Crystalfontz

INTELLIGENT LCD MODULE SPECIFICATIONS

Data Sheet Release 2014-10-13
for
[CFA835-TFK]
[CFA835-TML]
[CFA835-YYK]

Hardware Revision: 1.0
Firmware Revision: 0.6

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

FORWARD - - - - - 5
Revision Information - - - - - 5
Notices - - - - - 5
COLOR AND INTERFACE CHOICES - - - - - 7
Interface Choices - - - - - 7
Color Choices - - - - - 8
FEATURES LIST - - - - - 8
ADDITIONAL FEATURES WHEN USED WITH OPTIONAL CFA-FBSCABS - - - - - 10
CABLES FOR CFA835 - - - - - 13
Cables For USB Interface - - - - - 13
Cables For “Full Swing” RS232 Interface Using CFA-RS232 Serial Converter Board - - - - - 14
Cable For ATX Functionality - - - - - 14
Cable For Direct Connection To Host’s Power Supply - - - - - 15
BRACKETS AND SLEDS - - - - - 16
MECHANICAL SPECIFICATIONS - - - - - 17
Physical Characteristics - - - - - 17
Vibration - - - - - 17
Display Module Outline Drawing, Front & Side Views - - - - - 18
Display Module Outline Drawing, Back View & Pixel Details - - - - - 19
Keypad Detail Drawing - - - - - 20
ELECTRICAL SPECIFICATIONS - - - - - 21
System Block Diagram - - - - - 21
Display Duty And Bias - - - - - 22
Absolute Maximum Ratings - - - - - 23
DC Characteristics - - - - - 23
Logic Level GPIO +5 Volt Tolerant Pins For Serial Interface - - - - - 24
GPIO Current Limits - - - - - 24
Current Consumption - - - - - 25
CFA835-TFK (Dark On Near White) - - - - - 25
CFA835-TML (Near White On Blue) - - - - - 26
CFA835-YYK (Dark On Yellow-Green) - - - - - 27
ESD (Electro-Static Discharge) - - - - - 27
Backlight And Fan Criteria - - - - - 28
OPTICAL SPECIFICATIONS - - - - - 29
Optical Characteristics Tables - - - - - 29
CFA835-TFK (Dark On Near White) - - - - - 29
CFA835-TML (Near White On Blue) And CFA835-YYK (Dark On Yellow-Green) - - - - - 30
Optical Characteristics Test Conditions And Definitions - - - - - 30
LED BACKLIGHT INFORMATION - - - - - 34
CONNECTION INFORMATION - - - - - 34
Location Of Connectors - - - - - 34
H1 Connector Details - - - - - 35
H1 Pin Assignments - - - - - 35
Make Your Own H1 Cable - - - - - 35

CONTENTS, CONTINUED

Standard (+5v) Power Supply And Data Communications Through USB - 36
ATX Power Supply Power And Control Connections - 37
How to Set ATX Functionality Using cfTest - 38
FIRMWARE - 38
How To Identify Firmware Revision Number - 38
Possible Future Firmware Updates - 39
Create Your Own Firmware - 39
HOST COMMUNICATIONS - 39
Through USB - 39
Through Serial - 39
Multiple Port Communications - 40
Packet Structure - 40
About Handshaking - 41
Command Codes - 42
0 (0x00): Ping Command - 44
1 (0x01): Get Module Information - 44
2 (0x02): Write User Flash Area - 44
3 (0x03): Read User Flash Area - 45
4 (0x04): Store Current State As Boot State - 45
5 (0x05): Restart - 46
6 (0x06): Clear Display - 47
9 (0x09): Special Character Bitmaps - 47
11 (0x0B): Display Cursor Position - 48
12 (0x0C): Cursor Style - 48
13 (0x0D): Contrast - 49
14 (0x0E): Display And Keypad Backlights - 49
23 (0x17): Keypad Reporting - 50
24 (0x18): Read Keypad, Polled Mode - 51
28 (0x1C): ATX Functionality - 51
29 (0x1D): Watchdog - 53
31 (0x1F): Write Text To The Display - 53
32 (0x20): Read Text From The Display - 54
33 (0x21): Interface Options - 54
34 (0x22): GPIO Pin Levels - 55
36 (0x24): Interface Bridge - 57
37 (0x25): CFA-FBSCAB - 58
Subcommand 0: Read CFA-FBSCAB Information - 59
Subcommand 1: Fan Settings - 60
Subcommand 2: Read Fan Tachometers - 61
Subcommand 3: Read DOW Device Information - 62
Subcommand 4: Read WR-DOW-Y17 Temperature - 62
Subcommand 5: GPIO Pin Levels - 63
Subcommand 6: Reset And Search - 65
38 (0x26): Custom Fonts - 66

CONTENTS, CONTINUED

Subcommand 0: Load Custom Font Files From MicroSD Card - - - 66
Subcommand 1: Print Custom Font To Display - - - 66
39 (0x27): MicroSD File Operations - - - 67
Subcommand 0: Open/Close MicroSD File - - - 67
Subcommand 1: Position Seek - - - 67
Subcommand 2: Read File Data - - - 68
Subcommand 3: Write File Data - - - 68
Subcommand 4: Delete A File - - - 68
40 (0x32): Display Graphic Options - - - 69
Subcommand 0: Graphic Options - - - 69
Subcommand 1: Buffer Flush - - - 69
Subcommand 2: Send Image Data To Display From Host - - - 69
Subcommand 3: Display Image File From MicroSD Card On CFA835 - - - 70
Subcommand 4: Save Screenshot to MicroSD File - - - 71
Subcommand 5: Pixel Data - - - 71
Subcommand 6: Draw a Line - - - 72
Subcommand 7: Draw a Rectangle - - - 72
Subcommand 8: Draw a Circle - - - 73
41 (0x3A): Video Playback Control - - - 73
Subcommand 0: Load A Video From MicroSD Card - - - 73
Subcommand 1: Video Control - - - 73
Report Codes - - - 74
128 (0x80): Key Activity - - - 74
192 (0xC0): Extended Error Reporting - - - 75
CHARACTER GENERATOR ROM (CGROM) FOR STANDARD SET OF CHARACTERS - - - 76
DISPLAY MODULE RELIABILITY AND LONGEVITY - - - 77
Display Module Reliability - - - 77
Display Module Longevity (EOL / Replacement Policy) - - - 77
CARE AND HANDLING PRECAUTIONS - - - 78
Handling Cautions - - - 78
ESD (Electro-Static Discharge) Specifications - - - 79
Design And Mounting - - - 79
Avoid Shock, Impact, Torque, Or Tension - - - 80
Caution - - - 80
How To Clean - - - 80
Operation - - - 81
Storage And Recycling - - - 81
APPENDIX A: QUALITY ASSURANCE STANDARDS - - - 82
APPENDIX B: FREE DEMONSTRATION AND OTHER SOFTWARE - - - 86
CFA835 Utilities - - - 86
CFA835 Font Editor - - - 86
CFA835 Video Encoder - - - 87
CFA835 Graphic Test - - - 88
Demonstration Software - - - 88

CONTENTS, CONTINUED

cfTest - - - 88
CrystalControl2 (CC2) - - - 88
Linux CLI Examples - - - 88
Sample Code For RPM Calculation Information - - - 89
Sample Code for Temperature Sensor Report - - - 91
Sample Code for Font File Format - - - 92
Algorithms to Calculate the CRC - - - 93
Algorithm 1: “C” Table Implementation - - - 93
Algorithm 2: “C” Bit Shift Implementation - - - 94
Algorithm 2B: “C” Improved Bit Shift Implementation - - - 95
Algorithm 3: “PIC Assembly” Bit Shift Implementation - - - 95
Algorithm 4: “Visual Basic” Table Implementation - - - 97
Algorithm 5: “Java” Table Implementation - - - 99
Algorithm 6: “Perl” Table Implementation - - - 100
Algorithm 7: For PIC18F8722 or PIC18F2685 - - - 101
APPENDIX C: VIBRATION TEST REPORT - - - 104

FIGURES

Figure 1. CFA-RS232 Serial Converted Board Mounted On CFA835 - - - 7
Figure 2. Optional CFA-FBSCAB Connected To CFA835 With WR-EXT-Y37 Extension Cable - - - 10
Figure 3. Example Of Optional CFA-FBSCABs Daisy-Chained with WR-EXT-Y37 - - - 11
Figure 4. Bracket With Silver Brushed Anodized Aluminum Overlay - - - 16
Figure 5. SLED With Black Plastic Overlay - - - 16
Figure 6. Display Module Outline Drawing, Front And Side Views - - - 18
Figure 7. Display Module Outline Drawing, Back View And Pixel Details - - - 19
Figure 8. Keypad Detail Drawing - - - 20
Figure 9. System Block Diagram - - - 21
Figure 10. Definition Of Optimal Contrast Setting (Negative Image) - - - 31
Figure 11. Definition Of Optimal Contrast Setting (Positive Image) - - - 31
Figure 12. Definition Of Response Time (Tr, Tf) (Negative Image) - - - 32
Figure 13. Definition Of Response Time (Tr, Tf) (Positive Image) - - - 32
Figure 14. Definition Of 6:00 O'Clock And 12:00 O'Clock Viewing Angles - - - 33
Figure 15. Definition Of Horizontal And Vertical Viewing Angles (CR>2) - - - 33
Figure 16. Location Of CFA835 Connectors - - - 34
Figure 17. Pin Assignments on CFA835’s H1 Connector (Includes GPIOs) - - - 35
Figure 18. Micro-B USB Connection Pin Details - - - 36
Figure 19. ATX Power Supply and Control Connections Using The WR-PWR-Y25 Cable - - - 38
Figure 20. Character Generator ROM (CGROM) - - - 76
Figure 21. Handling Caution To Avoid Damaging Flat Flex Cable - - - 79

FORWARD

REVISION INFORMATION

Revision History For Data Sheet

Data Sheet Release: 2014-10-13
Data Sheet for new products.

CFA835 Hardware And Firmware Revisions

For information about firmware and hardware revisions for the CFA835, see Part Change Notifications (PCNs) and Product Update Notices (PUNs) under the Notices tab on the website page for each CFA835 part number.

To ensure that the appropriate people in your organization receive notices, please ask them to subscribe at www.crystalfontz.com/news/pcn.php.

NOTICES

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 CFA835 has nonvolatile memory.

Additional Fine Print

COLOR AND INTERFACE CHOICES

INTERFACE CHOICES

All CFA835 Intelligent Display Modules can simultaneously send and receive command packets using two interfaces: USB and a serial interface. When you order a CFA835-TFK, CFA835-TML, or CFA835-YYK on our website, you can choose one of two serial interfaces using the "Customize and Add to Cart" feature.
For the serial interface, you have two choices. You can choose:

USB and **"logic level"** serial interfaces.
or
USB and **"full swing" RS232** serial interfaces when the optional CFA-RS232 Serial Converter Board is purchased to customize your CFA835 order.

The special "bridged" interface mode allows use of an interface to communicate with other slave devices.

Difference Between The Two Serial Interface Choices

Both of the two serial interfaces use firmware that bring the two UART pins (Tx & Rx) of the CFA835's microcontroller to the CFA835's H1 connector.

"Logic Level" Serial

This is the default interface for the CFA835. This interface exposes the UART Tx & Rx ("logic level", 0v to +3.3v Tx nominal, 0v to +5.0v Rx nominal) on pin 1 and pin 2 of the CFA835's 16-pin H1 connector. If your embedded processor is close to the CFA835, you can cable its UART Rx and Tx pins directly to the CFA835's Tx and Rx pins. No RS232 level translators are required on either end.

"Full Swing" RS232 Serial (Requires Optional CFA-RS232 Serial Converter Board)

Bidirectional 9600 / 19200 / 115200 baud ESD protected RS232 is provided when you customize your web order with a mounted CFA-RS232 Serial Converter Board. This interface is the correct choice if your embedded controller or host system has a "full swing" RS232 serial port implemented with a UART plus RS232 level converter.

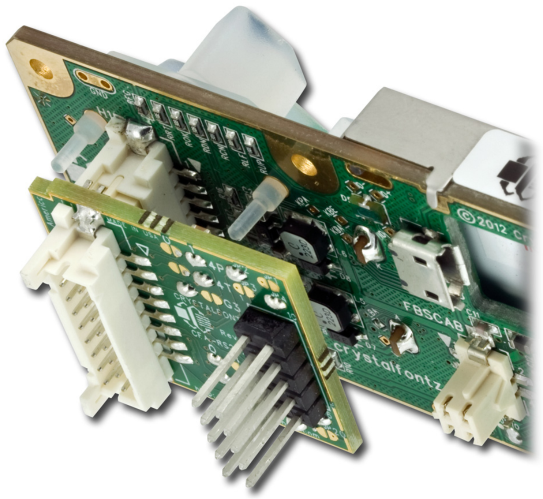


Figure 1. CFA-RS232 Serial Converted Board Mounted On CFA835

The CFA-RS232 Serial Converter Board has a 16-pin female connector J3 that mates with the male 16-pin connector H1 on the back of the CFA835. The CFA-RS232 converts the 0v to +5v (logic level) Rx and Tx signals from the CFA835's microcontroller to RS232 levels. For more information, download the Data Sheet on the CFA-RS232 website page.

COLOR CHOICES

The CFA835 Intelligent Display Modules have three color choices.

	3 Color Choices		
Part Number	**CFA835-TFK**	**CFA835-TML**	**CFA835-YYK**
Fluid[1]	FSTN	STN	STN
Glass Color	black	neutral	neutral
Image[2]	positive	negative	positive
Polarizer Film	transflective	transmissive	transflective
LED Backlight	white		yellow-green

[1]*FSTN has better contrast than STN.*

[2]*Positive Image: 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.*

Negative Image: Display can be read in normal office lighting and in dark areas. May be difficult to read in direct sunlight.

FEATURES LIST

- ❑ The CFA835 is a 244 x 68 graphic display module. The CFA835 can display 5-bit (32 shade) grayscale images from a host computer or a microSD card.
- ❑ Create your own custom made Unicode compatible fonts to fit as many as 80 small characters x 8 lines in any language. Or use our standard set of characters. See CHARACTER GENERATOR ROM (CGROM) FOR STANDARD SET OF CHARACTERS (Pg. 76).
- ❑ Free downloadable utilities package includes the CFA835 Font Editor, CFA835 Video Encoder, and CFA835 Graphic Test. See CFA835 Utilities (Pg. 86).
- ❑ Active Area is 77.97 (W) x 22.38 (H) millimeters. Pixel size is 0.300 (W) x 0.325 (H) millimeters.
- ❑ The CFA835 is mechanically similar to the CFA635 but not identical to it. Location of mounting holes, display, and keypad are the same for the CFA835 and CFA635. The CFA835 hardware is identical to the CFA735 hardware.
- ❑ The CFA835 command set supports most of the commands from the standard Crystalfontz Packet Command set. Certain commands have been expanded to support increased functionality, as well as the addition of many new useful commands. See Command Codes (Pg. 42). You can download the free software cfTest to experiment with this command set.
- ❑ Commands include tools to draw lines, rectangles, and circles. For example, create a chart to display temperature fluctuations over time.
- ❑ The CFA835 has a microSD card socket and is compatible with most FAT12/16/32 formatted SDHC microSD cards. Use the microSD card to display images, play videos, log data, and even update firmware.
- ❑ Fits nicely in a 1U rack mount case using an optional mounting bracket. Or use the optional SLED chassis that holds the CFA835, one optional CFA-FBSCAB, and a 3.5-inch disk drive (not included). Order a bracket or SLED using the "Customize and Add to Cart" feature on our website. Select from 4 overlay colors.

- ❑ Available in three colors, the edge-lit display is backlight with 12 LEDs, 6 per side and has an attractive stainless steel bezel.
- ❑ Six-button translucent silicone keypad with screened legend is backlit with LEDs. Fully decoded keypad: any key combination is valid and unique.
- ❑ Voltage regulated, adjustable contrast and backlight; backlight and contrast are insensitive to changes in voltage over the power supply range. The default contrast value for the CFA835 will be acceptable for many applications. If necessary, you can adjust the contrast using command 13 (0x0D): Contrast (Pg. 49).
- ❑ Only a single supply is needed. Wide power supply voltage range (V_{DD} = +3.3v to +5.5v) is perfect for most embedded systems.
- ❑ Wide Temperature operating range-20°C minimum to +70°C maximum.
- ❑ Viewing angle is 12:00 o'clock.
- ❑ DAC (Digital-to-Analog Converter) controls the constant current LED driver.
- ❑ The front of the display has four bicolor (red + green) LED status lights. Using constant current LED driver, the LEDs' brightness can be set by the host software, which allows smoothly adjusting the LEDs to produce other colors (for example, yellow and orange).
- ❑ The CFA835 is powered by an ST-Micro STM32F103 series 32-bit ARM-based microcontroller and Sitronix ST7529 driver/controller.
- ❑ Robust packet based communications protocol with 16-bit CRC.
- ❑ Nonvolatile memory (flash) capability:
  - Customize the "power-on" CFA835 settings.
  - 124-byte "scratch" register for storing IP, netmask, system serial number, etc.
- ❑ Five GPIOs can be used to control other devices or can be configured for optional ATX functionality.
- ❑ Optional ATX functionality allows the keypad buttons to replace the power and restart switches on your system, simplifying front panel design. The 16-pin WR-PWR-Y25 ATX power switch cable may be used for direct connection to the host's power supply. A perfect fit for 1U appliances, the CFA835 can be the only component in your front panel, simplifying mechanical, electrical, and software design.
- ❑ Hardware watchdog can restart host on host software failure.
- ❑ Additional free demonstration code is available. See APPENDIX B: FREE DEMONSTRATION AND OTHER SOFTWARE (Pg. 86).
- ❑ To download the most current Certificate of Compliance for ISO, RoHS, and REACH, go to the CFA835's Doc/Files tab on the part number's website page.

ADDITIONAL FEATURES WHEN USED WITH OPTIONAL CFA-FBSCABS

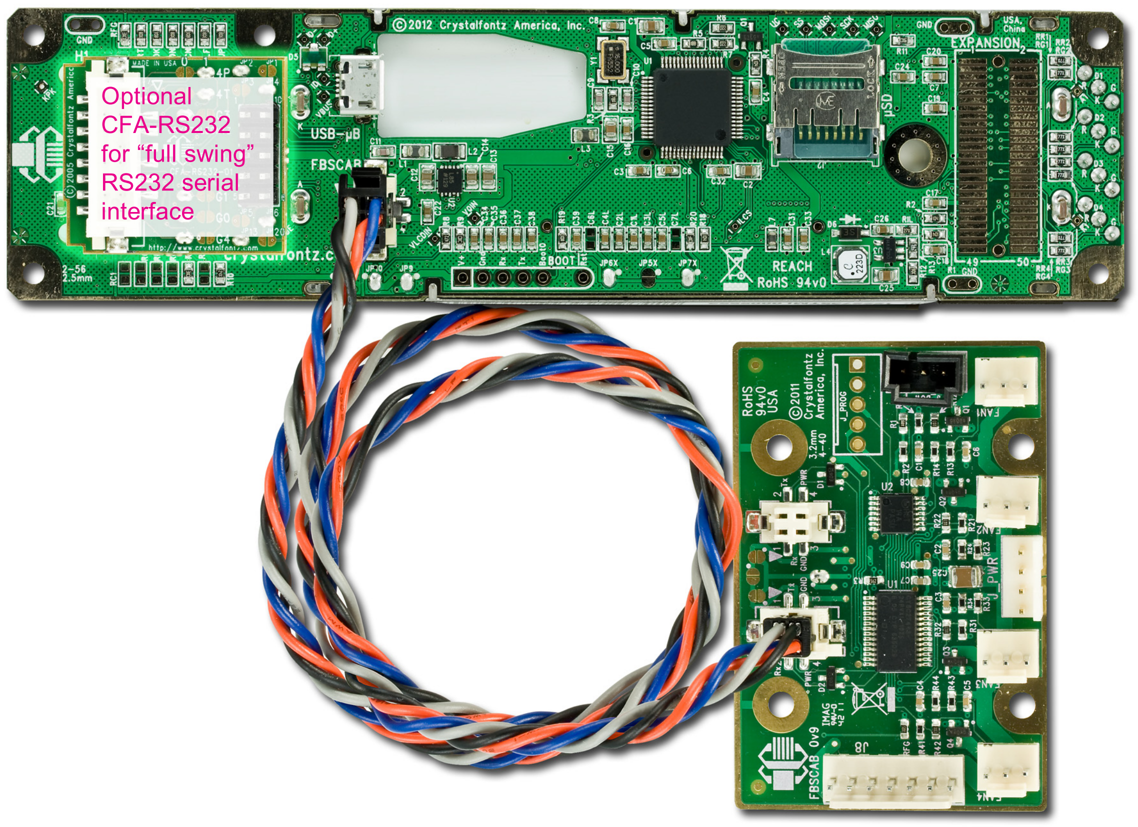


Figure 2. Optional CFA-FBSCAB Connected To CFA835 With WR-EXT-Y37 Extension Cable

To use all of the command set described in command 37 (0x25): CFA-FBSCAB (Pg. 58), at least one optional CFA-FBSCAB is required. Add up to 32 CFA-FBSCABs to your CFA835 order using the "Customize and Add to Cart" feature on our website. If you add optional CFA-FBSCABs, you will be prompted to add one WR-EXT-Y37 extension cable to your order.

