



INTELLIGENT LCD MODULE SPECIFICATIONS



Data Sheet Release Date 2014-03-18

for

[CFA635-TFE-KS](#)

[CFA635-TMF-KS](#)

[CFA635-YYE-KS](#)

Hardware Version: h1v2, Firmware Version: s2v0

Crystalfontz America, Incorporated

12412 East Saltese Avenue
Spokane Valley, WA 99216-0357

Phone: 888-206-9720

Fax: 509-892-1203

Email: support@crystalfontz.com

URL: www.crystalfontz.com



CONTENTS

INTRODUCTION	10
Comparison To Previous Versions Of The CFA635 Family	10
Comparison Of The CFA635 Family And The CFA735 Family	10
Difference Between The Two Serial Interfaces	10
TTL “Logic Level, Inverted” Serial (CFA635-xxx-KL)	10
“Full Swing” RS232 Serial (CFA635-xxx-KS)	11
Main Features	11
Explanation Of Part Number Codes In This Data Sheet	13
MECHANICAL SPECIFICATIONS	15
Physical Characteristics	15
Module Outline Drawing Front And Side Views	16
Module Outline Drawing Back View And Pixel Detail	17
Jumpers That Can Be Modified	18
Panel Mounting Application Cutout	19
Optional Mounting Bracket	20
Keypad Detail Drawing	21
CFA635-xxx-KS Has Two Parts: CFA635-xxx-KL LCD Module And CFA-RS232 Serial Converter	22
ELECTRICAL SPECIFICATIONS	24
System Block Diagram	24
LCD Duty And Bias	25
Absolute Maximum Ratings	25
DC Characteristics	26
Current Consumption	26
CFA635-TFE-Kx (Near-Black on White)	27
CFA635-TMF-Kx (Near-White on Blue)	27
CFA635-YYE-Kx (Near-Black on Yellow-Green)	28
GPIO Current Limits	28
ESD (Electro-Static Discharge) Specifications	28
Backlight And Fan Criteria	28
OPTICAL SPECIFICATIONS	29
CFA635-TFE-Kx (Near-Black on White)	29
CFA635-TMF-Kx (Near-White on Blue)	30
CFA635-YYE-Kx (Near-Black on Yellow-Green)	31
Test Conditions And Definitions For Optical Characteristics	31
Definition Of Optimal Contrast Setting	32
Definition Of Response Time (Tr, Tf)	33
Definition Of 6 O’Clock And 12:00 O’Clock Viewing Angles	34
Definition Of Vertical And Horizontal Viewing Angles (CR _{≥2})	34
LED BACKLIGHT INFORMATION	35
CONNECTION INFORMATION	36
Cables	36
Connectors On The Two Parts	37
Part 1: CFA-RS232 Serial Converter, Connectors J1, J2, And J3	38
Part 2: CFA635-xxx-KL Serial LCD Module, Connectors H1 And H2	39



CONTENTS, CONTINUED

Available Connectors When CFA-RS232 Is Mounted On CFA635-xxx-KL	40
CFA-RS232 J1 Connector Pin Assignments	40
J1 Connector Default RS232 Pin Assignments	40
J1 Connector Alternate RS232 Pin Assignments	42
CFA-RS232 J2 Connector Pin Assignments - Includes Five GPIOs	42
H2 Connector Pin Assignments - Includes Eight GPOs	44
Three Methods For Power Connection To Host	44
ATX Power Supply	45
ATX Power And Control Connections	45
ATX Connection With Optional SCAB Using WR-PWR-Y14 ATX Cable	47
ATX Connection Without Optional SCAB Using WR-PWR-Y25 ATX Cable	49
How to Set ATX Functionality Using cfTest	50
How To Connect The Optional SCAB	50
HOST COMMUNICATIONS	51
Packet Structure	51
About Handshaking	53
Command Codes	53
0 (0x00): Ping Command	53
1 (0x01): Get Hardware And Firmware Version	53
2 (0x02): Write User Flash Area	54
3 (0x03): Read User Flash Area	54
4 (0x04): Store Current State As Boot State	54
5 (0x05): Reboot CFA635, Reset Host, Or Power Off Host (ATX Required)	55
6 (0x06): Clear LCD Screen	57
7 (0x07): Deprecated (See command 31 (0x1F): Send Data To LCD (Pg. 69))	57
8 (0x08): Deprecated (See command 31 (0x1F): Send Data To LCD (Pg. 69))	57
9 (0x09): Set LCD Special Character Data	57
10 (0x0A): Read 8 Bytes Of LCD Memory	58
11 (0x0B): Set LCD Cursor Position	58
12 (0x0C): Set LCD Cursor Style	58
13 (0x0D): Set LCD Contrast	59
14 (0x0E): Set LCD And Keypad Backlights	59
15 (0x0F): Deprecated	59
16 (0x10): Set Up Fan Reporting (SCAB Required)	59
17 (0x11): Set Fan Power (SCAB Required)	60
18 (0x12): Read WR-DOW-Y17 Temperature Sensors (SCAB Required)	60
19 (0x13): Set Up WR-DOW-Y17 Temperature Reporting (SCAB Required)	61
20 (0x14): Arbitrary DOW Transaction (SCAB Required)	62
21 (0x15): (Not Accessible)	63
22 (0x16): Send Command Directly to the LCD Controller	63
23 (0x17): Configure Key Reporting	63
24 (0x18): Read Keypad, Polled Mode	63
25 (0x19): Set Fan Power Fail-Safe (SCAB Required)	64
26 (0x1A): Set Fan Tachometer Glitch Delay (SCAB Required)	65



27 (0x1B): Query Fan Power And Fail-Safe Mask (SCAB Required) -----	65
28 (0x1C): Set ATX Power Switch Functionality -----	66
29 (0x1D): Enable/Disable And Reset The Watchdog -----	68
30 (0x1E): Read Reporting And Status -----	68
31 (0x1F): Send Data To LCD -----	69
32 (0x20): Not Accessible (Reserved for CFA631 Key Legends) -----	69
33 (0x21): Set Baud Rate -----	69
34 (0x22): GPIO And GPO Settings (SCAB Required) -----	69
35 (0x23): Read GPIO And GPO Pin Levels And Configuration State (SCAB Required) -----	72
Report Codes -----	73
0x80: Key Activity -----	73
0x81: Fan Speed Report (SCAB Required) -----	74
0x82: Temperature Sensor Report (SCAB Required) -----	75
CHARACTER GENERATOR ROM (CGROM) -----	77
MODULE RELIABILITY AND LONGEVITY -----	78
Module Reliability -----	78
Module Longevity (EOL / Replacement Policy) -----	78
CARE AND HANDLING PRECAUTIONS -----	79
APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER (SCAB REQUIRED)-----	81
APPENDIX B: CONNECTING A DS1963S SHA IBUTTON (SCAB REQUIRED) -----	83
APPENDIX C: SAMPLE CODE (DEMONSTRATION SOFTWARE AND SAMPLE CODE) -----	86
Drivers -----	86
Demonstration And Test Programs -----	86
Algorithms to Calculate the CRC -----	86
Algorithm 1: "C" Table Implementation -----	86
Algorithm 2: "C" Bit Shift Implementation -----	87
Algorithm 2B: "C" Improved Bit Shift Implementation -----	89
Algorithm 3: "PIC Assembly" Bit Shift Implementation -----	89
Algorithm 4: "Visual Basic" Table Implementation -----	91
Algorithm 5: "Java" Table Implementation -----	92
Algorithm 6: "Perl" Table Implementation -----	94
Algorithm 7: For PIC18F8722 or PIC18F2685 -----	95
APPENDIX D: QUALITY ASSURANCE STANDARDS-----	98



LIST OF FIGURES

Figure 1. Module Outline Drawing Front And Side Views - - - - -	16
Figure 2. Module Outline Drawing Back View And Pixel Detail - - - - -	17
Figure 3. Jumpers that Can Be Modified - - - - -	18
Figure 4. Panel Mounting Application Cutout Drawing - - - - -	19
Figure 5. Optional Mounting Bracket Drawing - - - - -	20
Figure 6. Keypad Detail Drawing - - - - -	21
Figure 7. Front View Of CFA-RS232 Serial Converter - - - - -	22
Figure 8. CFA-RS232 Serial Converter Mounted On CFA635-xxx-KL LCD Module - - - - -	22
Figure 9. System Block Diagram - - - - -	24
Figure 10. Definition of Optimal Contrast Setting (Negative Image) - - - - -	32
Figure 11. Definition of Optimal Contrast Setting (Positive Image) - - - - -	32
Figure 12. Definition of Response Time (Tr, Tf) (Negative Image)- - - - -	33
Figure 13. Definition of Response Time (Tr, Tf) (Positive Image) - - - - -	33
Figure 14. Definition of 6:00 O'Clock and 12:00 O'Clock Viewing Angles - - - - -	34
Figure 15. Definition Of Horizontal And Vertical Viewing Angles (CR>2) - - - - -	34
Figure 16. CFA-RS232 Serial Converter (Side View)- - - - -	38
Figure 17. CFA-RS232 Serial Converter Connectors J1, J2, And J3- - - - -	38
Figure 18. H1 And H2 Connectors On CFA635-xxx-KL - - - - -	39
Figure 19. Available Connectors When CFA-RS232 Is Mounted On CFA635-xxx-KL - - - - -	40
Figure 20. CFA-RS232 J1 Connector Default RS232 Pin Assignments- - - - -	41
Figure 21. CFA-RS232 J1 Connector Alternate RS232 Pin Assignments - - - - -	42
Figure 22. CFA-RS232 J2 Connector Pin Assignments - - - - -	43
Figure 23. Pin Assignments On CFA635-xxx-KS H2 Connector - Includes Eight GPOs - - - - -	44
Figure 24. CFA635-xxx-KS Power Connection To Host And Optional SCAB - - - - -	45
Figure 25. ATX Connection With Optional SCAB Using WR-PWR-Y14 ATX Cable- - - - -	48
Figure 26. ATX Power Supply And Control Connections Using WR-PWR-Y25 ATX Cable - - - - -	49
Figure 27. CFA635-xxx-KS Connected To Optional SCAB With WR-EXT-Y19 Cable - - - - -	50
Figure 28. Character Generator ROM (CGROM) - - - - -	77
Appendix A Figure 1. Test Circuit Schematic- - - - -	81
Appendix B Figure 1. Connect to Maxim/Dallas DS19632 SHA iButton Using DS9094 iButton Clip - - - - -	83



Family Data Sheet Revision History

Data Sheet Release: 2014-03-18

- Added note to this section [About Volatility \(Pg. 8\)](#).
- In [Panel Mounting Application Cutout \(Pg. 19\)](#), removed incorrect Cutout Detail illustration. For mounting hole dimensions, see [Module Outline Drawing Front And Side Views \(Pg. 16\)](#).
- Minor changes to improve text and formatting.

Data Sheet Release: 2013-10-02

- Wherever listed, replaced reference to our free demonstration program 635_WinTest with the new demonstration program [cfTest](#). Please note that the steps for How to Set ATX Functionality Using cfTest are slightly different than the steps described in the previous Data Sheet for 635_WinTest.
- In Physical Characteristics, corrected weight for CFA635-xxx-KS from 71 to 68 grams. The correction is due to a more accurate scale. Module weight has not changed.
- Listed more accurate overall length specification for Cables. Some specifications are slightly longer and others are slightly shorter than what was listed in the previous Data Sheet.
- Corrected description of J1 Default and Alternate Pin Assignments. Pin assignments have not changed. See J1 Connector Default RS232 Pin Assignments and J1 Connector Alternate RS232 Pin Assignments.
- Expanded section on How To Clean. Please read carefully before you clean a module.
- Updated information in Demonstration And Test Programs.
- Made improvements to Data Sheet formatting and text.

Data Sheet Release: 2013-03-06

- Minor changes in descriptions of commands. See [PCN#10406](#) for change descriptions.
- Updated to current standards for Data Sheet template.
- Added [WR-PWR-Y38](#) cable and [UBERSCAB Kit](#) to descriptions for Cables.

Data Sheet Release: 2012-08-20 Preliminary

- Corrected pin descriptions in figure for H1 Connector Pin Assignments - Includes Eight GPIOs.
- Improvements to text for clarification, particularly in the sections describing connections.

Data Sheet Release: 2012-07-25 Preliminary (Available to customers by special request. Not published on website.)
Complete Data Sheet rewrite.

2010-10-07

Data Sheet Revision: v2.1

Changes since last revision (v2.0):

- Wherever listed or shown, improved explanation of module depth specifications with and without keypad. Module has not changed.
- In Absolute Maximum Ratings, added important note on specifications.
- Slightly modified specifications in Typical Current Consumption to reflect backlight improvement made 2008-07-01.
- In H1 Pin Assignments - Includes GPIO Connections figure, corrected from "F4P" to "LCD Tx / Host Rx" and from "F4T" to "LCD Rx/Host Tx". Only the USB version of the CFA635 has a fourth fan.
- In ATX Power Supply and Control Connections using Crystalfontz WR-PWR-Y25 Cable, corrected illustration so Y14 cable red and blue twisted wires are aligned correctly to the connector. Ground (blue) is on the far right. Cable connector order has not changed.
- In APPENDIX AC: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC), added Algorithm 7: For PIC18F8722 or PIC18F2685.
- Wherever needed, made minor modifications in text and illustrations to improve clarity. Includes how Hardware revisions are listed above.



Family Data Sheet Revision History (Continued)

2010-08-04	<p>Data Sheet Revision: v2.0 Changes since last revision (v1.0)</p> <ul style="list-style-type: none"> ● Added boxed note "Fine Print" at the bottom of this Revision History. Please read these disclaimers. ● Removed PDF sticky note titled "Changes" from this Revision History. The note read "2008-10-07 Corrected specification of GPIO pull-up/pull-down mode resistance values from "approximately 5Ω" to "approximately 5kΩ". Note is no longer needed because document has been revised. ● Reorganized content, improved drawings, and edited to improve readability. ● Wherever listed, deleted dash ("-") from module part numbers to match how they now appear on our website without the dash ("-"). ● Wherever listed, added dash ("-") to cable part numbers to match how they now appear on our website. ● Wherever needed, added information about which optional CrystalFontz cables to use. ● In Physical Characteristics, Module Outline Drawing Jumper Locations and Function, and Keypad Detail Drawing drawings, changed keypad dimensions from "10.5 mm" to "12 mm" height and module overall depth from "20.55 mm" to "22.05 mm". We started this gradual transition June 2010. ● In section Typical Current Consumption, made slight changes to specifications due to improved backlight. ● Added section ATX Power Supply Power and Control Connections. ● In section How to Connect the Optional SCAB, replaced photo. New photo does not include the bracket, which is optional. ● In Command 4 (0x04): Store Current State As Boot State, deleted reference to Command 32 which is reserved for CFA631 key legends. ● Deleted "SCAB Required" in the commands listed below. These commands can also be used when the module does not have the optional SCAB and is connected to the host with a CrystalFontz WR-PWR-Y25 cable. <ul style="list-style-type: none"> - Command 5 (0x05): Reboot CFA635, Reset Host, or Power Off Host. - Command 13 (0x0D): Set LCD Contrast, corrected contrast setting from " (0-255 valid)" and "126-255 = very dark" to "(0-254 valid)" and "126-254 = very dark". - Command 28 (0x1C): Set ATX Power Switch Functionality. - Command 29 (0x1D): Enable/Disable and Reset the Watchdog. ● In command 34 (0x22): Set or Set and Configure GPIO Pin, <ul style="list-style-type: none"> - Corrected specification of GPIO pull-up/pull-down mode resistance values from "approximately 5Ω" to "approximately 5kΩ". - Clarified parameter usage in GPIOs vs GPOs. ● In command 35 (0x23): Read GPIO Pin Levels and Configuration State, <ul style="list-style-type: none"> - Corrected from "data_length: 4" to "data_length: 1". - Index for "data[0]: index of GPIO to query" previously listed only 1-4. Added 5-12. - Corrected from "5-255: reserved" to "13-255: reserved". - Improved LED descriptions for H1 pins. Pins did not change. - Clarified parameter usage in GPIOs versus GPOs.
------------	---



Family Data Sheet Revision History (Continued)	
2010-08-04	<ul style="list-style-type: none"> ● Corrected references from J8 and J9 connectors (CFA633 connectors) to H1 and H2 (CFA635 connectors). Connectors were labeled correctly in drawings and have not changed. ● Please read expanded section MODULE RELIABILITY AND LONGEVITY. ● In APPENDIX C: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC), <ul style="list-style-type: none"> - Added section with hypertext links to our free downloadable code. - Added section Algorithm 2B: "C" Improved Bit Shift Implementation. This is a simplified algorithm that implements the CRC. - In sample code for Algorithm 1: "C" Table Implementation and Algorithm 2: "C" Bit Shift Implementation, added typedefs for "ubyte" and "word". - In Algorithm 6: "Perl" Table Implementation, corrected code from <code>my \$packet = \$type . \$length . \$data ;</code> to <code>my \$packet = chr(hex \$type) .chr(hex \$length) .chr(hex \$data);</code>. ● Reorganized content, improved drawings, and edited to improve readability.
2007-05-15	Data Sheet Revision: v1.0 New Data Sheet.

Hardware and Firmware Revisions

For information about firmware and hardware revisions for the this family of intelligent LCD modules, see Part Change Notifications (PCNs) under the Notices tab on the website page for each CFA635 part number.

About Variations

We work continuously to improve our products. Because display technologies are quickly evolving, these products may have component or process changes. Slight variations (for example, contrast, color, or intensity) between lots are normal. If you need the highest consistency, whenever possible, order and arrange delivery for your production runs at one time so your displays will be from the same lot.

About Volatility

The CFA635 family of Crystalfontz Intelligent LCD Modules have nonvolatile memory.



Additional 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.

All specifications in Data Sheets and on our website are, to the best of our knowledge, accurate but not guaranteed. Corrections to specifications are made as any inaccuracies are discovered.

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

Copyright © 2014 by CrystalFontz America, Inc., 12412 East Saltese Avenue, Spokane Valley, WA 99216-0357 U.S.A



INTRODUCTION

The CFA635 family of modules has three interface choices.

CFA635-xxx-KL	Serial interface “logic level, inverted”
CFA635-xxx-KS	Serial interface with “full swing” RS232
CFA635-xxx-KU	USB Interface

**This Data Sheet has information for RS232 Interface
CFA635-TFE-KS, CFA635-TMF-KS, and CFA635-YYE-KS**

When information in this Data Sheet applies to all three color choices, the term “CFA635-xxx-KS” or the shorter term “CFA635” is used.

COMPARISON TO PREVIOUS VERSIONS OF THE CFA635 FAMILY

For information about firmware and hardware revisions, see the Part Change Notifications (PCNs) under “News” in our website’s navigation bar. To see the most recent PCNs for the CFA635 family at the time of this Data Sheet release, see [PCN #10405](#) for hardware and [PCN#10406](#) for firmware.

COMPARISON OF THE CFA635 FAMILY AND THE CFA735 FAMILY

The CFA735 family of modules are similar to the CFA635 family and use CFA635 emulation software. The CFA735 can operate at +3.3v, two interfaces may be used simultaneously, and it has wide temperature range. The CFA635 family operates at +5v, cannot use two interfaces simultaneously, and has a normal temperature range. For a detailed comparison of the CFA635 and CFA735 families, see [CFA735_Migration_doc.pdf](#).

DIFFERENCE BETWEEN THE TWO SERIAL INTERFACES

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

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

TTL “Logic Level, Inverted” Serial (CFA635-xxx-KL)

The CFA635-xxx-KL simply exposes these UART Tx & Rx (inverted, logic level, 0v to 5v nominal) signals on pin 1 and pin 2 of the CFA635’s expansion connector H1. If your embedded processor is in close physical proximity to the CFA635, you can cable its UART Rx and Tx pins directly to the CFA635-xxx-KL’s Tx and Rx pins. No RS232 level translators are required on either end.



“Full Swing” RS232 Serial (CFA635-xxx-KS)

The CFA635-xxx-KS is a CFA635-xxx-KL with an RS232 level translator board ([CFA-RS232](#)) attached. The CFA635-xxx-KS 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).

Note

The USB connector on the CFA635-xxx-KL and CFA635-xxx-KS is disabled and cannot be used.

MAIN FEATURES

- Large easy-to-read LCD in a compact size can show 20 characters x 4 lines.
- Attractive stainless steel bezel.
- Fits nicely in a 1U rack mount case (37 mm overall height) using our optional mounting bracket. Or mount in a DS635, a SLED chassis that holds the display module, the optional [SCAB](#) and a 3.5-inch disk drive.
- Displays have a 12 o'clock viewing direction (polarizer viewing direction). See [Definition Of 6 O'Clock And 12:00 O'Clock Viewing Angles \(Pg. 34\)](#).
- 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 [Connectors On The Two Parts \(Pg. 37\)](#)).
- Six-button translucent silicone keypad with screened legend is backlit with LEDs. Fully decoded keypad: an key combination is valid and unique.
- Edge-lit backlit with 8 LEDs, 4 per side:
- The CFA635 family has three color (variant) choices.
 - *CFA635-TFE-KS*: White LED backlight with positive FSTN neutral transfective mode LCD. Displays dark (near-black) characters on light (near-white) background. The display can be read in normal office lighting, in dark areas, and in bright sunlight. Keypad is backlit with white LEDs.
 - *CFA635-TMF-KS*: White LED backlight with negative STN blue transmissive mode LCD. Displays light (near-white) characters on blue background. The display can be read in normal office lighting and in dark areas. May be difficult to read in direct sunlight. Keypad is backlit with blue LEDs.
 - *CFA635-YYE-KS*: Yellow-green LED backlight with positive STN yellow-green transfective mode LCD. Displays dark (near-black) characters on yellow-green background. The display can be read in normal office lighting, in dark areas, and in bright sunlight. Keypad is backlit with yellow-green LEDs.
- Adjustable contrast. The default contrast value for the module will be acceptable for many applications. If necessary, you can adjust the contrast using command [13 \(0x0D\): Set LCD Contrast \(Pg. 59\)](#).
- The front of the display has four bicolor (red + green) LED status lights. 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 CFA635-xxx-KS has a RockWorks RW1067 controller.
- Robust packet based communications protocol with 16-bit CRC.
- Optional ATX power supply control functionality allows the keypad buttons to replace the Power and Reset switches on your system, simplifying front panel design. Optional 16-pin Crystalfontz [WR-PWR-Y25](#) ATX cable may be used for direct connection to the host.
- Nonvolatile memory capability (EEPROM):
 - Customize the “power-on” display settings.
 - 16-byte “scratch” register for storing IP address, netmask, system serial number . . .
- Hardware watchdog can reset host on host software failure.
- The CFA635-xxx-KS may be used with our optional [SCAB](#) (System Cooling Accessory Board). The combination of the CFA635-xxx-KS with the optional SCAB (CFA635+SCAB) allows:



- Three functional fan connectors with RPM monitoring and variable PWM (Pulse Width Modulation) fan power control. Fail-safe fan power settings allows host to safely control three fans based on temperature. Commonly available PC cooling fans may be used. (Fans not sold by Crystalfontz.) A USB version of this module combined with a SCAB allows control of four fans. In this serial version, the connections for a fourth fan are remapped and used as the serial Rx and Tx connections. See Command [25 \(0x19\): Set Fan Power Fail-Safe \(SCAB Required\) \(Pg. 64\)](#).
- Add up to 32 Crystalfontz [WR-DOW-Y17](#) cables with DOW (Dallas 1-Wire) DS18B20 temperature sensors. Monitor temperatures at up to 0.5°C absolute accuracy.
- For ATX power supply control functionality when connected to a SCAB, buy the [WR-PWR-Y14](#) ATX power cable.
- For more information, see [ATX Power Supply \(Pg. 45\)](#) and the [SCAB](#) Data Sheet on our website.
- ❑ RoHS compliant. You can download the *Certificate of Compliance for ISO, RoHS, and REACH* from the Doc/Files tab on any CFA635 part number's website page.
- ❑ If you need a CE or MET (UL type) approved module, please see our [XES635 USB family](#). The CFA635-xxx-KS does not have CE certification because it is not an end product. The module requires power and communications from another system in order to operate.



