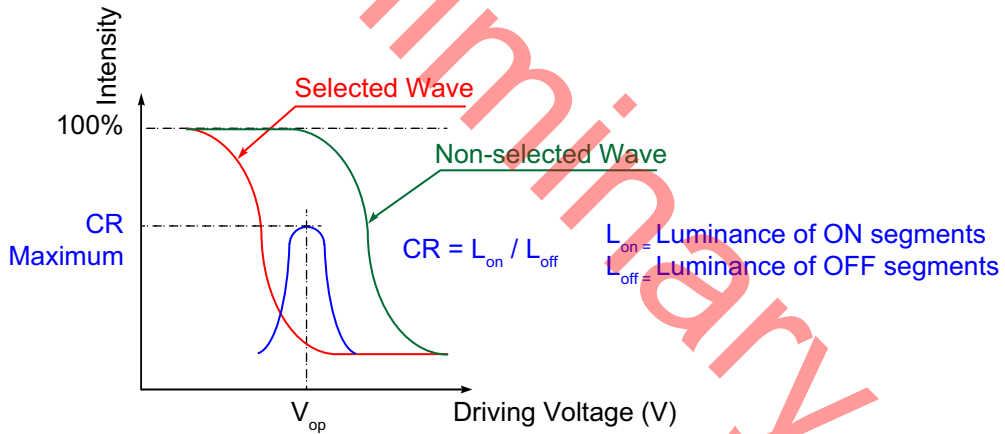


Figure 7. Definition of Operation Voltage (V_{OP}) (Positive Image)



Definition of Response Time (T_r , T_f)

CFA735-TML-Kx 

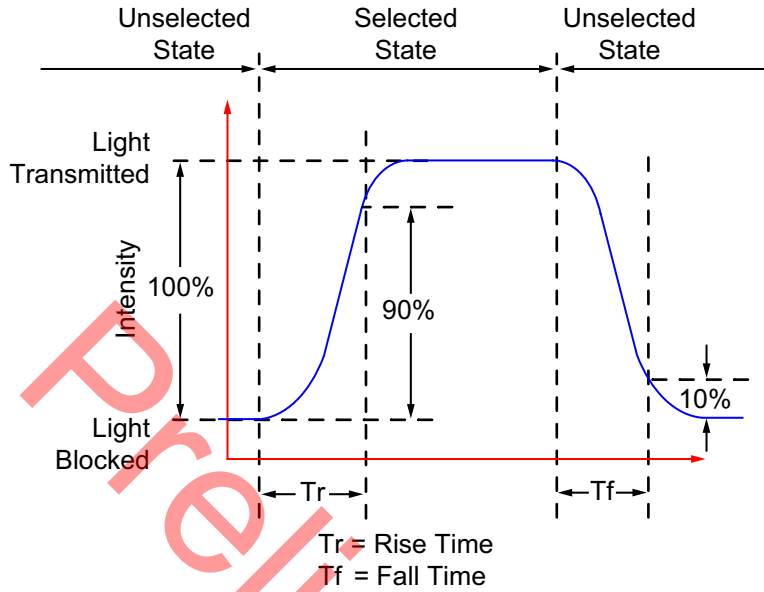




Figure 8. Definition of Response Time (T_r , T_f) (Negative Image)

CFA735-TFK-Kx  and CFA735-YYK-Kx 

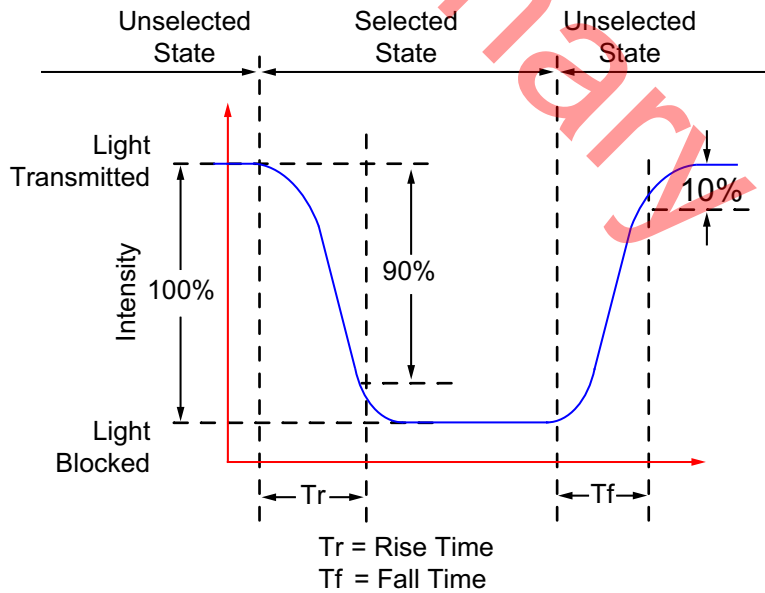


Figure 9. Definition of Response Time (T_r , T_f) (Positive Image)



Definition of Vertical and Horizontal Viewing Angles (CR_≥2)

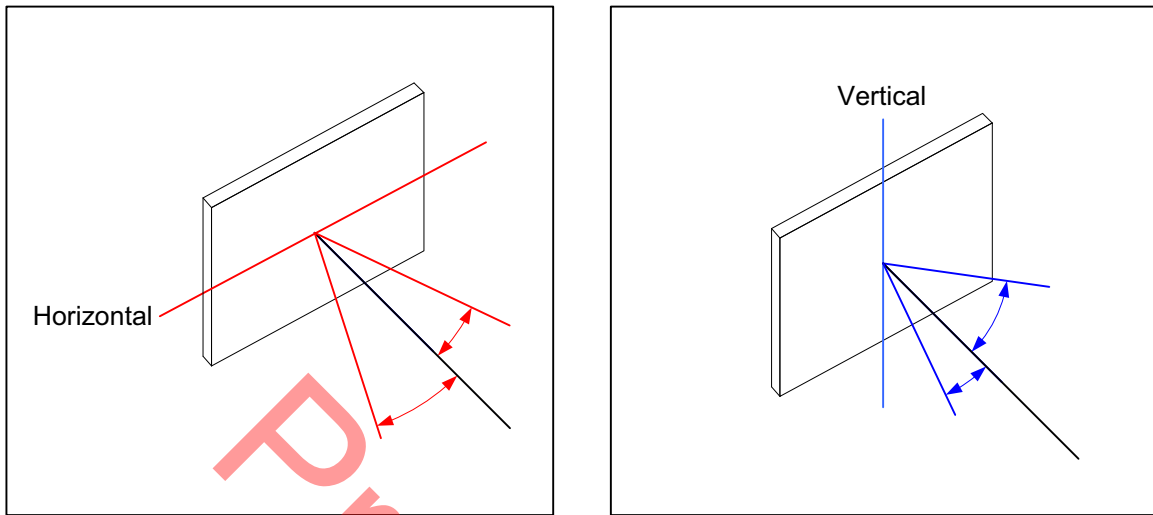


Figure 10. Definition of Horizontal and Vertical Viewing Angles (CR_≥2)

Definition of 6 O'Clock and 12:00 O'Clock Viewing Angles

This LCD module has a 12:00 o'clock viewing angle. A 6:00 o'clock viewing angle is a bottom viewing angle like what you would see when you look down at a cell phone or calculator. A 12:00 o'clock viewing angle is a top viewing angle like what you would see when you look up at the gauges in a vehicle.

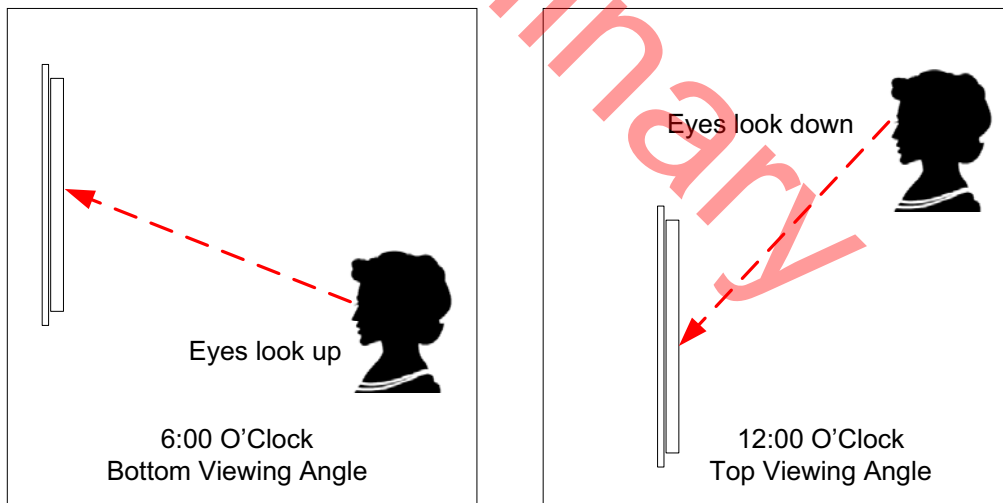


Figure 11. Definition of 6:00 O'Clock and 12:00 O'Clock Viewing Angles



BACKLIGHT INFORMATION

Backlight control is by DAC (Digital-to-Analog Converter) controlling the constant current LED driver. The LCD and keypad backlights are independently controlled.

NOTE

For modules with **white** backlights, we recommend that the backlight be dimmed or turned off during periods of inactivity to conserve the LEDs' lifetime.

Preliminary



CONNECTION INFORMATION

CABLES

Below this table are sections that describe different common connection configurations. Cable lengths in table below are approximate.

CrystalFontz Cable	Description All cables are RoHS compliant
USB	
WR-USB-Y27 6 feet	Connect cable's micro-B USB female connector to CFA735's micro-B USB connector. Connect cable's USB-A female connector to host's USB-A connector.
WR-USB-Y34 27.5 inches	Connect cable's micro-B USB female connector to CFA735's micro-B USB connector. Connect cable's single piece 4-pin 0.1" female connector to USB pins on host's motherboard.
<p><i>Note:</i> Keep the micro-B USB cable connector parallel to the CFA735 when plugging or unplugging the cable. Do not lift or pull up on the cable. Too much pressure may permanently damage the CFA735's micro-B USB connector.</p>	
"Full Swing" RS232 Serial	
WR-232-Y08 27 inches	Use this cable to supply communications to the CFA735 through the mounted CFA-RS232 converter. Connect cable's 10-pin female connector to CFA-RS232's J1 connector. Connect cable's RS232 DB9 9-pin female connector to host's external 9-pin serial port.
WR-232-Y22	Use this cable to supply communications to the CFA735 through the mounted CFA-RS232 converter. Connect cable's 0.1" 2x5 female connector to the CFA-RS232's J1 10-pin connector. Connect cable's second 0.1" 2x5 female connector to host's motherboard 10-pin connector (standard/alternate pinout).
WR-PWR-Y24 26 inches	Use this cable to supply power to the CFA735 through the mounted CFA-RS232 converter from the host's power supply. Connect cable's 16-pin female connector to CFA-RS232's J2 connector. Connect cable's 2-pin male connector directly to the host's power supply connector.
<p><i>Note:</i> See Connectors on CFA-RS232 Serial Converter (Pg. 35) for location of J1 and J2.</p>	
ATX Functionality Use CFA735 to power On, Power Off, or Reset the host with or without FBSCAB accessory.	
WR-PWR-Y25 11 inches	Connect cable's 16-pin female connector to CFA735's H1 connector. Connect cable's male connector directly to host's ATX power supply. Or connect cable's 4 separate female connectors to the 4 pins on host's motherboard.
WR-PWR-Y25 11 inches	Connect cable's 16-pin female connector to CFA-RS232's J2 connector. Connect cable's male connector directly to host's ATX power supply. Or connect cable's 4 separate female connectors to the 4 pins on host's motherboard.
<p><i>Note:</i> When using the CFA735 with an FBSCAB and ATX functionality, the ATX control is directly from the CFA735 and not from the FBSCAB. There is no GPIO pass through to the FBSCAB. Instead, the H1 connector on the CFA735 is used for GPIO.</p>	
FBSCAB (FB System Cooling Accessory Board)	
<p><i>Note:</i> The CFA735 does not supply power to the FBSCAB. The FBSCAB requires external power, typically supplied by a 4-pin 3.5-inch floppy drive power connector.</p>	
WR-PWR-Y12 13.3 inches	4-pin hard drive to floppy connector and hard drive splitter power cable. Power to FBSCAB. Requires +5v and +12v for fans.