A CFA835+one CFA-FBSCAB allows:

- Easy control of up to four fans with tachometer speed monitoring and variable PWM control per CFA-FBSCAB. Fail-safe fan power settings allows your host to safely control the fans based on temperature. Buy one 3-pin fan extension cable WR-FAN-X01 to connect each fan.
- Add up to 16 Crystalfontz WR-DOW-Y17 temperature sensor cables that have Maxim DS18B20 Programmable Resolution 1-Wire temperature sensors. The DS18B20 has an operating temperature range of -55°C to +125°C and is accurate to ±0.5°C over the range of -10°C to +85°C

- Up to 32 CFA-FBSCABs can be connected to each other (“daisy chained”). (“Daisy chain” means several devices connected in a linear series.) Maximum configuration will allow 128 fans and 512 temperature sensors.

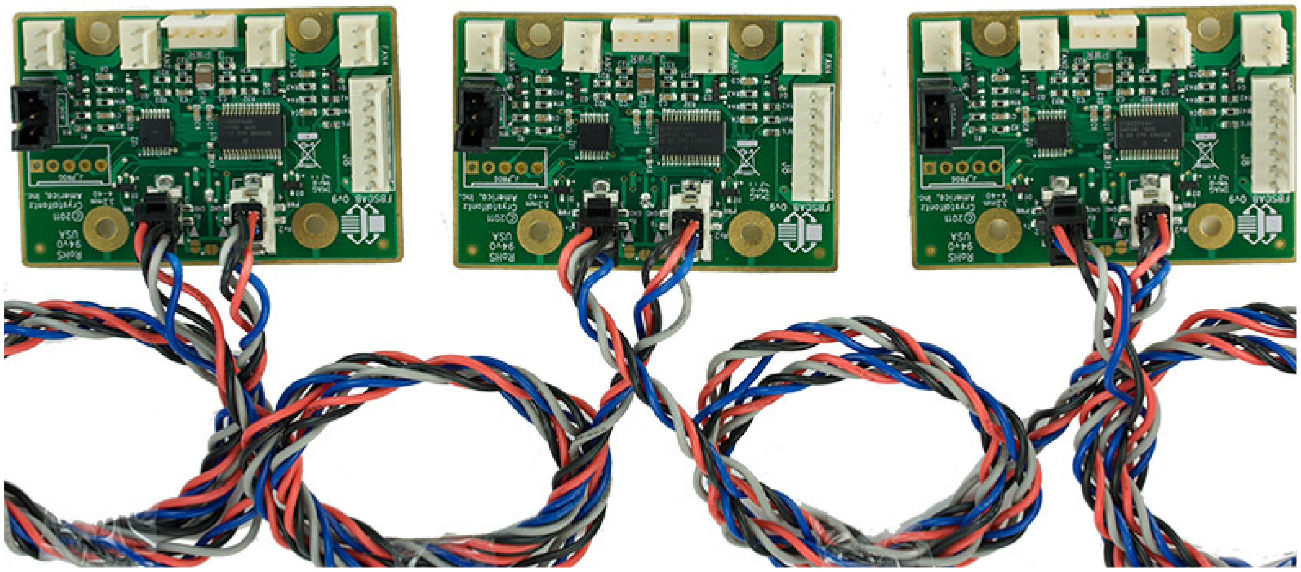


Figure 3. Example Of Optional CFA-FBSCABs Daisy-Chained with WR-EXT-Y37

- The CFA835+CFA-FBSCABs has no ATX functionality provided through the CFA-FBSCABs. However, ATX control is still available using the H1 connector on the CFA835.

> Note
> Remove power before connecting or disconnecting multiple CFA-FBSCABs. Connecting or disconnecting multiple CFA-FBSCABs while powered will cause addressing problems.

For more information, download the Data Sheet on the CFA-FBSCAB website page.

CABLES TO USE WITH OPTIONAL CFA-FBSCABS (FB SYSTEM COOLING ACCESSORY BOARDS)

The CFA835 does not supply power to the CFA-FBSCABs. The CFA-FBSCABs require external power, typically supplied by a 4-pin 3.5-inch floppy drive power connector. If you customize your CFA835 order by adding up to 32 optional CFA-FBSCABs, you will be prompted to add these cables to your order.

WR-EXT-Y37

The WR-EXT-Y37 is about 1 foot 6 inches long. Connect the cable’s 4-pin male connector to the CFA835’s connector labeled FBSCAB. Connect the cable’s 4-pin female connector to the first CFA-FBSCAB. This connector has the top left pin labeled Rx2.

WR-PWR-Y12

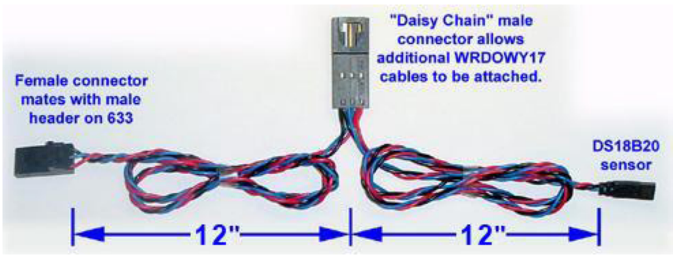
The WR-PWR-Y12 cable is about1 foot 0.55 inches long. This 4-pin hard drive to floppy connector and hard drive splitter power cable can be used to power the CFA-FBSCABs. Requires +5v and +12v for fans.

WR-FAN-X01

Up to four fans are supported per CFA-FBSCAB. Crystalfontz offers the WR-FAN-X01 extension cable to extend the range of your fans. The WR-FAN-X01 fan cable is about 1 foot, 4.30 inches long.

Connect the cable's 3-pin male connector to a CFA-FBSCAB's connectors labeled FAN1, FAN2, FAN3, or FAN4. Connect the cable's 3-pin female connector to a fan's connector. (Fans are not sold by Crystalfontz.)

WR-DOW-Y17 Temperature Sensor


"Daisy Chain" male connector allows additional WRDOWY17 cables to be attached.
Female connector mates with male header on 633
DS18B20 sensor
12"
12"

The WR-DOW-Y17 temperature sensor cable is about 1 foot 1.15 inches in length overall.

When an optional CFA-FBSCAB is connected to a CFA835, you can add a WR-DOW-Y17 to the CFA-FBSCAB 's connector labeled J_DOW. The WR-DOW-Y17 has a Maxim DS18B20 Programmable Resolution 1-Wire temperature sensor attached to a "daisy chainable" cable. If desired, connect the cable's 3-pin male connector to an additional temperature sensor. Up to 32 temperature sensors per CFA-FBSCAB can be connected ("daisy chained").

Connect the cable's 3-pin female connector to the CFA-FBSCAB's connector labeled J_DOW. The cable has a male expansion header midway along its length. If desired, connect the cable's 3-pin male connector to an additional temperature sensor.

The Maxim DS18B20 temperature sensor on the WR-DOW-Y17 has 0.5°C absolute accuracy. You can make your own temperature sensor cable using a Maxim DS1822 Econo 1-Wire Digital Thermometer with $\underline{+}2$°C accuracy.

Make Your Own CFA-FBSCAB Cable

To make your own CFA-FBSCAB cable, here are some typical connector parts manufactured by Hirose, available through Digi-Key:

Connection At CFA-FBSCAB

- Female housing on cable: Hirose DF11-4DS-2C / Digi-Key H2019-ND.
- Female crimp terminal in housing: Hirose DF11-2428SC / Digi-Key H1504TR-ND.
- For reference, mating male connector on CFA-FBSCAB: Hirose DF11GZ-4DP-2V(20) / Digi-Key H10259-ND.

Connection At CFA835

- Male housing on cable: Hirose DF11-4DEP-2C / Digi-Key H2913-ND.
- Male crimp terminal in housing: Hirose DF11-EP2428PC / Digi-Key H1506-ND.
- For reference, mating female connector on CFA835: Hirose DF11Z-4DS-2V(20) / Digi-Key H10197-ND.

Pre-crimped wires are also available from Digi-Key. Here is a link to a 12", 24ga, pin-to-socket in blue: Hirose H3ABT-10112-L4-ND / H3ABT-10112-L4-ND.

For more information, download the Data Sheet on the CFA-FBSCAB website page.

CABLES FOR CFA835

Most of the cables described below can be added to your CFA835 order using the “Customize and Add to Cart” feature on CFA835 web pages. See all of our cables listed at https://www.crystalfontz.com/products/lcd-display-cables.html. To make your own cable, see the part list under Make Your Own H1 Cable (Pg. 35).

CABLES FOR USB INTERFACE

Note: The CFA835 uses a nonstandard 2 mm low profile USB connector. USB cables with this type of connector are not readily available. If you do not have this cable and cannot build one, be sure to add one of these cables to your order.

Note: For these cables, keep the micro-B USB cable connector parallel to the CFA835 when plugging or unplugging the cable. Do not lift or pull up on the cable. Too much pressure may permanently damage the CFA835’s micro-B USB connector.

For more information, see Standard (+5v) Power Supply And Data Communications Through USB (Pg. 36).

WR-USB-Y27

The WR-USB-Y27 cable is about 6 feet 3.75 inches long. Connect the cable’s USB-A female connector to your host’s USB-A connector. Connect the cable’s micro-B USB female connector to the CFA835’s micro-B USB connector.

WR-USB-Y34

The WR-USB-Y34 cable is about 2 feet 2.35 inches long. Connect the cable’s micro-B USB female connector to CFA835’s micro-B USB connector. Connect the cable’s single piece 4-pin 0.1" female connector to USB pins on host’s motherboard. For correct orientation, note the +5v location on the 4-pin connector.

CABLES FOR “FULL SWING” RS232 INTERFACE USING CFA-RS232 SERIAL CONVERTER BOARD

If you customize your CFA835 order by adding the optional CFA-RS232 Serial Converter Board, you will be prompted to add RS232 cables to supply power and communications.

WR-232-Y08

The WR-232-Y08 cable is about 2 feet 2.55 inches long. Connect the cable’s 10-pin female connector to the CFA-RS232’s J1 connector. Connect cable’s RS232 DB9 9-pin female connector to your host’s external 9-pin serial port.

WR-232-Y22

The WR-232-Y22 cable is about 2 feet 1 inches long. Connect the cable’s 0.1" 2x5 female connector to the CFA-RS232’s J1 10-pin connector. Connect the cable’s second 0.1" 2x5 female connector to your host’s motherboard 10-pin connector. Choose standard or alternate pinout.

WR-232-Y23

The WR-232-Y23 cable is about 2 feet 1.75 inches long. Connect the cable’s 0.1" 2x5 female connector to the CFA-RS232’s J1 10-pin connector. Connect the cable’s RS232 DB9 9-pin female connector to your host’s external 9-pin serial port. Choose standard or alternate pinout. *Note*: This cable is not listed on the CFA835 “Customize and Add to Cart” feature. Add the cable as a separate item to your order.

CABLE FOR ATX FUNCTIONALITY

ATX power supply control functionality allows the buttons on the CFA835 to replace the power and restart button on your system, simplifying front panel design. For more information, see ATX Power Supply Power And Control Connections (Pg. 37) followed by How to Set ATX Functionality Using cfTest (Pg. 38).

Note: The CFA835 with optional CFA-FBSCABs has no ATX functionality provided through the CFA-FBSCABs. However, ATX control is still available on the CFA835’s H1 connector.

WR-PWR-Y25

About 11 inches long, connect the WR-PWR-Y25 ATX power switch cable’s 16-pin female connector to the CFA835’s H1 connector. Connect the cable’s male connector directly to your host’s ATX power supply. Or connect the cable’s 4 separate female connectors to the 4 pins on your host’s motherboard.

WR-PWR-Y44

Similar to the WR-PWR-Y25 above, this one meter long WR-PWR-Y44 ATX power cable is designed for use with rack mount chassis where extra length is necessary for routing and connectivity.

CABLE FOR DIRECT CONNECTION TO HOST'S POWER SUPPLY

WR-PWR-Y24

The WR-PWR-Y24 cable is about 2 feet 2.55 inches long. Connect power directly from your host's power supply.

For *"logic level" interface*, connect the cable's 16-pin female connector to the CFA835's 16-pin H1 connector. Connect the cable's 4-pin male connector to your host's power supply. *Note*: Rx/TX will not be available.

For *"full swing" RS232 interface,* use this cable to supply power (no communications) to the CFA835 through the optional mounted CFA-RS232 Serial Converter Board from the host's power supply. Connect the cable's 16-pin female connector to the mounted CFA-RS232's 16-pin male J2 connector. Connect the cable's 4-pin male connector directly to your host's power supply connector.

BRACKETS AND SLEDS

A bracket or SLED can be added to your CFA835 order using the “Customize and Add to Cart” feature on CFA835 web pages. A colored overlay with a display window of thick hard-coated polycarbonate is attached with adhesive to the front of the bracket or SLED.

Overlay choices are black brushed anodized aluminum, silver brushed anodized aluminum, beige plastic, and black plastic.

Drive Bay Bracket

A 5.25-inch half-height drive bay mounting bracket with your choice of four overlays. The bracket can hold a CFA835 and the optional CFA-FBSCAB.

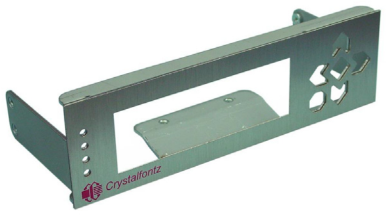


Figure 4. Bracket With Silver Brushed Anodized Aluminum Overlay

SLED

Our SLED is a chassis that fits in 5.25-inch half-height drive bay. The SLED can hold a CFA835, the optional CFA-FBSCAB, and a 3.5-inch hard disk drive. (Hard drive is not included.)

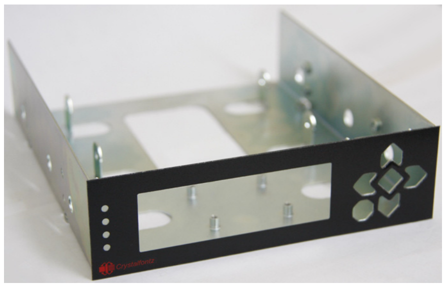


Figure 5. SLED With Black Plastic Overlay

MECHANICAL SPECIFICATIONS

PHYSICAL CHARACTERISTICS

PHYSICAL CHARACTERISTIC	SPECIFICATION
Display Module Overall Dimensions	
Width	142.00 (W) mm
Height	37.40 typical to 38.00 maximum (H) mm
Depth (Thickness) with keypad and connectors without keypad	 20.80 (D) mm 15.30 (D) mm
Viewing Area	83.00 (W) x 27.50 (H) mm
Pixel Array	79.27 (W) x 23.78 (H) mm
Active Area	77.97 (W) 22.38 (H) mm
Column Dots x Row Dots	244 x 68
5x7 Standard Character (see CHARACTER GENERATOR ROM (CGROM) FOR STANDARD SET OF CHARACTERS (Pg. 76))	3.225 (W) x 4.875 (H) mm
6x8 Character Matrix	3.900 (W) x 5.600 (H) mm
Pixel Size	0.300 (W) x 0.325 (H) mm
Pixel Pitch	0.325 (W) x 0.350 (H) mm
Keystroke Travel (approximate)	~2.4 mm
Weight including keypad and connectors if CFA835 is customized with CFA-RS232 Serial Converter Board	 55 grams (typical) 60 grams (typical)

VIBRATION

Test conditions:

- GR-63-CORE 5.4.2, Office Vibration, Alternative Test: 5-100-5 Hz at 1.0 g with a sweep rate of .25
- Octave/minute, 35 minutes per axis.
- MIL-STD 810F, Figure 514C-17, Random: 1 hour per axis.
- MIL-STD 810F, Figure 514C-18, Sine: 1 hour per axis.

For test details of the CFA835's CFA-10052 PCB in CFA735 mode, see APPENDIX C: Vibration Test Report (Pg. 104).