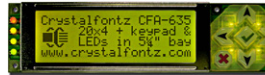


EXPLANATION OF PART NUMBER CODES IN THIS DATA SHEET

<u>CFA</u>	<u>635</u>	-	<u>X</u>	<u>X</u>	<u>X</u>	-	<u>K</u>	<u>S</u>	
①	②		③	④	⑤		⑥	⑦	⑧

①	Brand	Crystalfontz America, Inc.
②	Model Identifier	635
③	Backlight Type & Color	X = T – LED, white Y – LED, yellow-green
④	Fluid Type, Image (positive or negative), & LCD Glass Color	X = F – FSTN, positive, neutral M – STN, negative blue Y – STN, positive, yellow-green
⑤	Polarizer Film Type, Operating Temperature Range¹, & View Angle (O 'Clock)²	X = E – Transflective, NT, 12:00 F – Transmissive, NT, 12:00
<p>¹Normal Temperature operating range is 0°C minimum to +50°C maximum. If you need Wide Temperature operating range, (-20°C minimum to -70°C maximum) see the CFA735 family which is shipped with CFA635 emulation firmware.</p> <p>²For more information on Viewing Angle, see Definition Of Vertical And Horizontal Viewing Angles (CR>2)</p>		
⑥	Special Code 1	K – Manufacturer's code
⑦	Interface Code	S – Serial with "full swing" RS232
⑧	Customize Configuration Codes³	X = 1 or more characters ²
<p>³When you order a CFA635 through our website, you may be offered a choice of configurations (including accessories) to add to your order through our <i>Customize and Add to Cart</i> feature. Additional choices may be available through our Kit Configurator. Choices vary by interface.</p>		



Part Number	 CFA635-TFE-KS	 CFA635-TMF-KS	 CFA635-YYE-KS
Fluid	FSTN	STN	STN
LCD Glass Color	neutral	blue	yellow-green
Image	positive	negative	positive
Polarizer Film	transflective	transmissive	transflective
LEDs	Backlight: white Keypad: white	Backlight: white Keypad: blue	Backlight: yellow-green Keypad: yellow-green

Notes

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 silicone.

Additional modules in the CFA635 family are:

- A serial "logic level, inverted" 0v to +5v nominal interface. Part numbers end in "-KL". See www.crystalfontz.com/products/635serial.
- USB interface. Part numbers end in "-KU". See www.crystalfontz.com/products/635.
- An external enclosure with a captive USB "A" cable connection. See www.crystalfontz.com/family/XES635BK.



MECHANICAL SPECIFICATIONS

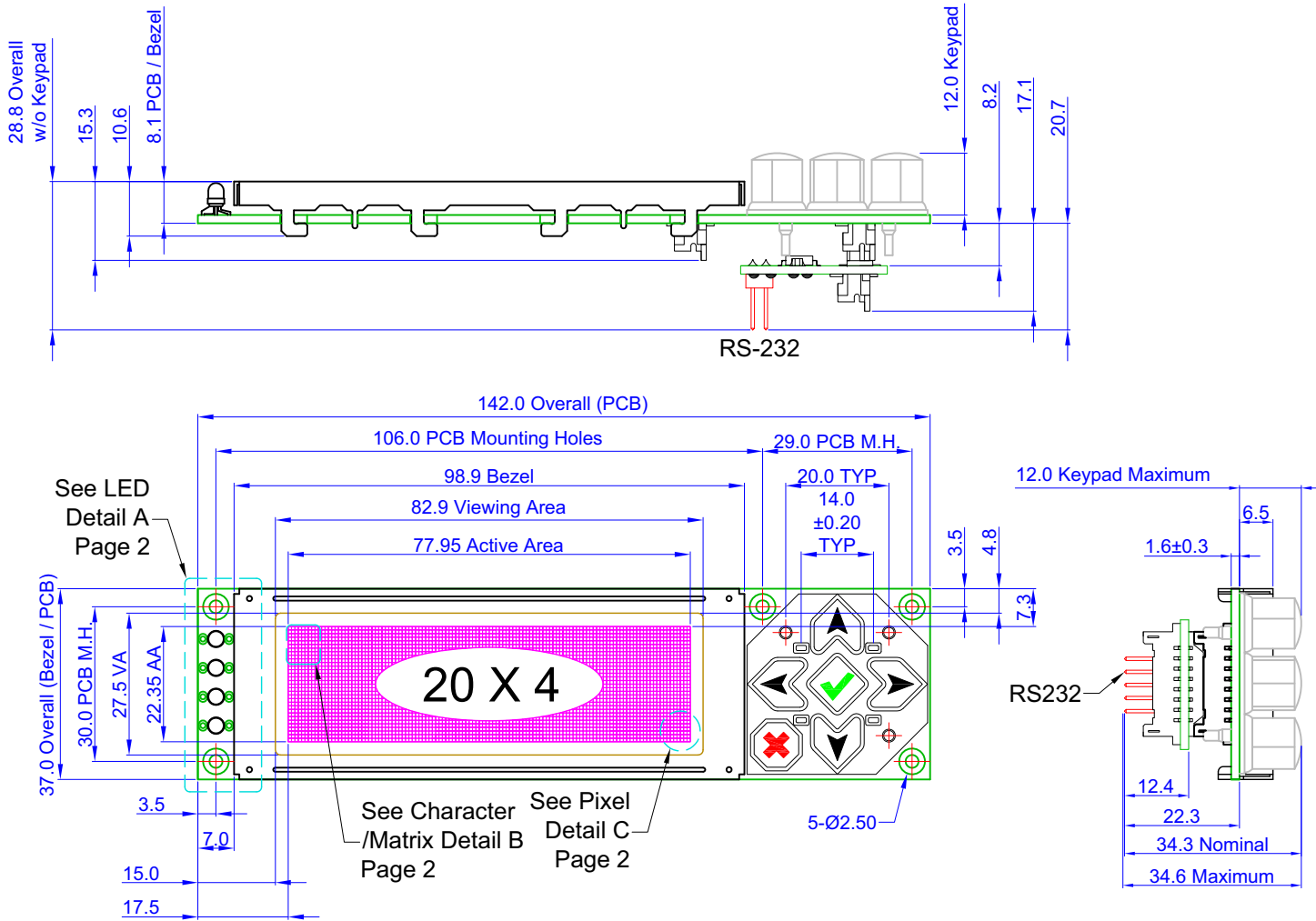
PHYSICAL CHARACTERISTICS

ITEM	SPECIFICATION
Pixels	
Pixel Size	Millimeters: 0.60 (W) x 0.65 (H) mm Inches: 0.024" (W) x 0.026" (H)
Pixel Pitch	Millimeters: 0.65 (W) x 0.70 (H) mm Inches: 0.026" (W) x 0.028" (H)
Active Area	Millimeters: 77.95 (W) x 22.35 (H) mm Inches: 3.07" (W) x 0.88" (H)
Viewing Area	Millimeters: 82.90 (W) x 27.50 (H) mm Inches: 3.26"(W) x 1.08" (H)
Character	
5x7 Standard Character Size	Millimeters: 3.20 (W) x 4.85 (H) mm Inches: 0.126" (W) x 0.190"(H)
6x8 Matrix (used for special characters or graphics)	Millimeters: 3.90 (W) X 5.60 (H) mm Inches: 0.15" (W) x 0.22" (H)
Module Overall Dimensions	
Width and Height	Millimeters: 142.00 (W) mm x 37.00 (H) mm Inches: 5.59" (W) mm x 1.46" (H) mm
Module Depth ¹ : with Keypad, with Connectors	Millimeters: 34.30 mm nominal, 34.60 mm maximum Inches: 1.35" nominal, 1.36" maximum
Weight ¹	68 grams (typical)
Keystroke Travel (approximate)	~2.4 mm
¹ Includes CFA-RS232 mounted to back of PCB.	



MODULE OUTLINE DRAWING FRONT AND SIDE VIEWS

Figure 1. Module Outline Drawing Front And Side Views



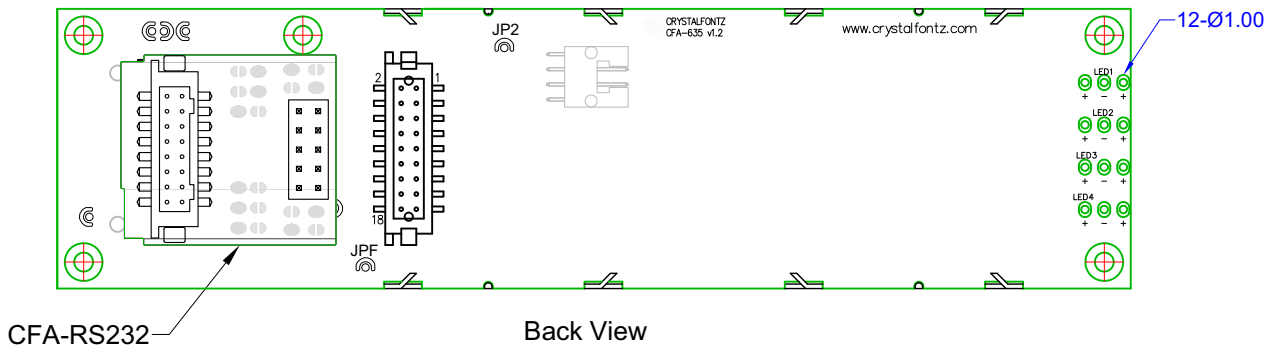
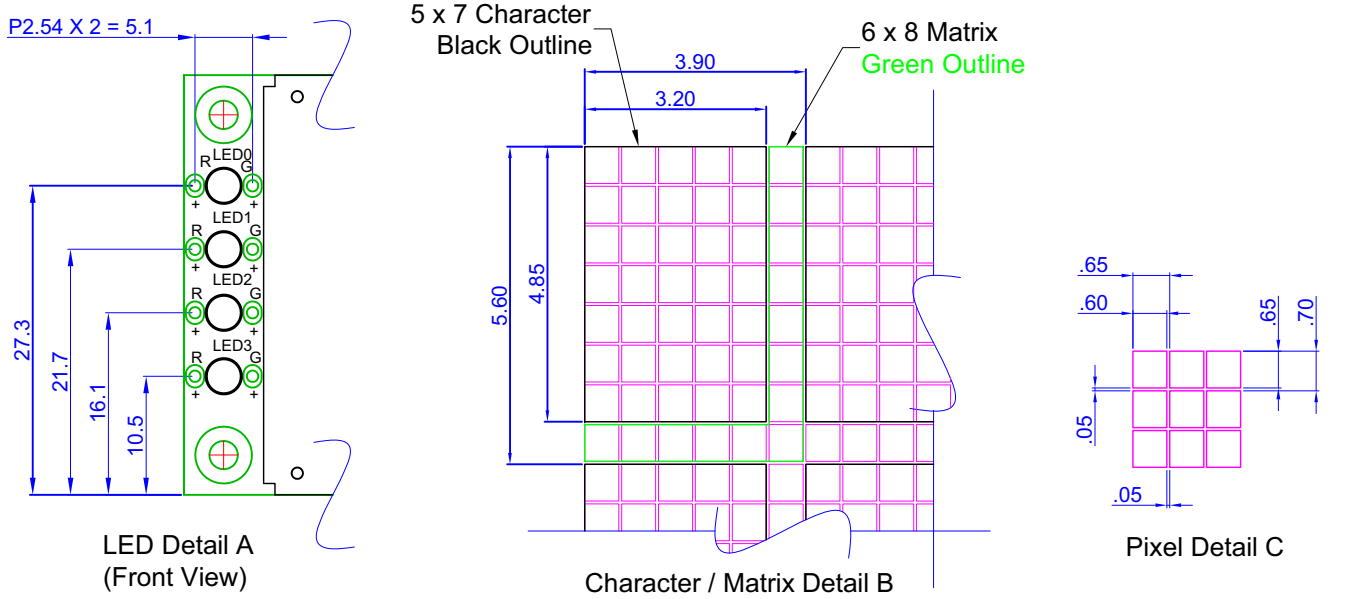
Note: Tolerance is ±0.5 mm unless specified.





MODULE OUTLINE DRAWING BACK VIEW AND PIXEL DETAIL

Figure 2. Module Outline Drawing Back View And Pixel Detail



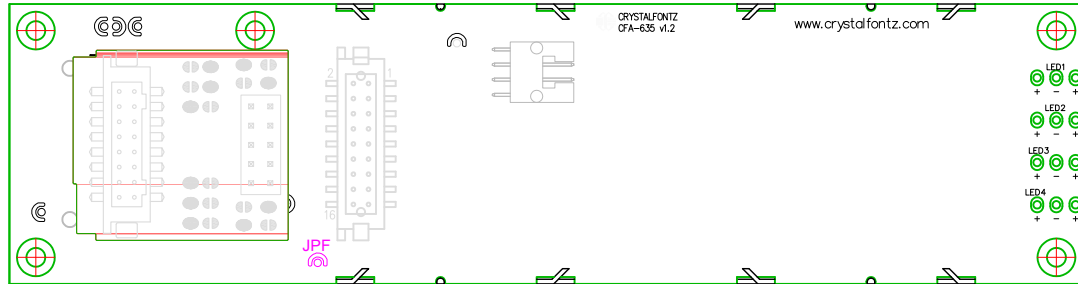
Note: Tolerance is ± 0.5 mm unless specified.





JUMPERS THAT CAN BE MODIFIED

This CFA635-xxx-KS has eight jumpers. Only JPF may be changed. To open a jumper, remove the solder. Solder wick works well for this. To close a jumper, melt solder across the gap.



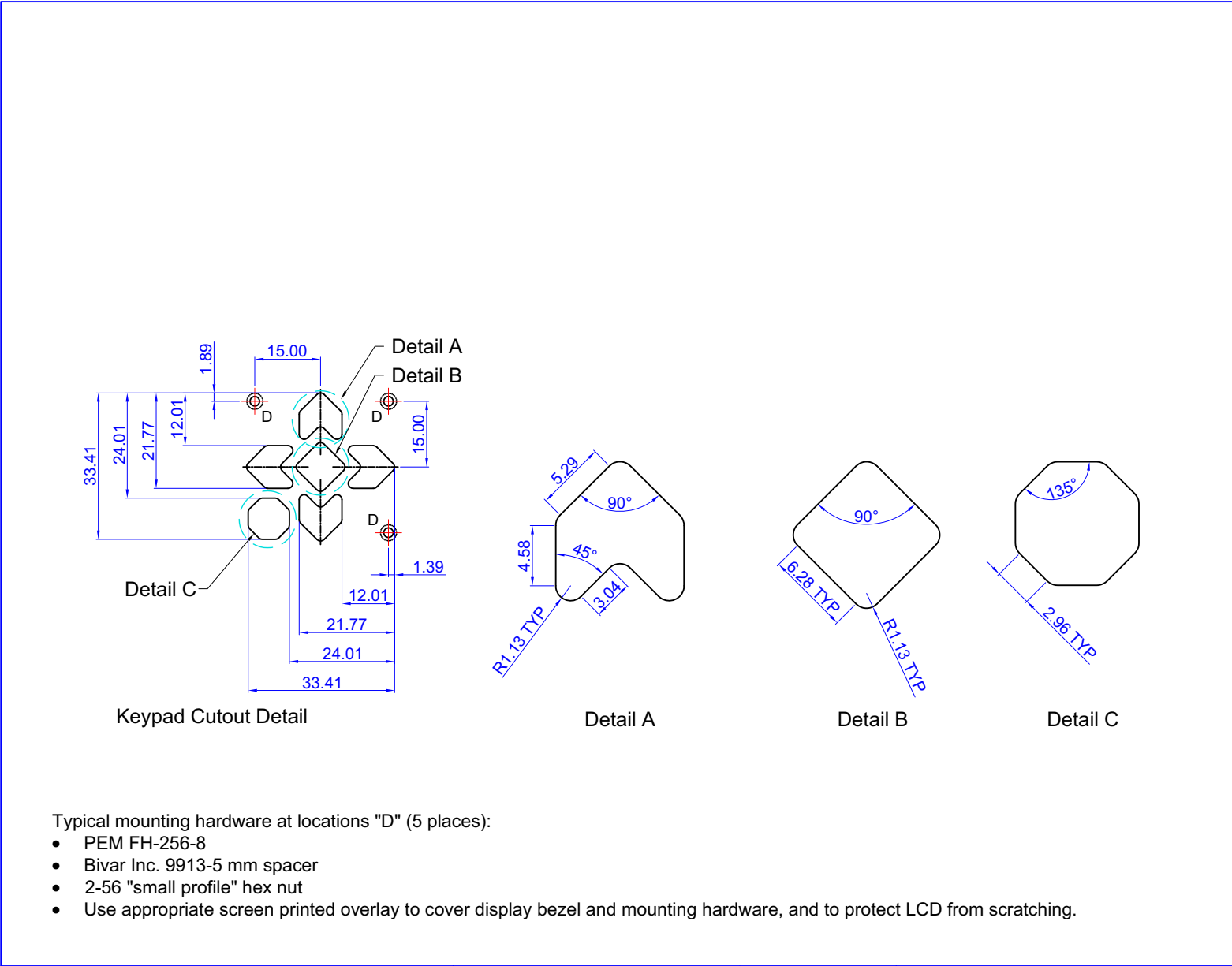
All other jumpers are factory build options. Do not change.

JPF	open	Shipped with JPF open. Frame ground is isolated from logic/USB ground.
	closed	Close JPF to connect frame ground to logic/USB ground.

Figure 3. Jumpers that Can Be Modified



PANEL MOUNTING APPLICATION CUTOUT



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

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



Crystalfontz America, Inc.

www.crystalfontz.com/products/

copyright © 2012 by

Part No.(s):

6-Button Keypad
Panel Mounting
Application Cutout

Scale:
Not to scale

Units:
Millimeters

Drawing Number:
6-Button_Panel_Cutout_Master

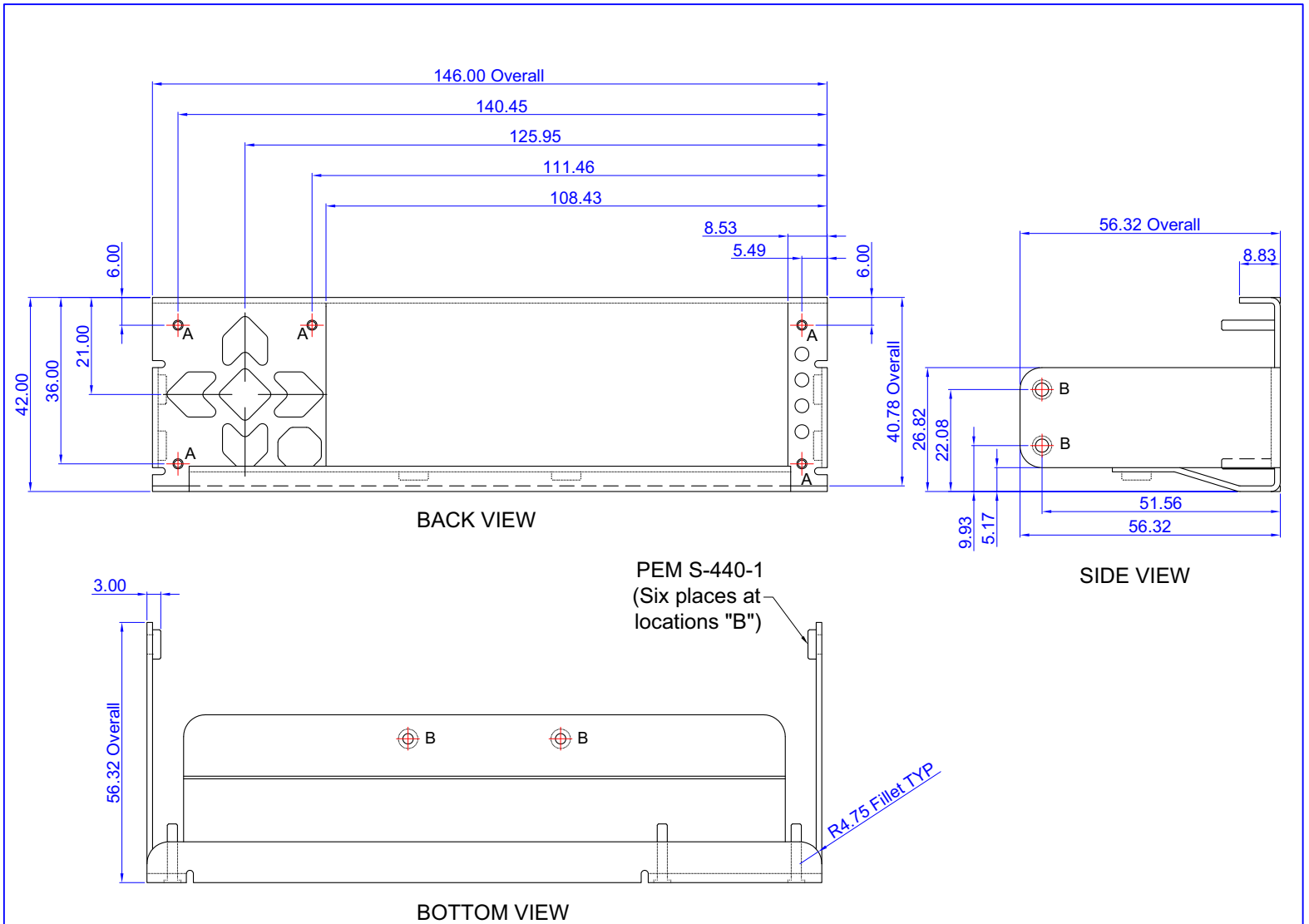
Date:
2014-03-14

Sheet:
1 of 1

Figure 4. Panel Mounting Application Cutout Drawing




OPTIONAL MOUNTING BRACKET



Note:
1. Tolerance is ± 0.30 mm unless specified.