Crystalfontz Cable	Description All cables are RoHS compliant
WR-EXT-Y37 18 inches	Connect cable's 4-pin male connector to the CFA735's connector labeled FBSCAB. Connect cable's 4-pin female connector to the FBSCAB. The connector on the FBSCAB has the top left pin labeled Rx2.
WR-FAN-X01 16 inches	Connect up to four cables to connect up to four fans. Connect cable's 3-pin male connector to FBSCAB's connectors labeled FAN1, FAN2, FAN3, or FAN4. Connect cable's 3-pin female connector to a fan's connector. (Fans are not sold by Crystalfontz.)
WR-DOW-Y17 12 inches + 12 inches between connectors	Connect up to sixteen of this Dallas One-Wire DS18B20 temperature sensor cable to one FBSCAB. ("Daisy chain".) Connect the cable's 3-pin female connector to the FBSCAB's connector labeled J_DOW. If desired, connect the cable's 3-pin male connector to an additional temperature sensor.

THREE CONNECTORS ON THE CFA735

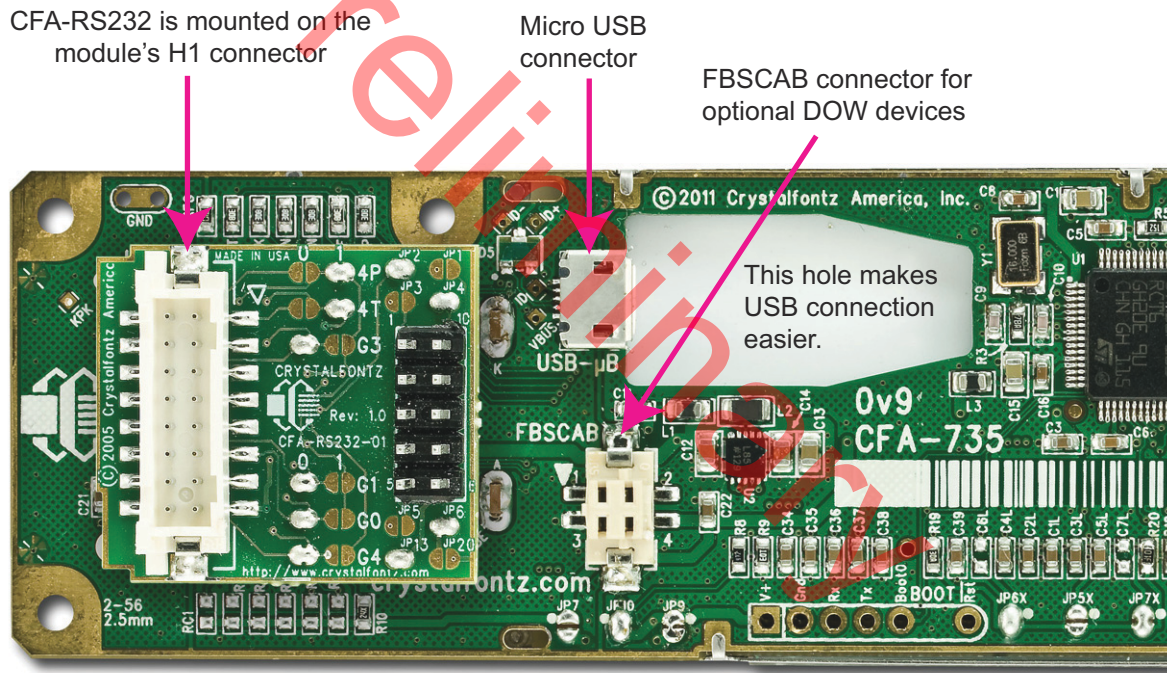


Figure 12. Location of the CFA735 Three Connectors

The module has three connectors on the back of the PCB: H1, USB, and [FBSCAB](#).

Note: The PCB pads labeled EXPANSION and BOOT are reserved for factory testing and programming.

Note: JP10 on the CFA735 is closed by factory default. If you are going to use USB interface AND power through H1, you must open JP10 to prevent backpowering the USB.

USB Connector

The USB connector is a micro USB 5-pin (F) B type. You may connect the module to one host using a USB interface while at the same time using a serial interface to a second host. See [Standard \(+5v\) Power Supply and Data Communications through USB \(Pg. 32\)](#). *Note:* Keep the micro-B USB cable connector parallel to the CFA735 when



plugging or unplugging the cable. Do not lift or pull up on the cable. Too much pressure may permanently damage the CFA735's micro-B USB connector.

FBSCAB Connector

The CFA735 has an optional system cooling accessory board [FBSCAB](#). Fans and DOW (Dallas One-Wire) devices such as temperature sensors may be connected to the FBSCAB. See [Connect Optional DOW \(Dallas One-Wire\) Devices to FBSCAB \(Pg. 40\)](#).

H1 Connector for CFA-RS232, "Full Swing" RS232 Serial Interface

A CFA-RS232 converter board is mounted to the CFA735's H1 connector. For more information, see [CFA-RS232 Serial Converter for RS232 "Full Swing" \(Pg. 34\)](#).

STANDARD (+5V) POWER SUPPLY AND DATA COMMUNICATIONS THROUGH USB

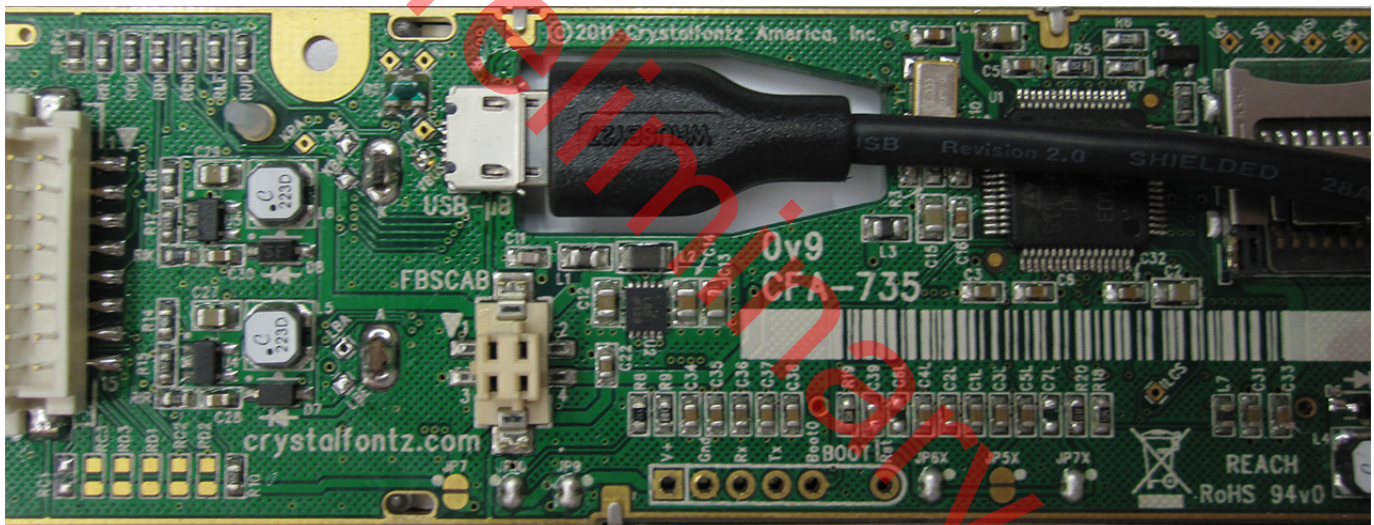


Figure 13. Standard (+5v) Power Supply and USB Data Communications through USB

By using the micro-B USB connector, the CFA735-xxx-KT requires only this one connection to the host for both data communications and power supply.

The micro-B USB connection and the hole in the PCB keeps the CFA735-xxx-KT profile as thin as possible.

Note: Keep the micro-B USB cable connector parallel to the CFA735-xxx-KT when plugging or unplugging the cable. Do not lift or pull up on the cable. Too much pressure may permanently damage CFA735-xxx-KT's micro-B USB connector.



ATX POWER SUPPLY POWER AND CONTROL CONNECTIONS

ATX power supply control functionality allows the buttons on the CFA735-xxx-KT to replace the power and reset button on your system, simplifying front panel design.

NOTE TO CUSTOMERS USING OPTIONAL FBSCAB

When using ATX functionality with the optional [FBSCAB](#) (System Cooling Accessory Board), ATX control is directly from the CFA735 and not from the FBSCAB. There is no GPIO pass through to the FBSCAB. Instead, the H1 connector on the CFA735 is used for GPIO.

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 \(Pg. 61\)](#).

GPIO[1] ATX Host Power Sense

Since the CFA735-xxx-KT 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 \(Pg. 57\)](#).

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 \(Pg. 57\)](#). This connection is also used for the hardware watchdog.

ATX Power Supply & Control Connections	Pins on Connector H1
V _{SB} (+5v)	Pin 16
Ground	Pin 15
GPIO[1] ATX Host POWER SENSE	Pin 12
GPIO[2] ATX Host POWER CONTROL	Pin 9
GPIO[3] ATX Host RESET CONTROL	Pin 10



Below is an illustration of how the optional Crystalfontz [WR-PWR-Y25](#) cable connects to the CFA735's connector H1 and your system motherboard and ATX power supply:

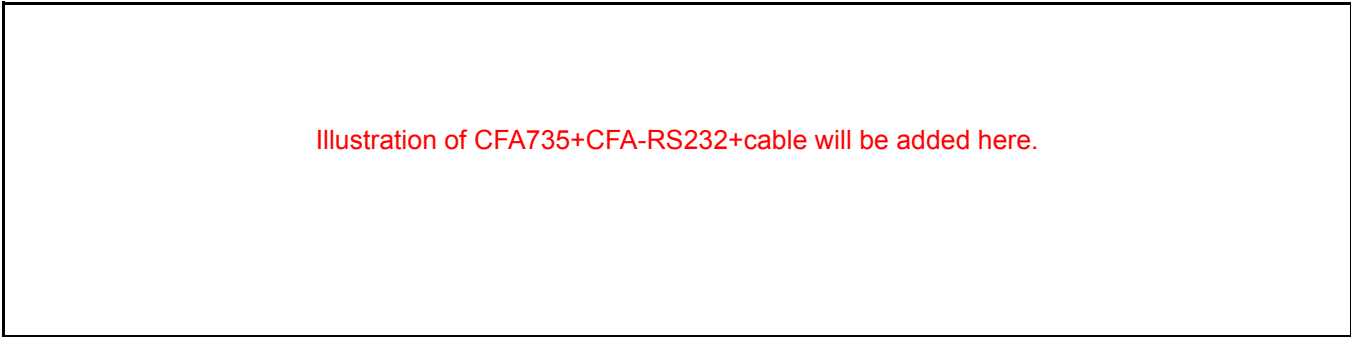


Figure 14. ATX Power Supply and Control Connections Using Crystalfontz WR-PWR-Y25 Cable

CFA-RS232 SERIAL CONVERTER FOR RS232 “FULL SWING”

Figure 15. Photo of CFA-RS232 mounted on CFA735-xxx-KT

The “Full Swing” RS232 interface on the CFA735-xxx-KT consists of two parts:

1. CFA735-xxx-KR Serial LCD Module
2. CFA-RS232 Serial Converter

The CFA-RS232 Serial Converter is a small printed circuit assembly mounted on the CFA735-TFK-KR LCD module. It has a 16-pin female connector J3 (see location of J3 connector on [Figure 19. on Pg. 36](#)) that mates with the male 16-pin connector H1 on the back of the CFA735-xxx-KT LCD module. The CFA-RS232 serial converter converts the 0v to +5v (logic level) Rx and Tx signals from the module's microcontroller to RS232 levels.