DISPLAY MODULE OUTLINE DRAWING, FRONT & SIDE VIEWS

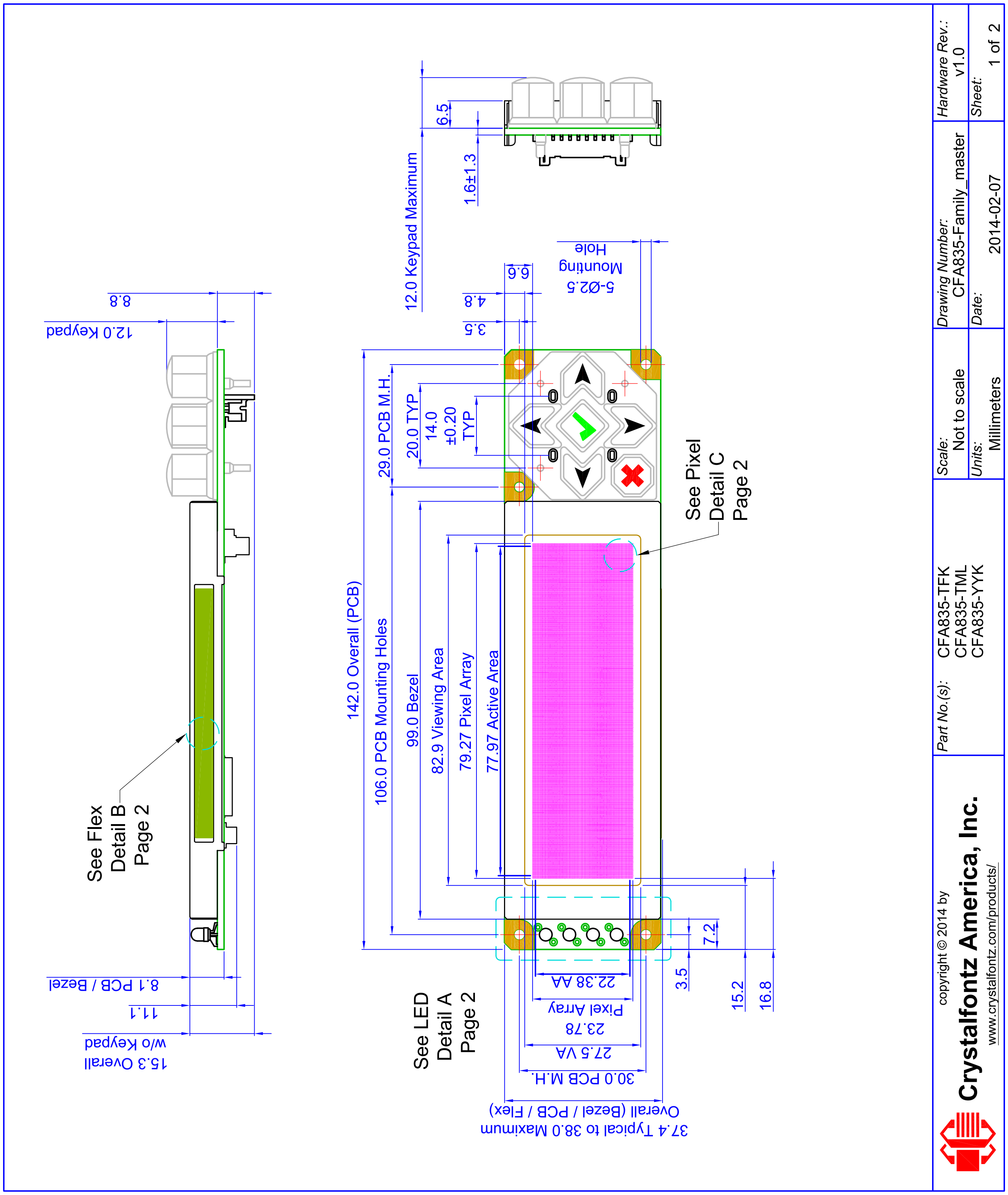


Figure 6. Display Module Outline Drawing, Front And Side Views

DISPLAY MODULE OUTLINE DRAWING, BACK VIEW & PIXEL DETAILS

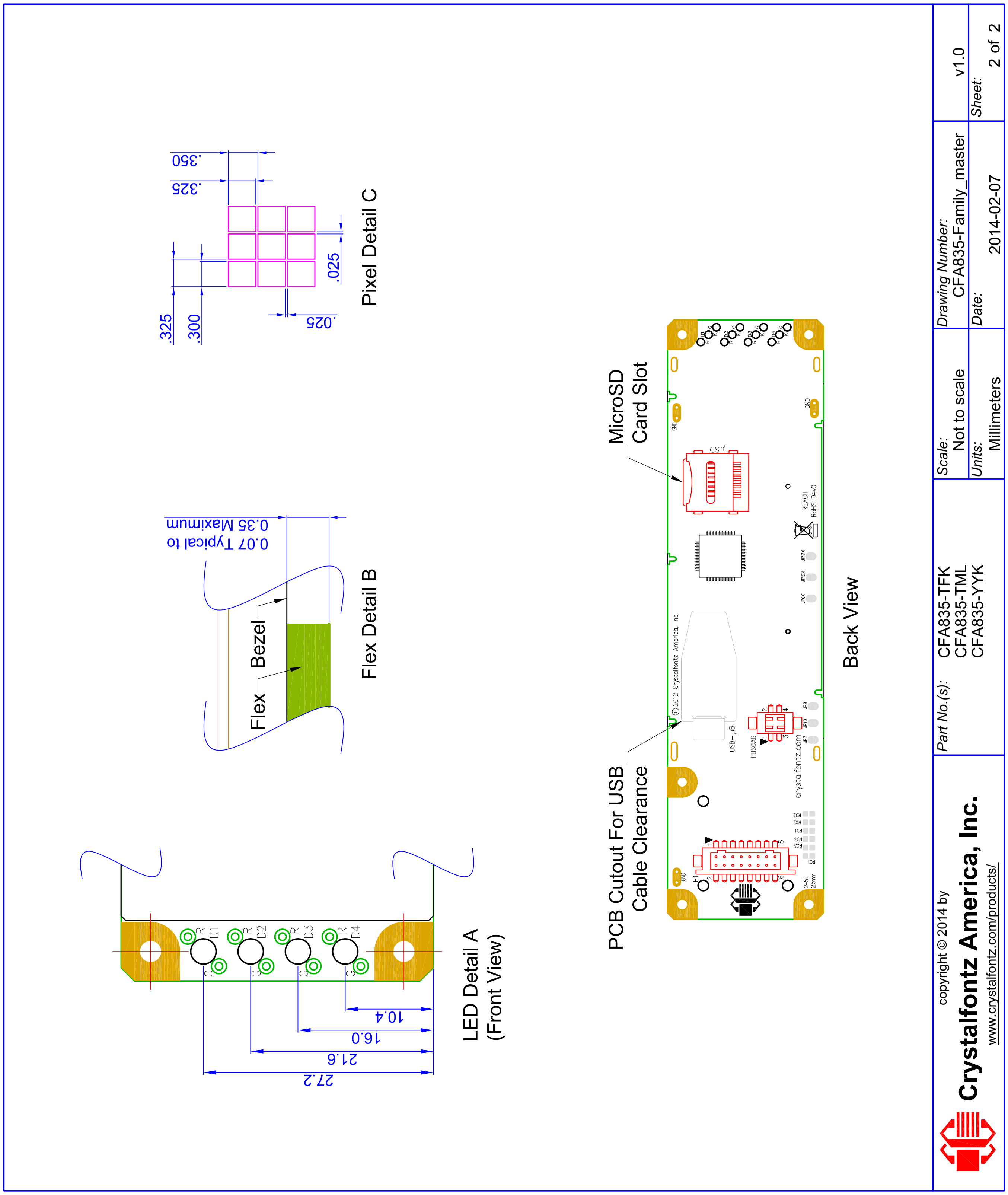
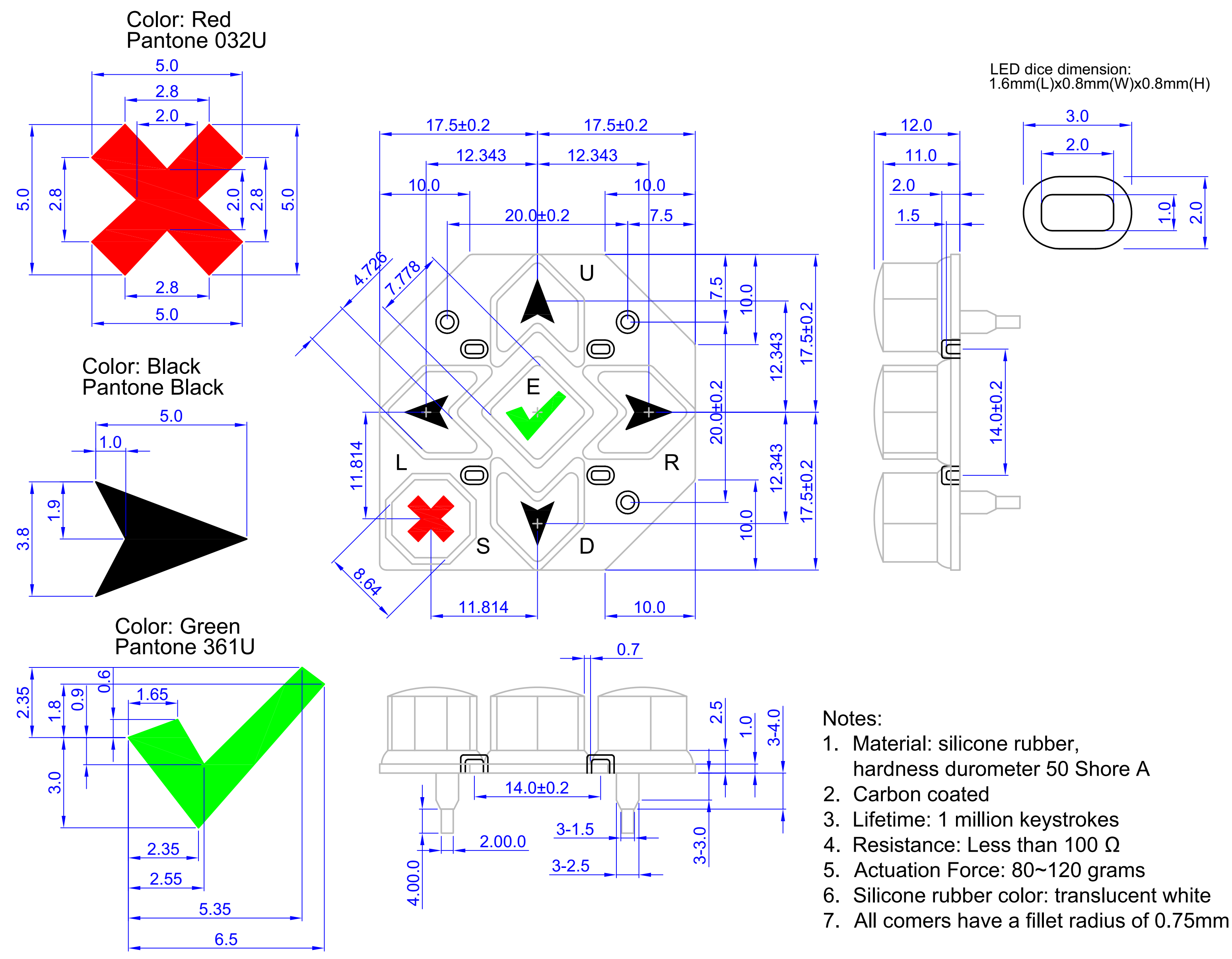


Figure 7. Display Module Outline Drawing, Back View And Pixel Details

KEYPAD DETAIL DRAWING



Notes:
1. Material: silicone rubber, hardness durometer 50 Shore A
2. Carbon coated
3. Lifetime: 1 million keystrokes
4. Resistance: Less than 100 Ω
5. Actuation Force: 80~120 grams
6. Silicone rubber color: translucent white
7. All comers have a fillet radius of 0.75mm

copyright © 2013 by Crystalfontz America, Inc. www.crystalfontz.com/products/	Part No.(s): 6 Button Keypad Detail	Scale: Not to scale	Drawing Number: CFA835_Family_master	
		Units: Millimeters	Date: 2013-11-08	Sheet: 1 of 1

Figure 8. Keypad Detail Drawing

ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

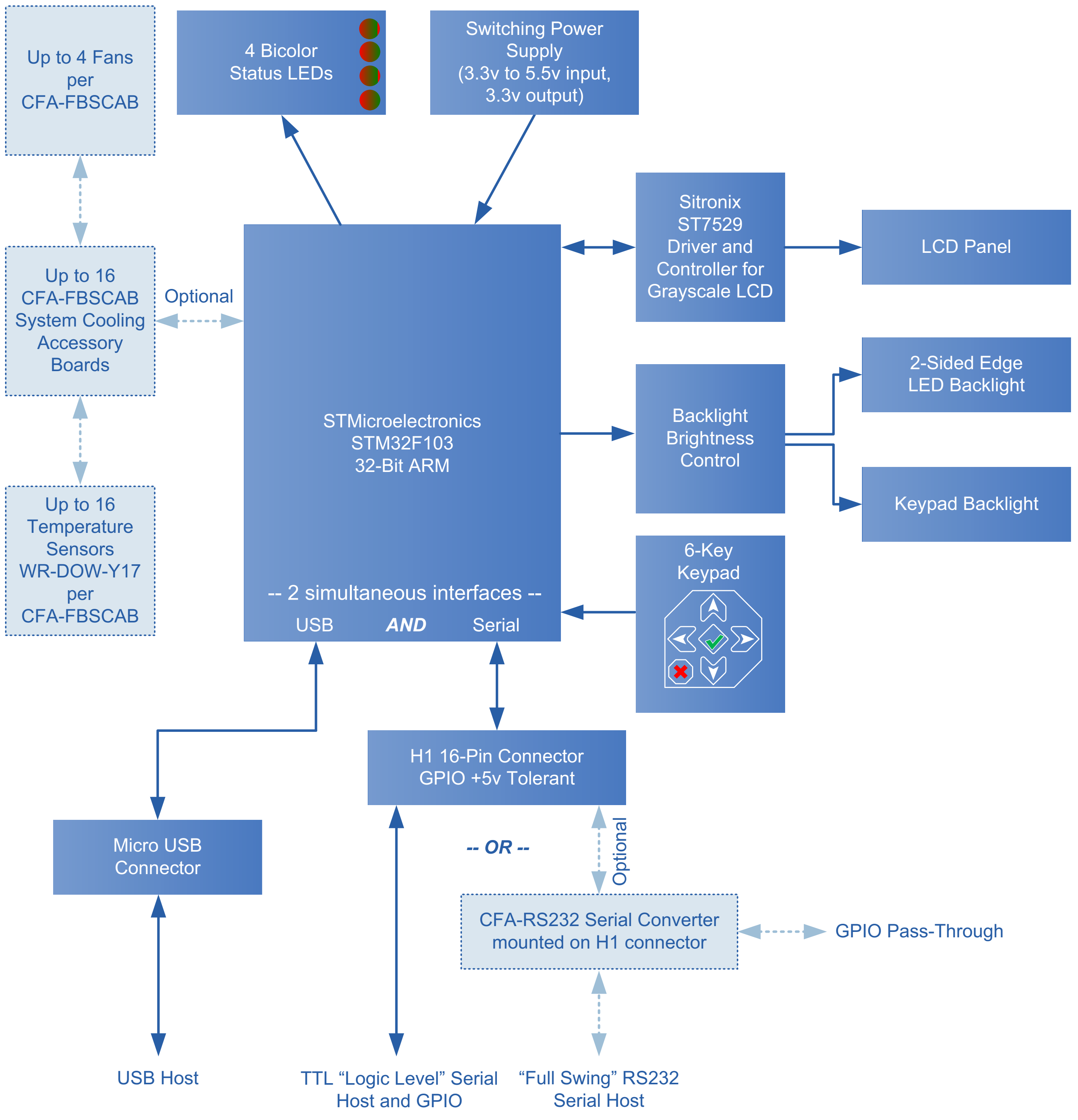

Up to 4 Fans per CFA-FBSCAB
4 Bicolor Status LEDs
Switching Power Supply (3.3v to 5.5v input, 3.3v output)
Sitronix ST7529 Driver and Controller for Grayscale LCD
LCD Panel
Up to 16 CFA-FBSCAB System Cooling Accessory Boards
Optional
STMicroelectronics STM32F103 32-Bit ARM
-- 2 simultaneous interfaces --
USB AND Serial
Backlight Brightness Control
2-Sided Edge LED Backlight
Keypad Backlight
6-Key Keypad
Up to 16 Temperature Sensors WR-DOW-Y17 per CFA-FBSCAB
H1 16-Pin Connector GPIO +5v Tolerant
Micro USB Connector
-- OR --
Optional
CFA-RS232 Serial Converter mounted on H1 connector
GPIO Pass-Through
USB Host
TTL “Logic Level” Serial Host and GPIO
“Full Swing” RS232 Serial Host

Figure 9. System Block Diagram

DISPLAY DUTY AND BIAS

DRIVING METHOD	SPECIFICATION
Duty[1]	1/160
Bias[2]	1/13

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

[2]The drive bias, also known as voltage margin, is related to the number of voltage levels used when driving the display. Bias is defined as 1/(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

All variants (all colors)

ABSOLUTE MAXIMUM RATINGS	SYMBOL	MINIMUM	MAXIMUM
Operating Temperature	T_{OP}	-20°C	+70°C
Storage Temperature	T_{ST}	-30°C	+80°C
Humidity Range (Noncondensing)	RH	10%	90%
Supply Voltage for Logic	V_{DD}	0v	+5.50v
If interface is "Full Swing" RS232 using optional CFA-RS232 Serial Converter Board:			
RS232 Input Pin	V_{RX}	-25v	+25v
RS232 Output Pin	V_{TX}	-13v	+13v
Notes: *These are stress ratings only. Extended exposure to the absolute maximum ratings listed above may affect reliability or cause permanent damage.* *Changes in temperature can result in changes in contrast.*			

DC CHARACTERISTICS

	SPECIFICATIONS	SYMBOL	MINIMUM	TYPICAL	MAXIMUM
INPUT SUPPLY	Supply Voltage For CFA835	V_{IH}	+3.3v	+5.0v	+5.5v
	Internal Processor And Logic			+3.3v	
'FULL SWING" RS232	Input Voltage Range (Rx)	V_{OH}	-25v		+25v
	Output Voltage Swing (Tx)	V_{OL}	$\pm$5.0v	$\pm$5.4v	

LOGIC LEVEL GPIO +5 VOLT TOLERANT PINS FOR SERIAL INTERFACE

The H1 connector on the CFA835 has five GPIO pins. See H1 Pin Assignments (Pg. 35).

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

The default serial interface for the CFA835 is logic level serial. If the optional CFA-RS232 Serial Converter is mounted to provide, “full swing” RS232, the H1 connector’s GPIO pins pass through to the CFA-RS232 J2 connector. For CFA-RS232 connection details, see the Data Sheet on the CFA-RS232 website page.

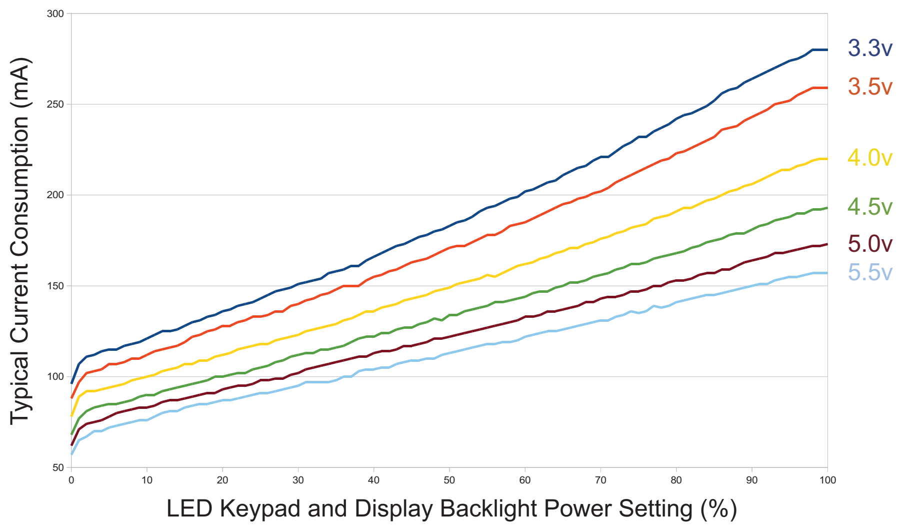
GPIO CURRENT LIMITS

TYPICAL GPIO CURRENT LIMITS	
Sink	8 mA
Source	8 mA

CURRENT CONSUMPTION

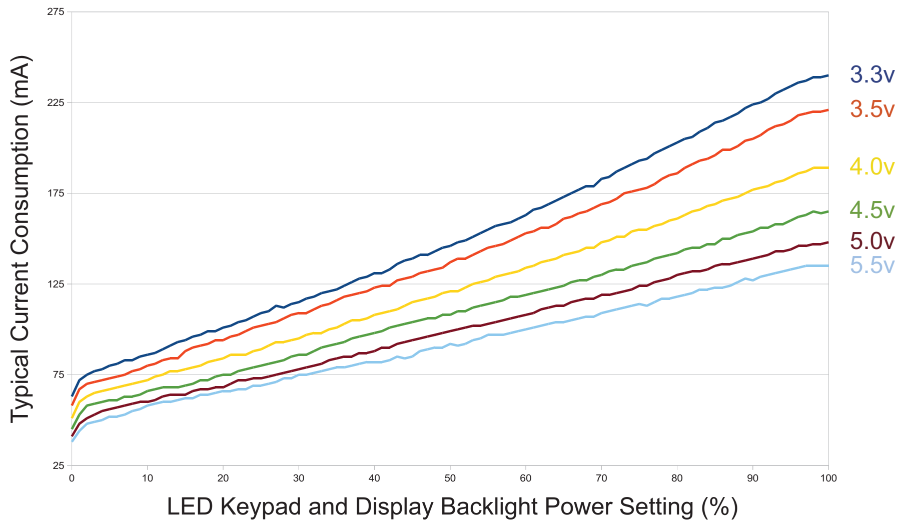
Current consumption varies by color choice. Current consumption is the same for all interfaces.