Figure 5: Optional Mounting Bracket Drawing

 <p>copyright © 2012 by CrystalFontz America, Inc. www.crystalfontz.com/products/</p>	<p>Part No.(s): CFA635 Family Mounting Bracket</p>	<p>Scale: Not to scale</p>	<p>Drawing Number: Bracket_master</p>	
		<p>Units: Millimeters</p>	<p>Date: 2012/02/21</p>	<p>Sheet: 1 of 1</p>



KEYPAD DETAIL DRAWING

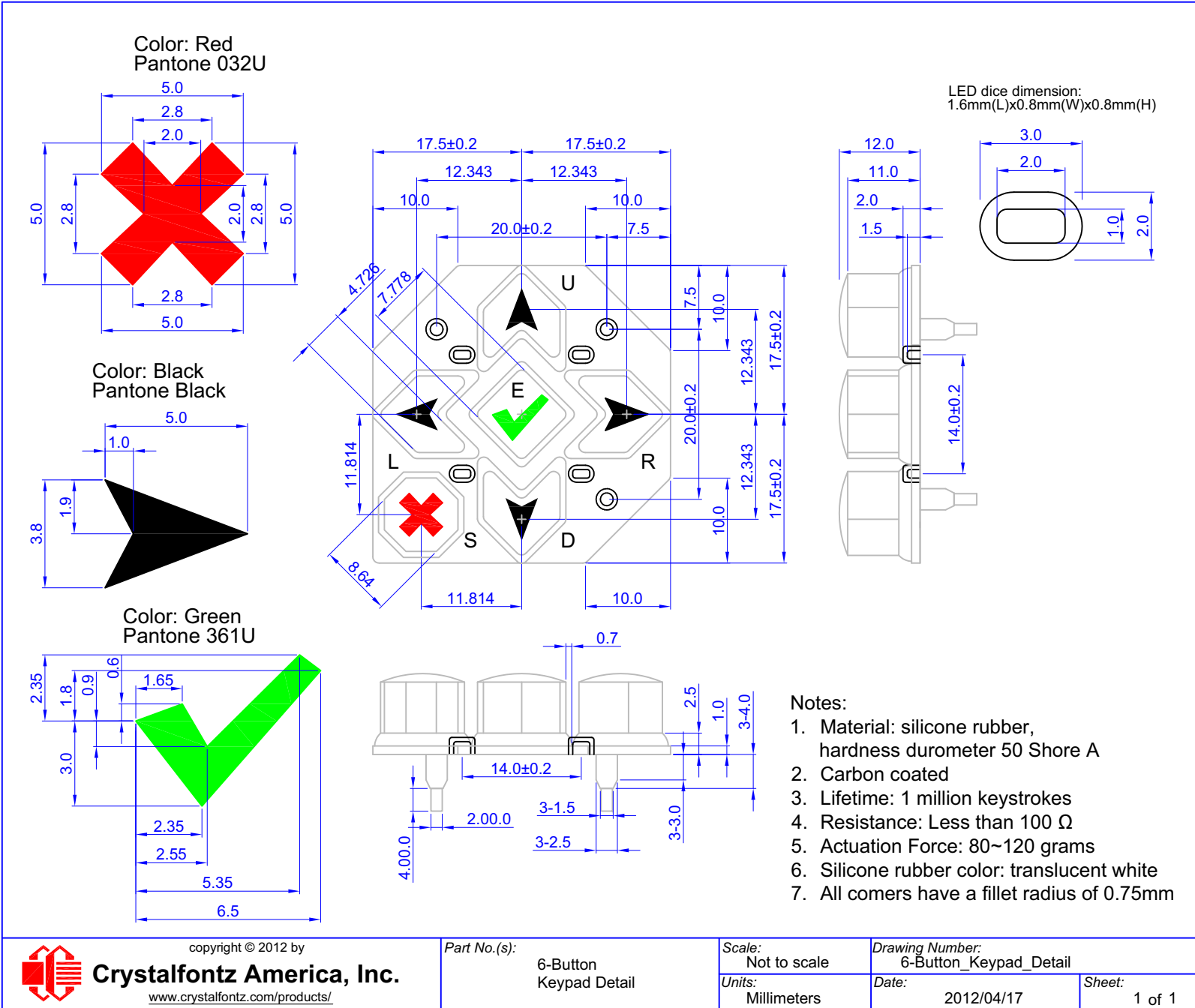


Figure 6. Keypad Detail Drawing



copyright © 2012 by
Crystalfontz America, Inc.
www.crystalfontz.com/products/

Part No.(s):
6-Button
Keypad Detail

Scale:
Not to scale
Units:
Millimeters

Drawing Number:
6-Button_Keypad_Detail
Date:
2012/04/17

Sheet:
1 of 1



CFA635-XXX-KS HAS TWO PARTS: CFA635-XXX-KL LCD MODULE AND CFA-RS232 SERIAL CONVERTER

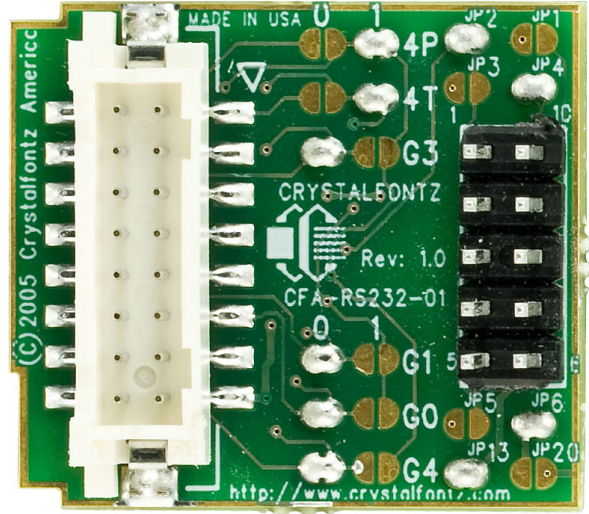


Figure 7. Front View Of CFA-RS232 Serial Converter

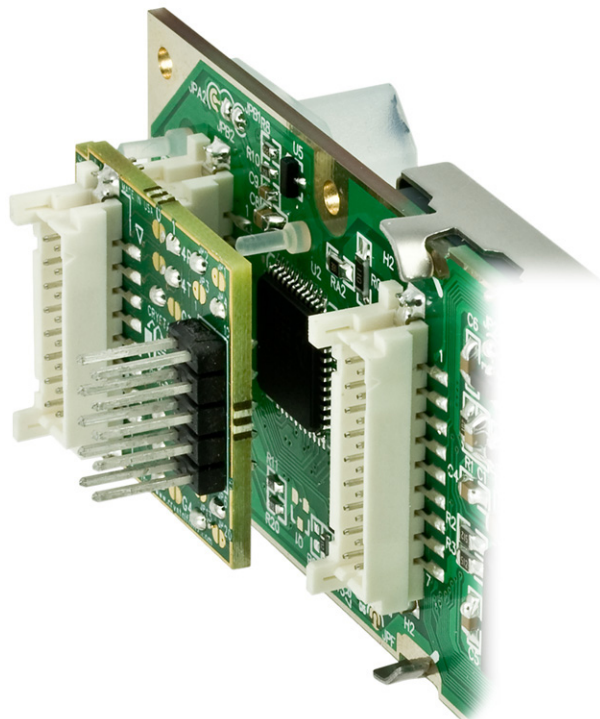


Figure 8. CFA-RS232 Serial Converter Mounted On CFA635-xxx-KL LCD Module



The CFA635-xxx-KS has two parts:

1. CFA635-xxx-KL Serial LCD Module

Physically, the CFA635-xxx-KL is identical to the USB version of this module CFA635-xxx-KU. However, the firmware on this serial module maps the Rx and Tx signals to connector H1 instead of to the USB chip.

2. CFA-RS232 Serial Converter

The CFA-RS232 Serial Converter is a small printed circuit assembly mounted on the CFA635-xxx-KL serial LCD module. The CFA-RS232 has a 16-pin female connector J3 (see location of J3 connector on [Figure 16. on Pg. 38](#)) that mates with the male 16-pin connector H1 on the back of the CFA635-xxx-KL. The CFA-RS232 Serial Converter converts the 0v to +5v (“logic level, inverted”) Rx and Tx signals from the CFA635-xxx-KL’s microcontroller to RS232 levels.

For more information, see [Three Methods For Power Connection To Host \(Pg. 44\)](#).



ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

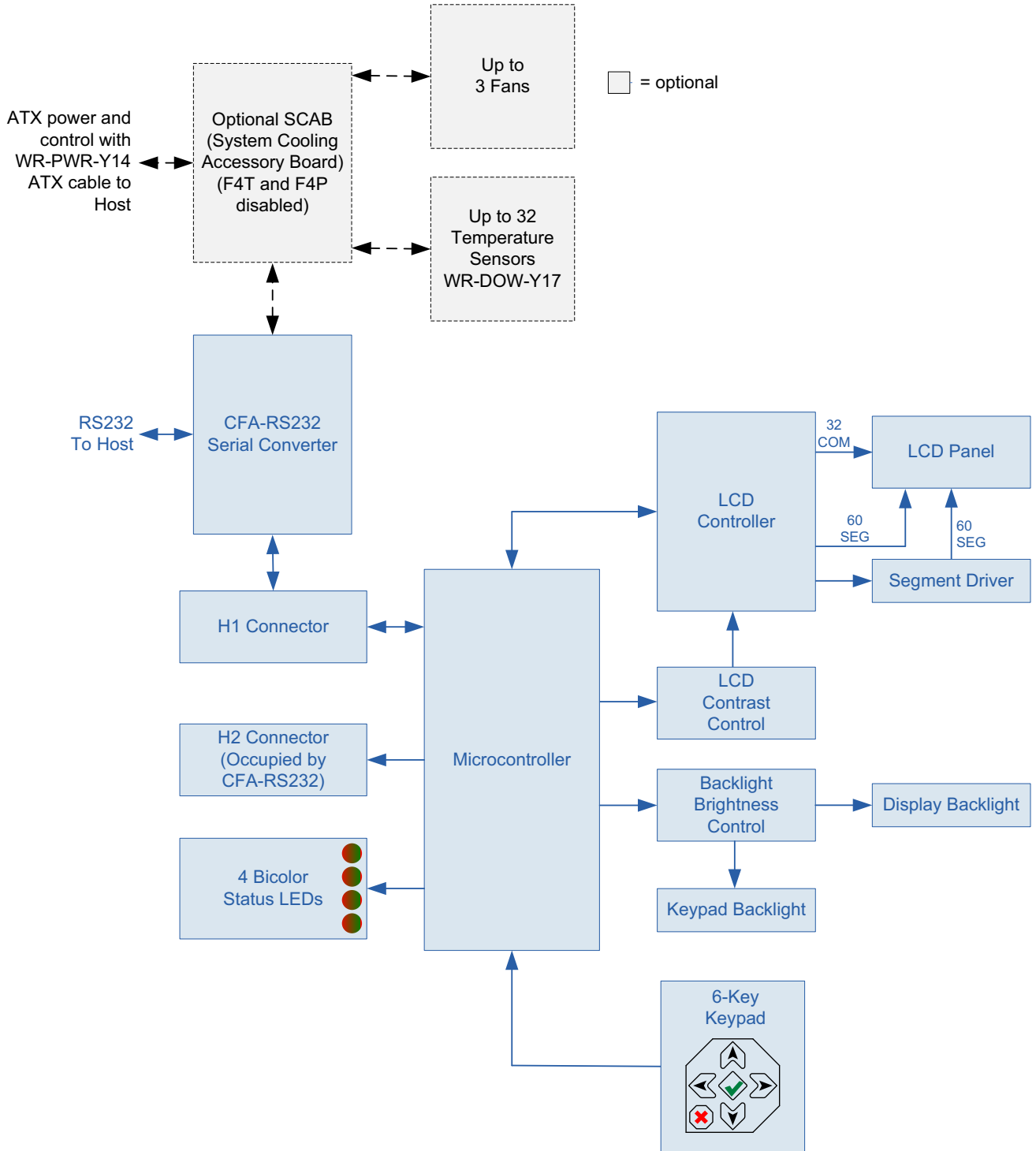


Figure 9. System Block Diagram



LCD DUTY AND BIAS

DRIVING METHOD	SPECIFICATION
Duty ¹	1/32
Bias ²	6.7

¹The duty cycle, also known as duty ratio or multiplex rate, is the fraction of total frame time that each row of the LCD is addressed.

²The drive bias, also known as voltage margin, is related to the number of voltage levels used when driving the LCD. Bias is defined as $1/(\text{number of voltage levels}-1)$. The more segments driven by each driver(1), the higher number of voltage levels are required. There is a direct relationship between the bias and the duty.

ABSOLUTE MAXIMUM RATINGS

ABSOLUTE MAXIMUM RATINGS	SYMBOL	MINIMUM	MAXIMUM
Operating Temperature	T _{OP}	0°C	+50°C
Storage Temperature	T _{ST}	-10°C	+60°C
Humidity Range (Noncondensing)	RH	10%	90%
Supply Voltage for Logic	V _{DD}	0v	+5.25v
Input and Output Pins for RS232 Serial (for CFA634-xxx-KS only)			
RS232 Input Pin	V _{RX}	-25v	+25v
RS232 Output Pin	V _{TX}	-13v	+13v
<p>Notes: <i>These are stress ratings only. Extended exposure to the absolute maximum ratings listed above may affect device reliability or cause permanent damage. Functional operation of the module at these conditions beyond those listed under DC Characteristics (Pg. 26) is not implied.</i></p> <p><i>Changes in temperature can result in changes in contrast.</i></p>			



DC CHARACTERISTICS

	SPECIFICATION	SYMBOL	MINIMUM	TYPICAL	MAXIMUM
CONTROLLER AND BOARD	RS232 Input Voltage Range	V_{IH}	-25v		+25v
	RS232 Input High Voltage	V_{IL}		+1.8v	+2.4v
	RS232 Input Low Voltage	V_{OH}	+0.8v	+1.5v	
	RS232 Output Voltage Swing	V_{OL}	$\pm 5.0v$	$\pm 5.4v$	

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 if the optional [SCAB](#) is attached to the module. Current consumption for all interfaces are the same: TTL "Logic Level, Inverted" Serial (CFA635-xxx-KL), "Full swing" RS232 Serial (CFA635-xxx-KS), and USB (CFA635-xxx-KU).



CFA635-TFE-Kx (Near-Black on White)



ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION	
Logic	LCD and Keypad Backlights at 100%	All Status LEDs 4 Red + 4 Green at 100%	V _{DD} = +4.75	V _{DD} = +5.25v
X	-	-	22 mA	24 mA
X	X	-	112 mA	136 mA
X	-	X	134 mA	156 mA
X	X	X	220 mA	262 mA

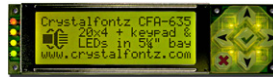
CFA635-TMF-Kx (Near-White on Blue)



ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION	
Logic	LCD and Keypad Backlights at 100%	All Status LEDs 4 Red + 4 Green at 100%	V _{DD} = +4.75	V _{DD} = +5.25v
X	-	-	20 mA	20 mA
X	X	-	110 mA	134 mA
X	-	X	132 mA	154 mA
X	X	X	220 mA	264 mA



CFA635-YYE-Kx (Near-Black on Yellow-Green)



ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION	
Logic	LCD and Keypad Backlights at 100%	All Status LEDs 4 Red + 4 Green at 100%	V _{DD} = +4.75	V _{DD} = +5.25v
X	-	-	20 mA	24 mA
X	X	-	134 mA	136 mA
X	-	X	132 mA	155 mA
X	X	X	240 mA	262 mA

GPIO CURRENT LIMITS

TYPICAL GPIO CURRENT LIMITS	
Sink	25 mA
Source	10 mA

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS

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

BACKLIGHT AND FAN CRITERIA

BACKLIGHT AND FAN ¹ CRITERIA	SPECIFICATION
Backlight PWM ² Frequency	300 Hz nominal
Fan Tachometer Speed Range (assuming two PPR ³)	600 RPM to 3,000,000 RPM
Fan Power Control PWM ² Frequency	18 Hz nominal

¹Optional SCAB is required to add fans.
²PWM is *Pulse Width Modulation*. PWM is a way to simulate intermediate levels by switching a level between full on and full off. PWM can be used to control the brightness of LED backlights, relying on the natural averaging done by the human eye, as well as for controlling fan power.
³PPR is *Pulses Per Revolution*, can also written as p/r.



OPTICAL SPECIFICATIONS

Viewing Direction	12 o'clock
-------------------	------------

CFA635-TFE-KX (NEAR-BLACK ON WHITE)



ITEM	SYMBOL	CONDITION	TYPICAL
<i>Text Condition: Ta = 25°C</i>			
Viewing Angle (12 o'clock)	Deg $\theta = 90^\circ$	CR \geq 2	60
	Deg $\theta = 270^\circ$		30
	Deg $\theta = 0^\circ$		40
	Deg $\theta = 180^\circ$		40
Contrast Ratio ¹	CR		6.3
LCD Response Time ^{2,3}	T rise	Ta = 25°C	180 ms
	T fall	Ta = 25°C	200 ms
¹ Contrast Ratio = (brightness with pixels light)/(brightness with pixels dark). ² Response Time: The amount of time it takes a liquid crystal cell to go from active to inactive or back again. ³ For reference only.			



CFA635-TMF-KX (NEAR-WHITE ON BLUE)



ITEM	SYMBOL	CONDITION	TYPICAL
Viewing Angle (12 o'clock)	Deg $\theta = 90^\circ$	CR \geq 2	40
	Deg $\theta = 270^\circ$		35
	Deg $\theta = 0^\circ$		40
	Deg $\theta = 180^\circ$		40
Contrast Ratio ¹	CR		5.0
LCD Response Time ^{2,3}	T rise	Ta = 25°C	180 ms
	T fall	Ta = 25°C	200 ms
¹ Contrast Ratio = (brightness with pixels light)/(brightness with pixels dark). ² Response Time: The amount of time it takes a liquid crystal cell to go from active to inactive or back again. ³ For reference only.			



CFA635-YYE-KX (NEAR-BLACK ON YELLOW-GREEN)



ITEM	SYMBOL	CONDITION	TYPICAL
Viewing Angle (12 o'clock)	Deg $\theta = 90^\circ$	CR \geq 2	40
	Deg $\theta = 270^\circ$		30
	Deg $\theta = 0^\circ$		40
	Deg $\theta = 180^\circ$		40
Contrast Ratio ¹	CR		5.0
LCD Response Time ^{2,3}	T rise	Ta = 25°C	180 ms
	T fall	Ta = 25°C	200 ms
¹ Contrast Ratio = (brightness with pixels light)/(brightness with pixels dark). ² Response Time: The amount of time it takes a liquid crystal cell to go from active to inactive or back again. ³ For reference only.			

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.

- 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 Optimal Contrast Setting

CFA635-TMF-Kx 

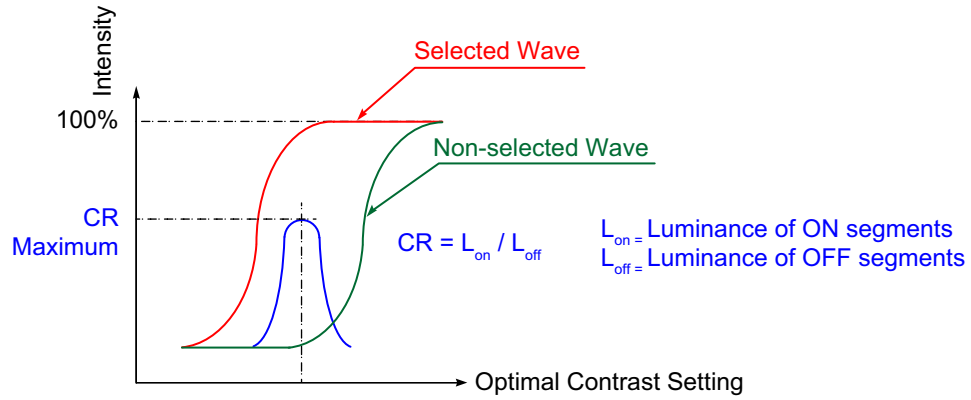


Figure 10. Definition of Optimal Contrast Setting (Negative Image)

CFA635-TFE-Kx  and CFA635-YYE-Kx 

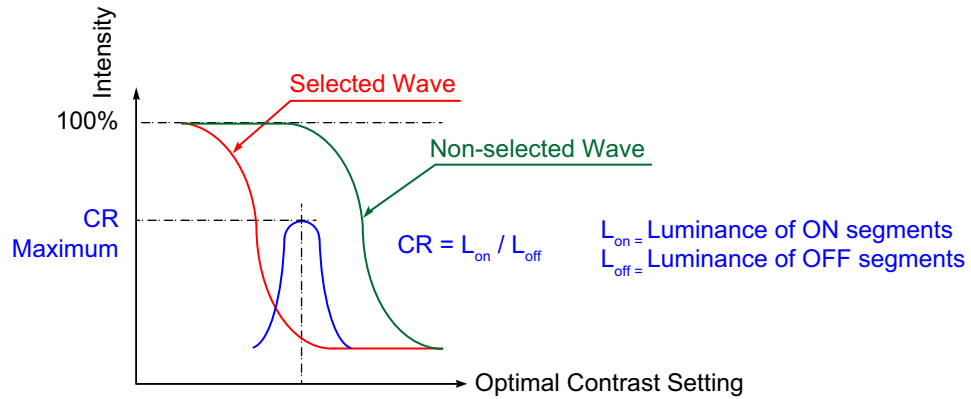


Figure 11. Definition of Optimal Contrast Setting (Positive Image)



Definition Of Response Time (T_r , T_f)

CFA635-TMF-Kx 

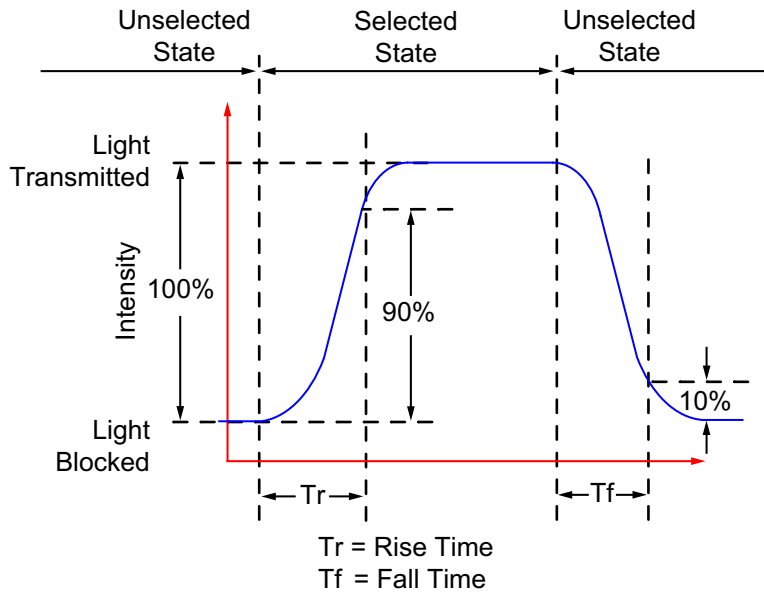


Figure 12. Definition of Response Time (T_r , T_f) (Negative Image)

CFA635-TFE-Kx  and CFA635-YYE-Kx 

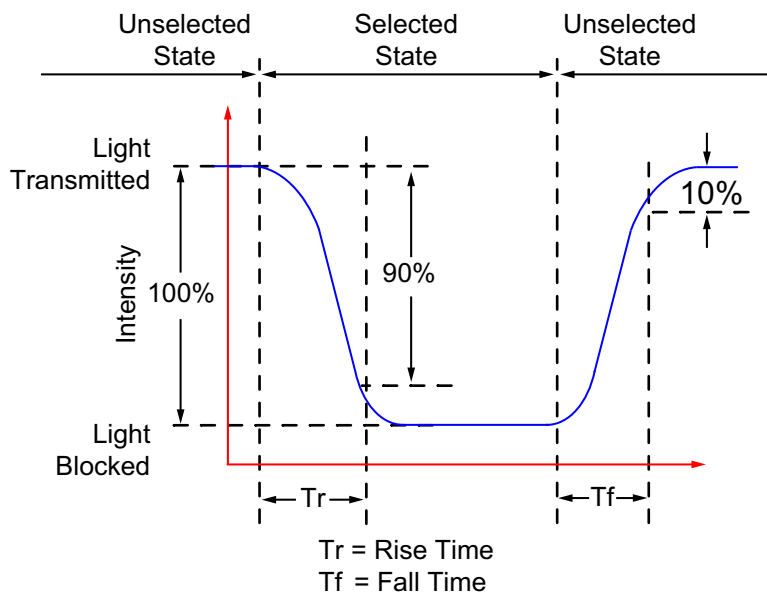


Figure 13. Definition of Response Time (T_r , T_f) (Positive Image)



Definition Of 6 O'Clock And 12:00 O'Clock Viewing Angles

This module has a 12:00 o'clock viewing angle.