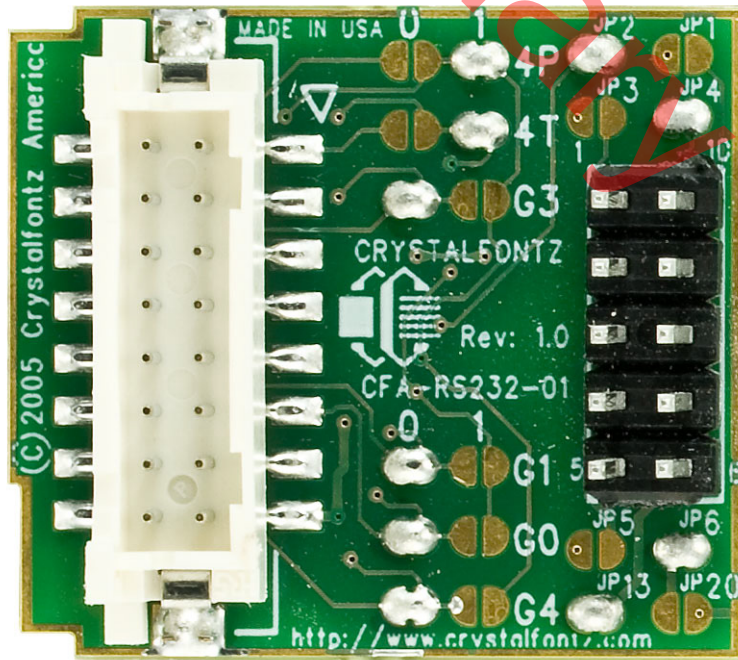
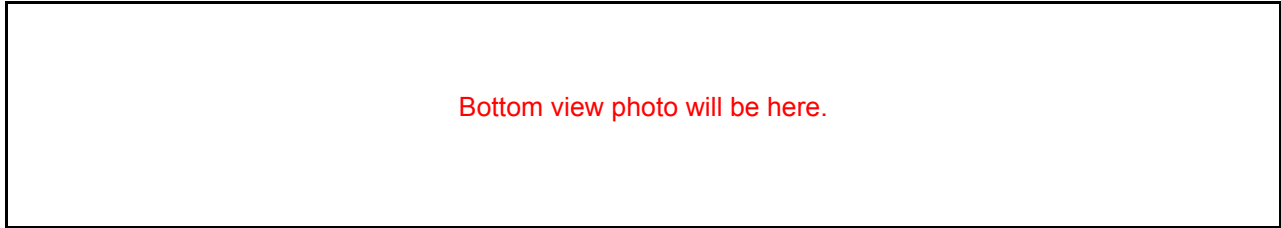


Figure 16. Top View of CFA-RS232



Bottom view photo will be here.

Figure 17. Bottom View of CFA-RS232

Connectors on CFA-RS232 Serial Converter

The top side of the CFA-RS232 serial converter has the Crystalfontz logo silk-screened onto it and has two connectors. The bottom side of the CFA-RS232 does not have the Crystalfontz logo and has one connector.

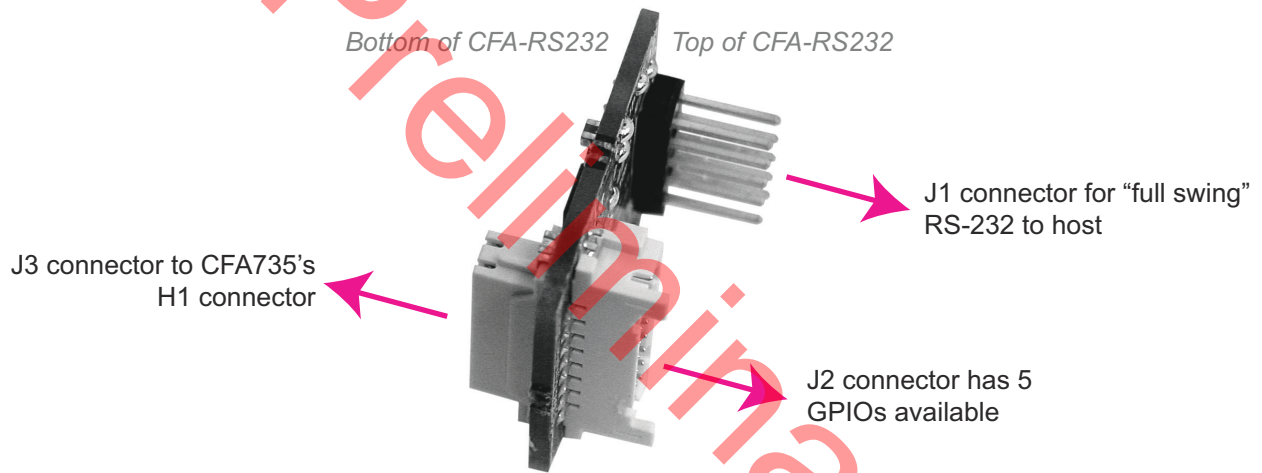


Figure 18. CFA-RS232 Serial Converter (Side View)



The J1 and J2 connectors are on the top side of the mounted CFA-RS232, facing away from the module. The J3 connector is on the bottom of the mounted CFA-RS232, facing towards the module.

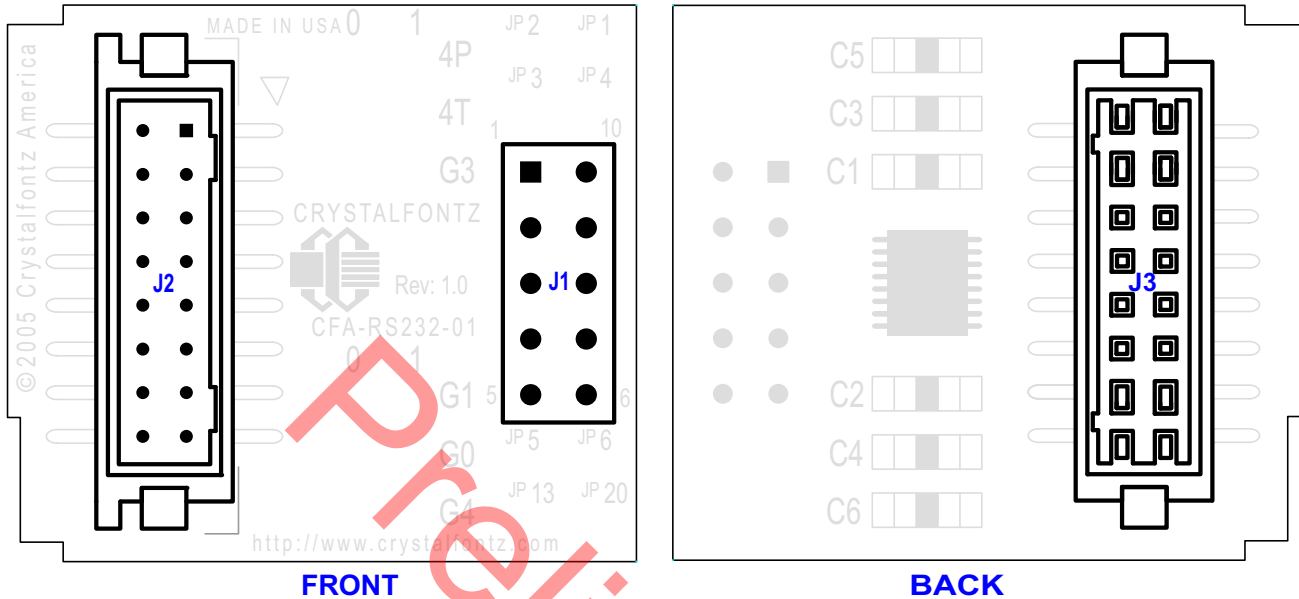


Figure 19. CFA-RS232 Serial Converter Connectors J1, J2, and J3

1. J1 is the male 10-pin (0.1" center) RS232 host communications connector on the top side. For pin assignments, see [CFA-RS232 J1 Connector Pin Assignments \(Default and Alternate\) \(Pg. 36\)](#).
2. J2 is the male 16-pin 2 mm "pass through" connector on the top side, passing through to the J3 female 16-pin 2 mm connector on the bottom side of the board. For pin assignments, see [CFA-RS232 J2 Connector Pin Assignments \(Includes GPIO Connections\) \(Pg. 38\)](#). An optional [FBSCAB](#) may be connected to the male 16-pin "pass through" connector J2.
3. J3 is the female 16-pin 2 mm connector on the bottom side that mates with H1 male 16-pin 2 mm connector on the CFA735 serial LCD Module.

CFA-RS232 J1 Connector Pin Assignments (Default and Alternate)

The pin order of your motherboard's header will determine if the CFA735-xxx-KT's pin assignments ("full swing" RS232 serial) need to be "Default" or "Alternate", as described below.

NOTE

The [WR-232-Y22](#) cable, when connected to the J1 of the CFA-RS232 Serial Converter, provides two connectors on its opposite end. The connector a few inches from the end has a "Default" pin assignment and the connector at the very end has an "Alternate" pin assignment. By using the [WR-232-Y22](#) cable, you can avoid changing jumpers on the CFA-RS232 Serial Converter.



On the CFA-RS232 Serial Converter, the jumpers JP2, JP4, and JP6 are closed by default at the factory, selecting the J1 connector "Default RS232 Pin Assignments". This default pin assignment allows a low cost ribbon cable ([WR-232-Y08](#)) to connect the CFA735-xxx-KT ("full swing" RS232 serial) to a PC's DB9 COM port.

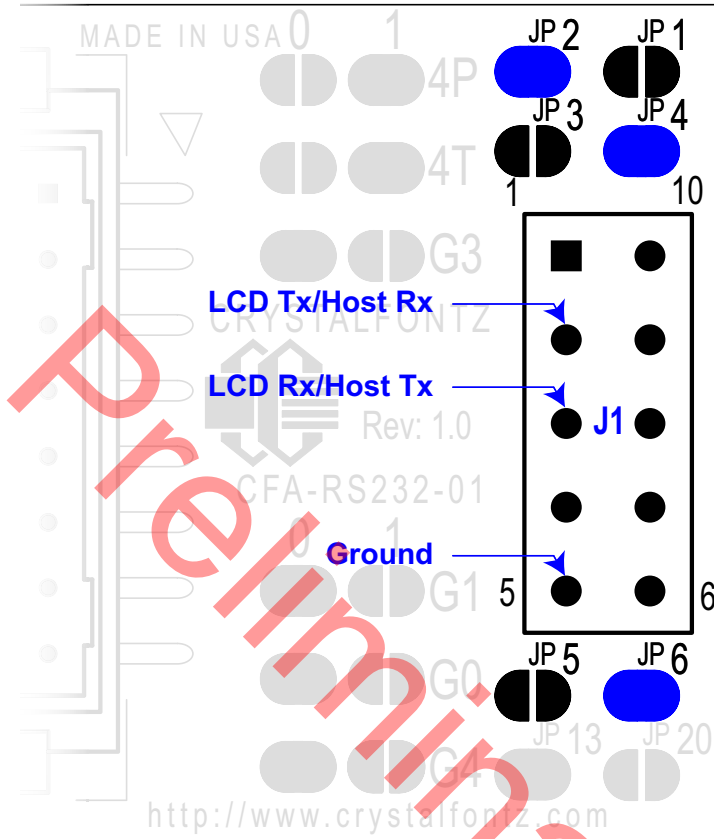


Figure 20. CFA-RS232 J1 Connector Default RS232 Pin Assignments



By opening jumpers JP2, JP4, and JP6 and closing JP1, JP3, and JP5, you can select the “Alternate RS232 Pin Assignments”.