CFA835-TFK (Dark On Near White)



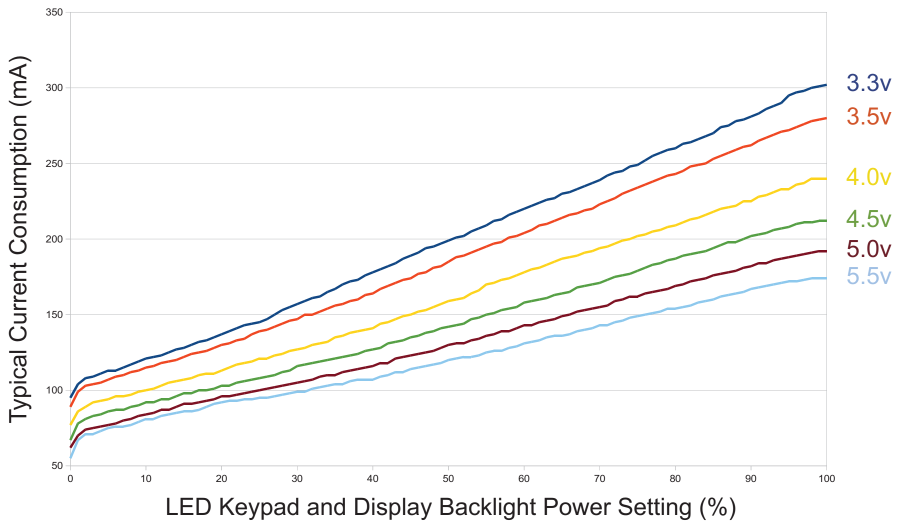
ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION	
Logic	Display and Keypad Backlights at 100%	All Status LEDs 4 Red + 4 Green at 100%	V_{DD} = +3.3	V_{DD} = +5.0v
X	-	-	70 mA	50 mA
X	X	-	245 mA	151 mA
X	-	X	193 mA	152 mA
X	X	X	383 mA	267 mA

CFA835-TML (Near White On Blue)



ITEMS ENABLED			**TYPICAL CURRENT CONSUMPTION**	
Logic	**Display and Keypad Backlights at 100%**	**All Status LEDs 4 Red + 4 Green at 100%**	**V_{DD} = +3.3v**	**V_{DD} = +5.0v**
X	-	-	73 mA	54 mA
X	X	-	248 mA	156 mA
X	-	X	205 mA	161 mA
X	X	X	393 mA	276 mA

CFA835-YYK (Dark On Yellow-Green)



ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION	
Logic	Display and Keypad Backlights at 100%	All Status LEDs 4 Red + 4 Green at 100%	V_{DD} = +3.3v	V_{DD} = +5.0v
X	-	-	62 mA	41 mA
X	X	-	266 mA	173 mA
X	-	X	195 mA	153 mA
X	X	X	411 mA	166 mA

ESD (ELECTRO-STATIC DISCHARGE)

Tx and Rx pins when serial interface is used:
+15 kV Human Body Model
+15 kV IEC1000-4-2 Air Discharge
+8 kV IEC1000-4-2 Contact Discharge

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

BACKLIGHT AND FAN CRITERIA

BACKLIGHT AND FAN CRITERIA	SPECIFICATION
Luminous Intensity Through Panel *CFA835-TFK* *CFA835-TML* *CFA835-YYK*	 486 cd/m^2 467 cd/m^2 577 cd/m^2
Backlight PWM Frequency	300 Hz nominal
Fan Power Control PWM Frequency	18 Hz nominal
Notes; One or more optional CFA-FBSCABs are required to add fans. *PWM* is *Pulse Width Modulation*. PWM is a way to simulate intermediate levels by switching a level between full on and full off. PWM can be used to control the brightness of LED backlights, relying on the natural averaging done by the human eye, as well as for controlling fan power.	

OPTICAL SPECIFICATIONS

OPTICAL CHARACTERISTICS TABLES

CFA835-TFK (Dark On Near White)

ITEM	SYMBOL	CONDITION	TYPICAL	MAXIMUM
Viewing Angle (12 o'clock)	Deg θ = 90°	CR≥2	35	
	Deg θ = 270°		60	
	Deg θ = 0°		45	
	Deg θ = 180°		45	
Contrast Ratio[1]	CR		3.8	5.0
LCD Response Time[2,3]	T rise	Ta = 25°C	180 ms	
	T fall		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.*

Viewing Direction: 12 o'clock

CFA835-TML (Near White On Blue) And CFA835-YYK (Dark On Yellow-Green)

ITEM	SYMBOL	CONDITION	TYPICAL	MAXIMUM
Viewing Angle (12 o'clock)	Deg θ = 90°	CR≥2	30	
	Deg θ = 270°		40	
	Deg θ = 0°		30	
	Deg θ = 180°		30	
Contrast Ratio[1]	CR		3.8	5.0
LCD Response Time[2,3]	T rise	Ta = 25°C	180 ms	
	T fall		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.*				
Viewing Direction: 12 o'clock				

OPTICAL CHARACTERISTICS TEST CONDITIONS AND DEFINITIONS

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

- Viewing Angle
  - Vertical (V)θ: 0°
  - Horizontal (H)φ: 0°
- Frame Frequency: 78 Hz
- Driving Waveform: 1/160 Duty, 1/13 Bias
- Ambient Temperature (Ta): 25°C

Definition Of Optimal Contrast Setting

CFA835-TML

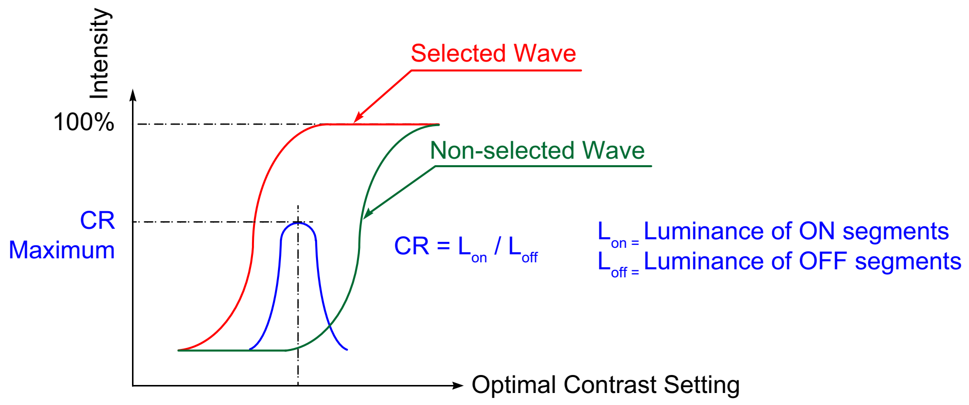


Figure 10. Definition Of Optimal Contrast Setting (Negative Image)

CFA835-TFK *and CFA835-YYK*

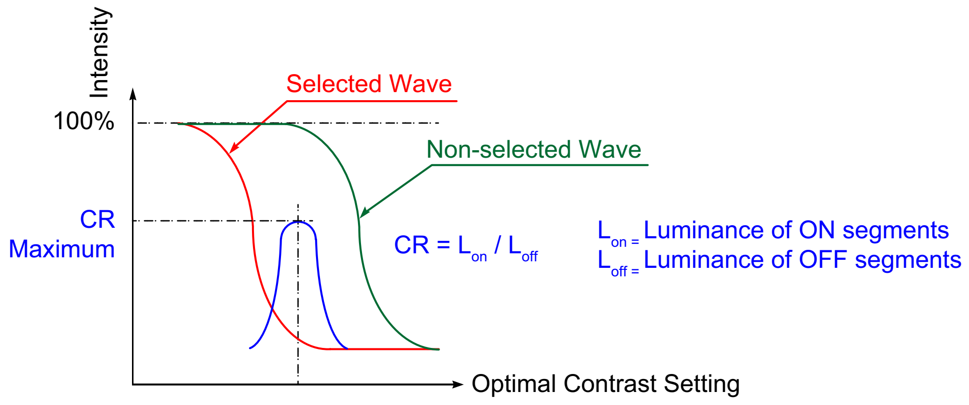


Figure 11. Definition Of Optimal Contrast Setting (Positive Image)

Definition Of Response Time (Tr, Tf)

CFA835-TML

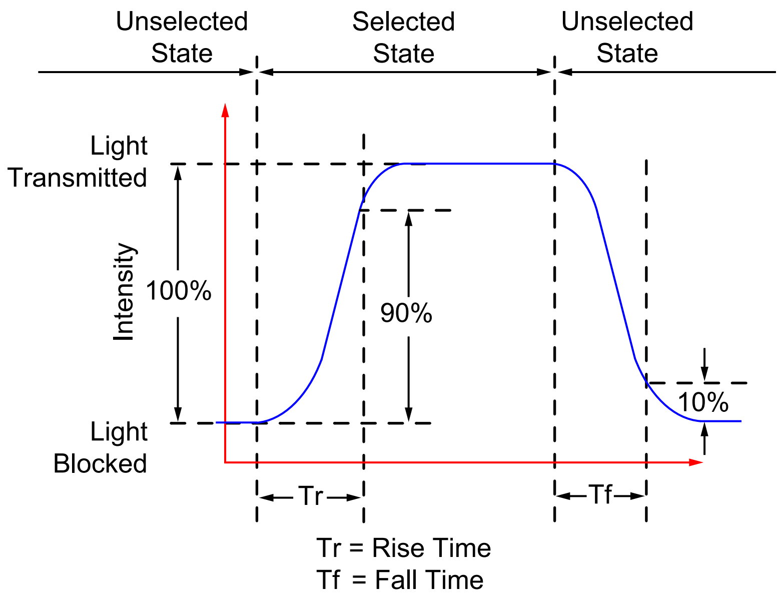


Figure 12. Definition Of Response Time (Tr, Tf) (Negative Image)

CFA835-TFK *and CFA835-YYK*

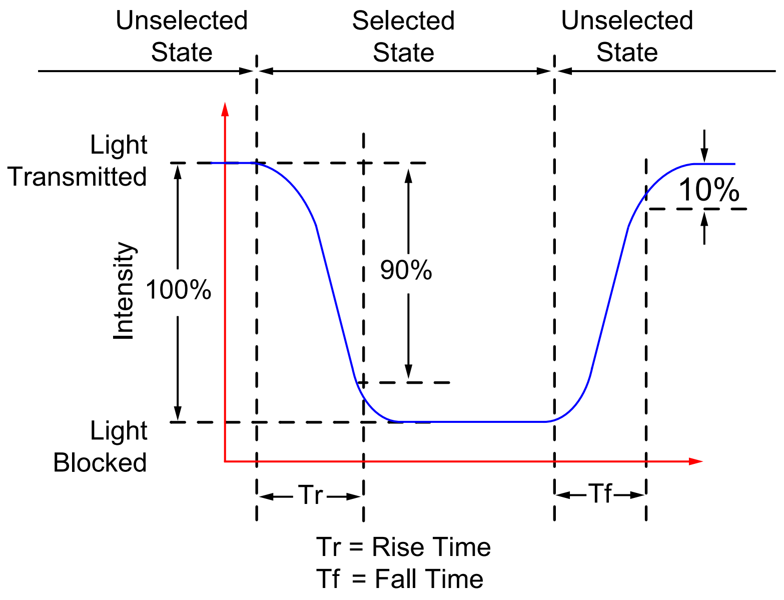


Figure 13. Definition Of Response Time (Tr, Tf) (Positive Image)

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

The CFA835 has a 12 o'clock viewing angle.

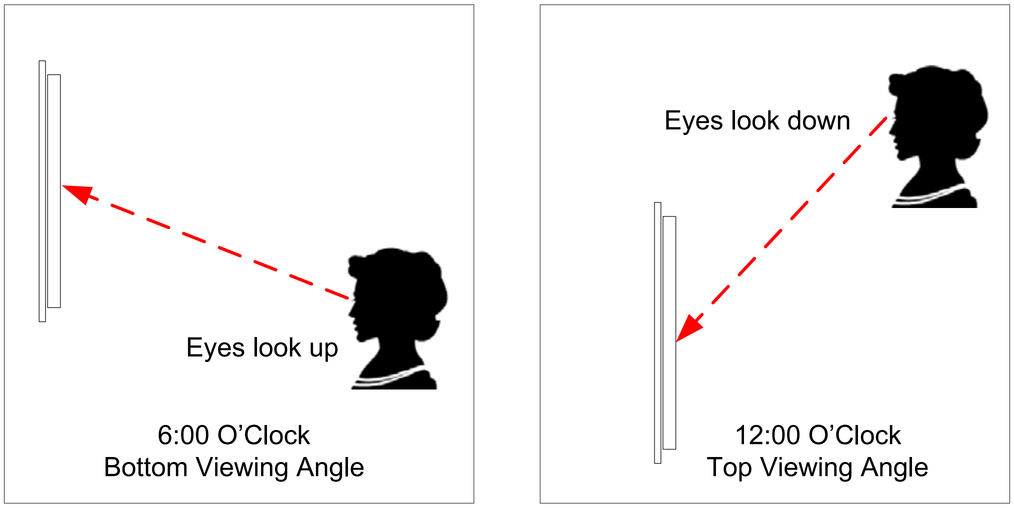


Figure 14. Definition Of 6:00 O'Clock And 12:00 O'Clock Viewing Angles

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

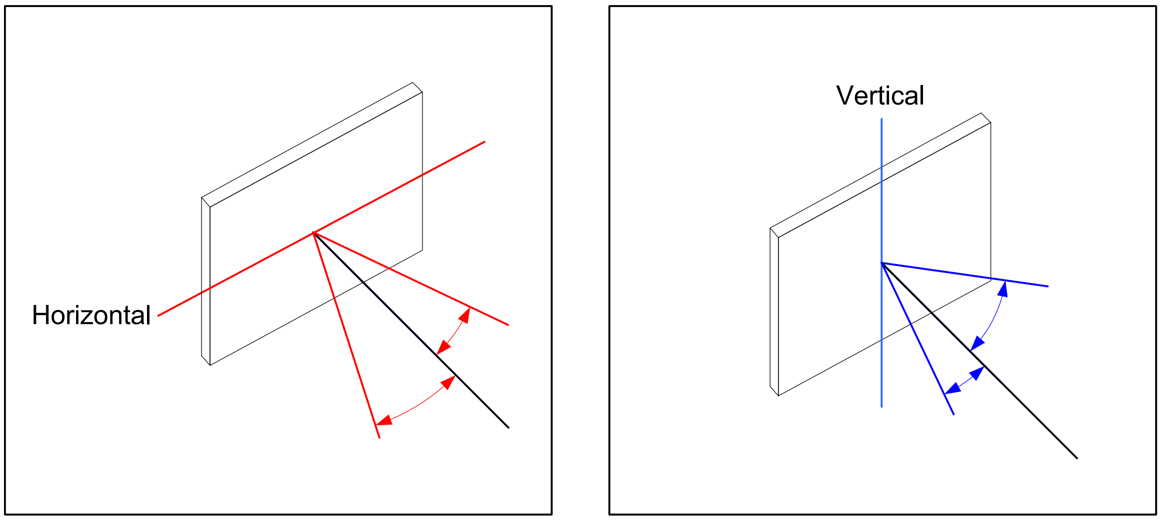


Figure 15. Definition Of Horizontal And Vertical Viewing Angles (CR>2)

LED BACKLIGHT INFORMATION

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

Note
For display modules with **white** backlights (CFA835-TML and CFA835-TFK), we recommend that the backlight be dimmed or turned off during periods of inactivity to conserve the LEDs' lifetime.

CONNECTION INFORMATION

LOCATION OF CONNECTORS

The CFA835 has three connectors on the back of the PCB: H1, USB, and FBSCAB. The H1 connector can be used for "logic level" serial interface and GPIO/ATX functionality. For "full swing" RS232 serial interface, the optional CFA-RS232 Serial Converter Board is mounted on H1.

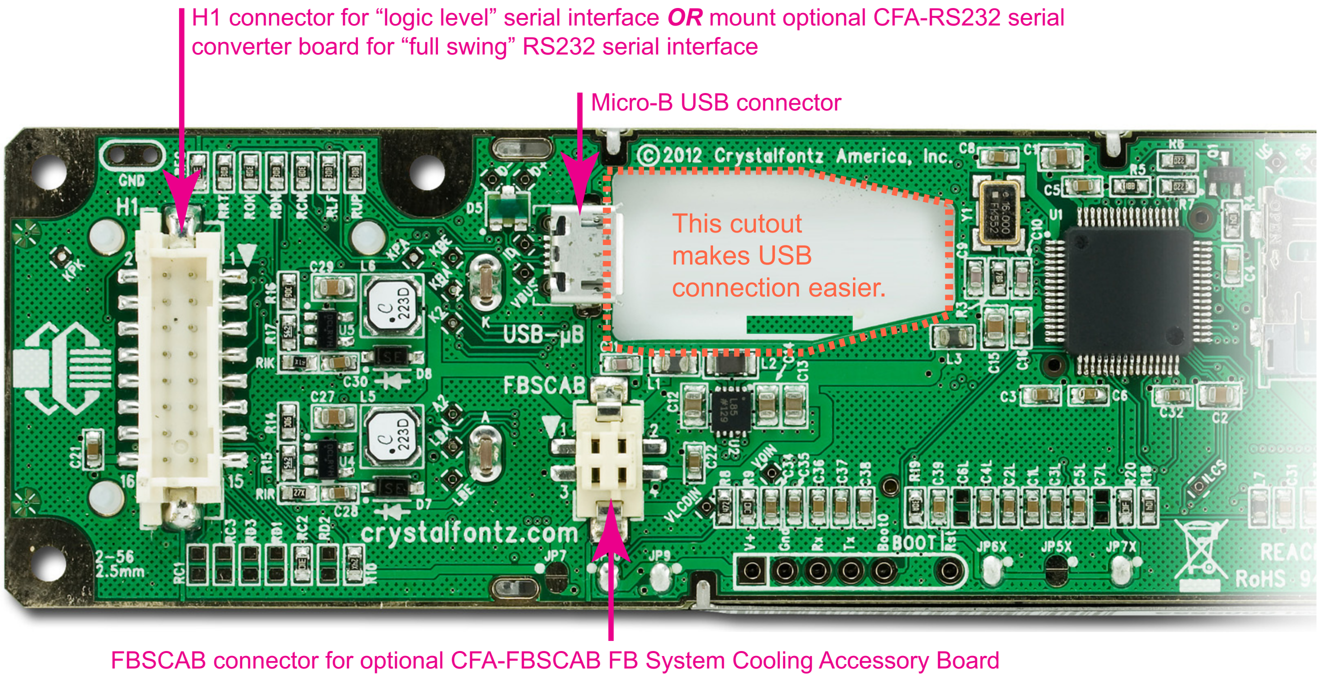

H1 connector for "logic level" serial interface ***OR*** mount optional CFA-RS232 serial converter board for "full swing" RS232 serial interface
Micro-B USB connector
This cutout makes USB connection easier.
FBSCAB connector for optional CFA-FBSCAB FB System Cooling Accessory Board