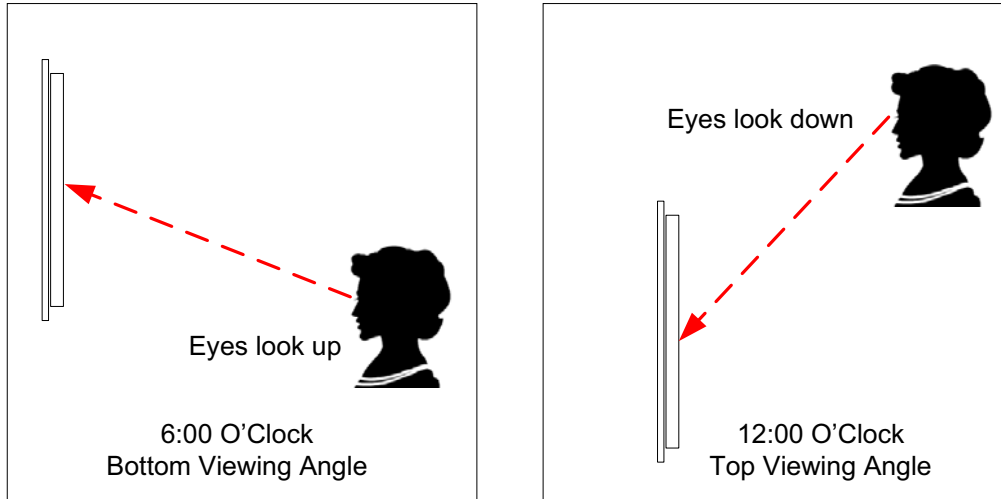


Figure 14. Definition of 6:00 O'Clock and 12:00 O'Clock Viewing Angles

Definition Of Vertical And Horizontal Viewing Angles (CR_≥2)

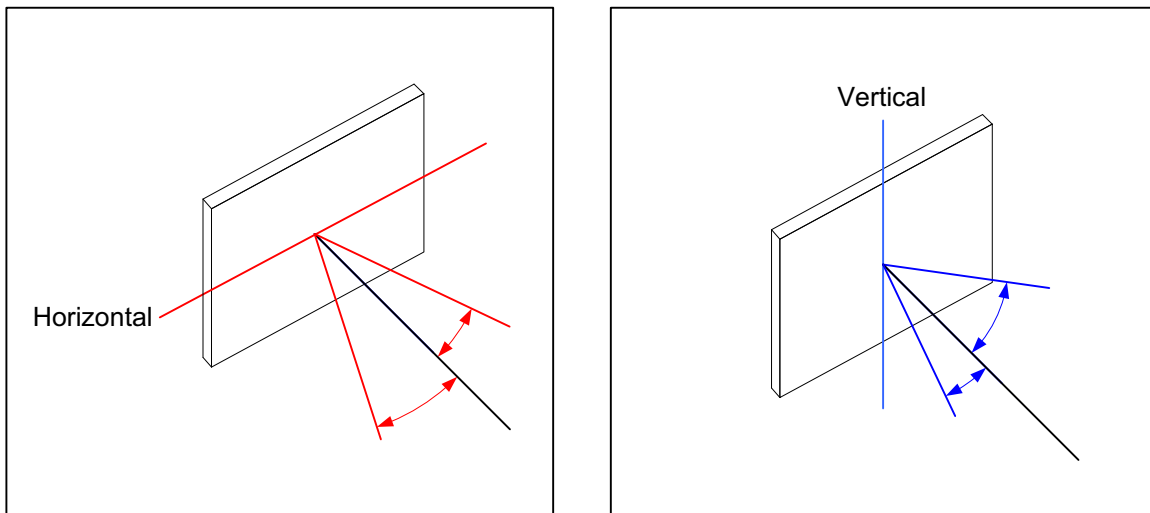




Figure 15. Definition Of Horizontal And Vertical Viewing Angles (CR>2)



LED BACKLIGHT INFORMATION

The backlight uses LEDs. The backlight is easy to use properly but it is also easily damaged by abuse.

Note

For modules with **white** backlights (CFA635-TFE-Kx  and CFA635-TMF-Kx ), we recommend that the backlight be dimmed or turned off during periods of inactivity to conserve the LEDs' lifetime.

LEDs are "current" devices. The brightness is controlled by the current flowing through it, not the voltage across it. Use a DC power supply with the correct current limiting resistance for optimum performance.



CONNECTION INFORMATION

CABLES

Following the table below are descriptions of common connection configurations. Cable lengths are approximate. When you order a CFA635-xxx-KS through our website, you are offered a choice of cables to add to your order through our “Customize and Add to Cart” feature. Choices vary by interface. Additional cables are on our website. Click on “Accessories” in the left navigation bar.

Crystalfontz Cable	Description All cables are RoHS compliant
“Full Swing” RS232 Serial	
WR-232-Y08 ~2 ft. 2.55 inches	Use this ribbon cable to supply communications to the CFA635 through the mounted CFA-RS232 converter. Connect cable’s 10-pin female connector to the mounted CFA-RS232’s 10-pin male J1 connector. Connect cable’s RS232 DB9 9-pin female connector to host’s external 9-pin serial port. Match the red stripe on the cable to pin 1 on the PCB’s silkscreen.
WR-232-Y22 ~2 ft. 1.00 inches	Use this cable to supply communications to the CFA635 through the mounted CFA-RS232 converter. This cable has three identical 10-pin female connectors, One connector is at each end and a third connector is a few inches from one end. Connect one of the cable’s end connectors to the mounted CFA-RS232’s J1 10-pin male connector. For <i>standard</i> pinout, connect the cable’s connector that is a few inches from the other end. For <i>alternate</i> pinout, connect the other end connector to the host’s motherboard 10-pin male connector.
WR-PWR-Y24 ~3 ft. 1.95 inches	Use this cable to supply power to the CFA635 through the mounted CFA-RS232 converter from the host’s power supply. Connect cable’s 16-pin female connector to mounted CFA-RS232’s 16-pin male J2 connector. Connect cable’s 4-pin male connector directly to the host’s power supply connector.
<i>Note:</i> See CFA-RS232 Serial Converter Connectors J1, J2, And J3 (Pg. 38) for location of J1 and J2.	
Cables for ATX Functionality: Use CFA635 to Power On, Power Off, or Reset the Host	
WR-PWR-Y14 ~1 ft. 11 inches	<i>With SCAB:</i> This cable allows ATX power control connections through the optional SCAB. Connect the cable’s 7-pin female connector to the SCAB’s 7-pin male J8 connector. Connect the cable’s labeled Reset, Power and 3-pin WOL connector to the host’s motherboard. Buy the WR-EXT-Y15 or WR-EXT-Y19 to connect the SCAB to the CFA635’s connector H1.
WR-PWR-Y25 ~11 inches	Connect cable’s 16-pin female connector to the mounted CFA-RS232’s 16-pin male J2 connector. Connect cable’s 4-pin male 4-pin connector directly to host’s ATX power supply. Connect cable’s 4 separate female connectors to the appropriate 4 pins on the host’s motherboard. (Cable pins are labeled.)
WR-PWR-Y38 ~2 ft. 11 inches	Identical to the WR-PWR-Y25 (described in row above) except length is longer.
Cables for Optional SCAB (System Cooling Accessory Board)	
<i>Note:</i> The CFA635-xxx-KS does not supply power to the SCAB. The SCAB requires external power, typically supplied by a 4-pin 3.5-inch floppy drive power connector.	
WR-PWR-Y14 ~1 ft. 11 inches	See description under ATX Functionality table section, above.



Crystalfontz Cable	Description (Continued) All cables are RoHS compliant
WR-EXT-Y15 ~1 ft. 5.70 inches	<p>Use this cable to mount the SCAB some distance away from the CFA635. For example, the SCAB could be mounted in a central location within the PC's case to the CFA635 mounted in a drive bay. Then the connections to the fans and temperature sensors would only need to be run to the SCAB, not all the way to the front panel where the LCD module is mounted.</p> <p>Connect one of the cable's two 16-pin female connectors to the CFA635's 16-pin H1 male connector. Connect one of the cable's two 16-pin female connectors to the mounted CFA-RS232's 16-pin male J2 connector. Connect the other 16-pin female connector to the SCAB's 16-pin male J1 connector.</p>
WR-EXT-Y19 ~3.5 inches	<p>Use this cable when the SCAB is mounted in close to the LCD module; for example, when the SCAB is fastened directly to the LCD module's optional mounting bracket.</p> <p>Connect the cable's other 16-pin female connector to the SCAB's 16-pin male J1 connector. Connect one of the cable's two 16-pin female connectors to the mounted CFA-RS232's 16-pin male J2 connector. Connect the other 16-pin female connector to the SCAB's 16-pin male J1 connector.</p>
WR-FAN-X01 ~16 inches	<p>Connect up to three cables to connect up to three fans. Connect cable's 3-pin male connector to SCAB's connectors labeled FAN1, FAN2, or FAN3, or FAN4. Connect cable's 3-pin female connector to a fan's connector. (Fans are not sold by Crystalfontz.)</p>
WR-DOW-Y17 ~12 inches + ~12 inches between connectors	<p>Connect ("daisy chain") up to 32 of these DOW (Dallas 1-Wire) DS18B20 temperature sensor cables to one SCAB. Connect the cable's 3-pin female connector to the SCAB's connector labeled J_DOW. If desired, connect the cable's 3-pin male connector to an additional temperature sensor.</p>
<p>UBERSCAB Kit (System Cooling Accessory Board + Cables)</p>	
<p>The module does not supply power to the SCAB. The SCAB requires external power, typically supplied by a 4-pin 3.5-inch floppy drive power connector. The UBERSCAB is a kit that includes one SCAB, four temperature cables (WR-DOW-Y17), four fan extension cables (WR-FAN-X01), one power cable splitter (WR-PWR-Y12), one 3.5-inch cable to connect SCAB to the module (WR-EXT-19), and one 16-inch cable to connect SCAB to the module (WR-EXT-Y15).</p>	

CONNECTORS ON THE TWO PARTS

CFA635-xxx-KS consists of two parts: CFA635-xxx-KL and CFA-RS232. These two parts have a total of six connectors. The location of five of these connectors (J1, J2, and J3 on the CFA-RS232 Serial Converter and H1 and H2 on the CFA635-xxx-KL Serial LCD Module) are shown in the figures below. The sixth connector is for USB connection and is not usable.



Part 1: CFA-RS232 Serial Converter, Connectors J1, J2, And J3

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

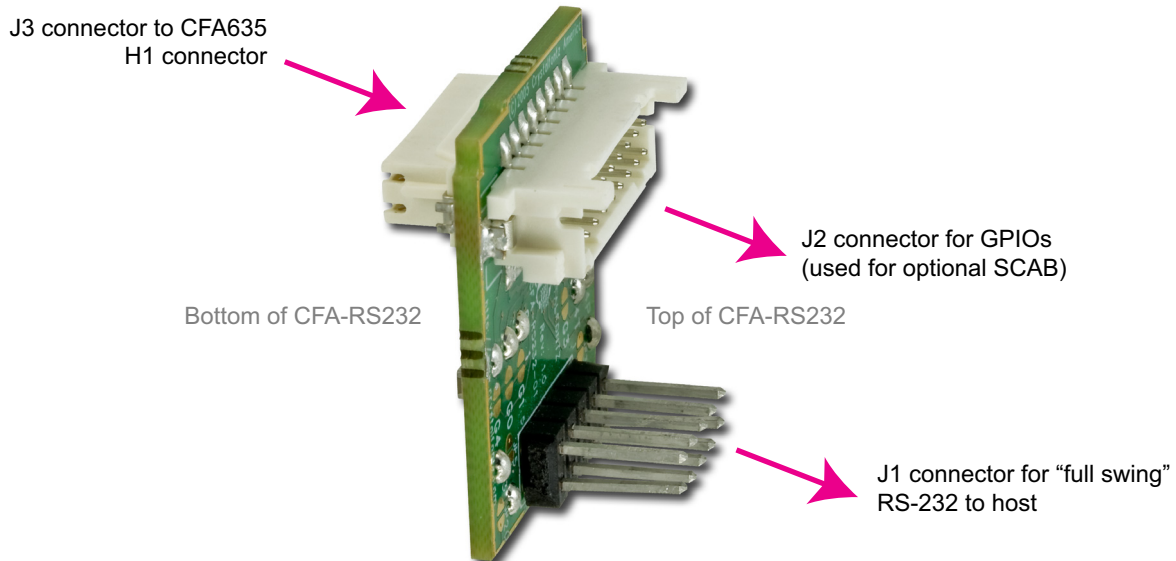


Figure 16. CFA-RS232 Serial Converter (Side View)

The J1 and J2 connectors are on the top side of the CFA-RS232 Serial Converter, facing away from the module when the CFA-RS232 is mounted on the CFA635-xxx-KL Serial LCD Module. The J3 connector is on the bottom side of the CFA-RS232 Serial Converter, facing towards the module when it is mounted on the CFA635-xxx-KS.

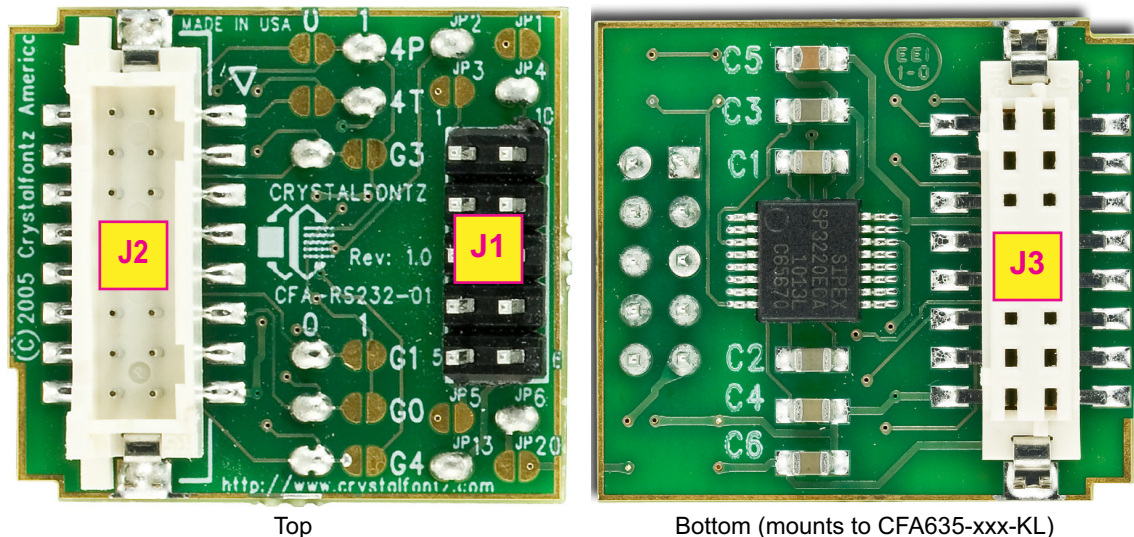


Figure 17. CFA-RS232 Serial Converter Connectors J1, J2, And J3



Top Side Connectors: J1 and J2

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 \(Pg. 40\)](#).
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 Five GPIOs \(Pg. 42\)](#). An optional [SCAB](#) may be connected to the J2.

Bottom Side Connector: J3

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 CFA635-xxx-KL.

Part 2: CFA635-xxx-KL Serial LCD Module, Connectors H1 And H2

The CFA635-xxx-KL has two connectors on its back side: *H1* and *H2*.

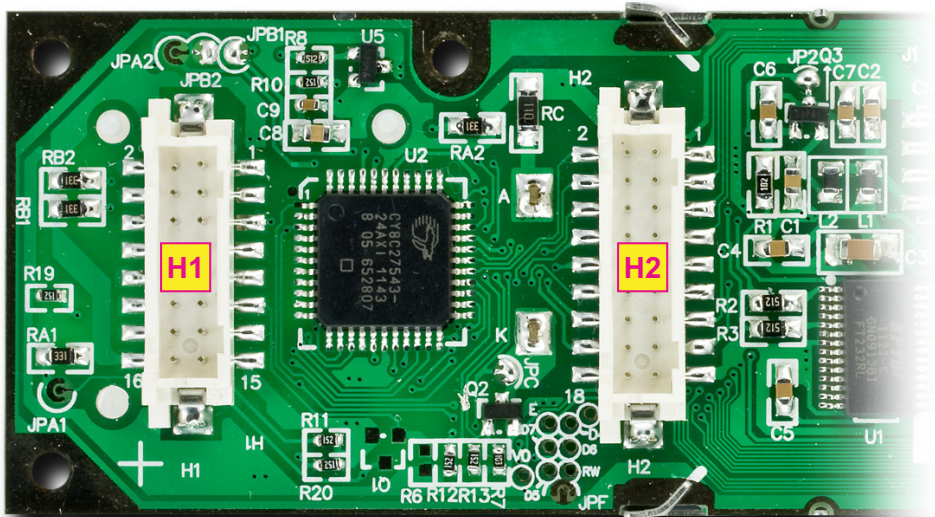


Figure 18. H1 And H2 Connectors On CFA635-xxx-KL

The *H1* male 16-pin 2 mm connector on the CFA635-xxx-KL mates with the J3 female 16-pin 2 mm connector on the bottom of the mounted CFA-RS232 Serial Converter.

The *H2* male 18-pin 2 mm connector on the LCD module provides GPIO connections, including those that drive the front panel status LEDs.



Available Connectors When CFA-RS232 Is Mounted On CFA635-xxx-KL

The CFA635-xxx-KS (CFA-RS232+CFA635-xxx-KL) has three available connectors on its back side: J1, and J2 on the mounted CFA-RS232 and H2 on the CFA635-xxx-KL.

CFA-RS232 is mounted on CFA635-xxx-KL's H1 connector.
J2 is a "pass through" connector for H1.

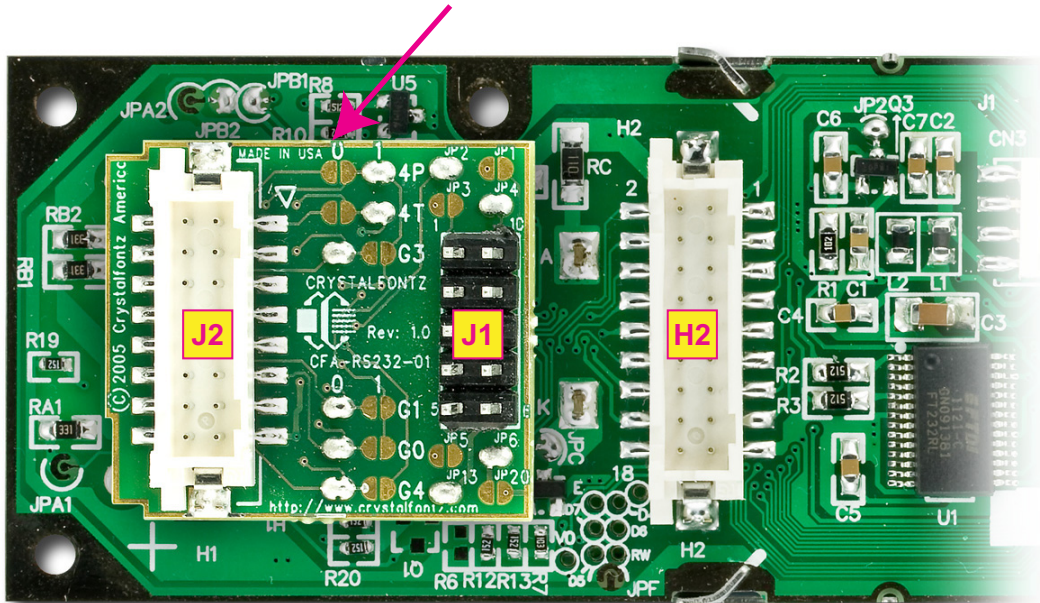


Figure 19. Available Connectors When CFA-RS232 Is Mounted On CFA635-xxx-KL

CFA-RS232 J1 CONNECTOR PIN ASSIGNMENTS

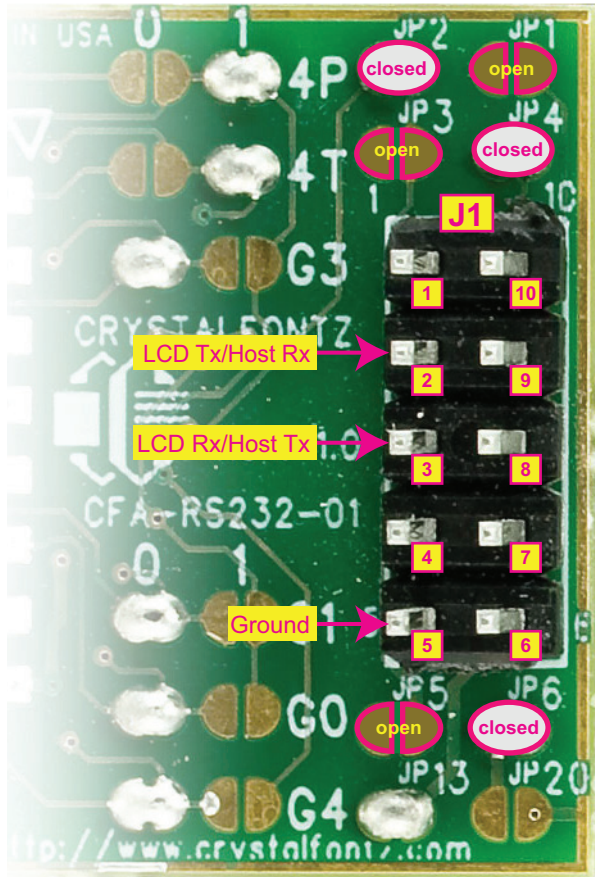
The pin order of your motherboard's header will determine if the CFA635-xxx-KS pin assignments need to be "Default" or "Alternate". Pin assignment are described below.