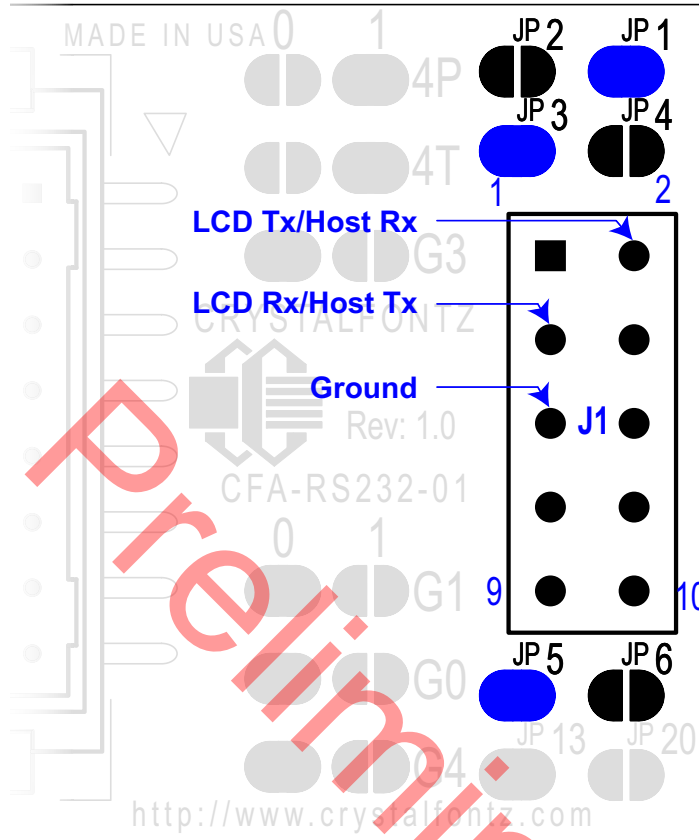


Figure 21. CFA-RS232 J1 Connector Alternate RS232 Pin Assignments

If there is a matching 0.1-inch center, 10-pin RS232 connector on your system's motherboard, then in most cases a simple straight-through ribbon cable such as CrystalFontz [WR-232-Y22](#) or CW Industries' [C3AAG-1018G-ND](#) cable (available from Digi-Key) can be used to connect from the CFA735-xxx-KT (“full swing” RS232 serial) to a motherboard's 10-pin header.

CFA-RS232 J2 Connector Pin Assignments (Includes GPIO Connections)

The CFA735 module has five pins that can be used for “General Purpose Input or Output (GPIO)s”. The pass-through header J2 on the CFA-RS232 Serial Converter. (See the GPIOs labeled in [Figure 20. on Pg. 37.](#)) These GPIOs can be accessed directly through J2.

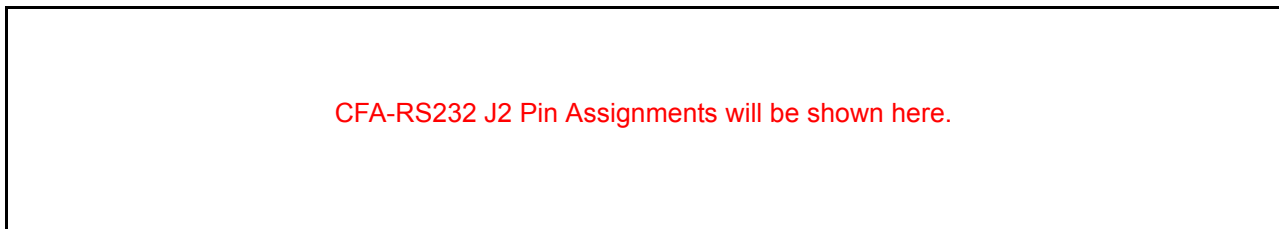


Figure 22. CFA-RS232 J2 Connector Pin Assignments



The following parts may be used to make a mating cable for J2:

- 16-position housing: Hirose DF11-16DS-2C / [Digi-Key H2025-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](#) (typical).

Figure 23.

CONNECT OPTIONAL FBSCAB

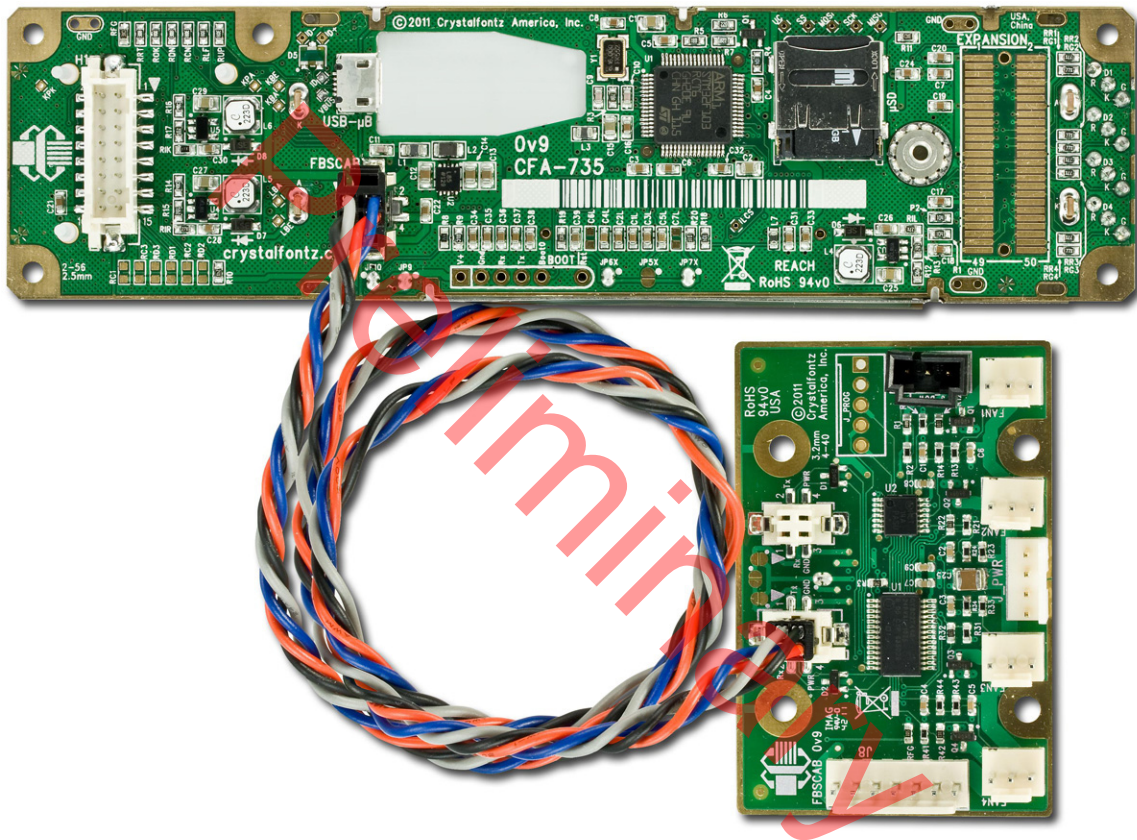


Figure 24. CFA735 Module Connected to Optional FBSCAB with WR-EXT-Y37 Cable

Note: Photo above shows a CFA735-xxx-KR. The CFA735-xxx-KT is a CFA735-xxx-KR with a CFA-RS232 serial converter board.

Note: When using the CFA735 with the optional [FBSCAB](#) (FB System Cooling Accessory Board) and ATX functionality, the ATX control is directly from the CFA735 and not from the FBSCAB. There is no GPIO pass through to the FBSCAB. Instead, the H1 connector on the CFA735 is used for GPIO.



CONNECT OPTIONAL DOW (DALLAS ONE-WIRE) DEVICES TO FBSCAB

Temperature Sensors

The Crystalfontz [WR-DOW-Y17](#) cable has a [DS18B20](#) Dallas Programmable Resolution One-Wire (DOW) temperature sensor attached to a “daisy chainable” cable. (“Daisy chain” means several devices connected in a linear series.) Connect the WR-DOW-Y17 to the connector labeled J_DOW on the FBSCAB. If desired, connect the cable’s 3-pin male connector to an additional temperature sensor. Up to 16 temperature sensors can be connected. (“Daisy chained”.)

The DS18B20 on the WR-DOW-Y17 has $\pm 0.5^{\circ}\text{C}$ accuracy. You can make a temperature sensor cable using a [DS1822](#) Dallas Econo One-Wire Digital Thermometer with $\pm 2^{\circ}\text{C}$ accuracy.

Other One-Wire Devices

Other [Dallas One-Wire Devices](#) may be connected to the one-wire bus, with the CFA735 acting as a bridge between the host system and the one-wire bus (see command [20 \(0x14\): Arbitrary DOW Transaction \(FBSCAB required\) \(Pg. 53\)](#)). The total number of one-wire devices supported is 16, including directly supported temperature sensors and any other user-provided one-wire devices.

The LCD module can send up to 15 bytes and receive up to 14 bytes. This will be sufficient for many devices but some devices require larger transactions and cannot be fully used with the module.

HOST COMMUNICATIONS FOR 635 EMULATION

NOTE

Because there is no difference in communications and commands for *serial* and *USB* differences, this Data Sheet section uses the term “CFA735” for the entire CFA735 family of modules.

THROUGH USB

The easiest and most common way for the host software to access the USB is through the Crystalfontz virtual COM port (VCP) drivers. A link to VCP drivers download and installation instructions can be found on the Crystalfontz website at [USB LCD Drivers](#). Using these drivers makes it appear to the host software as if there is an additional serial port (the VCP) on the host system when the CFA735 is connected. When communicating over USB, the VCP settings are accepted for compatibility reasons. The settings that are not used as the communication is pure USB data.

NOTE TO CFA635 CUSTOMERS

The previous CFA635 required different firmware for USB interface. The CFA735 supports serial *and* USB interface with the same firmware. *The CFA735 USB driver is not the same as the CFA635 USB driver. If you used the CFA635 USB driver, you will need to replace it with the CFA735 USB driver.*

THROUGH “FULL SWING” RS232

The CFA735 communicates with its host using the RS232 interface. The port settings are 115200 baud, 8 data bits, no parity, 1 stop bit by factory default. See [33 \(0x21\): Set Baud Rate \(Pg. 60\)](#) for other speeds.



PACKET STRUCTURE

All communication between the CFA735 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 CFA735 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).

NOTE

Reconciling packets is recommended rather than using delays when communicating with the LCD module. To reconcile your packets, please ensure that you have received the acknowledgement packet from the packet most recently sent before sending any additional packets to the LCD module. This practice will guarantee that you will not have any dropped packets or missed communication with the LCD module.

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 CFA735
01 = normal response from CFA735 to host
10 = normal report from CFA735 to host (not in
    direct response to a command from the host)
11 = error response from CFA735 to host (a packet
    with valid structure but illegal content
    was received by the CFA735)
```

`data_length` specifies the number of bytes that will follow in the data field. The valid range of `data_length` is 0 to 255.

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

`CRC` is a standard 16-bit CRC of all the bytes in the packet except the CRC itself. The CRC is sent LSB first. At the port, the CRC immediately follows the last used element of data []. See [Appendix B: Demonstration Software and Sample Code \(Pg. 76\)](#) for several examples of how to calculate the CRC in different programming languages.

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

```
typedef struct
{
    unsigned char
        command;
    unsigned char
        data_length;
    unsigned char
        data[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 CFA735's packets makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for a corresponding acknowledge packet from the CFA735 before sending the next command packet. The CFA735 will respond to all packets within 250 mS need to verify time. The host software should stop waiting and retry the packet if the CFA735 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 CFA735 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 CFA735 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 CFA735. For any modern PC or microcontroller using reasonably efficient software, this requirement will not be 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 CFA735 can be configured to report three items. The CFA735 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: Fan Speed Report ([FBSCAB](#) required), and (3) 0x82: Temperature Sensor Report. Details are below.

0x80: Key Activity

If a key is pressed or released, the CFA735 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 \(Pg. 54\)](#).

```
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 [CFA533](#) and [CFA633](#). Please note that the CFA631 will return codes 13 through 20. (For more details, see the [CFA631 Data Sheet](#) (this link is for CFA631-TFM-KU) on our website.)



0x81: Fan Speed Report (FBSCAB required)

If any of up to four fans connected to CFA735+[FBSCAB](#) is configured to report its speed information to the host, the CFA735 will send Fan Speed Reports for each selected fan every 1/2 second. See command [16 \(0x10\): Set Up Fan Reporting \(FBSCAB required\) \(Pg. 51\)](#).