Figure 16. Location Of CFA835 Connectors

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

H1 CONNECTOR DETAILS

H1 Pin Assignments

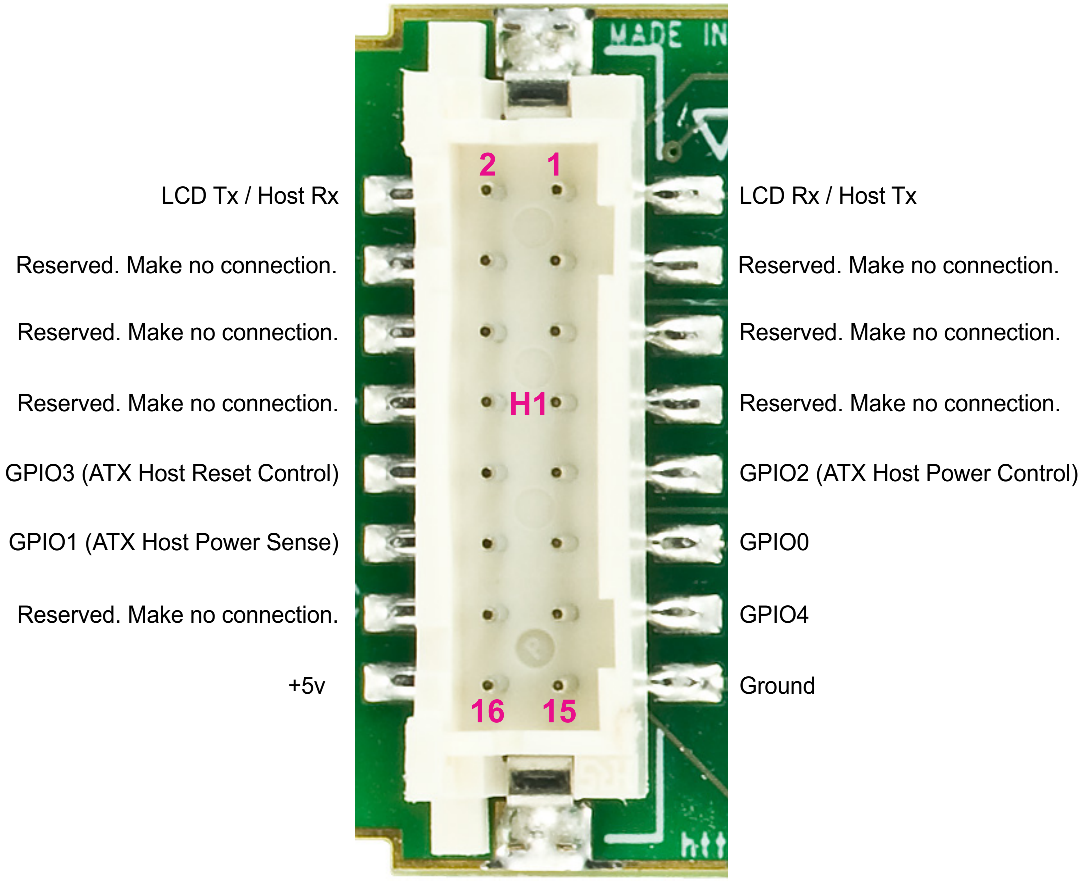

LCD Tx / Host Rx
Reserved. Make no connection.
Reserved. Make no connection.
Reserved. Make no connection.
GPIO3 (ATX Host Reset Control)
GPIO1 (ATX Host Power Sense)
Reserved. Make no connection.
+5v
2
1
H1
16
15
LCD Rx / Host Tx
Reserved. Make no connection.
Reserved. Make no connection.
Reserved. Make no connection.
GPIO2 (ATX Host Power Control)
GPIO0
GPIO4
Ground

Figure 17. Pin Assignments on CFA835's H1 Connector (Includes GPIOs)

Make Your Own H1 Cable

The following parts may be used to make your own cable to connect to the CFA835's H1 connector:

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

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

By using the micro USB 5-pin (F) B type connector, the CFA835 requires only one connection to the host for both data communications and power supply.

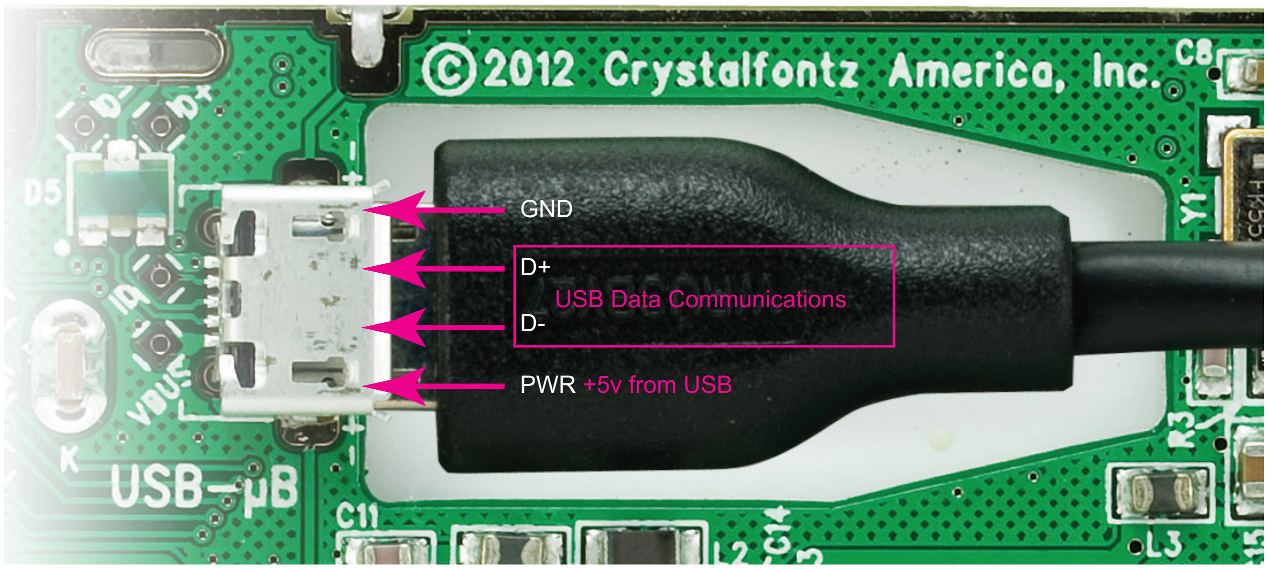
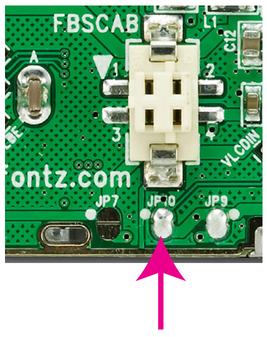


Figure 18. Micro-B USB Connection Pin Details

The micro-B USB connector and the cutout in the PCB keeps the CFA835 profile as thin as possible. You can connect the CFA835 to one host using a USB interface while at the same time using a serial interface to a second host.

Note: Keep the micro-B USB cable connector parallel to the CFA835 when plugging or unplugging the cable. Do not lift or pull up on the cable. Too much pressure may permanently damage micro-B USB connector.

If You Want To Use USB Interface While Supplying Power Through H1



JP10 on the CFA835 is closed by factory default. If you are going to use USB interface while supplying power through H1, you must **open JP10** to prevent back-powering the USB.

ATX POWER SUPPLY POWER AND CONTROL CONNECTIONS

ATX power supply control functionality allows the buttons on the CFA835 to replace the power and restart button on your system, simplifying front panel design.

For ATX: If You Add Optional CFA-FBSCABS (FB System Cooling Accessory Boards)
The CFA835+CFA-FBSCABs has no ATX functionality provided through the CFA-FBSCAB. However, ATX control is available using a WR-PWR-Y25 ATX power switch cable on the H1 connector of the CFA835.

For ATX: Do Not Change Dedicated GPIO Pins
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. See the note under command 28 (0x1C): ATX Functionality (Pg. 51) or command 34 (0x22): GPIO Pin Levels (Pg. 55),

GPIO[1] ATX Host Power Sense
Since the CFA835 must act differently depending on whether the host's power supply is on or off, you must also connect the host's “switched +5v” to GPIO[1]. This GPIO line functions as POWER SENSE. The POWER SENSE pin is configured as an input with a pull-down, 5kΩ nominal.

GPIO[2] ATX Host Power Control
The motherboard's power switch input is connected to GPIO[2]. This GPIO line functions as POWER CONTROL. The POWER CONTROL pin is configured as a high impedance input until the CFA835 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): ATX Functionality (Pg. 51).

GPIO[3] ATX Host Restart Control
The motherboard's restart switch input is connected to GPIO[3]. This GPIO line functions as RESTART. The RESTART pin is configured as a high-impedance input until the CFA835 wants to restart the host. Then it will change momentarily to low impedance output, driving either low or high depending on the setting of RESTART_INVERT. See command 28 (0x1C): ATX Functionality (Pg. 51). This connection is also used for the hardware watchdog.

ATX Power Supply & Control Connections	Pins on H1 Connector*
V_{SB} (+5v)	Pin 16
Ground	Pin 15
GPIO[1] ATX Host POWER SENSE	Pin 12
GPIO[2] ATX Host POWER CONTROL	Pin 9
GPIO[3] ATX Host RESTART CONTROL	Pin 10
**For “full swing” RS232 using the optional CFA-RS232 Serial Converter Board, the H1 pins are passed through to the CFA-RS232’s J1 connector.*	

Below is an illustration of how the optional WR-PWR-Y25 ATX power switch cable connects to the CFA835's connector H1 and your host's motherboard and ATX power supply

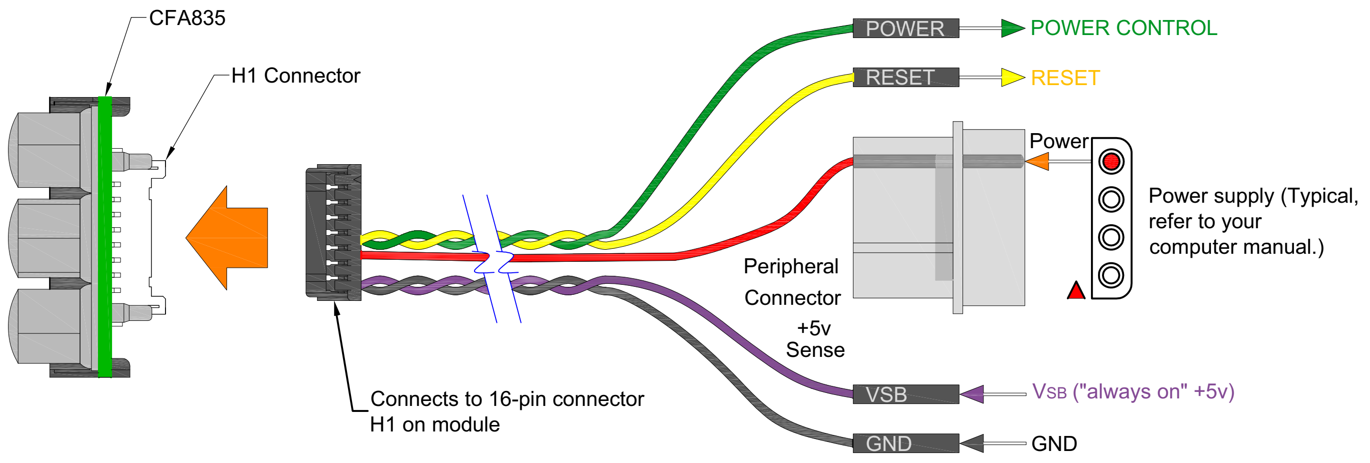


Figure 19. ATX Power Supply and Control Connections Using The WR-PWR-Y25 Cable

HOW TO SET ATX FUNCTIONALITY USING CFTEST

1. Download the cfTest application here: http://www.crystalfontz.com/software/CFTEST.html.
2. Connect the CFA835 to a Windows' based PC. You may want to connect the +5VSB and +5VSENSE so you will be able to see the CFA835 when it powers up.
3. Disable any applications that communicate with the CFA835 to free up the virtual COM port.
4. Launch cfTest. The application should automatically recognize the CFA835 and display it in the *Communications Port* dropdown list. If not, select your CFA835 from the dropdown list.
5. In the *Send Packet* section, select command 28 (0x1C): ATX Functionality from the dropdown list.
6. Type in the following value: "\240" into the *Data* field. The '\240" represents the bitmask value for data[0].
7. Click *Send Packet*.
8. Select command 4 (0x04): Store Current State As Boot State from *The PacketType* dropdown list.
9. Clear the *Data* text box.
10. Click *Send Packet*. This saves the current state set with ATX.

FIRMWARE

HOW TO IDENTIFY FIRMWARE REVISION NUMBER

Before you apply power to the CFA835, press the right arrow key on the keypad. Apply power, keeping the right arrow key depressed until the firmware revision displays. As long as the keypad is depressed, this information is displayed. When you release the right arrow key, the display clears after five seconds.

Or when coming out of restart, keep the right arrow key depressed until the firmware revision displays. As long as the keypad is depressed, this information is displayed. When you release the right arrow key, the display clears after five seconds.

An alternate method to identify revision number is by using command 1 (0x01): Get Module Information (Pg. 44).

POSSIBLE FUTURE FIRMWARE UPDATES

The CFA835 display modules are shipped with preinstalled firmware that performs the command functions described in this Data Sheet. We may make updates to the firmware in the future. Firmware updates are announced through our PCN (Part Change Notices).

Any updates to firmware will be available as free downloads under the Datasheets &F iles tab on the part number's web page.

Updated firmware is downloaded onto the CFA835 by copying the firmware file onto a microSD card which is then inserted in the CFA835's microSD card socket. A MS-DOS FAT-12/16/32 formatted microSD SDHC card must be used. Detailed instructions and description of firmware changes will be included in the firmware update package.

CREATE YOUR OWN FIRMWARE

The CFA835 uses a STMicroelectronics STM32F103R microcontroller.The microcontroller is preprogrammed with a bootloader that can load user created firmware into the microcontroller's flash memory from a microSD card. A MS-DOS FAT-12/16/32 formatted SDHC microSD card up to 4 GB must be used. You can also program the microcontroller by using a JTAG programmer interface.

Caution

If you load user created firmware, you will overwrite the Crystalfontz firmware. Functions for the Command Codes described in this Data Sheet will not work. There is no method to reinstall the supported firmware without returning the CFA835 to Crystalfontz. A reprogramming charge may apply.

Crystalfontz has no phone or email support for user code.

HOST COMMUNICATIONS

To quickly get up and running, download our free demonstration cfTest. cfTest includes all the commands needed to communicate with the CFA835 display module and showcase its functionality.

THROUGH USB

The easiest and most common way for the host software to access the USB is through the Crystalfontz virtual COM port (VCP) drivers. A link to VCP drivers download and installation instructions can be found on the Crystalfontz website. WHQL USB drivers are available under the Datasheets&Files tab for this product.Using these drivers makes it appear to the host system as if there is an additional serial port (the VCP) on the host system when the CFA835 is connected. When communicating over USB, the VCP settings are accepted for compatibility reasons. The virtual COM port settings such as baud rate (speed), stop bits, etc. are ignored as the communications occur as pure USB data.

THROUGH SERIAL

The CFA835 display modules are shipped with port settings 115200 baud, 8 data bits, no parity, 1 stop bit. Baud rate can be changed to 19200 or 9600 baud. See command 33 (0x21): Interface Options (Pg. 54).

MULTIPLE PORT COMMUNICATIONS

The CFA835 supports communication through two interfaces at the same time. Keypad report packets are sent to all available interfaces. All command reply packets are sent to the interface from which the command packet originated.

PACKET STRUCTURE

All communication between the CFA835 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 CFA835 and the host without the traditional problems that occur in a stream-based serial communication (such as having to send data in inefficient ASCII format, to “escape” certain “control characters”, or losing sync if a characters is corrupted, missing, or inserted).

Note

Reconciling packets is recommended rather than using delays when communicating with the CFA835. To reconcile your packets, please ensure that you have received the acknowledgment packet from the packet most recently sent before sending any additional packets to the CFA835. This practice will guarantee that you will not have any dropped packets or missed communication with the CFA835.

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

```
typedef struct
{
   unsigned char type;
   unsigned char data_length;
   unsigned char data[max_data_length];
   unsigned short CRC;
} COMMAND_PACKET;
```

All packets have the following structure:

```
<type><data_length><data><CRC>
```

`type` is one byte, and identifies the type and function of the packet:

```
TTcc cccc
|||| ||||--command, response, error or report code 0-63
||---------type:
             00 = normal command from host to CFA835
             01 = normal response from CFA835 to host
             10 = normal report from CFA835 to host (not in direct response to a command
                  from the host)
             11 = error response from CFA835 to host
```

`data_length` specifies the number of bytes that will follow in the data field. The valid range of `data_length` is 0 to 124.

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

`CRC` is a standard 16-bit CRC of all the bytes in the packet except the CRC itself. The CRC is sent LSB first. At the port, the CRC immediately follows the last used element of data `[]`. See APPENDIX B: FREE DEMONSTRATION AND OTHER SOFTWARE (Pg. 86) for several examples of how to calculate the CRC in different programming languages.

ABOUT HANDSHAKING

The nature of CFA835's packets makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for a corresponding acknowledge packet from the CFA835 before sending the next command packet. The CFA835 will respond to all packets within 250mS. The host software should report an error if a packet is not acknowledged within 250ms. This situation indicates a possible 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 CFA835 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 CFA835 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. 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

For your convenience, here is a list of command code links grouped by type. Below this list, commands are listed numerically, from 1 to 41.

COMMUNICATIONS
Command 0 (0x00): Ping Command (Pg. 44)
Command 1 (0x01): Get Module Information (Pg. 44)
Command 5 (0x05): Restart (Pg. 46) includes: Reload Boot Settings (Pg. 46) Restart Host (WR-PWR-Y25 ATX Power Switch Cable Required) (Pg. 46) Power Off Host (WR-PWR-Y25 ATX Power Switch Cable Required) (Pg. 46) CFA835 Restart (Pg. 46) CFA835 Restore Default Settings (Pg. 47)
Command 28 (0x1C): ATX Functionality (Pg. 51)
Command 29 (0x1D): Watchdog (Pg. 53)
Command 33 (0x21): Interface Options (Pg. 54)
Command 36 (0x24): Interface Bridge (Pg. 57)
DISPLAY / LCD
Command 6 (0x06): Clear Display (Pg. 47)
Command 9 (0x09): Special Character Bitmaps (Pg. 47)
Command 11 (0x0B): Display Cursor Position (Pg. 48)
Command 12 (0x0C): Cursor Style (Pg. 48)
Command 13 (0x0D): Contrast (Pg. 49)
Command 14 (0x0E): Display And Keypad Backlights (Pg. 49)
Command 31 (0x1F): Write Text To The Display (Pg. 53)
Command 32 (0x20): Read Text From The Display (Pg. 54)
Command 38 (0x26): Custom Fonts (Pg. 66) includes: Subcommand 0: Load Custom Font Files From MicroSD Card (Pg. 66) Subcommand 1: Print Custom Font To Display (Pg. 66)
Command 40 (0x32): Display Graphic Options (Pg. 69) includes: Subcommand 0: Graphic Options (Pg. 69) Subcommand 1: Buffer Flush (Pg. 69) Subcommand 2: Send Image Data To Display From Host (Pg. 69) Subcommand 3: Display Image File From MicroSD Card On CFA835 (Pg. 70) Subcommand 4: Save Screenshot to MicroSD File (Pg. 71) Subcommand 5: Pixel Data (Pg. 71) Subcommand 6: Draw a Line (Pg. 72) Subcommand 7: Draw a Rectangle (Pg. 72)

DISPLAY / LCD (Continued)
Subcommand 8: Draw a Circle (Pg. 73)
Command 41 (0x3A): Video Playback Control (Pg. 73) includes: Subcommand 0: Load A Video From MicroSD Card (Pg. 73) Subcommand 1: Video Control (Pg. 73)
GPIOS AND KEYPAD
Command 14 (0x0E): Display And Keypad Backlights (Pg. 49)
Command 23 (0x17): Keypad Reporting (Pg. 50)
Command 24 (0x18): Read Keypad, Polled Mode (Pg. 51)
Command 28 (0x1C): ATX Functionality (Pg. 51) includes: Function 1: KEYPAD_RESTART (Pg. 51) Function 2: KEYPAD_POWER_ON (Pg. 52) Function 3: KEYPAD_POWER_OFF (Pg. 52)
Command 34 (0x22): GPIO Pin Levels (Pg. 55)
Command 37, Subcommand 5: GPIO Pin Levels (Pg. 63)
FAN AND TEMPERATURE CONTROL / MONITORING
Command 34, Subcommand 2: Read Fan Tachometers (Pg. 61)
Command 37 (0x25): CFA-FBSCAB (Pg. 58) includes: Subcommand 0: Read CFA-FBSCAB Information (Pg. 59) Subcommand 1: Fan Settings (Pg. 60) includes Set Fan Power, Fail-Safe And Glitch information Subcommand 2: Read Fan Tachometers (Pg. 61) Subcommand 3: Read DOW Device Information (Pg. 62) Subcommand 4: Read WR-DOW-Y17 Temperature (Pg. 62) Subcommand 5: GPIO Pin Levels (Pg. 63) Subcommand 6: Reset And Search (Pg. 65)
MICROSD OPERATIONS
Command 38 (0x26): Custom Fonts (Pg. 66) includes: Subcommand 0: Load Custom Font Files From MicroSD Card (Pg. 66) Subcommand 1: Print Custom Font To Display (Pg. 66)
Command 39 (0x27): MicroSD File Operations (Pg. 67) includes: Subcommand 0: Open/Close MicroSD File (Pg. 67) Subcommand 1: Position Seek (Pg. 67) Subcommand 2: Read File Data (Pg. 68) Subcommand 3: Write File Data (Pg. 68) Subcommand 4: Delete A File (Pg. 68)
Command 40 (0x32): Display Graphic Options (Pg. 69) includes: Subcommand 3: Display Image File From MicroSD Card On CFA835 (Pg. 70) Subcommand 4: Save Screenshot to MicroSD File (Pg. 71)

MICROSD OPERATIONS (Continued)
Command 41, Subcommand 0: Load A Video From MicroSD Card (Pg. 73)
EEPROM OPERATIONS
Command 2 (0x02): Write User Flash Area (Pg. 44)
Command 3 (0x03): Read User Flash Area (Pg. 45)
Command 4 (0x04): Store Current State As Boot State (Pg. 45)

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.