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.



J1 Connector Default RS232 Pin Assignments

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 CFA635-xxx-KS to a PC's DB9 COM port.



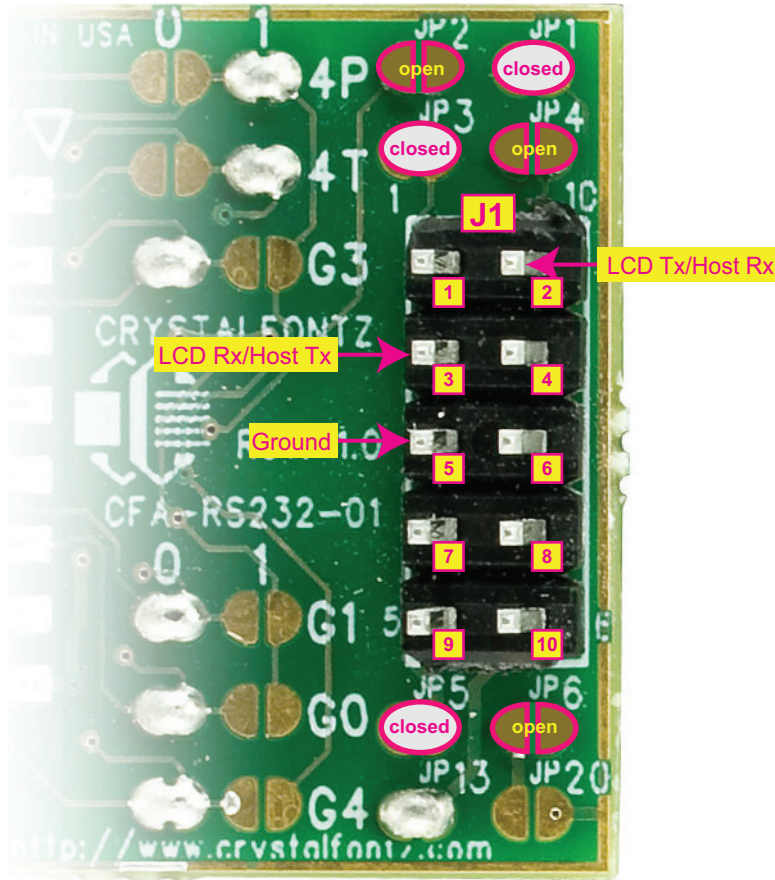
JP	STATE
JP1	open
JP2	closed
JP3	open
JP4	closed
JP5	open
JP6	closed

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



J1 Connector Alternate RS232 Pin Assignments

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



JP	STATE
JP1	closed
JP2	open
JP3	closed
JP4	open
JP5	closed
JP6	open

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 CFA635-xxx-KS to a motherboard's 10-pin header.

CFA-RS232 J2 CONNECTOR PIN ASSIGNMENTS - INCLUDES FIVE GPIOs

The CFA635-xxx-KS has five pins that can be used for “General Purpose Input or Output (GPIO)s” on the CFA-RS232 Serial Converter’s pass-through connector J2. These GPIOs can be accessed directly through J2 or through the optional [SCAB](#) connected to J2. The SCAB can be easily connected to the CFA635-xxx-KS by using either the [WR-EXT-Y15](#) or



[WR-EXT-Y19](#) cables. For more information on the SCAB, see [ATX Power Supply \(Pg. 45\)](#) and www.crystalfontz.com/product/SCAB.html.

Note

Please note that FAN4 (F4P & F4T) is disabled in the CFA635-xxx-KS. The connections for a fourth fan are remapped and used as the serial Rx and Tx connections. A USB version of the CFA635 (CFA635-xxx-KU) combined with a SCAB allows control of four fans.

Note: F1P through F3P and F1T through F3T are reserved for fans with optional SCAB.

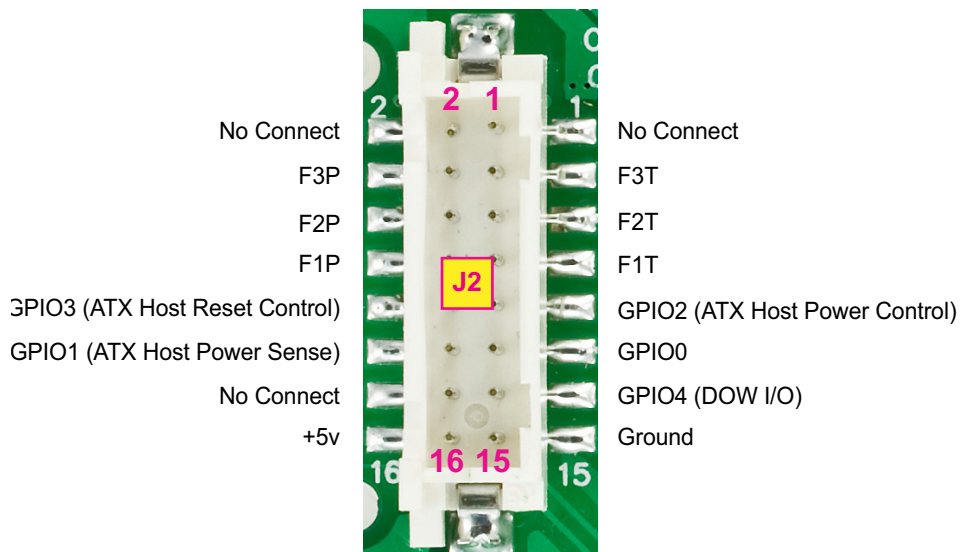


Figure 22. CFA-RS232 J2 Connector Pin Assignments

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).



H2 CONNECTOR PIN ASSIGNMENTS - INCLUDES EIGHT GPOS

The CFA635-xxx-KS has eight GPO connections available on connector H2. By factory default, these GPOs drive the front panel status LEDs. By removing the LEDs, these GPOs could be used for other purposes.

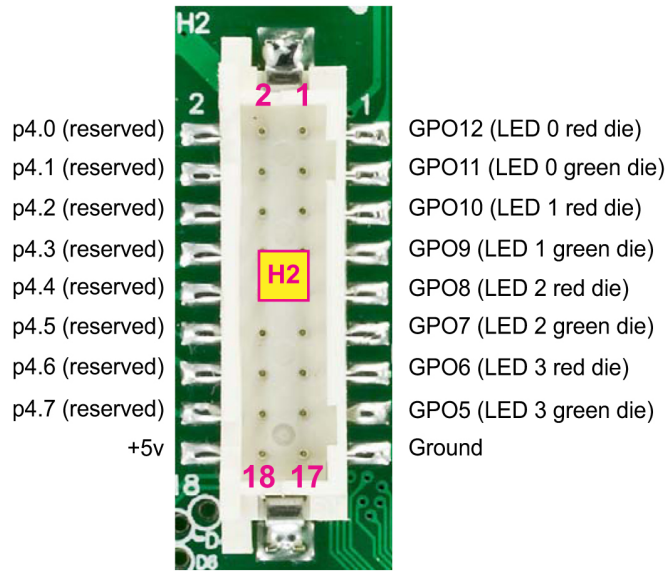


Figure 23. Pin Assignments On CFA635-xxx-KS H2 Connector - Includes Eight GPOs

Please see the commands [34 \(0x22\): GPIO And GPO Settings \(SCAB Required\) \(Pg. 69\)](#), and [35 \(0x23\): Read GPIO And GPO Pin Levels And Configuration State \(SCAB Required\) \(Pg. 72\)](#) for details on how to control the GPIOs.

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

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

THREE METHODS FOR POWER CONNECTION TO HOST

For the USB variants of the CFA635 (CFA635-xxx-KU), power as well as communications is supplied through the USB connection. Serial RS232 connections traditionally provide only communications, not power.

Choose one of these three methods to supply power to the CFA635-xxx-KS:

1. Use a [WR-PWR-Y24](#) cable or other cable / connection to provide power through J2 on the CFA-RS232 Serial Converter (typical of a PC or server installation).

or

2. Use the optional [SCAB](#) to provide power through J2 on the CFA-RS232 Serial Converter (typical of a PC or server installation).

or

3. Supply power through pin 4 of J1 on the CFA-RS232 Serial Converter (typical of an embedded system installation).

The first two options need no explanation. All that is involved is connecting a cable from the CFA635-xxx-KS to your PC or server's power supply.



For embedded systems or high volume production applications where you would like to minimize connections, you may want to use a single cable to carry both RS232 communications and power. The +5v power can be supplied through connector J1 on the CFA-RS232 Serial Converter, allowing a single cable to contain both power and data connections. If the "Default RS232 Pin Assignments" are selected, the four connections needed to operate the CFA635-xxx-KL are on a single column of pins on J1, which allows a single 0.1-inch spacing 4-conductor cable to connect between the CFA635-xxx-KS and your embedded system.

To enable +5v to be supplied through J1, jumper JP13 must be closed (default from the factory).

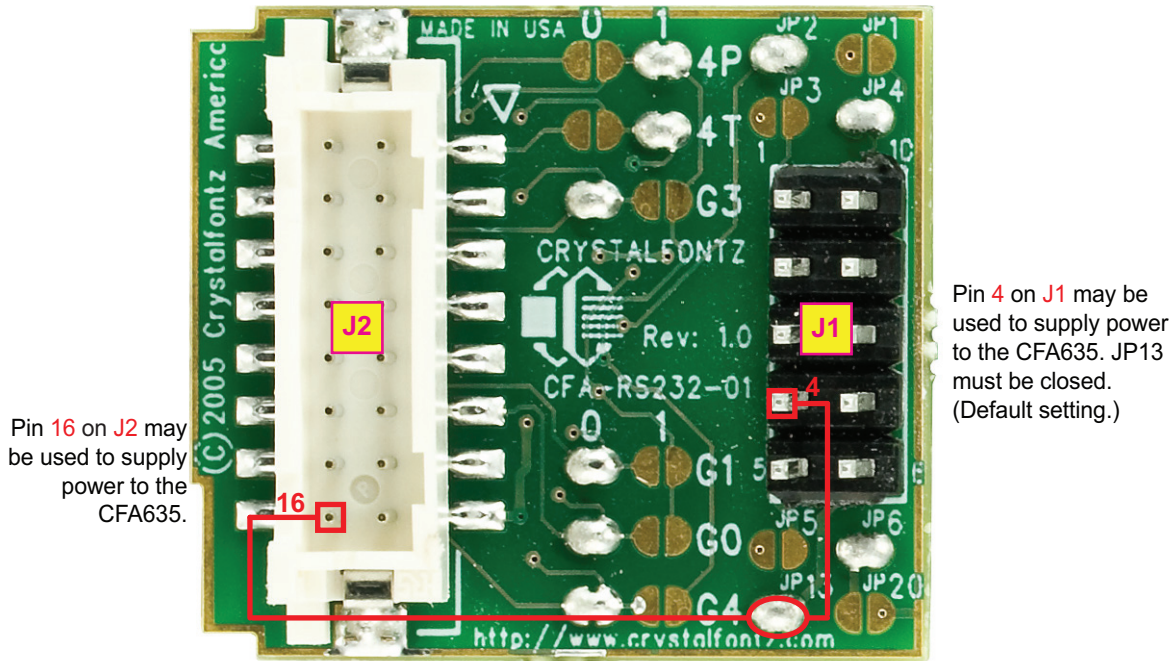


Figure 24. CFA635-xxx-KS Power Connection To Host And Optional SCAB

ATX POWER SUPPLY

ATX Power And Control Connections

ATX power supply control functionality allows the buttons on the CFA635-xxx-KS to replace the power and reset button on your system, simplifying front panel design. This ATX power supply control functionality can be accomplished with the optional [SCAB+WR-PWR-Y14](#) ATX power cable or use the [WR-PWR-Y25](#) or [WR-PWR-Y38](#) ATX power cable without the SCAB. The SCAB provides fan monitoring and control as well as DOW temperature sensor monitoring.

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 \(0x22\): GPIO And GPO Settings \(SCAB Required\) \(Pg. 69\)](#).



ATX configuration for the CFA635-xxx-KS is powered from the PC's V_{SB} signal, the "stand-by" or "always-on" +5v ATX power supply output, on pins 15 and 16 of the H1 connector. When using the optional SCAB, the +5 standby voltage is supplied on the 7-pin header pins labeled GND and +5v.

GPIO[1] ATX Host Power Sense

Since the CFA635-xxx-KS 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, 5k Ω 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 \(Pg. 66\)](#).

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 \(Pg. 66\)](#). This connection is also used for the hardware watchdog.

ATX Power Supply and Control Connections	With Optional SCAB*	Without Optional SCAB Pins on Connector H1
V_{SB} , +5v	SCAB's 7-pin header, +5v	Pin 16
V_{SB} , Ground	SCAB's 7-pin header, GND	Pin 15
GPIO[1] ATX Host Power Sense	SCAB's 4-pin power header, +5v	Pin 12
GPIO[2] ATX Host Power Control	SCAB's 7-pin power header, GPIO[2]	Pin 9
GPIO[3] ATX Host Reset Control	SCAB's's 7-pin power header, GPIO[3]	Pin 10
*SCAB's JP8 must be open and JP9 must be closed. For details, see the SCAB Data Sheet on www.crystalfontz.com/product/SCAB.html#docs .		



ATX Connection With Optional SCAB Using WR-PWR-Y14 ATX Cable

The Crystalfontz [WR-PWR-Y14](#) cable allows ATX power control connections through the optional [SCAB](#). This allows additional flexibility in cabling and overall functionality of the CFA635-xxx-KS in system control and monitoring. Buy the [WR-EXT-Y15](#) or [WR-EXT-Y19](#) to connect the SCAB to the CFA635-xxx-KS's connector H1.

Note

If the Crystalfontz [WR-PWR-Y14](#) cable and SCAB are ordered at the same time as the module, Crystalfontz will open JP8 and close JP9 on the SCAB, and send the following software configuration commands unless we are otherwise instructed.

Please note that once these changes are made, for the module to power up, power must be applied to the 7-pin header on the SCAB as well as the 4-pin power header.

```
command = 28 // Set ATX Switch Functionality
length = 1
data[0] = 240 // Enable:
                // KEYPAD_POWER_OFF
                // KEYPAD_POWER_ON
                // KEYPAD_RESET
                // LCD_OFF_IF_HOST_IS_OFF
command = 4 // Store current state as boot state
length = 0
```



The illustration below shows:

- ❑ CFA-RS232 serial converter mounted on CFA635-xxx-KL's H1 connector.
- ❑ How optional SCAB connects to the CFA-RS232 using a Crystalfontz cable (choose WR-EXT-19 or WR-EXT-Y15).
- ❑ How optional SCAB connects to your system's motherboard using a Crystalfontz WR-PWR-Y14 ATX cable.

SCAB Note: For ATX functionality,

- JP8 must be open.
- JP9 must be closed.

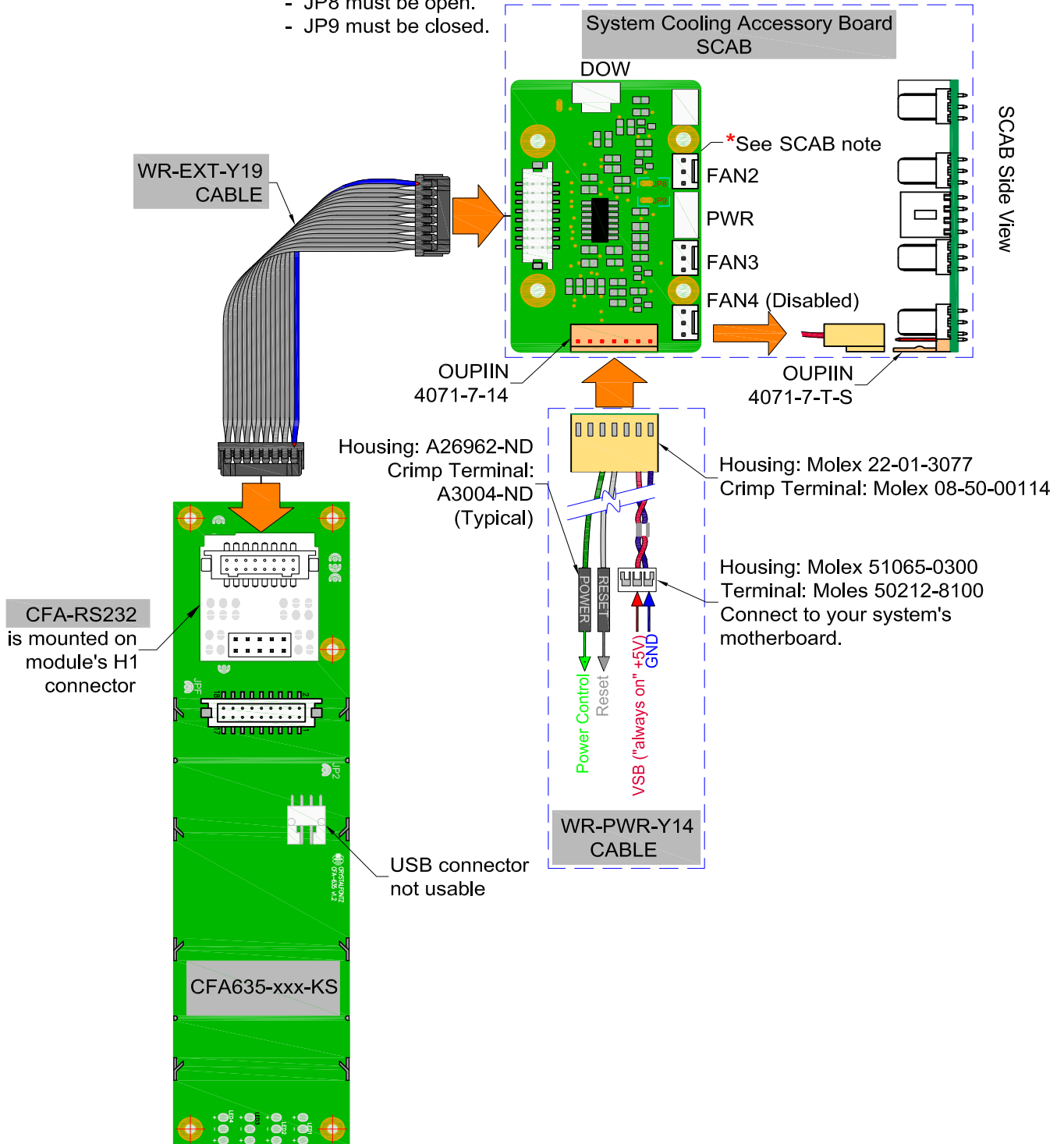


Figure 25. ATX Connection With Optional SCAB Using WR-PWR-Y14 ATX Cable



ATX Connection Without Optional SCAB Using WR-PWR-Y25 ATX Cable

The optional Crystalfontz [WR-PWR-Y25](#) cable simplifies ATX power control connections, allowing all ATX power supply control functionality through the CFA635-xxx-KS 's H1 connector.

Note

If you order the [WR-PWR-Y25](#) or [WR-PWR-Y38](#) ATX power cable at the same time as the module, please specify in the special instructions if you want Crystalfontz to run Command 28 for ATX functionality.

Crystalfontz will send the following software configuration commands. Please note that once these changes are made, for the module to power up, power must be applied to connector H1 with +5v applied to pin 16 and ground to pin 15.

```

command = 28 // Set ATX Switch Functionality
length = 1
data[0] = 240 // Enable:
                // KEYPAD_POWER_OFF
                // KEYPAD_POWER_ON
                // KEYPAD_RESET
                // LCD_OFF_IF_HOST_IS_OFF
command = 4 // Store current state as boot state
length = 0

```

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

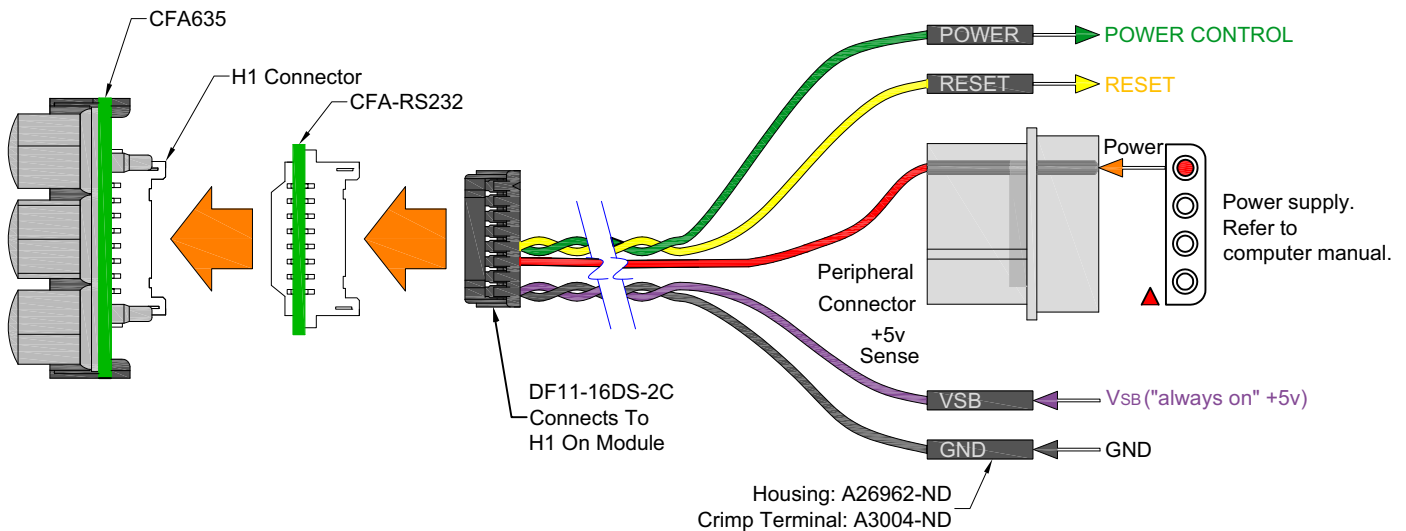


Figure 26. ATX Power Supply And Control Connections Using WR-PWR-Y25 ATX Cable



HOW TO SET ATX FUNCTIONALITY USING CFTEST

1. Download the [cfTest](http://www.crystalfontz.com/software/CFTEST.html) application here: <http://www.crystalfontz.com/software/CFTEST.html>.
2. Connect the module to a Windows based PC. You may want to connect the +5VSB and +5VSENSE so you will be able to see the module when it powers up.
3. Disable any applications that communicate with the module to free up the virtual COM port.
4. Launch [cfTest](#). The application should automatically recognize the module and display it in the Communications Port dropdown list. If not, select your module from the dropdown list.
5. To set the Backlight, click the LCD Display button. Slide the Brightness slider to the desired level.
6. In the Send Packet section, select Command 28 from the dropdown list.
7. Type in the following value: "\240" into the Data field. The "\240" represents the bitmask value for data[0].
8. Click Send Packet.
9. Select Command 4 from the The PacketType dropdown list.
10. Clear the data text box.
11. Click Send Packet. This saves the current state set with ATX.