```
type = 0x81
data_length = 4
data[0] is the index of the fan being reported:
    0 = FAN 1
    1 = FAN 2
    2 = FAN 3
    3 = FAN 4
data[1] is number of fan_tach_cycles
data[2] is the MSB of Fan_Timer_Ticks
data[3] is the LSB of Fan_Timer_Ticks
```

The following C function will decode the fan speed from a Fan Speed Report packet into RPM:

```
int OnReceivedFanReport(COMMAND_PACKET *packet, char * output)
{
    int
    return_value;
    return_value=0;

    int
    number_of_fan_tach_cycles;
    number_of_fan_tach_cycles=packet->data[1];

    if(number_of_fan_tach_cycles<3)
        sprintf(output," STOP");
    else if(number_of_fan_tach_cycles<4)
        sprintf(output," SLOW");
    else if(0xFF==number_of_fan_tach_cycles)
        sprintf(output," ----");
    else
    {
        //Specific to each fan, most commonly 2
        int
        pulses_per_revolution;
        pulses_per_revolution=2;

        int
        Fan_Timer_Ticks;
        Fan_Timer_Ticks=(*(unsigned short *)&(packet->data[2]));

        return_value=((27692308L/pulses_per_revolution)*
            (unsigned long)(number_of_fan_tach_cycles-3))/
            (Fan_Timer_Ticks);
        sprintf(output,"%5d",return_value);
    }
    return(return_value);
}
```



0x82: Temperature Sensor Report (FBSCAB and Temperature Sensor Accessories required)

If any of the up to 16 temperature sensors is configured to report to the host, the CFA735+[FBSCAB](#) will send Temperature Sensor Reports for each selected sensor every second. See the command [19 \(0x13\): Set Up Temperature Reporting \(FBSCAB required\) \(Pg. 53\)](#).

```

type = 0x82
data_length = 4
data[0] is the index of the temperature sensor being reported:
    0 = temperature sensor 1
    1 = temperature sensor 2
    .
    .
    15 = temperature sensor 16
data[1] is the LSB of Temperature_Sensor_Counts
data[2] is the MSB of Temperature_Sensor_Counts
data[3] is DOW_crc_status
  
```

The following C function will decode the Temperature Sensor Report packet into °C and °F:

```

void OnReceivedTempReport(COMMAND_PACKET *packet, char *output)
{
  //First check the DOW CRC return code from the CFA735
  if(packet->data[3]==0)
    strcpy(output, "BAD CRC");
  else
  {
    double
      degc;
    degc=(*(short *)&(packet->data[1]))/16.0;

    double
      degf;
    degf=(degc*9.0)/5.0+32.0;

    sprintf(output, "%9.4f°C =%9.4f°F",
            degc,
            degf);
  }
}
  
```

COMMAND CODES

Below is a list of valid commands for the CFA735. 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 CFA735 will return the Ping Command to the host.

```

type = 0x00 = 010
valid data_length is 0 to 255
data[0-(data_length-1)] can be filled with any arbitrary data
  
```

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

```

type = 0x40 | 0x00 = 0x40 = 6410
data_length = (identical to received packet)
data[0-(data_length-1)] = (identical to received packet)
  
```



1 (0x01): Get Hardware & Firmware Version and Module Information

The CFA735 will return the hardware and firmware version information to the host.

```
type = 0x01 = 110
valid data length is 0-1.
data[0] = Module information to return
0 = (optional) Hardware and Firmware Version, same as using a data length of 0
1 = Module Serial Number
```

The return packet will be:

```
type = 0x40 | 0x01 = 0x41 = 6510
data_length = 16
data[] = "CFA735:hXvX,fYvY"
```

XvX is the hardware revision, "0v9"
YvY is the firmware version, "0v9" for example

or for data[0] = 1 from the host

```
type = 0x40 | 0x01 = 0x41 = 6510
data_length = 20
data[] = "1134735TMI0000000001"
```

2 (0x02): Write User Flash Area

The CFA735 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 = 210
valid data length is 16
data[] = 16 bytes of arbitrary user data to be stored in the CFA735's non-volatile memory
```

The return packet will be:

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

3 (0x03): Read User Flash Area

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

```
type = 0x03 = 310
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 0x03 = 0x43 = 6710
data_length = 16
data[] = 16 bytes user data recalled from the CFA735's non-volatile memory
```

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

The CFA735 loads its power-up configuration from nonvolatile memory when power is applied. The CFA735 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 \(Pg. 49\)](#).
 - Command [31 \(0x1F\): Send Data to LCD \(Pg. 60\)](#).
- Special character font definitions (command [9 \(0x09\): Set LCD Special Character Data \(Pg. 49\)](#)).
- Cursor position (command [11 \(0x0B\): Set LCD Cursor Position \(Pg. 50\)](#)).



- Cursor style (command [12 \(0x0C\): Set LCD Cursor Style \(Pg. 50\)](#)).
- Contrast setting (command [13 \(0x0D\): Set LCD Contrast \(Pg. 50\)](#)).
- Backlight setting (command [14 \(0x0E\): Set LCD & Keypad Backlight \(Pg. 51\)](#)).
- Fan power settings (command [17 \(0x11\): Set Fan Power \(FBSCABrequired\) \(Pg. 52\)](#)).
- Key press and release masks (command [23 \(0x17\): Configure Key Reporting \(Pg. 54\)](#)).
- Fan glitch delay settings (command [26 \(0x1A\): Set Fan Tachometer Glitch Filter \(FBSCAB required\) \(Pg. 56\)](#)).
- ATX function enable and pulse length settings (command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 57\)](#)).
- Baud rate (command [33 \(0x21\): Set Baud Rate \(Pg. 60\)](#)).
- GPIO settings (command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 61\)](#)).
- The front panel LED/GPO settings (command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 61\)](#)).

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 = 410  
valid data_length is 0
```

The return packet will be:

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

5 (0x05): Reset Functions

Depending on the parameters you provide, this command provides five reset functions: (1) Reload Boot Settings, (2) Reset Host, (3) Power Off Host, (4) CFA735 Soft Reboot, or (5) CFA735 Soft Reboot and Settings Reset.

Reload Boot Settings

Reloads default settings stored in command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#). This has the same parameters and the same effect as the CFA635's (EOL) command 5 "Reboot CFA635".

Rebooting the CFA735 may be useful when testing the boot configuration. It may also be useful to re-enumerate the devices on the one-wire bus.

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

NOTE

The CFA735 will return the acknowledge packet immediately, then reload its settings. The module will respond to new commands immediately. Realize this may reconfigure the interface you are communicating through to its boot state. Part of this delay is the intentional staggered sequencing of turning on power to the fans. If you are not using fans, you can speed the boot process by setting the fan power to 0 (command [17 \(0x11\): Set Fan Power \(FBSCABrequired\)](#)) and saving this as the default boot state (command [4 \(0x04\): Store Current State As Boot State](#)).

Reset Host

This command instructs the CFA735+[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 \(Pg. 61\)](#).

To reset the host (CFA735+[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 \(Pg. 57\)](#), send the following packet:

```
type = 0x05 = 510
valid data length is 3
data[0] = 12
data[1] = 28
data[2] = 97
```

NOTE

The CFA735 will return the **acknowledge** packet immediately, then reset the host. After resetting the host (~1.5 seconds), the module will soft reboot as detailed below. Part of this delay is the intentional staggered sequencing of turning on power to the fans. If you are not using fans, you can speed the boot process by setting the fan power to 0 (command [17 \(0x11\): Set Fan Power \(FBSCABrequired\)](#)) and saving this as the default boot state (command [4 \(0x04\): Store Current State As Boot State](#)). Normally, the host will be recovering from its own reset, so the boot delay of the module will not be of consequence.

Power Off Host

This command instructs the CFA735+[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 \(Pg. 61\)](#).

To turn the host's power off (CFA735+[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 \(Pg. 57\)](#), send the following packet:

```
type = 0x05 = 510
valid data length is 3
data[0] = 3
data[1] = 11
data[2] = 95
```



NOTE

The CFA735 will return the acknowledge packet immediately, then power down the host. The power down length is dependent on the length of the power pulse (command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 57\)](#)). After powering down the host, the module will perform according to the ATX Settings defined by command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 57\)](#).

CFA735 Soft Reboot

Performs a soft reboot of the CFA735 module. If used as a USB device, CFA735 soft reboot will cause the module to disconnect and then reconnect (re-enumerate).

NOTE

The CFA735 will return the acknowledge packet immediately, then reboot itself. The module will not respond to new command packets for up to 3 seconds.

For CFA735 soft reboot, send the following packet:

```
type = 0x05 = 510
valid data_length is 3
data[0] = 8
data[1] = 25
data[2] = 48
```

CFA735 Soft Reboot and Settings Reset

Resets the system boot state to that of an “un-customized” CFA735 and then performs a CFA735 soft reboot. If used as a USB device, CFA735 soft reboot will cause the module to disconnect and then reconnect (re-enumerate).

NOTE

The CFA735 will return the acknowledge packet immediately, then reboot itself. The module will not respond to new command packets for up to 3 seconds.

For settings reset and CFA735 soft reboot, send the following packet:

```
type = 0x05 = 510
valid data_length is 3
data[0] = 10
data[1] = 8
data[2] = 98
```

NOTE

This command does not affect the user flash values input for command [2 \(0x02\): Write User Flash Area \(Pg. 45\)](#).

Return Packet for all Five Reset Functions

For any of the reset functions, the return packet will be:

```
type = 0x40 | 0x05 = 0x45 = 6910
data_length = 0
```




6 (0x06): Clear LCD Screen

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

```
type = 0x06 = 610  
valid data_length is 0
```

The return packet will be:

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

Clear LCD Screen is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).

9 (0x09): Set LCD Special Character Data

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

```
type = 0x09 = 910  
valid data_length is 9  
data[0] = index of special character that you would like to modify, 0-15 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 255, 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.

If you set bit 7 of any of the data bytes in the pixel row, the entire row will blink.
or
If you set bit 6, the first pixel in the row will blink.

The return packet will be:

```
type = 0x40 | 0x09 = 0x49 = 7310  
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 \(Pg. 45\)](#).

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 = 1010  
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 = 7410  
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 CFA735'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 \(Pg. 50\)](#).

```
type = 0x0B = 1110
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 = 7510
data_length = 0
```

Set LCD Cursor Position is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).

12 (0x0C): Set LCD Cursor Style

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

```
type = 0x0C = 1210
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 = 7610
data_length = 0
```

Set LCD Cursor Style is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).

13 (0x0D): Set LCD Contrast

This command sets the contrast of the display.

```
type = 0x0D = 1310
valid data_length is 1
data[0] = contrast setting (0-254 valid)
    0-65 = very light
    66 = light
    95 = about right
    125 = dark
    126-255 = very dark
```

The return packet will be:

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

Set LCD Contrast is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).



Reporting a fan will override the fan power setting to 100% for up to 1/8 of a second every 1/2 second. Please see Fan Connections in the [FBSCAB](#) Data Sheet) for a detailed description.

17 (0x11): Set Fan Power (FBSCABrequired)

This command will configure the power for the fan connectors. The fan power setting is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).

```
type = 0x11 = 1710
valid data_length is 4
data[0] = power level for FAN 1 (0-100 valid)
data[1] = power level for FAN 2 (0-100 valid)
data[2] = power level for FAN 3 (0-100 valid)
data[3] = power level for FAN 4 (0-100 valid)
```

The return packet will be:

```
type = 0x40 | 0x11 = 0x51 = 8110
data_length = 0
```

Set Fan Power is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).

18 (0x12): Read DOW Device Information (FBSCAB required)

When power is applied to the CFA735+[FBSCAB](#)+[WR-DOW-Y17](#) cable, it detects any devices connected to the DOW (Dallas One-Wire) bus and stores the device's information. This command will allow the host to read the device's information.

The first byte returned is the Family Code of the Dallas One-Wire devices. There is a list of the possible Dallas One-Wire device family codes available in [App Note 155: One-Wire Software Resource Guide](#) on the Maxim website.