You can experiment with these command by using our free download of cfTest.

0 (0x00): Ping Command

Used to verify communication with the CFA835. The CFA835 will echo the Ping Command to the host.

Command Packet:
```
type: 0x00 = 0₁₀
data_length: 0 to 124
data[]: any arbitrary data
```

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

1 (0x01): Get Module Information

The CFA835 will return the hardware and firmware revision or serial number to the host.

Command Packet:
```
type: 0x01 = 1₁₀
data_length: 0 or 1
data[0]: module information to return (optional)
   0 = (optional) hardware and firmware version
   1 = CFA835 module serial number
```

Response Packet (data_length=0 or data[0]=0):
```
type: 0x40 | 0x01 = 0x41 = 65₁₀
data_length: 16
data[]: "CFA835:hX.X,fY.Y"
```

Response Packet (data[0]=1):
```
type: 0x40 | 0x01 = 0x41 = 65₁₀
data_length: 17
data[]: "1134835TMI0000001"
```

2 (0x02): Write User Flash Area

The CFA835 reserves 124 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.

Command Packet:

```
type: 0x02 = 2₁₀
data_length: 1 to 124
data[]: arbitrary user data to be stored in nonvolatile memory
```

Response Packet:

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

3 (0x03): Read User Flash Area

Command Packet:

```
type: 0x03 = 3₁₀
data_length: 1
data[0]: number of bytes of data to be returned (1 to 124)
```

Response Packet:

```
type: 0x40 | 0x03 = 0x43 = 67₁₀
data_length: number of bytes specified in command
data[]: user data recalled from the CFA835's flash memory
```

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

The CFA835 loads its power-up configuration from nonvolatile memory when power is applied. The CFA835 is configured at the factory to display a bootscreen when power is applied. This command can be used to customize the bootscreen, as well as the following items:

- Characters shown on display, which are affected by:
  - Command 6 (0x06): Clear Display (Pg. 47).
  - Command 31 (0x1F): Write Text To The Display (Pg. 53).
  - Command 38, Subcommand 1: Print Custom Font To Display (Pg. 66).
  - Command 9 (0x09): Special Character Bitmaps (Pg. 47)
  - Command 11 (0x0B): Display Cursor Position (Pg. 48)).
- Command 12 (0x0C): Cursor Style (Pg. 48).
- Command 13 (0x0D): Contrast (Pg. 49)).
- Command 14 (0x0E): Display And Keypad Backlights (Pg. 49)).
- Command 23 (0x17): Keypad Reporting (Pg. 50)).
- Command 28 (0x1C): ATX Functionality (Pg. 51)).
- Command 33 (0x21): Interface Options (Pg. 54).
- Command 34 (0x22): GPIO Pin Levels (Pg. 55).
- Command 37 (0x25): CFA-FBSCAB (Pg. 58).

All FBSCAB settings are also saved, but are saved in the nonvolatile memory on the CFA-FBSCAB module itself.

Watchdog settings cannot be saved. The host software should enable these items once the system is initialized and ready to receive the data.

Command Packet:

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

Response Packet:

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

5 (0x05): Restart

Depending on the parameters you provide, this command provides five restart options: (1) Reload Boot Settings, (2) Restart Host, (3) Power Off Host, (4) CFA835 Restart, or (5) CFA835 Restore Default Settings.

When using both the USB and a serial interface simultaneously (logic level or “full swing” RS232 with mounted optional CFA-RS232 Serial Converter Board), you may notice that performing a restart from one interface will impact the other interface. The ATX related options to power down or restart the host using the CFA835 may be useful in many situations.

The ATX related options to power down or restart the host using the CFA835 may be useful in many situations. These options rely on the GPIO pins used for ATX control to be configured in their default drive modes in order for the ATX functions to work correctly. See command 28 (0x1C): ATX Functionality (Pg. 51).

Reload Boot Settings

Reloads the settings stored using command 4 (0x04): Store Current State As Boot State (Pg. 45). Reloading the boot settings may be useful when testing the boot configuration. It may also be useful to re-enumerate the devices on the 1-wire bus.

The CFA835 will return the acknowledge packet immediately, then reload its settings.

Command Packet:

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

Restart Host (WR-PWR-Y25 ATX Power Switch Cable Required)

This option instructs the CFA835 to restart the host via the WR-PWR-Y25 ATX power switch cable and then restart itself. This command will also restart any attached CFA-FBSCAB modules to the state saved in their nonvolatile memory.

The CFA835 will return the acknowledge packet before carrying out the actions.

Command Packet:

```
type = 0x05 = 5₁₀
data_length: 3
data[0]: 12
data[1]: 28
data[2]: 97
```

Power Off Host (WR-PWR-Y25 ATX Power Switch Cable Required)

This option instructs the CFA835 to power down the host via the WR-PWR-Y25 ATX power switch cable and then restart itself. This command will also restart any attached CFA-FBSCAB modules to the state saved in their nonvolatile memory.

Command Packet:

```
type = 0x05 = 5₁₀
data_length: 3
data[0]:   3
data[1]: 11
data[2]: 95
```

CFA835 Restart

Performs a software restart of the CFA835 module. If used with USB interface, this command will cause the CFA835 module to disconnect and then reconnect (re-enumerate). This command will also restart any attached FBSCAB modules to the state saved in their nonvolatile memory.

The CFA835 will return the acknowledge packet immediately, then restart itself. The CFA835 may not respond to new command packets for up to 3 seconds.

Command Packet:

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

CFA835 Restore Default Settings

Restarts the system boot state to that of a factory CFA835 and then performs a CFA835 restart. If used as a USB device, this command will cause the module to disconnect and then reconnect (re-enumerate). This command will also restart any attached CFA-FBSCAB to the state saved in their nonvolatile memory.

This option does not affect the user flash values set by command 2 (0x02): Write User Flash Area (Pg. 44).

The CFA835 will return the acknowledge packet immediately, then restart itself. The CFA835 may not respond to new command packets for up to 3 seconds.

Command Packet:

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

Response Packet For All Five Restart Options:

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

6 (0x06): Clear Display

Clears the CFA835's display, graphical display buffer, and character row/column buffer. It also moves the cursor to the left-most column of the top line, and stops any videos that are being played from an microSD card. See command 41 (0x3A): Video Playback Control (Pg. 73).

Command Packet:

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

Response Packet:

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

9 (0x09): Special Character Bitmaps

Sets the bitmap for one of the special characters in the CGRAM to be used with command 31 (0x1F): Write Text To The Display (Pg. 53). *Note:* special characters are not supported when using custom fonts. See command 38, Subcommand 0: Load Custom Font Files From MicroSD Card (Pg. 66) for details.

Command Packet (Read):

```
type: 0x09 = 9₁₀
data_length: 1
data[0]: index of special character to read, 0-7 are valid
```

Response Packet (Read):

```
type: 0x40 | 0x09 = 0x49 = 73₁₀
data_length: 9
data[0]: index of special character data
data[1-8]: bitmap of this special character
```

Command Packet (Write):

```
type: 0x09 = 9₁₀
data_length: 9
data[0]: index of special character that you would like to modify, 0-7 are valid
data[1-8]: bitmap of this special character
```

Response Packet (Write):

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

11 (0x0B): Display Cursor Position

This command allows the cursor to be placed at the desired location on the CFA835's display. If you want the cursor to be visible, you may also need to send command 12 (0x0C): Cursor Style (Pg. 48). The current cursor location can also be read using this command.

Command Packet (Read):

```
type: 0x0B = 11₁₀
data_length: 0
```

Response Packet (Read):

```
type = 0x40 | 0x0B = 0x4B = 75₁₀
data_length: 2
data[0]: column
data[1]: row
```

Command Packet (Write):

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

Response Packet (Write):

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

12 (0x0C): Cursor Style

This command allows you to either hide the cursor or select among four hardware generated cursor options. You can also read the current cursor style using this command.

Cursor Styles:

```
0 = no cursor
1 = blinking block cursor
2 = underscore cursor
3 = blinking block plus underscore
4 = inverting, blinking block
```

Command Packet (Read):

```
type = 0x0C = 12₁₀
data_length: 0
```

Response Packet (Read):

```
type = 0x40 | 0x0C = 0x4C = 76₁₀
data_length: 1
data[0]: cursor style
```

Command Packet (Write):

```
type = 0x0C = 12₁₀
data_length: 1
data[0]: cursor style
```

Response Packet (Write):

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

13 (0x0D): Contrast

This command sets the contrast of the display. This command can also be used to read the current display contrast.

Command Packet (Read):

```
type = 0x0D = 13₁₀
data_length: 0
```

Response Packet (Read):

```
type = 0x40 | 0x0D = 0x4D = 77₁₀
data_length: 1
data[0]: contrast setting (0-255 valid)
```

Command Packet (Write):

```
type = 0x0D = 13₁₀
data_length: 1
data[0]: contrast setting (0-255 valid)
     0-111 = very light
       112 = light
       127 = about right
       168 = dark
   169-255 = very dark (may be useful at cold temperatures)
```

Response Packet (Write):

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

14 (0x0E): Display And Keypad Backlights

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

If two bytes are supplied, the display is set to the brightness of the first byte, the keypad is set to the brightness of the second byte. This command can also be used to read the current brightness levels.

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

Command Packet (Read):

```
type: 0x0E = 14₁₀
data_length: 0
```

Response Packet (Read):

```
type: 0x40 | 0x0E = 0x4E = 78₁₀
data_length: 2
data[0]: current display brightness (0-100)
data[1]: current keypad brightness (0-100)
```

Command Packet (Write):

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

Response Packet (Write):

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

23 (0x17): Keypad Reporting

By default, the CFA835 reports any key event to the host. This command allows the key events to be enabled or disabled on an individual basis. This command can also be used to read the current key reporting masks.

Keypad Bitmasks:

```
bit0 - up key
bit1 - enter key
bit2 - cancel key
bit3 - left key
bit4 - right key
bit5 - down key
```

Command Packet (Read):

```
type = 0x17 = 23₁₀
data_length: 0
```

Response Packet (Read):

```
type = 0x40 | 0x17 = 0x57 = 87₁₀
data_length = 2
data[0]: current keypad press mask
data[1]: current keypad release mask
```

Command Packet (Write):

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

Response Packet (Write):

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

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the CFA835 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): Keypad Reporting (Pg. 50). 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.

Keypad Bitmasks:

```
bit0 - up key
bit1 - enter key
bit2 - cancel key
bit3 - left key
bit4 - right key
bit5 - down key
```

Command Packet:

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

Response Packet:

```
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.
```

28 (0x1C): ATX Functionality

The combination of the CFA835 with ATX can be used to replace the function of the power and restart switches in a standard ATX-compatible system.

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. Please read ATX Power Supply Power And Control Connections (Pg. 37) followed by How to Set ATX Functionality Using cfTest (Pg. 38).

The RESTART (GPIO[3]) and POWER CONTROL (GPIO[2]) lines on the CFA835 with ATX are normally high-impedance. Electrically, they appear to be disconnected or floating. When the CFA835 with ATX asserts the RESTART or POWER CONTROL lines, they are momentarily driven high or low (as determined by the RESTART_INVERT and POWER_INVERT bits, detailed below). To end the power or restart pulse, the CFA835 with ATX changes the lines back to high-impedance.

This command can also be used to read the current ATX power switch function settings.

FOUR FUNCTIONS ENABLED BY COMMAND 28

Function 1: KEYPAD_RESTART

If POWER-ON SENSE (GPIO[1]) is high, holding the green check key for 4 seconds will pulse RESTART (GPIO[3]) pin for 1 second. During the 1-second pulse, the CFA835 will show RESTART, and then the CFA835 will restart itself, showing its boot state as if it had just powered on. Once the pulse has finished, the CFA835 will not respond to any commands until after it has restart the host and itself.

Function 2: KEYPAD_POWER_ON

If POWER-ON SENSE (GPIO[1]) is low, pressing the green check key for 0.25 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified in data[1]. During this time the CFA835 will show POWER ON, then the CFA835 will restart itself.

Function 3: KEYPAD_POWER_OFF

If POWER-ON SENSE (GPIO[1]) is high, holding the red X key for 4 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified in data[1] If the user continues to hold the power key down, then the CFA835 will continue to drive the line for a maximum of 5 additional seconds. During this time the CFA835 will show POWER OFF.

Function 4: MODULE_MIMIC_HOST_POWER

If MODULE_MIMIC _HOST_POWER is set, the CFA835 will blank its display and turn off its backlight to simulate its power being off any time POWER-ON SENSE (GPIO[1]) is low. The CFA835 will still be active (since it is powered by V_{SB}), monitoring the keypad for a power-on keystroke. If +12v remains active (which would not be expected, since the host is “off”), the fans will remain on at their previous settings. Once POWER-ON SENSE (GPIO[1]) goes high, the CFA835 will restart as if power had just been applied to it.

ATX Bitmasks:

```
bit0 - AUTO_POLARITY: Automatically detects polarity for restart and power (recommended)
bit1 - RESTART_INVERT: Restart pin drives high instead of low (ignored if AUTO_POLARITY is
                      set)
bit2 - POWER_INVERT: Power pin drives high instead of low (ignored if AUTO_POLARITY is
                      set)
bit3 - LEDS_MIMIC_HOST_POWER: Turn off the LEDs also if the host is off (ignored if
                               MODULE_MIMIC_HOST_POWER is not set)
bit4 - MODULE_MIMIC_HOST_POWER: Turn off the display if the Host is off
bit5 - KEYPAD_RESTART
bit6 - KEYPAD_POWER_ON
bit7 - KEYPAD_POWER_OFF
```

Command Packet (Read):

```
type = 0x1C = 28₁₀
data_length = 0
```

Response Packet (Read):

```
type = 0x40 | 0x1C = 0x5C = 92₁₀
data_length: 2
data[0]: bitmask of enabled functions
data[1]: length of power on & off pulses in 1/32 second increments
```

Command Packet (Write):

```
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 second
           2 = 1/16 second
          16 = 1/2  second
         ...
         254 = 7.9  second
         255 = Hold until power sense change or 8 second, whichever is shorter (default)
```

Response Packet (Write):

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

29 (0x1D): 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 CFA835 with ATX (CFA835+WR-PWR-Y25 ATX power switch cable).

If the system monitor program fails to update the watchdog timer, the CFA835 with ATX will reset the host system and then itself as if command 5 (0x05): Restart (Pg. 46) was issued.

If the command is not reissued within the specified number of seconds, then the CFA835 with ATX will restart the host system (see command 28 (0x1C): ATX Functionality (Pg. 51) for details) and restart itself as if command 5 (0x05): Restart (Pg. 46) restart function was issued. Since the watchdog is off by default when it powers up, CFA835 with ATX will not issue another host restart until the host has once again enabled the watchdog.

To turn the watchdog off once it has been enabled, set data [0] = 0.

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. See the note under command 28 (0x1C): ATX Functionality (Pg. 51) or command 34 (0x22): GPIO Pin Levels (Pg. 55),

Command Packet (Read):

```
type = 0x1D = 29₁₀
data_length: 0
```

Response Packet (Read):

```
type = 0x40 | 0x1D = 0x5D = 93₁₀
data_length: 1
data[0]: watchdog timeout in seconds (0=disabled)
```

Command Packet (Write):

```
type = 0x1D = 29₁₀
data_length: 1
data[0]: enable counter
         0 = watchdog is disabled
         1-255 = timeout in seconds
```

Response Packet (Write):

```
type = 0x40 | 0x1D = 0x5D = 93₁₀
data_length: 0
```

31 (0x1F): Write Text To The Display

This command allows text and special characters to be placed at any position on the display. The text is displayed in the default font, unless overridden by command 38, Subcommand 0: Load Custom Font Files From MicroSD Card (Pg. 66). See default font standard set of characters at CHARACTER GENERATOR ROM (CGROM) FOR STANDARD SET OF CHARACTERS (Pg. 76).

Command Packet:

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

Response Packet:

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

32 (0x20): Read Text From The Display

This command allows the host to read back text that is displayed on the CFA835.

Note: This command will only read text displayed by command 31 (0x1F): Write Text To The Display (Pg. 53) It cannot be used to read text written by custom font command 38, Subcommand 0: Load Custom Font Files From MicroSD Card (Pg. 66).

Command Packet:

```
type = 0x20 = 32₁₀
data_length = 3
data[0]: column position (x = 0 to 19)
data[1]: row position (y = 0 to 3)
data[2]: length of text to read in characters (1 - 20)
```

Response Packet:

```
type = 0x40 | 0x20 = 0x60 = 96₁₀
data_length = 1 to 20
data[] = read text
```

33 (0x21): Interface Options

The CFA835 has a logic level serial interface located on pins 1 (Tx) and 2 (Rx) of the H1 connector. For “full swing” RS232 using the optional CFA-RS232 Serial Converter Board, the H1 pins are passed through to the CFA-RS232’s J1 connector.

After sending this command, the host should wait for a positive acknowledgment from the CFA835 at the old baud rate. The host can then begin communicating at the new baud rate.

The baud rate must be saved by command 4 (0x04): Store Current State As Boot State (Pg. 45) if you want the CFA835 to power-up/restart using the new baud rate. The factory default baud rate is 115200.

This command is also used to read the current interface options.