HOW TO CONNECT THE OPTIONAL SCAB

The optional [SCAB](#) is designed to connect to the CFA635-xxx-KS's J2 connector. The SCAB will receive the correct signals to operate from the module.

Here is a photo showing the CFA635-xxx-KS connected to the optional SCAB using the [WR-EXT-Y19](#) cable:

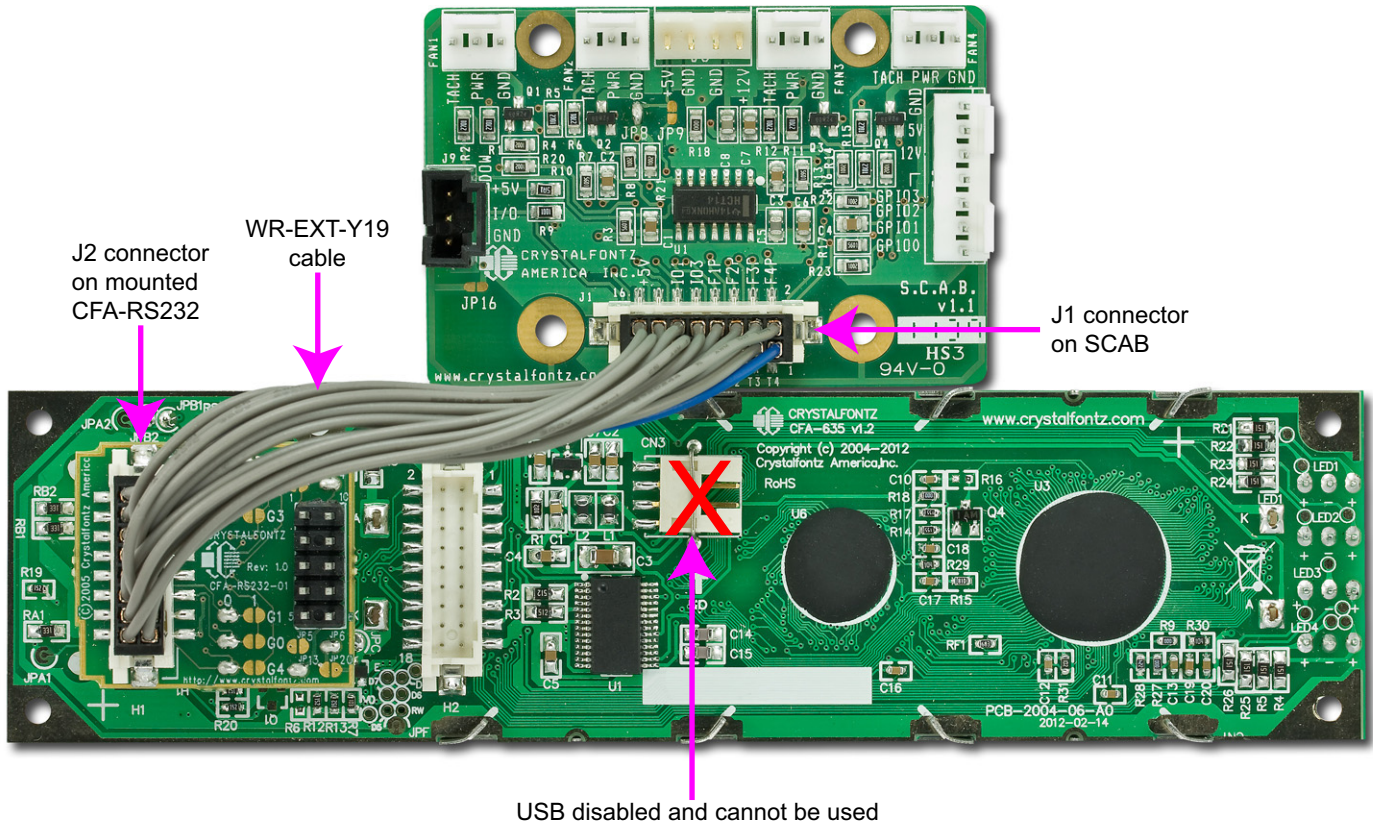


Figure 27. CFA635-xxx-KS Connected To Optional SCAB With WR-EXT-Y19 Cable



Two Crystalfontz cables are available to make the connection between the optional SCAB and the CFA635-xxx-KS:

1. [WR-EXT-Y15](#) SCAB Cable (~16-inch)

This cable allows the SCAB to be mounted some distance away from the CFA635-xxx-KS. For instance, the SCAB could be mounted in a central location within a PC's case. The WR-EXT-Y15 could connect the SCAB from this central location to the CFA635-xxx-KS mounted in a drive bay. The connections to the three fans and temperature sensors only need to be run to the SCAB, not all the way to the front panel where the CFA635 is mounted.

2. [WR-EXT-Y19](#) SCAB Cable (~3.5-inch)

This cable can be used when the SCAB is mounted in close proximity to the CFA635-xxx-KS, as is the case when the SCAB is fastened directly to the LCD module's mounting bracket. (See the photo above.)

Notes

1. Fan 4 (F4P and F4T) will not be available through the SCAB when it is used with the CFA635-xxx-KS.
2. Because the serial CFA635-xxx-KS has firmware distinct from the USB CFA635-xxx-KU, it will not work if you attempt to use it as a USB LCD module.

HOST COMMUNICATIONS

Notes on Terms Used in Command Descriptions Below

1. Where there is no difference in commands for the USB (CFA635-xxx-KU) and the two serial interfaces (CFA635-xxx-KL and CFA635-xxx-KS), the command descriptions in this Data Sheet use the shorter term "CFA635".

The one difference is that up to *four* fans can be used with the optional [SCAB](#) when connected to a CFA635-xxx-KU while up to *three* fans can be used with the SCAB when connected to a CFA635-xxx-KL and CFA635-xxx-KS.

2. Where "CFA635 with ATX" is described, you can use:
CFA635+[WR-PWR-Y25](#) ATX power cable
or
CFA635+[WR-PWR-Y38](#) ATX power cable
or
CFA635+optional [SCAB](#)+[WR-PWR-Y14](#) power ATX cable

The CFA635-xxx-KS communicates with its host using the RS232 interface. The host's RS232 communications port should be opened at 115200 baud, 8 data bits, no parity, 1 stop bit.

PACKET STRUCTURE

All communication between the CFA635 and the host takes place in the form of a simple and robust CRC checked packet. The packet format allows for very reliable communications between the CFA635 and the host without the



ABOUT HANDSHAKING

The nature of CFA635's packets makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for a corresponding acknowledge packet from the CFA635 before sending the next command packet. The CFA635 will respond to all packets within 250 mS. The host software should stop waiting and retry the packet if the CFA635 fails to respond within 250 mS. The host software should report an error if a packet is not acknowledged after several retries. This situation indicates a hardware problem — for example, a disconnected cable.

Please note that some operating systems may introduce delays between when the data arrives at the physical port from the CFA635 until it is available to the user program. In this case, the host program may have to increase its timeout window to account for the additional overhead of the operating system.

The CFA635 can be configured to send several types of report packets along with regular acknowledge packets. The host should be able to buffer several incoming packets and must guarantee that it can process and remove packets from its input buffer faster than the packets can arrive given the 115200 equivalent baud rate of the VCP and the reporting configuration of the CFA635. For any modern PC or microcontroller using reasonably efficient software, this requirement will not 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.

COMMAND CODES

Below is a list of valid commands for the CFA635. Each command packet is answered by either a response packet or an error packet. The low 6 bits of the `type` field of the response or error packet is the same as the low 6 bits of the `type` field of the command packet being acknowledged.

0 (0x00): Ping Command

The CFA635 will return the Ping Command to the host.

```
type = 0x00 = 010  
valid data_length is 0 to 16  
data[0-(data_length-1)] can be filled with any arbitrary data
```

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

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

1 (0x01): Get Hardware And Firmware Version

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

```
type = 0x01 = 110  
valid data_length is 0
```



The return packet will be:

```
type = 0x40 | 0x01 = 0x41 = 6510  
data_length = 16  
data[] = "CFA635:hXvX,sYvY"
```

hXvX is the hardware revision.
sYvY is the firmware version.

2 (0x02): Write User Flash Area

The CFA635 reserves 16 bytes of nonvolatile memory for arbitrary use by the host. This memory can be used to store a serial number, IP address, gateway address, netmask, or any other data required. All 16 bytes must be supplied.

```
type = 0x02 = 210  
valid data_length is 16  
data[] = 16 bytes of arbitrary user data to be stored in the CFA635's nonvolatile 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 CFA635's nonvolatile memory
```

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

The CFA635 loads its power-up configuration from nonvolatile memory when power is applied. The CFA635 is configured at the factory to display a "welcome" screen when power is applied. This command can be used to customize the "welcome" screen, as well as the following items:

- Characters shown on LCD, which are affected by:
 - Command [6 \(0x06\): Clear LCD Screen \(Pg. 57\)](#).
 - Command [31 \(0x1F\): Send Data To LCD \(Pg. 69\)](#).
- Special character font definitions (command [9 \(0x09\): Set LCD Special Character Data \(Pg. 57\)](#)).
- Cursor position (command [11 \(0x0B\): Set LCD Cursor Position \(Pg. 58\)](#)).
- Cursor style (command [12 \(0x0C\): Set LCD Cursor Style \(Pg. 58\)](#)).
- Contrast setting (command [13 \(0x0D\): Set LCD Contrast \(Pg. 59\)](#)).
- Backlight setting (command [14 \(0x0E\): Set LCD And Keypad Backlights \(Pg. 59\)](#)).
- Fan power settings (command [17 \(0x11\): Set Fan Power \(SCAB Required\) \(Pg. 60\)](#)).
- Key press and release masks (command [23 \(0x17\): Configure Key Reporting \(Pg. 63\)](#)).
- Fan glitch delay settings (command [26 \(0x1A\): Set Fan Tachometer Glitch Delay \(SCAB Required\) \(Pg. 65\)](#)).
- ATX function enable and pulse length settings (command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 66\)](#)).
- Baud rate (command [33 \(0x21\): Set Baud Rate \(Pg. 69\)](#)).
- GPIO settings and GPO settings for front panel status LEDs (command [34 \(0x22\): GPIO And GPO Settings \(SCAB Required\) \(Pg. 69\)](#)).



You cannot store the fan or temperature reporting, 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
```

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

5 (0x05): Reboot CFA635, Reset Host, Or Power Off Host (ATX Required)

For ATX, a [WR-PWR-Y25](#), [WR-PWR-Y38](#) ATX power cable or the optional [SCAB+WR-PWR-Y14](#) ATX power cable is required.

This command instructs the CFA635 with ATX 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.

**Advanced Configuration and Power Interface) is an industry specification for the efficient handling of power consumption in desktop and mobile computers.*

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 \(0x22\): GPIO And GPO Settings \(SCAB Required\) \(Pg. 69\)](#).

Rebooting the CFA635 may be useful when testing the boot configuration. It may also be useful to re-enumerate the optional [WR-DOW-Y17](#) temperature sensors on the 1-Wire bus (optional SCAB required).

To *reboot the CFA635*, send the following packet:

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



Note on Bootup Delay if using Fans (Optional SCAB Required)

The reboot command may take up to 3 seconds to return its acknowledge packet.

At bootup, there is up to a 500ms (1/2 second) delay between turning on fans. By default, all fans are set to "on" at 100%. If you are not using a fan, set power to 0% (command [17 \(0x11\): Set Fan Power \(SCAB Required\) \(Pg. 60\)](#)) and save this setting as the default boot state (command [4 \(0x04\): Store Current State As Boot State \(Pg. 54\)](#)). This will reduce the boot time.

# of Fans Powered On	Expected Boot Time
0 to 1	300ms - 500ms
2	800ms - 1,000ms
3	1.3s - 1.5s

To reset the host using CFA635 with ATX, assuming the host's reset line is connected to GPIO[3] as described in command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 66\)](#), send the following packet:

```

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

```

Note

The CFA635 will return the acknowledge packet immediately, then reset the host. After resetting the host (~1.5 seconds), the module will reboot itself. The module will not respond to new command packets for up to 3 seconds (~4.5 seconds overall) after its reboot. 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 \(SCAB Required\) \(Pg. 60\)](#)) and saving this as the default boot state (command [4 \(0x04\): Store Current State As Boot State \(Pg. 54\)](#)). Normally, the host will be recovering from its own reset, so the boot delay of the module will not be of consequence.

To turn the host's power off using CFA635 with ATX, assuming the host's power control line is connected to GPIO[2] as described in command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 66\)](#), send the following packet:

```

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

```



Note

The CFA635 will return the acknowledge packet immediately, then reset the host. After resetting the host (~1.5 seconds), the module will reboot itself. The module will not respond to new command packets for up to 3 seconds (~4.5 seconds overall) after its reboot. 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 \(SCAB Required\) \(Pg. 60\)](#) and saving this as the default boot state (command [4 \(0x04\): Store Current State As Boot State \(Pg. 54\)](#)). Normally, the host will be recovering from its own reset, so the boot delay of the module will not be of consequence.

In any of the above cases, 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 \(Pg. 54\)](#).

7 (0x07): Deprecated (See command [31 \(0x1F\): Send Data To LCD \(Pg. 69\)](#))

8 (0x08): Deprecated (See command [31 \(0x1F\): Send Data To LCD \(Pg. 69\)](#))

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, 0-7 are valid
data[1-8] = bitmap of the new font for this character
```

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

If you set bit 7 of any of the data bytes, the entire line of pixels within this character will blink.

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

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 \(Pg. 54\)](#).

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
line 2
0xE0 (224) to 0xF3 (243) for DDRAM, line 3
0xC0 (192) to 0xD3 (211) for DDRAM,
```

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 CFA635's LCD screen. If you want the cursor to be visible, you may also need to send a command [12 \(0x0C\): Set LCD Cursor Style \(Pg. 58\)](#).

```
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 \(Pg. 54\)](#).

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 = blinking underscore (Behavior is different from previous CFA635 versions
firmware s/u1.6 and earlier.)
```

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 \(Pg. 54\)](#).



13 (0x0D): Set LCD Contrast

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

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

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 \(Pg. 54\)](#).

14 (0x0E): Set LCD And Keypad Backlights

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

```
type = 0x0E = 1410  
valid data_length is 1  
data[0] = backlight power setting (0-100 valid)  
0 = off  
1-99 = variable brightness  
100 = on
```

The return packet will be:

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

Set LCD & Keypad Backlight is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 54\)](#).

15 (0x0F): Deprecated

16 (0x10): Set Up Fan Reporting (SCAB Required)

Note: Fan 4 is disabled and unused in the CFA635-xxx-KS+[SCAB](#).

This command will configure the CFA635-xxx-KS+[SCAB](#) to report the fan speed information to the host every 500 mS.

```
type = 0x10 = 1610  
valid data_length is 1  
data[0] = bitmask indicating which fans are enabled to report (0-15 valid)  
---- 8421 Enable Reporting of this Fan's Tach Input  
|| | | | | | | | | | | | | | | | | | |  
| | | | | | | | | | | | | | | | | | |  
|-- Fan 1: 1 = enable, 0 = disable  
| | | | | | | | | | | | | | | | | | |  
| | | | | | | | | | | | | | | | | | |  
|--- Fan 2: 1 = enable, 0 = disable  
| | | | | | | | | | | | | | | | | | |  
| | | | | | | | | | | | | | | | | | |  
|---- Fan 3: 1 = enable, 0 = disable  
| | | | | | | | | | | | | | | | | | |  
| | | | | | | | | | | | | | | | | | |  
|----- Fan 4: 1 = enable, 0 = disable (not functional for this module)
```

The return packet will be:

```
type = 0x40 | 0x10 = 0x50 = 8010  
data_length = 0
```

If data [0] is not 0, then the CFA635-xxx-KS+SCAB will start sending 0x81: Fan Speed Report packets for each enabled fan every 500 mS. (See [0x81: Fan Speed Report \(SCAB Required\) \(Pg. 74\)](#).) Each of the report packets is staggered by 1/8 of a second.



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 [SCAB](#) Data Sheet for a detailed description.

17 (0x11): Set Fan Power (SCAB Required)

Note: Fan 4 is disabled and unused in the CFA635-xxx-KS+[SCAB](#).

This command will configure the power for the fan connectors.

```
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) (not functional for this module)
```

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 \(Pg. 54\)](#).

18 (0x12): Read WR-DOW-Y17 Temperature Sensors (SCAB Required)

When power is applied to the CFA635+[SCAB](#)+[WR-DOW-Y17](#) temperature sensors, it detects any devices (WR-DOW-Y17) connected to the Dallas Semiconductor 1-Wire (DOW) bus and stores the device's information. This command will allow the host to read the device's information.

Note

The GPIO pin used for DOW must not be configured as user GPIO. It must be configured to its default drive mode in order for the DOW functions to work correctly.

These settings are factory default but may be changed by the user. Please see command [34 \(0x22\): GPIO And GPO Settings \(SCAB Required\) \(Pg. 69\)](#).

In order for the DOW subsystem to be enabled and operate correctly, user GPIO[4] must be configured as:

```
DDD = "111: 1=Hi-Z, 0=Slow, Strong Drive Down".
F = "0: Port unused for user GPIO."
```

This state is the factory default, but it can be changed and saved by the user. To ensure that GPIO[4] is set correctly and the DOW operation is enabled, send the following command:

```
command = 34
length = 3
data[0] = 4
data[1] = 100
data[2] = 7
```

This setting must be saved as the boot state, so when the CFA635+SCAB reboots, it will detect the WR-DOW-Y17 temperature sensors.

```
type = 0x12 = 1810
valid data_length is 1
data[0] = device index (0-31 valid)
```




Sensor enabled must have been detected as 0x28 (WR-DOW-Y17 temperature sensor) during DOW enumeration. This can be verified by using the command [18 \(0x12\): Read WR-DOW-Y17 Temperature Sensors \(SCAB Required\) \(Pg. 60\)](#).

The return packet will be:

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

20 (0x14): Arbitrary DOW Transaction (SCAB Required)

The CFA635-xxx-KS+[SCAB](#) can function as an RS232 to Dallas 1-Wire bridge. The CFA635-xxx-KS+SCAB 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 CFA635-xxx-KS+SCAB. This command allows you to specify arbitrary transactions on the 1-Wire bus. The 1-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
```

Please see [APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER \(SCAB REQUIRED\) \(Pg. 81\)](#) for an example of using this command.

```
type = 0x14 = 2010
valid data_length is 2 to 16
data[0] = device_index (0-32 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 32, then no address phase will be executed. If `device_index` is in the range of 0 to 31, and a 1-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 2, 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 2, then `data_length-2` bytes of `data_to_be_written` will be written to the 1-Wire bus immediately after the address phase.

If `number_of_bytes_to_read` is 0, then no read phase will be executed. If `number_of_bytes_to_read` is not 0 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 (0-31 valid)
data[data_length-2] = Data read from the 1-Wire bus. This is the same
                    as number_of_bytes_to_read from the command.
data[data_length-1] = 1-Wire CRC
```



21 (0x15): (Not Accessible)

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

This command allows you to access the CFA635's LCD controller directly. Note: It is possible to corrupt the CFA635 display using this command.

Note

Any command sent specifically to the controller Samsung S6A0073 will need to be reviewed / modified for the commands / registers of the Rockworks RW1067. Please contact the CrystalFontz Engineering Support Team at support@crystalfontz.com for the RW1067 datasheet.

```
type = 0x16 = 2210
data_length = 2
data[0]: location code
    0 = "Data" register
    1 = "Control" register, RE=0
    2 = "Control" register, RE=1
data[1]: data to write to the selected register
```

The return packet will be:

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

23 (0x17): Configure Key Reporting

By default, the CFA635 reports any key event to the host. This command allows the key events to be enabled or disabled on an individual basis.

```
#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
data[1]: release mask
```

Valid values of the mask are \000-\063.

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 \(Pg. 54\)](#).

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the CFA635 for key activity. This command allows the host to detect which keys are currently pressed, which keys have been pressed since the last poll, and which keys have been released since the last poll.



This command is independent of the key reporting masks set by command [23 \(0x17\): Configure Key Reporting \(Pg. 63\)](#). 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] = bitmask showing the keys currently pressed
data[1] = bitmask showing the keys that have been pressed since the last poll
data[2] = bitmask showing the keys that have been released since the last poll
```

25 (0x19): Set Fan Power Fail-Safe (SCAB Required)

Note: Fan 4 is disabled and unused in the CFA635-xxx-KS+[SCAB](#).

The combination of the CFA635-xxx-KS+[SCAB](#)+[WR-FAN-X01](#) cable can be used as part of an active cooling system. The fans can be slowed down to reduce noise when a system is idle or when the ambient temperature is low. The fans speed up when the system is under heavy load or the ambient temperature is high.

Since there are a 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 CFA635-xxx-KS'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 (not functional for this module)
```

```
type = 0x19 = 2510
data_length = 2
data[0] = bitmask 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 Delay (SCAB Required)

Note: Fan 4 is disabled and unused in the CFA635-xxx-KS+[SCAB](#).

The combination of the CFA635-xxx-KS+[SCAB](#)+[WR-FAN-X01](#) cable controls fan speed by using PWM. Using PWM turns the power to a fan on and off quickly to change the average power delivered to the fan. The CFA635 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 CFA635 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 \(Pg. 54\)](#).

```
type = 0x1A = 2610
data_length = 4

data[0] = delay count of fan 1
data[1] = delay count of fan 2
data[2] = delay count of fan 3
data[3] = delay count of fan 4 (not functional for this module)
```

The return packet will be:

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

27 (0x1B): Query Fan Power And Fail-Safe Mask (SCAB Required)

Note: Fan 4 is disabled and unused in the CFA635-xxx-KS+[SCAB](#).

The combination of the CFA635-xxx-KS+[SCAB](#)+[WR-FAN-X01](#) cable is required to use this command. 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 (not functional for this module)
```

```
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 (not functional for this module)
data[4] = bitmask of fans with fail-safe set
```



28 (0x1C): Set ATX Power Switch Functionality

For ATX, [WR-PWR-Y25](#), [WR-PWR-Y38](#) ATX power cable or the optional [SCAB+WR-PWR-Y14](#) ATX power cable is required.

The combination of the CFA635 with ATX 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 \(Pg. 54\)](#).

Note

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 \(0x22\): GPIO And GPO Settings \(SCAB Required\) \(Pg. 69\)](#). These settings must be saved as the boot state.