```
type = 0x12 = 1810
valid data_length is 1
data[0] = device index (0-15 valid)
```

The return packet will be:

```
type = 0x40 | 0x12 = 0x52 = 8210
data_length = 9
data[0] = device index (0-15 valid)
data[1-8] = ROM ID of the device
```

If data[1] is 0x22 ([DS1822](#) Dallas Econo One-Wire Digital Thermometer temperature sensor) or 0x28 ([DS18B20](#) Dallas Programmable Resolution One-Wire Digital Thermometer temperature sensor used on our [WR-DOW-Y17](#)), then that device can be set up to automatically convert and report the temperature every second. See the command [19 \(0x13\): Set Up Temperature Reporting \(FBSCAB required\) \(Pg. 53\)](#).



19 (0x13): Set Up Temperature Reporting (FBSCAB required)

This command will configure the CFA735 to report the temperature information to the host every second.

```
type = 0x13 = 1910
valid_data_length is 4
data[0] = 32-bit bitmask indicating which temperature sensor fans are enabled to report
          (0-255 valid in each location)
data [2-3] = 0
```

```
data[0]
08 07 06 05 04 03 02 01 Enable Reporting of sensor with
| | | | | | | | device index of:
| | | | | | | | -- 0: 1 = enable, 0 = disable
| | | | | | | | --- 1: 1 = enable, 0 = disable
| | | | | | | | ---- 2: 1 = enable, 0 = disable
| | | | | | | | ----- 3: 1 = enable, 0 = disable
| | | | | | | | ----- 4: 1 = enable, 0 = disable
| | | | | | | | ----- 5: 1 = enable, 0 = disable
| | | | | | | | ----- 6: 1 = enable, 0 = disable
| | | | | | | | ----- 7: 1 = enable, 0 = disable
```

```
data[1]
16 15 14 13 12 11 10 09 Enable Reporting of sensor with
| | | | | | | | device index of:
| | | | | | | | -- 8: 1 = enable, 0 = disable
| | | | | | | | --- 9: 1 = enable, 0 = disable
| | | | | | | | ---- 10: 1 = enable, 0 = disable
| | | | | | | | ----- 11: 1 = enable, 0 = disable
| | | | | | | | ----- 12: 1 = enable, 0 = disable
| | | | | | | | ----- 13: 1 = enable, 0 = disable
| | | | | | | | ----- 14: 1 = enable, 0 = disable
| | | | | | | | ----- 15: 1 = enable, 0 = disable
```

```
data[2] = 0
data[3] = 0
```

Any sensor enabled must have been detected as a 0x22 (DS1822 temperature sensor) or 0x28 (DS18B20 temperature sensor) during DOW enumeration. This can be verified by using the command [18 \(0x12\): Read DOW Device Information \(FBSCAB required\) \(Pg. 52\)](#).

The return packet will be:

```
type = 0x40 | 0x13 = 0x53 = 8310
data_length = 0
```

20 (0x14): Arbitrary DOW Transaction (FBSCAB required)

The CFA735+[FBSCAB](#) can function as an RS232 to Dallas One-Wire bridge. The CFA735+FBSCAB can send up to 15 bytes and receive up to 14 bytes. This will be sufficient for many devices, but some devices require larger transactions and cannot be fully used with the CFA735+FBSCAB.

This command allows you to specify arbitrary transactions on the one-wire bus. One-wire commands follow this basic layout:

```
<bus reset> //Required
<address_phase> //Must be "Match ROM" or "Skip ROM"
<write_phase> //optional, but at least one of write_phase or read_phase must be sent
<read_phase> //optional, but at least one of write_phase or read_phase must be sent
```



```
type = 0x14 = 2010
valid data_length is 2 to 16
data[0] = device_index (0-16 valid)
data[1] = number_of_bytes_to_read (0-14 valid)
data[2-15] = data_to_be_written[data_length-2]
```

If `device_index` is 16, then no address phase will be executed. If `device_index` is in the range of 0 to 16, and a one-wire device was detected for that `device_index` at power on, then the write cycle will be prefixed with a "Match ROM" command and the address information for that device.

If `data_length` is two, then no specific write phase will be executed (although address information may be written independently of `data_length` depending on the value of `device_index`).

If `data_length` is greater than two, then `data_length-2` bytes of `data_to_be_written` will be written to the one-wire bus immediately after the address phase.

If `number_of_bytes_to_read` is zero, then no read phase will be executed. If `number_of_bytes_to_read` is not zero then `number_of_bytes_to_read` will be read from the bus and loaded into the response packet.

The return packet will be:

```
type = 0x40 | 0x14 = 0x54 = 8410
data_length = 2 to 16
data[0] = device_index (0-16 valid)
data[data_length-2] = Data read from the one-wire bus. This is the same
                    as number_of_bytes_to_read from the command.
data[data_length-1] = one-wire CRC
```

23 (0x17): Configure Key Reporting

By default, the CFA735 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 \(Pg. 45\)](#).

```
#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 = 2310
data_length = 2
data[0]: press mask (valid 0-63 or 0x3F)
data[1]: release mask
```

The return packet will be:

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

Configure Key Reporting is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the CFA735 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 \(Pg. 54\)](#). 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 = 2410
data_length = 0
```

The return packet will be:

```
type = 0x40 | 0x18 = 0x58 = 8810
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): Set Fan Power Fail-Safe (FBSCAB required)

The combination of the CFA735+[FBSCAB](#) can be used as part of an active cooling system. For instance, the fans in a system can be slowed down to reduce noise when a system is idle or when the ambient temperature is low, and sped up when the system is under heavy load or the ambient temperature is high.

Since there are a very large number of ways to control the speed of the fans (thresholds, thermostat, proportional, PID, multiple temperature sensors "contributing" to the speed of several fans . . .) there was no way to foresee the particular requirements of your system and include an algorithm in the CFA735's firmware that would be an optimal fit for your application.

Varying fan speeds under host software control gives the ultimate flexibility in system design but would typically have a fatal flaw: a host software or hardware failure could cause the cooling system to fail. If the fans were set at a slow speed when the host software failed, system components may be damaged due to inadequate cooling.

The fan power fail-safe command allows host control of the fans without compromising safety. When the fan control software activates, it should set the fans that are under its control to fail-safe mode with an appropriate timeout value. If for any reason the host fails to update the power of the fans before the timeout expires, the fans previously set to fail-safe mode will be forced to 100% power.

```
#define FAN_1      0x01
#define FAN_2      0x02
#define FAN_3      0x04
#define FAN_4      0x08

type = 0x19 = 2510
data_length = 2
data[0] = bit mask of fans set to fail-safe (0-15 valid)
data[1] = timeout value in 1/8 second ticks:
          1 = 1/8 second
          2 = 1/4 second
          255 = 31 7/8 seconds
```

The return packet will be:

```
type = 0x40 | 0x19 = 0x59 = 8910
data_length = 0
```



26 (0x1A): Set Fan Tachometer Glitch Filter (FBSCAB required)

The combination of the CFA735+[FBSCAB](#) cable controls fan speed by using DAC (Digital-to-Analog Converter) controlling the constant current LED driver. Using PWM turns the power to a fan on and off quickly to change the average power delivered to the fan. The CFA735 uses approximately 18 Hz for the PWM repetition rate. The fan's tachometer output is only valid if power is applied to the fan. Most fans produce a valid tachometer output very quickly after the fan has been turned back on but some fans take time after being turned on before their tachometer output is valid.

This command allows you to set a variable-length delay after the fan has been turned on before the CFA735 will recognize transitions on the tachometer line. The delay is specified in counts, each count being nominally 552.5 μ S long (1/100 of one period of the 18 Hz PWM repetition rate).

In practice, most fans will not need the delay to be changed from the default length of 1 count. If a fan's tachometer output is not stable when its PWM setting is other than 100%, simply increase the delay until the reading is stable. Typically you would (1) start at a delay count of 50 or 100, (2) reduce it until the problem reappears, and then (3) slightly increase the delay count to give it some margin.

Setting the glitch delay to higher values will make the RPM monitoring slightly more intrusive at low power settings. Also, the higher values will increase the lowest speed that a fan with RPM reporting enabled will "seek" at "0%" power setting.

The Fan Glitch Delay is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).

```
type = 0x1A = 2610
data_length = 4
data[0] = delay count (1-100) of fan 1
data[1] = delay count (1-100) of fan 2
data[2] = delay count (1-100) of fan 3
data[3] = delay count (1-100) of fan 4
```

The return packet will be:

```
type = 0x40 | 0x1A = 0x5A = 9010
data_length = 0
```

27 (0x1B): Query Fan Power & Fail-Safe Mask (FBSCAB required)

This command can be used to verify the current fan power and verify which fans are set to fail-safe mode.

```
#define FAN_1      0x01
#define FAN_2      0x02
#define FAN_3      0x04
#define FAN_4      0x08
```

```
type = 0x1B = 2710
data_length = 0
```

The return packet will be:

```
type = 0x40 | 0x1B = 0x5B = 9110
data_length = 5
data[0] = fan 1 power
data[1] = fan 2 power
data[2] = fan 3 power
data[3] = fan 4 power
data[4] = bit mask of fans with fail-safe set
```




28 (0x1C): Set ATX Power Switch Functionality

The combination of the CFA735+[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 \(Pg. 45\)](#).

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 \(Pg. 61\)](#). 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=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 CFA735+[WR-PWR-Y25](#) cable are normally high-impedance. Electrically, they appear to be disconnected or floating. When the CFA735+[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 CFA735+[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 CFA735 will show RESET, and then the CFA735 will reset itself, showing its boot state as if it had just powered on. Once the pulse has finished, the CFA735 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 in data[1]. During this time the CFA735 will show "POWER ON", then the CFA735 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 in data[1]. If the user continues to hold the power key down, then the CFA735 will continue to drive the line for a maximum of 5 additional seconds. During this time the CFA735 will show "POWER OFF".

Function 4: LCD_OFF_IF_HOST_IS_OFF

If LCD_OFF_IF_HOST_IS_OFF is set, the CFA735 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 CFA735 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 CFA735 will reboot as if power had just been applied to it.

```
#define AUTO_POLARITY          0x01 //Automatically detects polarity for reset and
                                //power (recommended)
#define RESET_INVERT          0x02 //Reset pin drives high instead of low (disregard
                                //if AUTO_POLARITY)
#define POWER_INVERT          0x04 //Power pin drives high instead of low (disregard
                                //if AUTO_POLARITY)
#define LEDS_FOLLOW_MODULE_LOOK 0x08 // Turn off the LEDs also if the host is off
                                // (ignored if MODULE_LOOK_FOLLOWS_HOST is not set)
#define MODULE_LOOK_FOLLOWS_HOST 0x10 // Turn off the LCD if the Host is off
#define KEYPAD_RESET          0x20
#define KEYPAD_POWER_ON       0x40
#define KEYPAD_POWER_OFF      0x80

type = 0x1C = 2810
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 increments
         1 = 1/32 sec
         2 = 1/16 sec
         16 = 1/2 sec
         ...
         254 = 7 30/32 sec
         255 = Hold until power sense change or 8 sec, whichever is shorter (default)
```

The return packet will be:

```
type = 0x40 | 0x1C = 0x5C = 9210
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 CFA735+[WR-PWR-Y25](#) cable. If the system monitor program fails to reset the watchdog timer, the CFA735+[WR-PWR-Y25](#) cable will reset the host system and soft reboot as if command [5 \(0x05\): Reset Functions \(Pg. 46\)](#) soft reboot function was issued.