Baud Rate:

```
0 = 19200
1 = 115200
2 = 9600
```

Option Flags:

```
bit0 = enable interface
       note: USB interface cannot be fully disabled
bit1 = command interpreter enabled.
       note: CFA835 will accept packets on this interface. interface must be enabled for
             interpreter on an interface to be enabled. normal reply packets are only sent
```

```
                to the originating interface. the following options are only available if the
                interpreter is enabled
bit2 = CFA835 will transmit report packets on this interface (reports 128)
bit3 = CFA835 will transmit errors from commands received on this interface
bit4 = CFA835 will transmit errors from commands received on either interface
bit5 = CFA835 will transmit extended error information
```

Command Packet (Read):

```
type = 0x21 = 33₁₀
data_length = 1
data[0]: interface
         0 = serial
         1 = USB
```

Response Packet (Read):

```
type = 0x40 | 0x21 = 0x61 = 97₁₀

SERIAL INTERFACE:
data_length: 3
data[0]: 0 (serial)
data[1]: option flags
data[2]: baud rate

USB INTERFACE:
data_length: 2
data[0]: 1 (USB)
data[1]: option flags
```

Command Packet (Write):

```
type = 0x21 = 33₁₀
data_length = 2 or 3
data[0]: interface
         0 = serial
         1 = USB
data[1]: option flags
```

If `data[0]: interface` is `0 = serial`

```
data_length = 3
data[2] = baud rate
```

If `data[0]: interface` is `1 = USB`

```
data_length = 2
No extra options.
```

Response Packet (Write):

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

34 (0x22): GPIO Pin Levels

The CFA835 has five pins for user-definable general purpose input / output (GPIO). These pins are shared with the ATX functions. Be careful when you configure the GPIO if you want to use the ATX at the same time.

The architecture of the CFA835 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.

The default GPIO mode uses PWM and a suitable current limiting resistor to control the LEDs on the front of the module. They can be turned on and 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 CFA835 continuously polls the GPIOs as inputs at 50 Hz.The present level can be queried by the host software at a lower rate. The CFA835 also keeps track of whether there were rising or falling edges since the last host query (subject to the resolution of the 50 Hz sampling). This means that the host is not forced to poll quickly in order to detect short events. The algorithm used by the CFA835 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 40kΩ.Typical GPIO current limits when sinking or sourcing all five GPIO pins simultaneously are 8 mA. If you need more information, see the ST-Micro STM32F103 datasheet.

Note

The GPIO pins may also be used for ATX control through the H1 connector using the WR-PWR-Y25 ATX power switch cable. By factory default, the GPIO output setting, function, and drive mode are set correctly to enable operation of the ATX function. **The GPIO output setting, function, and drive mode must be set to the correct values in order for the ATX function to function properly.** Our free demonstration software cfTest may be used to easily check and restart the GPIO configuration to the default state so the ATX and DOW WR-DOW-Y17 temperature sensor cable functions will work.

Command Packet (Read):

```
type: 0x22 = 34₁₀
data_length: 1
data[0]: index of GPIO/GPO to read (0-12 valid)
```

Response Packet (Read):

```
type = 0x40 | 0x22 = 0x62 = 98₁₀
data_length = 4
data[0]: index of GPIO/GPO
data[1]: pin output state
data[2]: pin PWM output value
data[3]: pin function select and drive mode
```

Command Packet (Write):

```
type: 0x22 = 34₁₀
data_length:
            2 bytes to change value only
            3 bytes to change value and configure function and drive mode

data[0]: index of GPIO/GPO to modify (0-12 valid)
         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 Restart Control)
         4  = GPIO[4]: H1, pin 13
         5  = GPO[ 5]: LED 3 (bottom) green die
         6  = GPO[ 6]: LED 3 (bottom) red die
         7  = GPO[ 7]: LED 2 green die
         8  = GPO[ 8]: LED 2 red die
         9  = GPO[ 9]: LED 1 green die
         10 = GPO[10]: LED 1 red die
         11 = GPO[11]: LED 0 (top) green die
         12 = GPO[12]: LED 0 (top) red die

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

data[2] = pin function select and drive mode (optional, 0-15 valid except for 6 and 14)
     0    only meaningful for GPIOs (index 0-4). GPOs (index of 5-12) will ignore

  ---- FDDD
  ||||  |||-- DDD = drive mode (based on output state of 1 or 0)
  ||||  |     =======================================================
  ||||  |     000: 1=strong drive up, 0=resistive pull down
  ||||  |     001: 1=strong drive up, 0=strong drive down
  ||||  |     010: hi-Z, use for input
  ||||  |     011: 1=resistive pull up, 0=strong drive down
  ||||  |     100: 1=strong drive up, 0=hi-z
  ||||  |     101: 1=strong drive up, 0=strong drive down
  ||||  |     110: reserved, do not use -- error returned
  ||||  |     111: 1=hi-Z,0=strong drive down
  ||||  |
  ||||  |----- F = function (only valid for GPIOs, index of 0-4)
  ||||        =======================================================
  ||||        0: port unused for GPIO. it will take on the default
  ||||           function such as ATX, DOW (on CFA-FBSCABs), 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
```

Response Packet (Write):

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

36 (0x24): Interface Bridge

Interfaces:

```
0 = serial
1 = USB
```

The CFA835 has two interfaces: USB and a serial interface (logic level or “full swing” RS232 with mounted optional CFA-RS232).

By default, all interfaces on the CFA835 have the command interpreter enabled and are used by the host (or hosts) to send/receive command packets to and from the CFA835. If the command interpreter is disabled for an interface using command 33 (0x21): Interface Options (Pg. 54), that interface can be used to forward and receive raw data using this command.

For example, a host connected to the CFA835's USB interface could send raw data to the serial interface buffer. Incoming raw data on the serial interface is buffered and can be read from the buffer using the USB interface.

Note: This command will return an error if the interface being written to or read from has the command interpreter enabled.

Serial Interface

If the command interpreter is turned off, incoming bytes will be buffered in a circular buffer. If the buffer is allowed to wrap, it will overwrite the oldest data first. If the circular buffer does wrap, the next write/read command response will have the buffer overflow flag set. `data[1]` is treated as a timeout and the CFA835 will wait this long for the specified amount of data before aborting and throwing an error.

USB Interface

Because the USB to host interface has flow control, if the CFA835's incoming USB data buffer becomes full, the CFA835 will request the host not to send any more data. The overflow flag will never be set.

Command Packet:

```
type: 0x24 = 36₁₀
data_length: 4 + write data length
data[0]: interface
data[1]: delay/timeout
         0 = no delay/timeout, only return data that is already in the buffer
         1 to 50 = time in milliseconds / 10 (up to a value of 500mS)
data[2]: clear receive buffer options
         0x0 = do not clear
         0x1 = clear before read
         0x2 = clear after read
         0x3 = clear before and after
data[3]: requested read bytes
data[4-123]: data to be written to specified interface
```

Response Packet:
If there are less bytes available in the circular buffer than are requested, a smaller amount of data may be returned, as indicated by the read data length.

```
type: 0x40 | 0x24 = 0x64 = 100₁₀
data_length: 2 + read data length
data[0]: interface
data[1]: interface buffer status flags
         bit 0 = buffer overflow
         bit 1 = more data is available
data[2-123]: data read from interface buffer
```

37 (0x25): CFA-FBSCAB

The CFA835 supports fans, temperature sensors, and additional GPIOs through the addition of one or more CFA-FBSCABs. This command group contains all of the subcommands necessary to interact with the attached CFA-FBSCABs including reading and writing from the CFA-FBSCAB's fans, temperature sensors, and GPIO pins. As many as 32 CFA-FBSCABs can be attached by daisy-chaining them with WR-EXT-Y37 communication cables.

Fan Power Definition
Percentage value 0 to 100 PWM fan power.

Fail-Safe Definition

The combination of the CFA835 + one or more CFA-FBSCABs 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 is 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 CFA835’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.

Fail-Safe Bitmask Definitions

```
bit0 - fan_1
bit1 - fan_2
bit2 - fan_3
bit3 - fan_4
```

Glitch Definition

The CFA835 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. However, 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 CFA835 will recognize transitions on the tachometer line, The delay is specified in counts, each count being nominally 552.5 μS 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 fan tachometer monitoring slightly more intrusive at low power settings. Also, the higher values will increase the lowest speed that a fan with tachometer reporting enabled will “seek” at "0%" power setting.

Subcommand 0: Read CFA-FBSCAB Information

This subcommand returns the quantity of CFA-FBSCABs detected by the CFA835 or the serial number of a specified CFA-FBSCAB.

Command Packet (Query Number Of CFA-FBSCABs):

type: 0x25 = 37_{10}
```
data_length: 1
data[0]: 0 (read FBSCAB information)
```

Response Packet (Query Number Of CFA-FBSCABs):

```
type: 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 2
data[0]: 0 (read FBSCAB information)
data[1]: number of attached FBSCABs
```

Command Packet (Query CFA-FBSCAB Serial Number):

```
type: 0x25 = 37₁₀
data_length: 2
data[0]: 0 (Read FBSCAB Information)
data[1]: FBSCAB index
```

Response Packet (Query CFA-FBSCAB Serial Number):

```
type: 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 18
data[0]: 0 (read FBSCAB Information)
data[1]: index of queried FBSCAB
data[2-18]: serial number of specified FBSCAB module (text)
```

Subcommand 1: Fan Settings

This command will configure or read the power settings for the fan connectors on the specified CFA-FBSCAB module.

Command Packet (Set Fan Power):

```
type = 0x25 = 37₁₀
data_length: 6
data[0]: 1 (Set/Read FBSCAB Fan Settings)
data[1]: FBSCAB module index
data[2]: power level for FAN 1 (0-100 valid)
data[3]: power level for FAN 2 (0-100 valid)
data[4]: power level for FAN 3 (0-100 valid)
data[5]: power level for FAN 4 (0-100 valid)
```

Response Packet (Set Fan Power):

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 1
data[0]: 1 (Set/Read FBSCAB Fan Settings)
```

Command Packet (Set Fan Power and Fail-Safe):

```
type = 0x25 = 37₁₀
data_length: 8
data[0]: 1 (Set/Read FBSCAB Fan Settings)
data[1]: FBSCAB module index
data[2]: power level for FAN 1 (0-100 valid)
data[3]: power level for FAN 2 (0-100 valid)
data[4]: power level for FAN 3 (0-100 valid)
data[5]: power level for FAN 4 (0-100 valid)
data[6]: fail-safe enabled for these fans’ bitmask
data[7]: fan power update must happen within this many 1/8 second periods
```

Response Packet (Set Fan Power and Fail-Safe):

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 1
data[0]: 1 (Set/Read FBSCAB Fan Settings)
```

Command Packet (Set Fan Power, Fail-Safe and Glitch):

```
type = 0x25 = 37₁₀
data_length: 12
data[0]: 1 (Set/Read FBSCAB Fan Settings)
data[1]: FBSCAB module index
data[2]: power level for FAN 1 (0-100 valid)
data[3]: power level for FAN 2 (0-100 valid)
data[4]: power level for FAN 3 (0-100 valid)
data[5]: power level for FAN 4 (0-100 valid)
data[6]: fail-safe enabled for these fans bitmask
data[7]: fan power update must happen within this many 1/8 second periods
data[8]: glitch delay for FAN 1 (1-100 valid)
data[9]: glitch delay for FAN 2 (1-100 valid)
data[10]: glitch delay for FAN 3 (1-100 valid)
data[11]: glitch delay for FAN 4 (1-100 valid)
```

Response Packet (Set Fan Power, Fail-Safe and Glitch):

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 1
data[0]: 1 (Set/Read FBSCAB Fan Settings)
```

Command Packet (Read Fan Settings):

```
type = 0x25 = 37₁₀
data_length: 2
data[0]: 1 (Set/Read FBSCAB Fan Settings)
data[1]: FBSCAB module index
```

Response Packet (Read Fan Settings):

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 12
data[0]: 1 (Set/Read FBSCAB Fan Settings)
data[1]: FBSCAB module index
data[2]: power level for FAN 1
data[3]: power level for FAN 2
data[4]: power level for FAN 3
data[5]: power level for FAN 4
data[6]: fail-safe enabled for these fans bitmask
data[7]: fan power update 1/8 second periods
data[8]: glitch delay for FAN 1
data[9]: glitch delay for FAN 2
data[10]: glitch delay for FAN 3
data[11]: glitch delay for FAN 4
```

Subcommand 2: Read Fan Tachometers

This command will read the last fan tachometer's information from the specified CFA-FBSCAB module.

Note: This command must be executed every 60 seconds or less to read fan speed information from a CFA-FBSCAB module. If the command is not re-executed within 60 seconds, fan speed readings will be disabled by the CFA835 (to reduce fan noise) until the next "Read Fan Tachometers" subcommand is issued.

See Sample Code For RPM Calculation Information (Pg. 89).

Command Packet:

```
type: 0x25 = 37₁₀
data_length: 3
data[0]: 2 (read fan tachometer speed)
data[1]: FBSCAB module index
```

Response Packet:

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 14
data[0]: 2 (read fan tachometer speed)
data[1]: FBSCAB module index
data[2]:  fan 1 number of fan tach cycles
data[3]:  fan 1 LSB of fan timer ticks
data[4]:  fan 1 MSB of fan timer ticks
data[5]:  fan 2 number of fan tach cycles
data[6]:  fan 2 LSB of fan timer ticks
data[7]:  fan 2 MSB of fan timer ticks
data[8]:  fan 3 number of fan tach cycles
data[9]:  fan 3 LSB of fan timer ticks
data[10]: fan 3 MSB of fan timer ticks
data[11]: fan 4 number of fan tach cycles
data[12]: fan 4 LSB of fan timer ticks
data[13]: fan 4 MSB of fan timer ticks
```

Subcommand 3: Read DOW Device Information

This command returns the ROM ID of the specified DOW device attached to the specified CFA-FBSCAB module. This is used to confirm the attached device is a WR-DOW-Y17 temperature sensor cable.

Command Packet:

```
type: 0x25 = 37₁₀
data_length: 3
data[0]: 3 (read DOW device information)
data[1]: FBSCAB module index
data[2]: DOW device index (0-15)
```

Response Packet:

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 11
data[0]: 3 (read DOW device information)
data[1]: FBSCAB module index
data[2]: DOW device index
data[3-10]: DOW ROM ID
```

Subcommand 4: Read WR-DOW-Y17 Temperature

This command will return the temperature of the specified DOW device on the specified CFA-FBSCAB module. The specified DOW device must be of type 0x22 or 0x28 (WR-DOW-Y17 with temperature sensor) as read by command 37, Subcommand 3: Read DOW Device Information (Pg. 62).

Temperature Data (MSB/LSB) Return Format:

```
cc ss s ttt tttt tttt
|| || | ||| |||| ||||-- 11 bit temperature value in degrees C * 16
|| || |---------------- Sign extension (2's complement)
||--------------------- DOW_CRC_status:
00 means CRC was checked and passed
01 means CRC was checked and failed
10 means no sensor detected in this slot
11 means valid sensor but no data yet
```

Command Packet:

```
type: 0x25 = 37₁₀
data_length: 3
data[0]: 4 (read WR-DOW-Y17 temperature)
data[1]: FBSCAB module index
data[2]: DOW device index (0-15)
```

Response Packet:

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 5
data[0]: 4 (read WR-DOW-Y17 temperature)
data[1]: FBSCAB module index
data[2]: DOW device index (0-15)
data[3]: LSB of temperature data
data[4]: MSB of temperature data
```

Subcommand 5: GPIO Pin Levels

The architecture of the CFA-FBSCABs 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 CFA-FBSCAB continuously polls the GPIOs as inputs. The present level can be queried by the host software at a lower rate. The CFA-FBSCAB also keeps track of whether there were rising or falling edges since the last host query (subject to the resolution of the 50 Hz sampling). This means that the host is not forced to poll quickly in order to detect short events. The algorithm used by the CFA-FBSCABs 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 40kΩ. Typical GPIO current limits when sinking or sourcing all five GPIO pins simultaneously are 8 mA.

Command Packet (Set Pin Value):

```
type: 0x25 = 37₁₀
data_length: 4
data[0]: 5 (Set/Read GPIO Pin Configuration & Value)
data[1]: FBSCAB module index
data[2]: index of GPIO to modify
         0 = GPIO[0] = J8, Pin 7
         1 = GPIO[1] = J8, Pin 6
         2 = GPIO[2] = J8, Pin 5
         3 = GPIO[3] = J8, Pin 4
         4 = GPIO[4] = J9, Pin 2 = DOW I/O (always has 1K hardware pull-up)
data[3]: pin output state (behavior depends on drive mode):
         0 = output set to low
      1-99 = output duty cycle percentage (100Hz nominal)
       100 = output set to high
   101-255 =invalid
```

Response Packet (Set Pin Value):

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 0
```

Command Packet (Set Pin Value & Configuration):

```
type: 0x25 = 37₁₀
data_length: 5
data[0]: 5 (Set/Read GPIO Pin Configuration & Value)
data[1]: FBSCAB module index
data[2]: index of GPIO to modify
           0 = GPIO[0] = J8, pin 7
           1 = GPIO[1] = J8, pin 6
           2 = GPIO[2] = J8, pin 5
           3 = GPIO[3] = J8, pin 4
           4 = GPIO[4] = J9, pin 2 = DOW I/O (always has 1K hardware pull-up)
data[3]: pin output state (behavior depends on drive mode):
           0 = output set to low
        1-99 = output duty cycle percentage (100Hz nominal)
         100 = output set to high
     101-255 = invalid
data[4] = pin function select and drive mode
  ---- FDDD
  ||||  |||-- DDD = drive mode (based on output state of 1 or 0)
  ||||  |     =======================================================
  ||||  |     000: 1=strong drive up, 0=resistive pull down
  ||||  |     001: 1=strong drive up, 0=strong drive down
  ||||  |     010: hi-Z, use for input
  ||||  |     011: 1=resistive pull up, 0=strong drive down
  ||||  |     100: 1=strong drive up, 0=hi-Z
  ||||  |     101: 1=strong drive up, 0=strong drive down
  ||||  |     110: reserved, do not use -- error returned
  ||||  |     111: 1=hi-Z,0=strong drive down
  ||||  |
  ||||  |----- F = function
  ||||        =======================================================
              0:port unused for GPIO. it will take on the default function such as ATX,
                 DOW (on CFA-FBSCABs), 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 (may be an extension of F in future versions)
```

Response Packet (Set Pin Value & Configuration):

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 0
```

Command Packet (Read Pin Value & Configuration):

```
type = 0x25 = 37₀₁
data_length: 3
data[0]: 5 (Set/Read GPIO Pin Configuration & Value)
data[1]: FBSCAB module \index
data[2]: index of GPIO
```

Response Packet (Read Pin Value & Configuration):

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 6
data[0]: 5 (Set/Read GPIO Pin Configuration & Value)
data[1]: FBSCAB module index
data[2]: index of GPIO
data[3]: pin state & changes since last poll
    -----RFS enable reporting of this fan's tach input
    |||| |||-- S = state at the last reading
    |||| ||--- F = at least one falling edge has been detected since the last poll
    |||| |---- R = at least one rising edge has been detected since the last poll
    |||| |----- reserved
    this reading is the actual pin state, which may or may not agree with the pin
    setting, depending on drive mode and the load presented by external circuitry.
    the pins are polled at approximately 32Hz asynchronously with respect to this
    command. transients that happen between polls will not be detected
data[4]: 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[5]: pin function select and drive mode\
---- FDDD
  |||| |||-- DDD = drive mode (based on output state of 1 or 0)
  |||| |     ======================================================
  |||| |     000: 1=strong drive up, 0=resistive pull down
  |||| |     001: 1=strong drive up, 0=strong drive down
  |||| |     010: hi-Z, use for input
  |||| |     011: 1=resistive pull up, 0=strong drive down
  |||| |     100: 1=strong drive up, 0=hi-Z
  |||| |     101: 1=strong drive up, 0=strong drive down
  |||| |     110: reserved, do not use -- error returned
  |||| |     111: 1=hi-Z,0=strong drive down
  |||| |
  |||| |----- F = function
  ||||       ======================================================
             0: port unused for GPIO. It will take on the default function such as ATX,
                DOW (on CFA-FBSCABs), 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,
```

Subcommand 6: Reset And Search

This command sends a reset instruction to all attached CFA-FBSCAB modules. This will revert the CFA-FBSCAB modules back to their saved power-on state. After the reset instructions have been sent, the CFA835 re-searches for attached CFA-FBSCAB modules.