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

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

This configuration can be assured by sending the following command:

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

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

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

This configuration can be assured by sending the following command:

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

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

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

This configuration can be assured by sending the following command:

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

These settings must be saved as the boot state.

The RESET (GPIO[3]) and POWER CONTROL (GPIO[2]) lines on the CFA635 with ATX are normally high-impedance. Electrically, they appear to be disconnected or floating. When the CFA635 with ATX asserts the RESET or POWER



CONTROL lines, they are momentarily driven high or low (as determined by the RESET_INVERT and POWER_INVERT bits, detailed below). To end the power or reset pulse, the CFA635 with ATX changes the lines back to high-impedance.

FOUR FUNCTIONS ENABLED BY COMMAND 28

Function 1: KEYPAD_RESET

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

Function 2: KEYPAD_POWER_ON

If POWER-ON SENSE (GPIO[1]) is low, pressing the green check key for 0.25 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified by in data[1] or the default of 1 second735 omits "or the default of 1 second". During this time the CFA635 will show POWER ON, then the CFA635 will reset itself.

Function 3: KEYPAD_POWER_OFF

If POWER-ON SENSE (GPIO[1]) is high, holding the red X key for 4 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified in data[1] or the default of 1 second735 omits "or the default of 1 second". If the user continues to hold the power key down, then the CFA635 will continue to drive the line for a maximum of 5 additional seconds. During this time the CFA635 will show POWER OFF.

Function 4: LCD_OFF_IF_HOST_IS_OFF

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

```
#define AUTO_POLARITY           0x01 //Automatically detects polarity for reset and
                                //power (recommended)
#define RESET_INVERT           0x02 //Reset pin drives high instead of low (ignored if
                                AUTO_POLARITY is set)
#define POWER_INVERT           0x04 //Power pin drives high instead of low (ignored if
                                AUTO_POLARITY is set)
#define LCD_OFF_IF_HOST_IS_OFF 0x10
#define KEYPAD_RESET           0x20
#define KEYPAD_POWER_ON        0x40
#define KEYPAD_POWER_OFF       0x80

type = 0x1C = 2810
data_length = 1 or 2
data[0]: bitmask 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 = 7.9 seconds
        255 = Assert power control line until host power state changes
```

The return packet will be:

```
type = 0x40 | 0x1C = 0x5C = 9210
data_length = 0
```



29 (0x1D): Enable/Disable And Reset The Watchdog

Some systems use hardware watchdog timers to ensure that a software or hardware failure does not result in an extended system outage. Once the host system has booted, a system monitor program is started. The system monitor program would enable the watchdog timer on the CFA635 with ATX. If the system monitor program fails to reset the watchdog timer, the CFA635 with ATX will reset the host system.

Note

The GPIO pins used for ATX control must not be configured as user GPIO. They must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are factory default, but may be changed by the user. Please see the note under command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 66\)](#) or command [34 \(0x22\): GPIO And GPO Settings \(SCAB Required\) \(Pg. 69\)](#).

```
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 CFA635 with ATX will reset the host (see command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 66\)](#) for details). Since the watchdog is off by default when the it powers up, the CFA635 with ATX 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 And Status

Note: Fan 4 is disabled and unused in the CFA635-xxx-KS+[SCAB](#).

This command can be used to verify the current items configured to report to the host, as well as some other miscellaneous status information. The combination of CFA635-xxx-KS+[SCAB](#)+[WR-DOW-Y17](#) temperature sensors is required to report the temperature information. The combination of the CFA635-xxx-KS+[SCAB](#)+[WR-FAN-X01](#) cable is required to control fans.

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




The return packet will be:

```
type = 0x40 | 0x1E = 0x5E = 9410
data_length = 15
data[0] = Fan 1-3 reporting status (as set by command 16)
data[1] = Temperatures 1-8 reporting status (as set by command 19)
data[2] = Temperatures 9-15 reporting status (as set by command 19)
data[3] = Temperatures 16-23 reporting status (as set by command 19)
data[4] = Temperatures 24-32 reporting status (as set by command 19)
data[5] = Key presses (as set by command 23)
data[6] = Key releases (as set by command 23)
data[7] = ATX Power Switch Functionality (as set by command 28)
data[8] = Current watchdog counter (as set by command 29)
data[9] = 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) (not functional for this module)
data[13] = Contrast setting (as set by command 13)
data[14] = Backlights setting (as set by command 14)
```

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

31 (0x1F): Send Data To LCD

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

```
type = 0x1F = 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 \(Pg. 54\)](#).

32 (0x20): Not Accessible (Reserved for CFA631 Key Legends)

33 (0x21): Set Baud Rate

This command will change the CFA635's baud rate. The CFA635 will send the acknowledge packet for this command and change its baud rate to the new value. The host should send the baud rate command, wait for a positive acknowledge from the CFA635 at the old baud rate, and then switch itself to the new baud rate. The baud rate must be saved by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 54\)](#) if you want the CFA635 to power up at the new baud rate.

The factory default baud rate is 115200.

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

The return packet will be:

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

34 (0x22): GPIO And GPO Settings (SCAB Required)



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

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

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

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

The GPIOs also have "pull-up" and "pull-down" modes. These modes can be useful when using the GPIO as an input connected to a switch since no external pull-up or pull-down resistor is needed. For instance, the GPIO can be set to pull up. Then when a switch connected between the GPIO and ground is open, reading the GPIO will return a "1". When the switch is closed, the input will return a "0".

Pull-up/pull-down resistance values are approximately 5kΩ/735:40k. Do not exceed current of 25 mA/735:8 mA per GPIO 735: adds Tech Support phone # because more complicated.

Note

The GPIO pins may be used for ATX control through the [SCAB](#)'s 7-pin connector and [WR-DOW-Y17](#) temperature sensing through the SCAB's DOW header. By factory default, the GPIO output setting, function, and drive mode are set correctly to enable operation of the ATX and DOW functions. **The GPIO output setting, function, and drive mode must be set to the correct values in order for the ATX and DOW functions to work. Improper use of this command can disable the ATX and DOW functions.** Our free [cfTest](#) demonstration program may be used to easily check and reset the GPIO configuration to the default state so the ATX and DOW functions will work.

The GPIO configuration is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 54\)](#).

```

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 on optional SCAB's connector when using
CFA635+SCAB+WR-PWR-Y14. GPOs are for the 4 status LEDs on the module's front panel.
  0 = GPIO[0] = H1, Pin 11
  1 = GPIO[1] = H1, Pin 12 (default is ATX Host Power Sense)
  2 = GPIO[2] = H1, Pin 9 (default is ATX Host Power Control)
  3 = GPIO[3] = H1, Pin 10 (default is ATX Host Reset Control)
  4 = GPIO[4] = H1, Pin 13 (default is DOW I/O--has 1 KΩ hardware pull-up on SCAB)
  5 = GPO[ 5] = H2, Pin 15 = LED 3 (bottom) green die
  6 = GPO[ 6] = H2, Pin 13 = LED 3 (bottom) red die
  7 = GPO[ 7] = H2, Pin 11 = LED 2 green die
  8 = GPO[ 8] = H2, Pin 9 = LED 2 red die
  9 = GPO[ 9] = H2, Pin 7 = LED 1 green die
 10 = GPO[10] = H2, Pin 5 = LED 1 red die
 11 = GPO[11] = H2, Pin 3 = LED 0 (top) green die
 12 = GPO[12] = H2, Pin 1 = LED 0 (top) red die

```



13-255 = not accessible

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

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

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

```

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

----- F = Function (only valid for GPIOs, index of 0-4)
Only meaningful for GPIOs (index 0-4). GPOs (index of 5-12) will ignore.
=====
0: Port unused for GPIO. It will take on the default
   function such as ATX, DOW or unused. The user is
   responsible for setting the drive to the correct
   value in order for the default function to work
   correctly.
1: Port used for GPIO under user control. The user is
   responsible for setting the drive to the correct
   value in order for the desired GPIO mode to work
   correctly.
----- reserved, must be 0

```

The return packet will be:

type = 0x40 | 0x22 = 0x62 = 98₁₀
data_length = 0



35 (0x23): Read GPIO And GPO Pin Levels And Configuration State (SCAB Required)

Please see command [34 \(0x22\): GPIO And GPO Settings \(SCAB Required\) \(Pg. 69\)](#) for details on the GPIO and GPO 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 (default is DOW I/O--always has 1 KΩ hardware pull-up on SCAB.)
5 = GPO[ 5] = H2, Pin 15 = LED 3 (bottom) green die
6 = GPO[ 6] = H2, Pin 13 = LED 3 (bottom) red die
7 = GPO[ 7] = H2, Pin 11 = LED 2 green die
8 = GPO[ 8] = H2, Pin 9 = LED 2 red die
9 = GPO[ 9] = H2, Pin 7 = LED 1 green die
10 = GPO[10] = H2, Pin 5 = LED 1 red die
11 = GPO[11] = H2, Pin 3 = LED 0 (top) green die
12 = GPO[12] = H2, Pin 1 = LED 0 (top) red die
13-255 = not accessible
```

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

The return packet will be:

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



```

returns:
data[0] = index of GPIO to read

data[1] = Pin state & changes since last poll
          Only useful for GPIOs (index 0-4). GPOs (index of 5-12) will return 0.
          ---- -RFS Enable Reporting of this Fan's Tach Input (optional SCAB required)
          |||||  |||  |-- S = state at the last reading
          |||||  |||  |-- F = at least one falling edge has been detected since the last poll
          |||||  |||  |-- R = at least one rising edge has been detected since the last poll
          |||||  |||  |-- reserved
          (This reading is the actual pin state, which may or may not agree with the pin setting, depending on drive mode and the load presented by external circuitry. Transients that happen between polls will not be detected.)

data[2] = Requested Pin level/PWM level
          0-100: Output duty cycle percentage
          (This value is the requested PWM duty cycle. The actual pin may or may not be toggling in agreement with this value, depending on the drive mode and the load presented by external circuitry.)

data[3] = Pin function select and drive mode
          ---- FDDD
          |||||  |||  |-- DDD = Drive Mode
          =====
          000: 1=Fast, Strong Drive Up, 0=Resistive Pull Down
          001: 1=Fast, Strong Drive Up, 0=Fast, Strong Drive Down
          010: Hi-Z, use for input
          011: 1=Resistive Pull Up,      0=Fast, Strong Drive Down
          100: 1=Slow, Strong Drive Up, 0=Hi-Z
          101: 1=Slow, Strong Drive Up, 0=Slow, Strong Drive Down
          110: reserved
          111: 1=Hi-Z,                  0=Slow, Strong Drive Down

          ----- F = Function
          Only meaningful for GPIOs (index 0-4). GPOs (index of 5-12) will return 0
          =====
          0: Port unused for GPIO. It will take on the default function such as ATX, DOW or unused. The user is responsible for setting the drive to the correct value in order for the default function to work correctly.
          1: Port used for GPIO under user control. The user is responsible for setting the drive to the correct value in order for the desired GPIO mode to work correctly.
          ----- reserved, will return 0

```

REPORT CODES

The CFA635 can be configured to report three items. The CFA635 sends reports automatically when the data becomes available. Reports are not sent in response to a particular packet received from the host. The three report types are (1) 0x80: Key Activity, (2) 0x81: Fan Speed Report (SCAB Required), and (3) 0x82: Temperature Sensor Report (SCAB Required). The three report types are:

0x80: Key Activity

If a key is pressed or released, the CFA635 sends a Key Activity report packet to the host. Key event reporting may be individually enabled or disabled by command [23 \(0x17\): Configure Key Reporting \(Pg. 63\)](#).



```
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
```

0x81: Fan Speed Report (SCAB Required)

If any of up to three fans connected to CFA635-xxx-KS+[SCAB](#) is configured to report its speed information to the host, the CFA635-xxx-KS will send Fan Speed Reports for each selected fan every 1/2 second. See command [16 \(0x10\): Set Up Fan Reporting \(SCAB Required\) \(Pg. 59\)](#).

```
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 (not functional for this module)
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 (SCAB Required)

If any of the up to 32 temperature sensors is configured to report to the host, the CFA635+[SCAB](#) will send Temperature Sensor Reports for each selected sensor every second. See the command [19 \(0x13\): Set Up WR-DOW-Y17 Temperature Reporting \(SCAB Required\) \(Pg. 61\)](#).

```
type = 0x82
data_length = 4
data[0] is the index of the temperature sensor being reported:
    0 = temperature sensor 1
    1 = temperature sensor 2
    . . .
    31 = temperature sensor 32
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 CFA635
    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);
    }
}
```




CHARACTER GENERATOR ROM (CGROM)

To find the code for a given character, add the two numbers that are shown in bold for its row and column. For example, the superscript "9" is in the column labeled "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.

Character Generator ROM (CGROM) for Crystalfontz CFA-635

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

Figure 28. Character Generator ROM (CGROM)



MODULE RELIABILITY AND LONGEVITY

Note: 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.

MODULE RELIABILITY

ITEM	SPECIFICATION	
LCD portion (excluding Keypad, bicolor status LEDs, and Backlights)	50,000 to 100,000 hours	
Keypad	1,000,000 keystrokes	
Bicolor status LEDs	50,000 to 100,000 hours	
Yellow-green LED Display and Keypad Backlights (CFA635-YYE-Kx)	50,000 to 100,000 hours	
White LED Display and Blue LED Keypad Backlights (CFA635-TMF-Kx) <i>Note: We recommend that the backlights of white LED backlit modules be dimmed or turned off during periods of inactivity to conserve the white LED backlight lifetime.</i>	Power-On Hours	% of Initial Brightness (New Module)
	<10,000	>70%
	<50,000	>50%
White LED Display and White LED Keypad Backlights (CFA635-TFE-Kx) <i>Note: We recommend that the backlights of white LED backlit modules be dimmed or turned off during periods of inactivity to conserve the white LED backlight lifetime.</i>	Power-On Hours	% of Initial Brightness (New Module)
	<10,000 hours	>70%
	<50,000 hours	>50%
<i>Note: For modules with white LED backlights (CFA635-TFE-Kx and CFA635-TMF-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.</i>		
<i>Note: Values listed above are approximate and represent typical lifetime.</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 publish Part Change Notices (PCN) as soon as possible.

CARE AND HANDLING PRECAUTIONS

For optimum operation of the CFA635-xxx-KS and to prolong its life, please follow the precautions described below. Excessive voltage will shorten the life of the module. You must drive the display within the specified voltage limit. See [Absolute Maximum Ratings \(Pg. 25\)](#).

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS

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

DESIGN AND MOUNTING

- The exposed surface of the LCD “glass” is actually a polarizer laminated on top of the glass. To protect the polarizer from damage, the CFA635-xxx-KS 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, polycarbonate or glass) in front of the CFA635-xxx-KS, leaving a small gap between the plate and the display surface. We recommend HP-92 Lexan which is readily available and works well.
- Do not disassemble or modify the module.
- Do not modify the eight tabs of the metal bezel or make connections to them.
- Solder only to the I/O terminals. Use care when removing solder so you do not damage the PCB. Use care when removing solder so you do not damage the PCB. Use care to keep the exposed terminals clean. Contamination, including fingerprints, may make soldering difficult and the reliability of the soldered connection poor.
- 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

All electronics may contain harmful substances. Avoid contamination by using care to avoid damage during handling. If any residues, gases, powders, liquids, or broken fragments come in contact with your skin, eyes, mouth, or lungs, immediately contact your local poison control or emergency medical center.

HOW TO CLEAN

1. Turn display off.
2. 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”).
3. If the polarizer is dusty, you may carefully blow it off with clean, dry, oil-free compressed air.
4. If you must clean with a liquid, never use glass cleaners, as they may contain ammonia or alcohol that will damage the polarizer over time. Never apply liquids directly on the polarizer. Long contact with moisture may permanently spot or stain the polarizer. Use filtered water to slightly moisten a clean lint-free microfiber cloth designed for cleaning optics. (For example, use a cloth sold for cleaning plastic eyeglasses.)
5. The plastic is easily scratched or damaged. Use a light touch as you clean the polarizer. Wipe gently.
6. Use a dry microfiber cloth to remove any trace of moisture before turning on the module.
7. Gently wash the microfiber cloths in warm, soapy water and air dry before reuse.

OPERATION

- Your circuit should be designed to protect the module from ESD and power supply transients.
- Observe the operating temperature limitations: a minimum of 0°C to a maximum of +50°C noncondensing with minimal fluctuation. Operation outside of these limits may shorten life and/or harm display. Changes in temperature can result in changes in contrast.
 - 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 LEDs (CFA635-TFE-Kx and CFA635-TMF-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 with humidity less than 90% noncondensing.
- Observe the storage temperature limitations: a minimum of -10°C minimum to +60°C noncondensing 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.
- To discard, please recycle your modules at an approved facility.

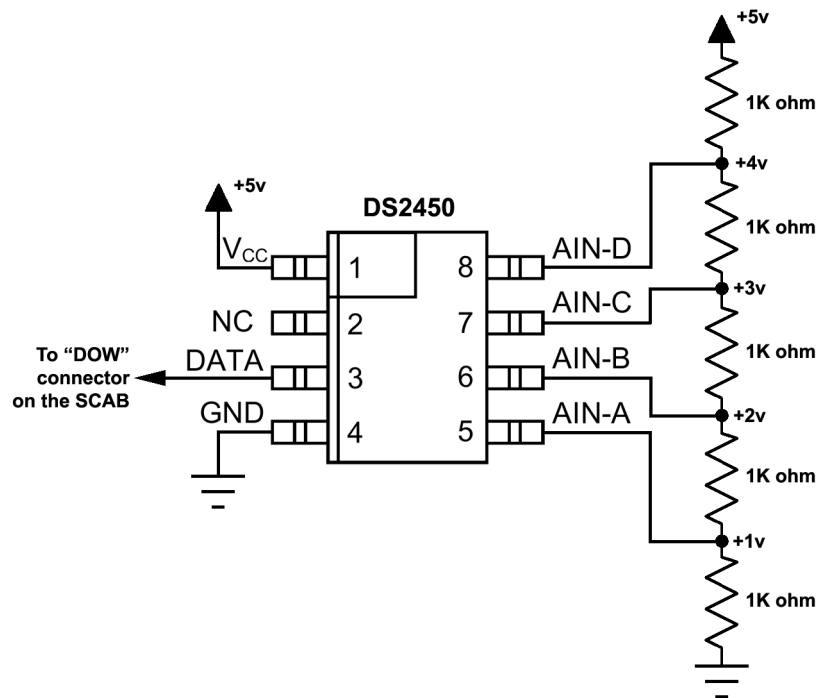


APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER (SCAB REQUIRED)

This appendix describes a simple test circuit that demonstrates how to connect a Dallas Semiconductor DS2450 4-channel ADC to the SCAB's DOW (Dallas One Wire) connector. It also gives a sample command sequence to initialize and read the ADC.

Up to 32 DOW devices can be connected to the CFA635+[SCAB](#). In this example the DS2450 appears at device index 0. Your software should query the connected devices using command [18 \(0x12\): Read WR-DOW-Y17 Temperature Sensors \(SCAB Required\) \(Pg. 60\)](#) to verify the locations and types of DOW devices connected in your application.

Please refer to the [DS2450 Data Sheet](#) and the description for command [20 \(0x14\): Arbitrary DOW Transaction \(SCAB Required\) \(Pg. 62\)](#) more information.



Appendix A Figure 1. Test Circuit Schematic

Start [cfTest](#) and open the Packet Debugger dialog.

Select Command 20 = Arbitrary DOW Transaction, then paste each string below into the data field and send the packet. The response should be similar to what is shown.



```
//Write 0x40 (=64) to address 0x1C (=28) to leave analog circuitry on
//(see page 6 of the data sheet)
<command 20> \000\002\085\028\000\064
<response> C=84(d=0):2E,05,22 //16 bit "i-button" CRC + 8-bit "DOW" CRC
//Consult "i-button" docs to check 16-bit CRC
//DOW CRC is probably useless for this device.

//Write all 8 channels of control/status (16 bits, 5.10v range)
<command 20> \000\002\085\008\000\000 // address = 8, channel A low
<response> C=84(d=0):6F,F1,68 // 16-bits, output off

<command 20> \000\002\085\009\000\001 // address = 9, channel A high
<response> C=84(d=0):FF,F1,AB // no alarms, 5.1v

<command 20> \000\002\085\010\000\000 // address = 10, channel B low
<response> C=84(d=0):CE,31,88 // 16-bits, output off

<command 20> \000\002\085\011\000\001 // address = 11, channel B high
<response> C=84(d=0):5E,31,4B // no alarms, 5.1v

<command 20> \000\002\085\012\000\000 // address = 12, channel C low
<response> C=84(d=0):2E,30,A3 // 16-bits, output off

<command 20> \000\002\085\013\000\001 // address = 13, channel C high
<response> C=84(d=0):BE,30,60 // no alarms, 5.1v

<command 20> \000\002\085\014\000\000 // address = 14, channel D low
<response> C=84(d=0):8F,F0,43 // 16-bits, output off

<command 20> \000\002\085\015\000\001 // address = 15, channel D high
<response> C=84(d=0):1F,F0,80 // no alarms, 5.1v

//Read all 4 channels of control/status (check only)
<command 20> \000\010\170\008\000
<response> C=84(d=0):00,01,00,01,00,01,00,01,E0,CF,01

//Repeat next two commands for each conversion (two cycles shown)

//Start conversion on all channels
<command 20> \000\002\060\015\000
<response> C=84(d=0):3A,03,28

//Read all 8 channels
<command 20> \000\010\170\000\000
<response> C=84(d=0):00,33,DF,64,84,96,6A,C8,5A,6B,BE

//Decoded response:
0x3300 = 130561.016015625 volts (channel A)
0x64DF = 258232.009541321 volts (channel B)
0x9684 = 385322.998553467 volts (channel C)
0xC86A = 513063.992623901 volts (channel D)

//Start conversion on all channels
<command 20> \000\002\060\015\000
<response> C=84(d=0):3A,03,28

//Read all 8 channels
<command 20> \000\010\170\000\000
<response> C=84(d=0):6B,33,B2,64,97,96,42,C8,0F,C9,0A

//Decoded response:
0x336B = 131631.024342346 volts (channel A)
0x64B2 = 257782.006039429 volts (channel B)
0x9697 = 385513.000032043 volts (channel C)
0xC842 = 512663.989511108 volts (channel D)
```



APPENDIX B: CONNECTING A DS1963S SHA IBUTTON (SCAB REQUIRED)

This appendix describes connecting a Dallas Semiconductor DS1963S Monetary iButton with SHA-1 Challenge Response Algorithm and 4KB of nonvolatile RAM to the CFA635+SCAB's DOW (Dallas One Wire) connector. It also gives a sample command sequence to read and write the DS1963S's scratch memory.