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 \(Pg. 57\)](#) or command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 61\)](#).

```
type = 0x1D = 2910  
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 CFA735+[WR-PWR-Y25](#) cable will reset the host system (see command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 57\)](#) for details) and soft reboot itself as if command [5 \(0x05\): Reset Functions \(Pg. 46\)](#) soft reboot function was issued. Since the watchdog is off by default when the it powers up, CFA735+[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 = 9310  
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 = 3010  
data_length = 0
```



The return packet will be:

```
type = 0x40 | 0x1E = 0x5E = 9410
data_length = 15
data[0] = Fan reporting status (as set by command 16)
data[1] = Temperatures 1-8 reporting status (as set by command 19)
data[2] = Temperatures 9-16 reporting status (as set by command 19)
data[3] = 0
data[4] = 0
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] = Fan RPM glitch delay[0] (as set by command 26)
data[10] = Fan RPM glitch delay[1] (as set by command 26)
data[11] = Fan RPM glitch delay[2] (as set by command 26)
data[12] = Fan RPM glitch delay[3] (as set by command 26)
data[13] = Contrast setting (as set by command 13)
data[14] = LCD backlight setting (as set by command 14)
```

31 (0x1F): Send Data to LCD

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

```
type = 0x1F = 3110
data_length = 3 to 22
data[0]: col = x = 0 to 19
data[1]: row = y = 0 to 3
data[2-21]: text to place on the LCD, variable from 1 to 20 characters
```

The return packet will be:

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

Send Data to LCD is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).

33 (0x21): Set Baud Rate

For CFA735-xxx-KR with TTL “logic level” RS232 interface and CFA735-xxx-KT with “full swing” RS232 serial interface, this command will change the CFA735’s baud rate per port basis. For CFA735-xxx-KR with USB interface, baud rate is ignored.

The CFA735 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 CFA735 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 \(Pg. 45\)](#) if you want the CFA735 to power up at the new baud rate.

The factory default baud rate is 115200.

```
type = 0x21 = 3310
data_length = 0
data[0]: 0 = 19200 baud
         1 = 115200 baud
         2 = 9600 baud
```

The return packet will be:

```
type = 0x40 | 0x21 = 0x61 = 9710
data_length = 0
```



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

The CFA735 has five pins for user-definable general purpose input / output (GPIO). These pins are shared with the ATX functions. Be careful when you configure the GPIO if you want to use the ATX at the same time.

The architecture of the CFA735 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 CFA735 continuously polls the GPIOs as inputs at 32 Hz. The present level can be queried by the host software at a lower rate. The CFA735 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 CFA735 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 40kΩ. Typical GPIO current limits when sinking or sourcing all five GPIO pins simultaneously are 8 mA. If you need more information, please contact (888) 206-9720 or (509) 892-1200.

NOTE ON SETTING AND CONFIGURING GPIO PINS and ATX

The GPIO pins may also be used for ATX control through the H1 connector using the Crystalfontz cable [WR-PWR-Y25](#). By factory default, the GPIO output setting, function, and drive mode are set correctly to enable operation of the ATX function. **The GPIO output setting, function, and drive mode must be set to the correct values in order for the ATX function to function properly.** The [635_WinTest](#) may be used to easily check and reset the GPIO configuration to the default state so the ATX and DOW functions will work.

When using the CFA735 with an [FBSCAB](#) and ATX functionality, the ATX control is directly from the CFA735 and not from the FBSCAB. **There is no GPIO pass through to the FBSCAB. Instead, the H1 connector on the CFA735 is used for GPIO.**

The GPIO configuration is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 45\)](#).



```
type: 0x22 = 3410
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
  5 = GPO[ 5] = LED 3 (bottom) green die
  6 = GPO[ 6] = LED 3 (bottom) red die
  7 = GPO[ 7] = LED 2 green die
  8 = GPO[ 8] = LED 2 red die
  9 = GPO[ 9] = LED 1 green die
 10 = GPO[10] = LED 1 red die
 11 = GPO[11] = LED 0 (top) green die
 12 = GPO[12] = LED 0 (top) red die
13-255 = reserved
```

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

```
data[1] = Pin output state (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 except for 6 and 14)
  0 Only meaningful for GPIOs (index 0-4). GPOs (index of 5-12) will ignore.
```

```
---- FDDD
||||-- DDD = Drive Mode (based on output state of 1 or 0)
=====
000: 1=Strong Drive Up, 0=Resistive Pull Down
001: 1=Strong Drive Up, 0=Strong Drive Down
010: Hi-Z, use for input
011: 1=Resistive Pull Up, 0=Strong Drive Down
100: 1=Strong Drive Up, 0=Hi-Z
101: 1=Strong Drive Up, 0=Strong Drive Down
110: reserved, do not use -- error returned
111: 1=Hi-Z,0=Strong Drive Down

----- F = Function (only valid for GPIOs, index of 0-4)
=====
0: Port unused for GPIO. It will take on the default
  function such as ATX 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 = 9810
data_length = 0
```



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

Please see command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 61\)](#) for details on the GPIO architecture.

```
type: 0x23 = 3510
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
5 = GPO[ 5] = LED 3 (bottom) green die
6 = GPO[ 6] = LED 3 (bottom) red die
7 = GPO[ 7] = LED 2 green die
8 = GPO[ 8] = LED 2 red die
9 = GPO[ 9] = LED 1 green die
10 = GPO[10] = LED 1 red die
11 = GPO[11] = LED 0 (top) green die
12 = GPO[12] = LED 0 (top) red die
13-255 = reserved
```

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

The return packet will be:

```
type = 0x40 | 0x23 = 0x63 = 9910
data_length = 4
```




CHARACTER GENERATOR

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 "128_d" and in the row labeled "9_d". 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.




upper 4 bits lower 4 bits	0 _d 0000.	16 _d 0001.	32 _d 0010.	48 _d 0011.	64 _d 0100.	80 _d 0101.	96 _d 0110.	112 _d 0111.	128 _d 1000.	144 _d 1001.	160 _d 1010.	176 _d 1011.	192 _d 1100.	208 _d 1101.	224 _d 1110.	240 _d 1111.
0 _d 0000.	CGRAM [0]															
1 _d 0001.	CGRAM [1]															
2 _d 0010.	CGRAM [2]															
3 _d 0011.	CGRAM [3]															
4 _d 0100.	CGRAM [4]															
5 _d 0101.	CGRAM [5]															
6 _d 0110.	CGRAM [6]															
7 _d 0111.	CGRAM [7]															
8 _d 1000.	CGRAM [0]															
9 _d 1001.	CGRAM [1]															
10 _d 1010.	CGRAM [2]															
11 _d 1011.	CGRAM [3]															
12 _d 1100.	CGRAM [4]															
13 _d 1101.	CGRAM [5]															
14 _d 1110.	CGRAM [6]															
15 _d 1111.	CGRAM [7]															

Figure 25. Character Generator



MODULE RELIABILITY AND LONGEVITY

MODULE RELIABILITY

ITEM		RELIABILITY SPECIFICATION	
LCD portion (excluding keypad, status LEDs, and backlights)		50,000 to 100,000 hours (typical)	
Keypad		1,000,000 keystrokes	
Bicolor LED status lights		50,000 to 100,000 hours (typical)	
 CFA735-TFK-Kx (white LED display backlight and white LED keypad backlight)	<i>Power-On Hours</i>	<i>% of Initial Brightness (New Module)</i>	
	10,000 hours	>90%	
	<50,000 hours	>50%	
 CFA735-TML-Kx (white LED display backlight and blue LED keypad backlight)	<i>Power-On Hours</i>	<i>% of Initial Brightness (New Module)</i>	
	<10,000	>90%	
	<50,000	>50%	
 CFA735-YYK-Kx (yellow-green LED display backlight and yellow-green LED keypad backlight)	50,000 to 100,000 hours (typical)		
<i>Note: For modules with white LED backlights (CFA735-TFK-Kx and CFA735-TML-Kx), adjust backlight brightness so the display is readable but not too bright. Dim or turn off the backlight during periods of inactivity to conserve the white LED backlight life-time.</i>			

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 module and to prolong its life, please follow the precautions described below.

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS

“Full Swing” RS232 Serial Interface

Tx and Rx pins of connector RS232 only:
+15 kV Human Body Model
+15 kV IEC1000-4-2 Air Discharge
+8 kV IEC1000-4-2 Contact Discharge

The remainder of 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.

USB Interface

D+ and D- pins of USB connector only: Electrostatic Discharge Voltage ($I < 1 \mu\text{A}$): +/- 2000 v.

The remainder of 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.

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 module 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 module, leaving a small gap between the plate and the display surface. We recommend GE HP-92 Lexan, which is readily available and works well.
- For USB interface, keep the micro-B USB cable connector parallel to the CFA735 when plugging or unplugging the cable. Do not lift or pull up on the cable. Too much pressure may permanently damage the CFA735's micro-B USB connector.
- 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 image 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.
- *Module 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.
- The microSD card slot may be used for future firmware updates. Firmware updates are announced through our PCN. (Part Change Notices). To ensure that the appropriate people in your organization receive notices, please ask them to subscribe at www.crystalfontz.com/news/pcn.php. **Leave the supplied dummy microSD card in the slot.** If you remove the dummy card and leave the slot empty when the module is powered on, you can irreparably damage the module.
- Observe the operating temperature limitations: a minimum of -20°C to a maximum of 70°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.
- For modules with white LED backlights (CFA735-TFK-Kx and CFA735-TML-Kx), adjust backlight brightness so the display is readable but not too bright. Dim or turn off the backlight during periods of inactivity to conserve the white LED backlight lifetime.

STORAGE AND RECYCLING

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



- Observe the storage temperature limitations: a minimum of -30°C minimum to +80°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.

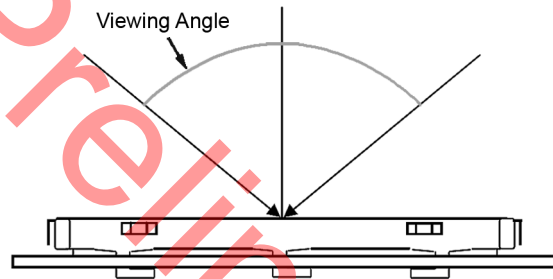
Preliminary



APPENDIX A: QUALITY ASSURANCE STANDARDS

INSPECTION CONDITIONS

- Environment
 - Temperature: $25 \pm 5^\circ\text{C}$
 - 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)
 - Viewable angle: inspect at 45° angle of vertical line right and left, top and bottom

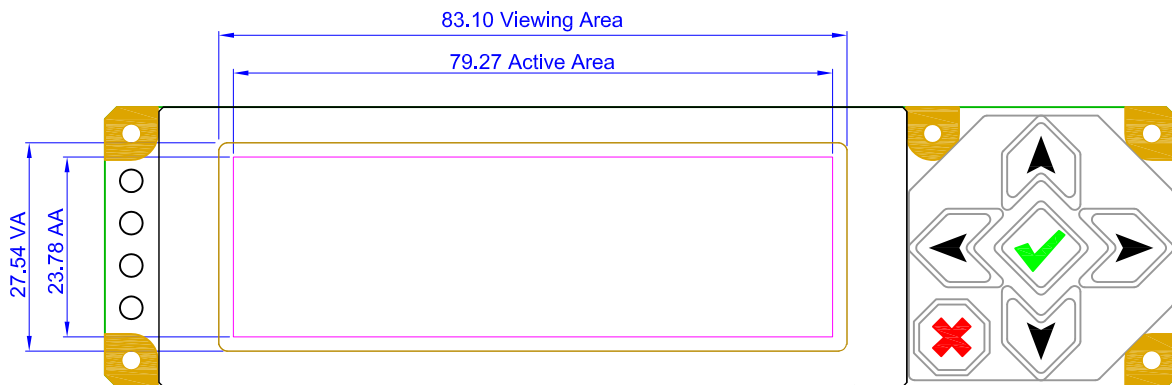


COLOR DEFINITIONS

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

Using CFA635 emulation, Active Area is 77.95 (W) x 22.35 (H) mm and Viewing Area is 82.95 (W) x 27.50 (H) mm.





ACCEPTANCE SAMPLING

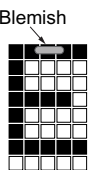
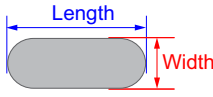
DEFECT TYPE	AQL*
Major	≤0.65%
Minor	≤1.00%
*Acceptable Quality Level: maximum allowable error rate or variation from standard	

DEFECTS CLASSIFICATION

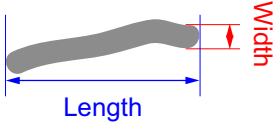
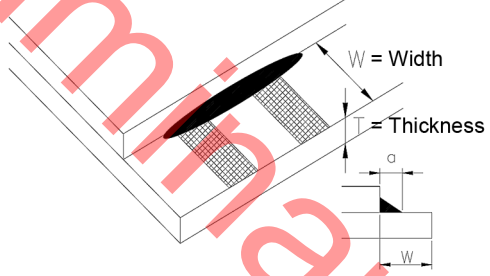
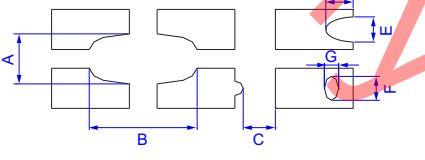
Defects are defined as:

- Major Defect: results in failure or substantially reduces usability of unit for its intended purpose
- Minor Defect: deviates from standards but is not likely 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. 70)).		Major	
3	Contrast adjustment defect	Contrast adjustment fails or malfunctions.		Major	
4	Blemishes or foreign matter on display segments		<i>Defect Size (mm)</i>	<i>Acceptable Qty</i>	Minor
			≤0.3	3	
			≤2 defects within 10 mm of each other		
5	Other blemishes or foreign matter outside of display segments	Defect size = (A + B)/2 	<i>Defect Size (mm)</i>	<i>Acceptable Qty</i>	Minor
			≤0.15	Ignore	
			0.15 to 0.20	3	
			0.20 to 0.25	2	
			0.25 to 0.30	1	



#	DEFECT TYPE	CRITERIA			MAJOR / MINOR
6	Dark lines or scratches in display area 	<i>Defect Width (mm)</i>	<i>Defect Length (mm)</i>	<i>Acceptable Qty</i>	Minor
		≤0.03	≤3.0	3	
		0.03 to 0.05	≤2.0	2	
		0.05 to 0.08	≤2.0	1	
		0.08 to 0.10	≤3.0	0	
		≥0.10	>3.0	0	
7	Bubbles between polarizer film and glass	<i>Defect Size (mm)</i>	<i>Acceptable Qty</i>	Minor	
		≤0.20	Ignore		
		0.20 to 0.40	3		
		0.40 to 0.60	2		
		≥0.60	0		
8	Glass rest defect				Minor
9	Display pattern defect 	<i>Dot Size (mm)</i>	<i>Acceptable Qty</i>		Minor
		$((A+B)/2) \leq 0.2$	≤3 total defects ≤2 pinholes per digit		
		C > 0			
		$((D+E)/2) \leq 0.25$			
		$((F+G)/2) \leq 0.25$			



#	DEFECT TYPE	CRITERIA	MAJOR / MINOR												
10	Chip in corner		Minor												
		<table border="1"> <thead> <tr> <th><i>a</i></th> <th><i>b</i></th> <th><i>c</i></th> <th>Acceptable Qty</th> </tr> </thead> <tbody> <tr> <td>$<4 \text{ mm}$</td> <td>$\leq W$</td> <td>$c \leq T$</td> <td>3</td> </tr> </tbody> </table>		<i>a</i>	<i>b</i>	<i>c</i>	Acceptable Qty	$<4 \text{ mm}$	$\leq W$	$c \leq T$	3				
<i>a</i>	<i>b</i>	<i>c</i>	Acceptable Qty												
$<4 \text{ mm}$	$\leq W$	$c \leq T$	3												
11	Chip on "non-contact" edge of LCD		Minor												
		<table border="1"> <thead> <tr> <th><i>a</i></th> <th><i>b</i></th> <th><i>c</i></th> </tr> </thead> <tbody> <tr> <td>$\leq 3 \text{ mm}$</td> <td>$\leq 1 \text{ mm}$</td> <td>$\leq T$</td> </tr> <tr> <td>$\leq 4 \text{ mm}$</td> <td>$\leq 1.5 \text{ mm}$</td> <td>$\leq T$</td> </tr> </tbody> </table>		<i>a</i>	<i>b</i>	<i>c</i>	$\leq 3 \text{ mm}$	$\leq 1 \text{ mm}$	$\leq T$	$\leq 4 \text{ mm}$	$\leq 1.5 \text{ mm}$	$\leq T$			
		<i>a</i>		<i>b</i>	<i>c</i>										
$\leq 3 \text{ mm}$	$\leq 1 \text{ mm}$	$\leq T$													
$\leq 4 \text{ mm}$	$\leq 1.5 \text{ mm}$	$\leq T$													
<table border="1"> <thead> <tr> <th><i>a</i></th> <th><i>b</i></th> <th><i>c</i></th> </tr> </thead> <tbody> <tr> <td>$\leq 3 \text{ mm}$</td> <td>$\leq 1.5 \text{ mm}$</td> <td>$\leq T$</td> </tr> </tbody> </table>	<i>a</i>	<i>b</i>	<i>c</i>	$\leq 3 \text{ mm}$	$\leq 1.5 \text{ mm}$	$\leq T$									
<i>a</i>	<i>b</i>	<i>c</i>													
$\leq 3 \text{ mm}$	$\leq 1.5 \text{ mm}$	$\leq T$													
12	Chip on "contact" edge of LCD, on the active side		Minor												
		<table border="1"> <thead> <tr> <th><i>a</i></th> <th><i>b</i></th> <th><i>c</i></th> <th>Acceptable Qty</th> </tr> </thead> <tbody> <tr> <td>$\leq 2 \text{ mm}$</td> <td>$\leq W/4$</td> <td>$\leq T$</td> <td>Ignore</td> </tr> <tr> <td>$\leq 3 \text{ mm}$</td> <td>$\leq W/4$</td> <td>$\leq T$</td> <td>3</td> </tr> </tbody> </table>		<i>a</i>	<i>b</i>	<i>c</i>	Acceptable Qty	$\leq 2 \text{ mm}$	$\leq W/4$	$\leq T$	Ignore	$\leq 3 \text{ mm}$	$\leq W/4$	$\leq T$	3
		<i>a</i>		<i>b</i>	<i>c</i>	Acceptable Qty									
$\leq 2 \text{ mm}$	$\leq W/4$	$\leq T$	Ignore												
$\leq 3 \text{ mm}$	$\leq W/4$	$\leq T$	3												
<table border="1"> <thead> <tr> <th><i>a</i></th> <th><i>b</i></th> <th><i>c</i></th> <th>Acceptable Qty</th> </tr> </thead> <tbody> <tr> <td>$\leq 2 \text{ mm}$</td> <td>$\leq W/4$</td> <td>$\leq T$</td> <td>Ignore</td> </tr> <tr> <td>$\leq 3 \text{ mm}$</td> <td>$\leq W/4$</td> <td>$\leq T$</td> <td>3</td> </tr> </tbody> </table>	<i>a</i>	<i>b</i>	<i>c</i>	Acceptable Qty	$\leq 2 \text{ mm}$	$\leq W/4$	$\leq T$	Ignore	$\leq 3 \text{ mm}$	$\leq W/4$	$\leq T$	3			
<i>a</i>	<i>b</i>	<i>c</i>	Acceptable Qty												
$\leq 2 \text{ mm}$	$\leq W/4$	$\leq T$	Ignore												
$\leq 3 \text{ mm}$	$\leq W/4$	$\leq T$	3												



#	DEFECT TYPE	CRITERIA	MAJOR / MINOR												
13	Chip on "contact" edge of LCD, on the inactive side		Minor												
		<table border="1"> <thead> <tr> <th>a</th> <th>b</th> <th>c</th> <th>Acceptable Qty</th> </tr> </thead> <tbody> <tr> <td>≤3 mm</td> <td>≤1 mm</td> <td>≤T</td> <td>Ignore</td> </tr> <tr> <td>≤4 mm</td> <td>≤1.5 mm</td> <td>≤T</td> <td>3</td> </tr> </tbody> </table>		a	b	c	Acceptable Qty	≤3 mm	≤1 mm	≤T	Ignore	≤4 mm	≤1.5 mm	≤T	3
		a		b	c	Acceptable Qty									
		≤3 mm		≤1 mm	≤T	Ignore									
≤4 mm	≤1.5 mm	≤T	3												
<table border="1"> <thead> <tr> <th>a</th> <th>b</th> <th>c</th> <th>Acceptable Qty</th> </tr> </thead> <tbody> <tr> <td><3 mm</td> <td>≤1.5 mm</td> <td>≤1/2 T</td> <td>3</td> </tr> </tbody> </table>	a	b	c	Acceptable Qty	<3 mm	≤1.5 mm	≤1/2 T	3							
a	b	c	Acceptable Qty												
<3 mm	≤1.5 mm	≤1/2 T	3												
Unacceptable if c > 50% of glass thickness or if the seal area is damaged.	Major														
14	Chip in seal area	<p>a = length b = width c = thickness</p>	Minor												
		<table border="1"> <thead> <tr> <th>a</th> <th>b</th> <th>c</th> <th>Acceptable Qty</th> </tr> </thead> <tbody> <tr> <td><3 mm</td> <td>≤1.5 mm</td> <td>≤1/2 T</td> <td>3</td> </tr> </tbody> </table>		a	b	c	Acceptable Qty	<3 mm	≤1.5 mm	≤1/2 T	3				
		a		b	c	Acceptable Qty									
<3 mm	≤1.5 mm	≤1/2 T	3												
Unacceptable if c > 50% of glass thickness or if the seal area is damaged.	Major														
15	Backlight defects	<ol style="list-style-type: none"> 1. Light fails or flickers.* 2. Color and luminance do not correspond to specifications.* 3. Exceeds standards for display's blemishes or foreign matter (see test 5, Pg. 71), and dark lines or scratches (see test 6, Pg. 72). <p>*Minor if display functions correctly. Major if the display fails.</p>	Minor												
16	COB defects	<ol style="list-style-type: none"> 1. Pinholes >0.2 mm. 2. Seal surface has pinholes through to the IC. 3. More than 3 locations of sealant beyond 2 mm of the sealed areas. 	Minor												
17	PCB defects	<ol style="list-style-type: none"> 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. <p>*Minor if display functions correctly. Major if the display fails.</p>	Minor												



#	DEFECT TYPE	CRITERIA	MAJOR / MINOR
18	Soldering defects	<ol style="list-style-type: none">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. <p><i>*Minor if display functions correctly. Major if the display fails.</i></p>	Minor

Preliminary



APPENDIX B: DEMONSTRATION SOFTWARE AND SAMPLE CODE

DEMONSTRATION SOFTWARE

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. CFA735 uses CFA635 emulation.
<http://www.crystalfontz.com/product/635WinTest>
- ❑ Linux compatible command-line demonstration program with C source code. 8K.
http://www.crystalfontz.com/product/linux_cli_examples.html
- ❑ Supported by CC2 (CrystalControl2) 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 CFA735 packet. Some of the algorithms were contributed by forum members and originally written for the CFA631. The CRC used in the CFA735 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,0x0E416,0x090A9,0x08120,0x0B3BE,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, originally for the CFA635.

```

;=====
; CrystalFontz CFA735 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
;
main      org          20          ; start of main program
main      movlw       seedhi      ; setup initial 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     clr      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 CFA735 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,0x0AC42,0x09ED9,0x08F50,0x0FBF7,0x0EA66,0x0D8FD,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 = 0xFFFF;
  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, 0x0DBFD, 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   dsplyCRCDData,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
dsplyCRCDData 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
    movwf    TXBUF2         ; command to send data to LCD
                                ; 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
    MOVLFS  upper stringname, TBLPTRU
    MOVLFS  high stringname, TBLPTRH
    MOVLFS  low stringname, TBLPTRL
    call     MOVE_STR
endm

;=====
MOVE_STR:
    tblrd   *+
    movf    TABLAT,w
    bz     ms1b
    movwf   POSTINC1
    incf    SendCount
    goto   MOVE_STR

ms1b:
    return

;=====
```