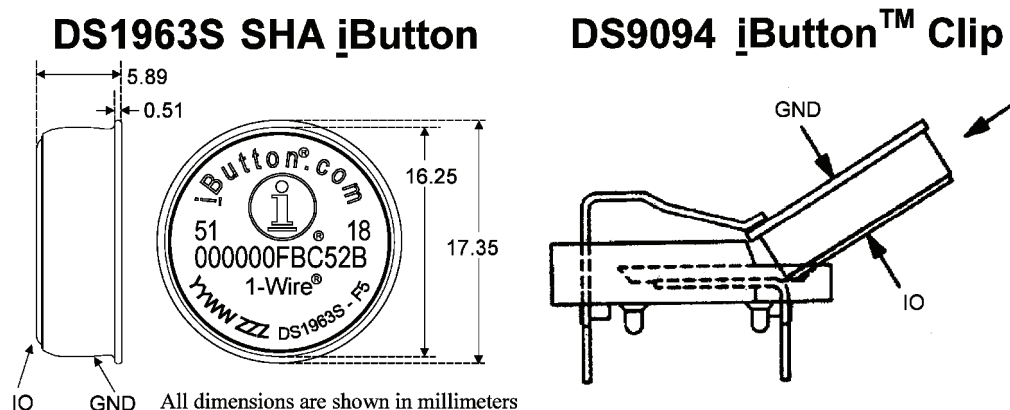
The DS1963S can be used as a secure dongle to protect your system's application software from being copied. Even if the communication channel is compromised or the host is not authentic, the SHA algorithm ensures that the data is still secure. Please see the following Maxim/Dallas white papers and application notes for more information:

- [White Paper 1: SHA Devices Used in Small Cash Systems](#)
- [White Paper 2: Using the 1-Wire Public-Domain Kit](#)
- [White Paper 3: Why are 1-Wire SHA-1 Devices Secure?](#)
- [White Paper 4: Glossary of 1-Wire SHA-1 Terms](#)
- [White Paper 8: 1-Wire SHA-1 Overview](#)
- [App Note 150: Small Message Encryption using SHA Devices](#)
- [App Note 152: SHA iButton Secrets and Challenges](#)
- [App Note 154: Passwords in SHA Authentication](#)
- [App Note 156: DS1963S SHA 1-Wire API Users Guide](#)
- [App Note 157: SHA iButton API Overview](#)
- [App Note 190: Challenge and Response with 1-Wire SHA devices](#)

Up to 32 DOW devices can be connected to the CFA635+SCAB. In this example the DS1963S appears at device index 0. Your software should query the connected devices using command [18 \(0x12\): Read WR-DOW-Y17 Temperature Sensors \(SCAB Required\) \(Pg. 60\)](#) to verify the locations and types of DOW devices connected in your application.

Please refer to the [DS1963S Data Sheet](#) and the description for command [20 \(0x14\): Arbitrary DOW Transaction \(SCAB Required\) \(Pg. 62\)](#) for more information.

To connect the DS1963S to the CFA635+SCAB, simply make one connection between the DS1963S's "GND" terminal and the CFA635+SCAB DOW connector's GND pin, and a second connection between the DS1963S's "IO" pin and the CFA635+SCAB DOW connector's I/O pin. By using a DS9094 iButton Clip, the connection is easy.



Appendix B Figure 1. Connect to Maxim/Dallas DS19632 SHA iButton Using DS9094 iButton Clip



To demonstrate reading and writing the scratch memory on DS1963S, open the [cfTest](#) Packet Debugger dialog and use it to experiment with the following commands: Erase Scratchpad, Read Scratchpad, and Write Scratchpad.

To use the full power of the DS1963S, a program based on the Dallas/Maxim application notes listed above is needed. The challenge/response sequence would be unwieldy to demonstrate using the cfTest Packet Debugger dialog.

First read the address of the DS1963S as detected by the CFA635 at boot. Since only one device is connected, you only need to query index 0. In a production situation, query all 32 indices to get a complete picture of the devices available on the DOW bus.

```
Command:
 18 = Read DOW Device Information
Data sent:
 \000
Data received:
 C=82 (d=0) : 18, CC, D2, 19; 00, 00, 00, 9E
```

The first byte returned is the Family Code of the Dallas One Wire / iButton device. 0x18 indicates that this device is a DS1963. A list of the possible Dallas One Wire / iButton device family codes is available in [App Note 155: 1-Wire Software Resource Guide](#) on the Maxim/Dallas website.

Erase Scratchpad Command (quote from the Maxim/Dallas [DS1963S Data Sheet](#)):

Erase Scratchpad [C3h]

The purpose of this command is to clear the HIDE flag and to wipe out data that might have been left in the scratchpad from a previous operation. After having issued the command code the bus master transmits a target address, as with the write scratchpad command, but no data. Next the whole scratchpad will be automatically filled with FFh bytes, regardless of the target address. This process takes approximately 32 μs during which the master reads 1's. After this the master reads a pattern of alternating 0's and 1's indicating that the command has completed. The master must read at least 8 bits of this alternating pattern. Otherwise the device might not properly respond to a subsequent Reset Pulse.

```
Command:
 20 = Arbitrary DOW transaction
Data sent:
 \000\014\xC3\000\000
Data received:
 C=84 (d=0) : FF, AA, AA, AA, AA, AA, AA, AA, AA, AA, AA, AA, AA, AA, AA, 9F
```

The "AA" bytes read are the pattern of alternating 0's and 1's indicating that the command has completed.

Read Scratchpad Command (quote from the Maxim/Dallas [DS1963S Data Sheet](#))

Read Scratchpad Command [AAh]

HIDE = 0:

The Read Scratchpad command allows verifying the target address, ending offset and the integrity of the scratchpad data. After issuing the command code the master begins reading. The first 2 bytes will be the target address. The next byte will be the ending offset/data status byte (E/S) followed by the scratchpad data beginning at the byte offset (T4: T0). The master may read data until the end of the scratchpad after which it will receive the inverted CRC generated by the DS1963S. If the master continues reading after the CRC all data will be logic 1's.

```
Command:
 20 = Arbitrary DOW transaction
Data sent:
 \000\014\xAA
Data received:
 C=84 (d=0) : 00, 00, 1F, FF, FF, FF, FF, FF, FF, FF, FF, FF, FF, FF, 07
```

Since you did an "Erase Scratchpad" as the previous command, the "Read Scratchpad" returns 0xFF bytes as expected.



Write Scratchpad Command (quote from the Maxim/Dallas [DS1963S Data Sheet](#))

Write Scratchpad Command [0Fh]

HIDE = 0, Target Address range 0000h to 01FFh only

After issuing the write scratchpad command, the master must first provide the 2-byte target address, followed by the data to be written to the scratchpad. The data will be written to the scratchpad starting at the byte offset (T4:T0). The ending offset (E4: E0) will be the byte offset at which the master stops writing data. Only full data bytes are accepted. If the last data byte is incomplete its content will be ignored and the partial byte flag PF will be set.

When executing the Write Scratchpad command the CRC generator inside the DS1963S (see Figure 12) calculates a CRC of the entire data stream, starting at the command code and ending at the last data byte sent by the master. This CRC is generated using the CRC16 polynomial by first clearing the CRC generator and then shifting in the command code (0FH) of the Write Scratchpad command, the Target Addresses TA1 and TA2 as supplied by the master and all the data bytes. The master may end the Write Scratchpad command at any time. However, if the ending offset is 11111b, the master may send 16 read time slots and will receive the CRC generated by the DS1963S.

Write 10 bytes of identifiable test data {0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA} to the scratch pad in location 0:0

Command:

20 = Arbitrary DOW transaction

Data sent:

\000\000\x0F\x00\x00\x11\x22\x33\x44\x55\x66\x77\x88\x99\xAA

Data received:

C=84 (d=0) :00

Use the Read Scratchpad Command [AAh] to read back the data.

Command:

20 = Arbitrary DOW transaction

Data sent:

\000\013\xAA

Data received:

C=84 (d=0) :00,00,09,11,22,33,44,55,66,77,88,99,AA,1E

Now write 10 bytes of identifiable test data {0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89, 0x9A, 0xAB} to the scratch pad in location 0:0x0A

Command:

20 = Arbitrary DOW transaction

Data sent:

\000\000\x0F\x0A\x00\x12\x23\x34\x45\x56\x67\x78\x89\x9A\xAB

Data received:

C=84 (d=0) :00

Use the Read Scratchpad Command [AAh] to read back the data.

Command:

20 = Arbitrary DOW transaction

Data sent:

\000\013\xAA

Data received:

C=84 (d=0) :00,02,09,12,23,34,45,56,67,78,89,9A,AB,62

Reading and writing to the scratch pad is the first step required to communicate with the DS1863S. In order to fully use the DS1963S for a dongle application that securely protects your software from copying, become familiar with the SHA algorithm as it applies to the SHA iButton by studying the Maxim/Dallas white papers and application notes listed above. Then create a software application that implements the secure challenge/response protocol as outlined in the application notes.



APPENDIX C: SAMPLE CODE (DEMONSTRATION SOFTWARE AND SAMPLE CODE)

DRIVERS

- ❑ Windows USB driver and installation instructions are here: www.crystalfontz.com/product/USB_LCD_Driver.html.
- ❑ See <http://lcdproc.omnipotent.net/hardware.php3> for Linux LCD drivers. LCDproc is an open source project that supports many of the CrystalFontz displays.

DEMONSTRATION AND TEST PROGRAMS

The following programs are available for free download on our website for CFA635-xxx-KS and CFA635-xxx-KU:

- ❑ [cfTest](#) demonstrates features of CrystalFontz Intelligent LCD modules.
- ❑ [Linux cli examples](#) is Linux-compatible demonstration software. The C source code is included.
- ❑ [CrystalControl2](#) can display system information from your PC and the programs it is running on your CrystalFontz display. See the [CrystalControl2 manual](#) for a description of this full-featured program.

ALGORITHMS TO CALCULATE THE CRC

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

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

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

Algorithm 1: "C" Table Implementation

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

```
//This code is from the IRDA LAP documentation, which appears to
//have been copied from PPP:
//
// http://irda.affiniscap.com/associations/2494/files/Specifications/IrLAP11_Plus_Er-
//rata.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] =
        {0x0000,0x01189,0x02312,0x0329B,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, 0x07CF7,
0x09489, 0x08500, 0x0B79B, 0x0A612, 0x0D2AD, 0x0C324, 0x0F1BF, 0x0E036,
0x018C1, 0x00948, 0x03BD3, 0x02A5A, 0x05EE5, 0x04F6C, 0x07DF7, 0x06C7E,
0x0A50A, 0x0B483, 0x08618, 0x09791, 0x0E32E, 0x0F2A7, 0x0C03C, 0x0D1B5,
0x02942, 0x038CB, 0x00A50, 0x01BD9, 0x06F66, 0x07EEF, 0x04C74, 0x05DFD,
0x0B58B, 0x0A402, 0x09699, 0x08710, 0x0F3AF, 0x0E226, 0x0D0BD, 0x0C134,
0x039C3, 0x0284A, 0x01AD1, 0x00B58, 0x07FE7, 0x06E6E, 0x05CF5, 0x04D7C,
0x0C60C, 0x0D785, 0x0E51E, 0x0F497, 0x08028, 0x091A1, 0x0A33A, 0x0B2B3,
0x04A44, 0x05BCD, 0x06956, 0x078DF, 0x00C60, 0x01DE9, 0x02F72, 0x03EFB,
0x0D68D, 0x0C704, 0x0F59F, 0x0E416, 0x090A9, 0x08120, 0x0B3BB, 0x0A232,
0x05AC5, 0x04B4C, 0x079D7, 0x0685E, 0x01CE1, 0x00D68, 0x03FF3, 0x02E7A,
0x0E70E, 0x0F687, 0x0C41C, 0x0D595, 0x0A12A, 0x0B0A3, 0x08238, 0x093B1,
0x06B46, 0x07ACF, 0x04854, 0x059DD, 0x02D62, 0x03CEB, 0x00E70, 0x01FF9,
0x0F78F, 0x0E606, 0x0D49D, 0x0C514, 0x0B1AB, 0x0A022, 0x092B9, 0x08330,
0x07BC7, 0x06A4E, 0x058D5, 0x0495C, 0x03DE3, 0x02C6A, 0x01EF1, 0x00F78};

```

```

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

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

```

Algorithm 2: "C" Bit Shift Implementation

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

```

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

```



```
//Get the next byte in the stream.
data=*bufptr++;
//Push this byte's bits through a software
//implementation of a hardware shift & xor.
for(bit_count=0;bit_count<=7;bit_count++)
{
    //Shift the CRC accumulator
    newCRC>>=1;

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

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

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

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



Algorithm 2B: "C" Improved Bit Shift Implementation

This is a simplified algorithm that implements the CRC.

```

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

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

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

```

Algorithm 3: "PIC Assembly" Bit Shift Implementation

This routine was graciously donated by one of our customers.

```

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

```



```

savchr      equ      46h          ; BYTE - temp storage for CRC routine
;
seedlo      equ      021h         ; initial seed for CRC reg lo byte
seedhi      equ      0F3h         ; initial seed for CRC reg hi byte
;
polyL       equ      008h         ; polynomial low byte
polyH       equ      084h         ; polynomial high byte
;=====
; CRC Test Program
;-----
          org      0              ; reset vector = 0000H
;
          clr     PCLATH          ; ensure upper bits of PC are cleared
          clr     STATUS          ; ensure page bits are cleared
          goto    main            ; jump to start of program
;
; ISR Vector
;
          org      4              ; start of ISR
          goto    $              ; jump to ISR when coded
;
          org      20             ; start of main program
main
          movlw   seedhi          ; setup intial CRC seed value.
          movwf   accumh          ; This must be done prior to
          movlw   seedlo          ; sending string to CRC routine.
          movwf   accuml          ;
          clr     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    ;
        btfs    STATUS,C    ; skip jump if if carry
        goto    _notset     ; otherwise goto next bit
        movlw   polyL      ; XOR poly mask with CRC register
        xorwf   accuml,F    ;
        movlw   polyH      ;
        xorwf   accumh,F    ;
_notset
        decfsz  j,F        ; decrement bit counter
        goto    _loop      ; loop if not complete
        movfw   savchr     ; restore the input character
        return  ; return to calling routine
;=====
; USER SUPPLIED Serial port transmit routine
;-----
SENDUART
        return           ; put serial xmit routine here
;=====
; test string storage
;-----
        org     0100h
;
InputStr
        addwf   PCL,f
        dt     7h,10h,"This is a test. ",0
;
;=====
        end

```

Algorithm 4: “Visual Basic” Table Implementation

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

```

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

```

```

Private Type WORD
    Lo As Byte
    Hi As Byte
End Type

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

Dim crcLookupTable(256) As WORD

Private Sub MSComm_OnComm()
'Leave this here
End Sub

```



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

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

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

Algorithm 5: “Java” Table Implementation

This [code was posted in our forum](#) by user “norm” as a working example of a Java CRC calculation.

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



```

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

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

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

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

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

```



```

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

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

```



```

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

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

my $valid_crc = '5584' ;

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

my $crc = 0xFFFF ;

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

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

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

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

```

Algorithm 7: For PIC18F8722 or PIC18F2685

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

```

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

```



```

        bcf     STATUS,C           ; clear carry for CRC register shift
        rrcf   dsplyCRCDATA,f     ; perform shift of data into CRC
                                   ;register

        rrcf   dsplyCRC,F
        rrcf   dsplyCRC+1,F
        btfss  STATUS,C           ; skip jump if carry
        goto   _notset           ; otherwise goto next bit
        movlw  0x84
        xorwf  dsplyCRC,F
        movlw  0x08               ; XOR poly mask with CRC register
        xorwf  dsplyCRC+1,F

_notset:
        decfsz dsplyCRCCOUNT,F    ; decrement bit counter
        bra   _cloop             ; loop if not complete
        return

;=====
; example to clear screen
dsplyFSR1_TEMP equ    0x83A      ; 16-bit save for FSR1 for display
                                   ; message handler
dsplyCRC       equ    0x83C      ; 16-bit CRC (H/L)
dsplyLPCNT     equ    0x83E      ; 8-bit save for display message
                                   ; length - CRC
dsplyCRCDATA   equ    0x83F      ; 8-bit CRC data for display use
dsplyCRCCOUNT equ    0x840      ; 8-bit CRC count for display use
SendCount      equ    0x841      ; 8-bit byte count for sending to
                                   ; display
RXBUF2         equ    0x8C0      ; 32-byte receive buffer for
                                   ; Display
TXBUF2         equ    0x8E0      ; 32-byte transmit buffer for
                                   ; Display

;-----
ClearScreen:
        movlb  RAM8
        movlw  .0
        movwf  SendCount
        movlw  0xF3
        movwf  dsplyCRC           ; seed ho for CRC calculation
        movlw  0x21
        movwf  dsplyCRC+1        ; seen lo for CRC calculation
        call   ClaimFSR1
        movlw  0x06
        movwf  TXBUF2
        LFSR  FSR1,TXBUF2
        movf  SendCount,w
        movwf  TXBUF2+1          ; message data length
        call   BMD1
        goto   SendMsg

;=====
; send message via interrupt routine. The code is made complex due
; to the limited FSR registers and extended memory space used
;
; example of sending a string to column 0, row 0
;-----
SignOnL1:
        call   ClaimFSR1
        lfsr  FSR1,TXBUF2+4      ; set data string position
        SHOW  COR0,BusName       ; move string to TXBUF2
        movlw .2
        addwf SendCount
        movff SendCount,TXBUF2+1 ; insert message data length

        call   BuildMsgDSPLY
        call   SendMsg
        return

;=====
; BuildMsgDSPLY used to send a string to LCD
;-----
BuildMsgDSPLY:

```



```
    movlw    0xF3
    movwf    dsplyCRC        ; seed hi for CRC calculation
    movlw    0x21
    movwf    dsplyCRC+1     ; seed lo for CRC calculation
    LFSR     FSR1, TXBUF2    ; point at transmit buffer
    movlw    0x1F           ; command to send data to LCD
    movwf    TXBUF2        ; insert command byte from us to
                            ; CFA-635

    BMD1     movlw    .2
    ddwf     SendCount,w    ; + overhead
    call     ComputeCRC2    ; compute CRC of transmit message
    movf     dsplyCRC+1,w
    movwf    POSTINC1      ; append CRC byte
    movf     dsplyCRC,w
    movwf    POSTINC1      ; append CRC byte
    return

;=====
SendMsg:
    call     ReleaseFSR1
    LFSR     FSR0, TXBUF2
    movff    FSR0H, irptFSR0
    movff    FSR0L, irptFSR0+1
                            ; save interrupt use of FSR0
    movff    SendCount, TXBUSY2
    bsf     PIE2, TX2IE
                            ; set transmit interrupt enable
                            ; (bit 4)

    return

;=====
; macro to move string to transmit buffer
SHOW macro src, stringname
    call     src
    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

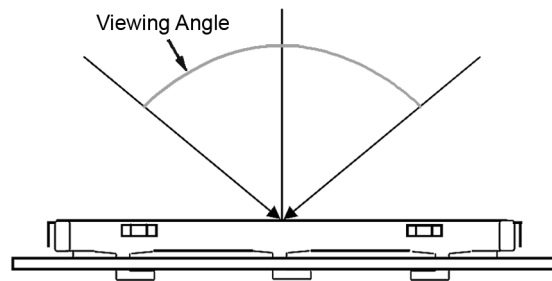
;=====
```




APPENDIX D: 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)
 - Viewing angle: inspect at 45° angle of normal line right and left, top and bottom

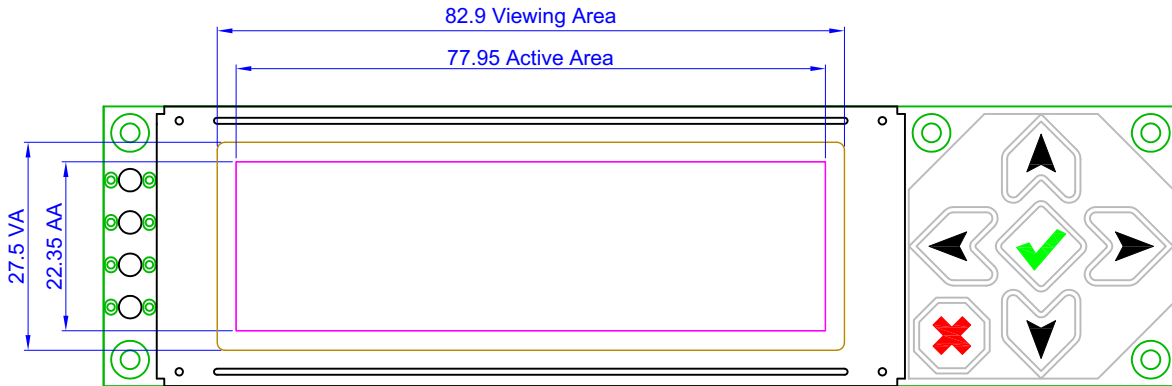


COLOR DEFINITIONS

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



DEFINITION OF ACTIVE AREA AND VIEWING AREA



ACCEPTANCE SAMPLING

DEFECT TYPE	AQL*
Major	≤.65%
Minor	<1.0%

* Acceptable Quality Level: maximum allowable error rate or variation from standard

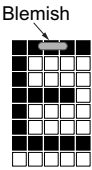
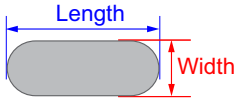
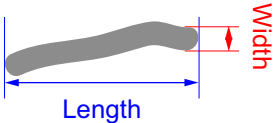
DEFECTS CLASSIFICATION

Defects are defined as:

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



ACCEPTANCE STANDARDS

#	DEFECT TYPE	ACCEPTANCE STANDARDS 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. 98) .)			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		
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		



#	DEFECT TYPE	ACCEPTANCE STANDARDS CRITERIA (Continued)	MAJOR / MINOR							
8	Display pattern defect		Minor							
		<table border="1"> <tr> <th>Dot Size (mm)</th> <th>Acceptable Qty</th> </tr> <tr> <td>$((A+B)/2) \leq 0.2$</td> <td rowspan="5"> ≤ 3 total defects ≤ 2 pinholes per digit </td> </tr> <tr> <td>$C > 0$</td> </tr> <tr> <td>$((D+E)/2) \leq 0.25$</td> </tr> <tr> <td>$((F+G)/2) \leq 0.25$</td> </tr> </table>		Dot Size (mm)	Acceptable Qty	$((A+B)/2) \leq 0.2$	≤ 3 total defects ≤ 2 pinholes per digit	$C > 0$	$((D+E)/2) \leq 0.25$	$((F+G)/2) \leq 0.25$
		Dot Size (mm)		Acceptable Qty						
		$((A+B)/2) \leq 0.2$		≤ 3 total defects ≤ 2 pinholes per digit						
		$C > 0$								
$((D+E)/2) \leq 0.25$										
$((F+G)/2) \leq 0.25$										
9	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. 100), and dark lines or scratches (see test 6, Pg. 100). <p><i>*Minor if display functions correctly. Major if the display fails.</i></p>	Minor							
10	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							
11	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><i>*Minor if display functions correctly. Major if the display fails.</i></p>	Minor							
12	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. Solder balls. <p><i>*Minor if display functions correctly. Major if the display fails.</i></p>	Minor							