Note: For one attached CFA-FBSCAB, this command takes approximately 400 mS to complete and return the response packet. If multiple CFA-FBSCABs are attached, searching may take longer, up to 2 additional seconds.

Command Packet:

```
type = 0x25 = 37₀₁
data_length: 1
data[0]: 6 (Reset & Search)
```

Response Packet:

```
type = 0x40 | 0x25 = 0x65 = 101₁₀
data_length: 1
data[0]: 6 (Reset & Search)
```

38 (0x26): Custom Fonts

The CFA835 is the first in our intelligent product line with a monochrome graphic LCD. It supports printing text using most any custom font in most any language. To support this exciting new functionality, we've developed a utility to convert fonts to the new CFA835 font structure. Using this utility, fonts can be created from scratch or imported from the Windows library and modified for export. Custom fonts can then be transfered to the CFA835 using the on board microSD card. The CFA835 supports using up to 4 custom fonts simultaneously.

Subcommand 0: Load Custom Font Files From MicroSD Card

This command loads custom font files from the inserted microSD card. Your custom font files must be created using the CFA835 Font Editor (Pg. 86). The loaded font is printed to the display using the subcommand immediately below, Subcommand 1: Print Custom Font To Display.

The CFA835 supports using up to 4 individual custom font files at a time (four “slots”).

User defined characters as set by command 9 (0x09): Special Character Bitmaps (Pg. 47) are not supported by this command or the subcommand immediately below, Subcommand 1: Print Custom Font To Display.

Command 31 (0x1F): Write Text To The Display (Pg. 53) supports a special replacement mode using a custom font. Replacement mode is activated by loading a custom font into slot 0 with data[2]:bit 1 set to 1.

To disable replacement mode, load a custom font into slot 0 with data[2]:bit 1 set to 0.

Replacement mode can only use a custom font in slot 0; attempting to set data[2]:bit 1 for a custom font loaded in any other slot will throw an error.

Command Packet:
```
type = 0x26 = 38₁₀
data_length: 4 to 124
data[0]: 0 (Load Custom Font Files From MicroSD Card)
data[1]: font slot (0 to 3)
data[2]: option flags
         bit 0 = forced monospace (ignore proportional flag in font file header).
         bit 1 = use font for 31 (0x1F): Write Text To The Display (utf-8 only, must be a
                 monospace font or forced monospace)
         bit 2 = 0=utf-8, 1=utf-16
data[3-123]: file name of the font file located on the microSD card.
```

Response Packet:
```
type = 0x40 | 0x26 = 0x46 = 102₁₀
data_length: 1
data[0]: 0 (Load Custom Font Files From MicroSD Card)
```

Subcommand 1: Print Custom Font To Display

This command prints the specified string to the display using the font slot set by the subcommand immediately above, Subcommand 0: Load Custom Font Files From MicroSD Card.

Command Packet:

```
type = 0x26 = 38₁₀
data_length: 4 to 124
data[0]: 1 (Print Custom Font to Display)
data[1]: font slot (0 to 3)
data[2]: character placement style
         0 = char/row
         1 = pixel x/y
         column value only used if font is monospaced or forced monospaced.
         pixel x/y is top left pixel of the first character
data[3]: column or x-pixel position of the top-left of first character
data[4]: row or y-pixel position of the top-left of first character
data[5-123]: utf-8 or utf-16 text string
```

Response Packet:

```
type = 0x40 | 0x26 = 0x46 = 102₁₀
data_length: 2
data[0]: 1 (Print Custom Font to Display)
data[1]: length of the printed text in pixels
```

39 (0x27): MicroSD File Operations

Subcommand 0: Open/Close MicroSD File

This command opens the specified file on the inserted microSD card for reading/writing. Only one file on the microSD card may be accessed at a time. The subcommands 1 through 4 operate on the opened file.

data[1] options 1 and 2 will set the file pointer position to the start of the file (position 0).
data[1] option 2 will set the file pointer position to the end of the file.

Command Packet:

```
type: 0x27 = 39₁₀
data_length: 2 to 124
data[0]: 0 (Open/Close File)
data[1]: options
         0 = close currently opened file (file name does not need to be specified)
         1 = open file for reading
         2 = open file for reading and writing (truncates existing file)
         3 = open file for reading and writing (appends to existing file)
data[2-123]: file name of the file located on the microSD card
```

Response Packet:

```
type: 0x40 | 0x27 = 0x67 = 103₁₀
data_length: 5
data[0]: 0 (Open/Close File)
data[1-4]: file size in bytes
```

Subcommand 1: Position Seek

This command seeks (sets the file pointer) to the location specified in the file opened with the subcommand immediately above, Subcommand 0: Open/Close MicroSD File.

Command Packet:

```
type: 0x27 = 39₁₀
data_length: 5
data[0]: 1 (Position Seek)
data[1-4]: 32 bit location of byte position in the file (LSB first)
```

Response Packet:

```
type: 0x40 | 0x27 = 0x67 = 103₁₀
data_length: 1
data[0]: 1 (Position Seek)
```

Subcommand 2: Read File Data

Read data from the file opened by command 39, Subcommand 0: Open/Close MicroSD File. Data is read from the current file pointer location.

The file pointer position will be incremented by the amount of data read by this command. To read data from elsewhere in the file, use command immediately above, Subcommand 1: Position Seek first.

Command Packet:

```
type: 0x27 = 39₁₀
data_length: 2
data[0]: 2 (Read File Data)
data[1]: number of bytes to read (1 to 124)
```

Response Packet:

```
type: 0x40 | 0x27 = 0x67 = 103₁₀
data_length: 1 to 124 (length of data read from the file if data_length is less than
             requested, then the end-of-file has been reached)
data[0]: 2 (Read File Data)
data[1-123]: data read from the file
```

Subcommand 3: Write File Data

Writes data to the file opened by command 39, Subcommand 0: Open/Close MicroSD File. Data is written at the current file pointer location.

Command Packet:

```
type: 0x2F = 47
data_length: 2 to 124
data[0]: 3 (Write File Data)
data[1-123]: data to write to the file
```

Response Packet:

```
type: 0x40 | 0x27 = 0x67 = 103₁₀
data_length: 1
data[0]: 3 (Write File Data)
```

Subcommand 4: Delete A File

This command deletes the specified file from the microSD card.

Command Packet:

```
type: 0x27 = 39₁₀
data_length: 2 to 124
data[0]: 4 (Delete a File)
data[1-123]: file name of the file located on the microSD card
```

Response Packet:

```
type: 0x40 | 0x27 = 0x67 = 103₁₀
data_length: 1
data[0]: 4 (Delete a File)
```

40 (0x32): Display Graphic Options

The CFA835 supports the ability to update the display either directly or using a buffer that can be flushed manually.

This option is enabled or disabled using Subcommand 0: Graphic Options (see immediately below).

Valid ranges for all the subcommands in this command group are:

```
X pixels = 0-243
Y pixels = 0-67
shade = 0-255
```

Subcommand 0: Graphic Options

This command controls two of the options related to the CFA835's graphical display capabilities:

1. Buffer Flush
   When enabled, display graphical commands (except command 31 (0x1F): Write Text To The Display (Pg. 53)) are buffered and only written to display when using the subcommand immediately below, Subcommand 1: Buffer Flush (Pg. 69).
2. Gamma Correction
   When enabled, graphics and fonts written to the display will have gamma correction applied. This option does not affect command 31 (0x1F): Write Text To The Display (Pg. 53).

Command Packet:

```
type: 0x28 = 40
data_length: 2
data[0]: 0 (Graphics Options)
data[1]: option flags
         bit 0 = buffer flush (0 = automatic, 1 = manual)
         bit 1 = gamma correction (1 = enabled, 0 = disabled)
```

Response Packet:

```
type: 0x40 | 0x28 = 0x68 = 104₁₀
data_length: 1
data[0]: 0 (Graphics Options)
```

Subcommand 1: Buffer Flush

This command flushes the memory of the graphical buffer to the CFA835's display. This command has no effect unless subcommand immediately above, Subcommand 0: Graphic Options (Pg. 69) `bit 0` is set to `manual`.

Command Packet:

```
type: 0x28 = 40₁₀
data_length: 1
data[0]: 1 (Buffer Flush)
```

Response Packet:

```
type: 0x40 | 0x28 = 0x68 = 104₁₀
data_length: 1
data[0]: 1 (Buffer Flush)
```

Subcommand 2: Send Image Data To Display From Host

This command supports a special "data streaming" mode unique to this command. After this packet has been sent to the CFA835, raw pixel data (not in normal packet format) is sent to the CFA835.

Note: As graphical data is not sent in packets, it is not CRC checked. Any data transmission errors will result in an incorrect image being displayed on the CFA835.

Note: A return acknowledge packet will not be sent by the CFA835 to the host until transmission of the graphical data is complete.

Note: If “manual buffer flush” is enabled (see command 40, Subcommand 0: Graphic Options (Pg. 69)), the image will not be drawn until the subcommand immediately above, Subcommand 1: Buffer Flush (Pg. 69) is executed.

Note: This command has no support for directly interpreting jpg/png/bmp/etc. file formats – only raw pixel data. cfTest includes functionality to convert an image (many different formats) into raw data which is then sent to the CFA835.

The raw pixel data transfer must be completed within 500 ms from the USB interface or 2 seconds from any other interface. Failure to do so will result in the CFA835 returning an error packet and ignoring any following raw data.

Raw pixel data is in the format of one byte per pixel. The display is capable of displaying 32 shades of grey (most significant 5 bits of the byte). The least significant 3 bits of shade is ignored. Pixel data is interpreted in order: left to right, top to bottom.

RLE compression removes repetitive values. Here is an example:

RLE Compression Example (values in hexidecimal)											
Byte Order	0	1	2	3	4	5	6	7	8	9	10
Original Pixel Data	0x00	0xF8	0x30	0x30	0x30	0x30	0x30	0x30	0x30	0xF8	0x00
Sent RLE Data	0x00	0xF8	0x03	0x07	0x30	0xF8	0x00				
Displayed Pixel Data	0x00	0xF8	0x30	0x30	0x30	0x30	0x30	0x30	0x30	0xF8	0x00

Command Packet:

```
type: 0x28 = 40₁₀
data_length: 6
data[0]: 2 (Send Image Data To Display From Host)
data[1]: option flags
         bit 0 = enable transparency (pixel value 0 is transparent)
         bit 1 = invert image color (will invert transparency value also)
         bit 2 = RLE compression (format: 0x03, length, value)
data[2]: x pixel location to start
data[3]: y pixel location to start
data[4]: width of image in pixels
data[5]: height of image in pixels
```

Response Packet:

```
type: 0x40 | 0x28 = 0x68 = 104₁₀
data_length: 1
data[0]: 2 (Send Image Data To Display From Host)
```

Subcommand 3: Display Image File From MicroSD Card On CFA835

This command displays a BMP formatted image file located on the inserted microSD card. The BMP file must be grayscale, 8 bits/pixel, no compression, Microsoft Windows format only.

Note: If “manual buffer flush” is enabled (see command 40, Subcommand 0: Graphic Options (Pg. 69)), the pixel will not be drawn until command 40, Subcommand 1: Buffer Flush (Pg. 69) is executed.

Command Packet:

```
type: 0x28 = 40₁₀
data_length: 6 to 124
data[0]: 3 (Display Image File From MicroSD Card On CFA835)
data[1]: option flags
         bit 0 = enable transparency (pixel value 0 is transparent)
         bit 1 = invert image shade (will invert transparency value also)
data[2]: x pixel location to start
data[3]: y pixel location to start
data[4-123]: name of the image file located on the microSD card
```

Response Packet:

```
type: 0x40 | 0x28 = 0x68 = 104₁₀
data_length: 1
data[0]: 3 (Display Image File From MicroSD card on CFA835)
```

Subcommand 4: Save Screenshot to MicroSD File

This command saves a screenshot of the current image to a BMP file of the specified name on the microSD card. If a file with the specified name already exists, it will be overwritten. The BMP file will be saved in Microsoft format, 8bits/pixel, greyscale, with no compression, and is 17,670 bytes in size.

Note: If "manual buffer flush" is enabled (see command 40, Subcommand 0: Graphic Options (Pg. 69)), the image stored will be the image currently in the buffer.

Command Packet:

```
type: 0x28 = 40₁₀
data_length: 2 to 124
data[0]: 4 (Save Screenshot to MicroSD File)
data[1-123]: name of the file to create on the microSD card
```

Response Packet:

```
type: 0x40 | 0x28 = 0x68 = 104₁₀
data_length: 1
data[0]: 4 (Save Screenshot to MicroSD File)
```

Subcommand 5: Pixel Data

This command sets or reads the value of the specified individual pixel on the display.

Note: If "manual buffer flush" is enabled by command 40, Subcommand 0: Graphic Options (Pg. 69), the value returned is the pixel value in the buffer.

Command Packet (Write):

```
type: 0x28 = 40₁₀
data_length: 4
data[0]: 5 (Pixel Data)
data[1]: x pixel location (0-243)
data[2]: y pixel location (0-67)
data[3]: new pixel shade
```

Response Packet (Write):

```
type: 0x40 | 0x28 = 0x68 = 104₁₀
data_length: 1
data[0]: 5 (Pixel Data)
```

Command Packet (Read):

```
type: 0x28 = 40₁₀
data_length: 3
data[0]: 5 (Pixel Data)
data[1]: x pixel location (0-243)
data[2]: y pixel location (0-67)
```

Response Packet (Read):

```
type: 0x40 | 0x28 = 0x68 = 1₀₄
data_length: 2
data[0]: 5 (Pixel Data)
data[1]: pixel shade value
```

Subcommand 6: Draw a Line

This command draws a line of the specified shade from point a to point b.

Note: If “manual buffer flush” is enabled (see command 40, Subcommand 0: Graphic Options (Pg. 69)), the line will not be displayed onto the CFA835 until command 40, Subcommand 1: Buffer Flush (Pg. 69) is executed.

Command Packet:

```
type: 0x28 = 40₁₀
data_length: 6
data[0]: 6 (Draw a Line)
data[1]: x pixel location to start
data[2]: y pixel location to start
data[3]: x pixel location to finish
data[4]: y pixel location to finish
data[5]: line shade value
```

Response Packet:

```
type: 0x40 | 0x28 = 0x68 = 104₁₀
data_length: 1
data[0]: 6 (Draw a Line)
```

Subcommand 7: Draw a Rectangle

This command draws a rectangle to the CFA835's display.

Note: If “manual buffer flush” is enabled (see command 40, Subcommand 0: Graphic Options (Pg. 69)), the rectangle will not be displayed onto the CFA835 until command 40, Subcommand 1: Buffer Flush (Pg. 69) is executed.

Command Packet:

```
type: 0x28 = 40₁₀
data_length: 7
data[0]: 7 (Draw a Rectangle)
data[1]: x pixel location (top-left)
data[2]: y pixel location (top-left)
data[3]: rectangle width
data[4]: rectangle height
data[5]: line shade
data[6]: fill shade (0 is transparent)
```

Response Packet:

```
type: 0x40 | 0x28 = 0x68 = 104₁₀
data_length: 1
data[0]: 7 (Draw a Rectangle)
```

Subcommand 8: Draw a Circle

This command draws a circle of the specified radius using the specified x,y pair as its center point.

Note: If "manual buffer flush" is enabled (see command 40, Subcommand 0: Graphic Options (Pg. 69)), the circle will not be displayed onto the CFA835 until command 40, Subcommand 1: Buffer Flush (Pg. 69) is executed.

Command Packet:

```
type: 0x28 = 40₁₀
data_length: 6
data[0]: 8 (Draw a Circle)
data[1]: x of circle
data[2]: y position center of circle
data[3]: circle radius
data[4]: line shade
data[5]: fill shade (0 is transparent)
```

Response Packet:

```
type: 0x40 | 0x28 = 0x68 = 104₁₀
data_length: 1
data[0]: 8 (Draw a Circle)
```

41 (0x3A): Video Playback Control

Subcommand 0: Load A Video From MicroSD Card

The CFA835 can play up to four independent video files (four "slots") to the CFA835 at a time. Video slots are drawn in order of slot number, so a video in slot 1 will be displayed over the top of a video in slot 0. Each video can be controlled independently using the subcommand immediately below, Subcommand 1: Video Control (Pg. 73). The video files must be encoded using the CFA835 Video Encoder utility. See CFA835 Video Encoder (Pg. 87).

Note: Playing a video directly on top of another video may result in flicker. We recommend against doing this. If your project solution depends on playing multiple videos layered over each other, compression must be disabled during encoding and the videos must have the same frame rate.

Command Packet:

```
type: 0x29 = 41₁₀
data_length: 3 to 124
data[0]: 0 (Load A Video From MicroSD Card)
data[1]: video slot number (0 to 3)
data[2-123]: name of the video file on the microSD card
```

Response Packet:

```
type: 0x40 | 0x29 = 0x69 = 105₁₀
data_length: 1
data[0]: 0 (Load A Video From MicroSD Card)
```

Subcommand 1: Video Control

This command controls the video(s) opened using the subcommand immediately above, Subcommand 0: Load A Video From MicroSD Card (Pg. 73).

Note: Attempting to play a video outside of the display's graphical limits will result in an error being returned.

Command Packet:

```
type: 0x29 = 41₁₀
data_length: 3 or 6
data[0]: 1 (Video Control)
data[1]: video slot number (0 to 3)
data[2]: control option
         0 = play
         1 = stop (data[3-5] not required for this option)
         2 = toggle pause (data[3-5] not required for this option)
data[3]: play video X times in loop (up to 255) (0x00 = continuously)
data[4]: x pixel location
data[5]: y pixel location
```

Response Packet:

```
type: 0x40 | 0x29 = 0x69 = 105₁₀
data_length: 1
data[0]: 1 (Video Control)
```

REPORT CODES

The CFA835 can be configured to report information automatically when data becomes available. Reports are not sent in response to a particular packet received from the host. Details are below.

128 (0x80): Key Activity

If a key is pressed or released, the CFA835 sends a Key Activity report packet to the host. Key event reporting may be individually enabled or disabled by command 23 (0x17): Keypad Reporting (Pg. 50).

```
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
```

192 (0xC0): Extended Error Reporting

If enabled by command 33 (0x21): Interface Options (Pg. 54), error packets have the format:

```
type: 0xC0
data length = 2
data[0]: originating command interface
         0 = serial
         1 = USB
data[1]: ID of extended error information
         Errors are:
```

Error #	Description
1	Unknown Error
2	Unknown Command
3	Invalid Command Length/Options
4	Writing Flash Mem Failed
5	Reading Flash Mem Failed
6	FBSCAB Not Present At Index
7	FBSCAB Did Not Reply To Req
8	MicroSD Not Inserted Or Bad
9	MicroSD Not Formatted
10	MicroSD File Could Not Be Found/Opened
11	MicroSD Unknown Error
12	MicroSD File Could Not Be Read
13	MicroSD File Could Not Be Written
14	File Header Is Invalid
15	MicroSD File Is Already Open
16	MicroSD File Operation Failed
17	MicroSD File Has Not Been Opened
18	GFX Stream Already Started
19	GFX Is Out Of LCD Bounds
20	Video Is Not Open In Slot
21	GFX Stream Has Timed Out
22	GPIO Not Set For ATX Use
23	Interface Not Enabled
24	Interface Not Available

CHARACTER GENERATOR ROM (CGROM) FOR STANDARD SET OF CHARACTERS

To find the code for a font, add the two numbers that are shown in bold for its row and column. For example, the superscript "9" is in the column labeled “128d” and in the row labeled “9d”. Add 128 + 9 to get 137. When you send a byte with the value of 137 to the display, then a superscript "9" will be shown.

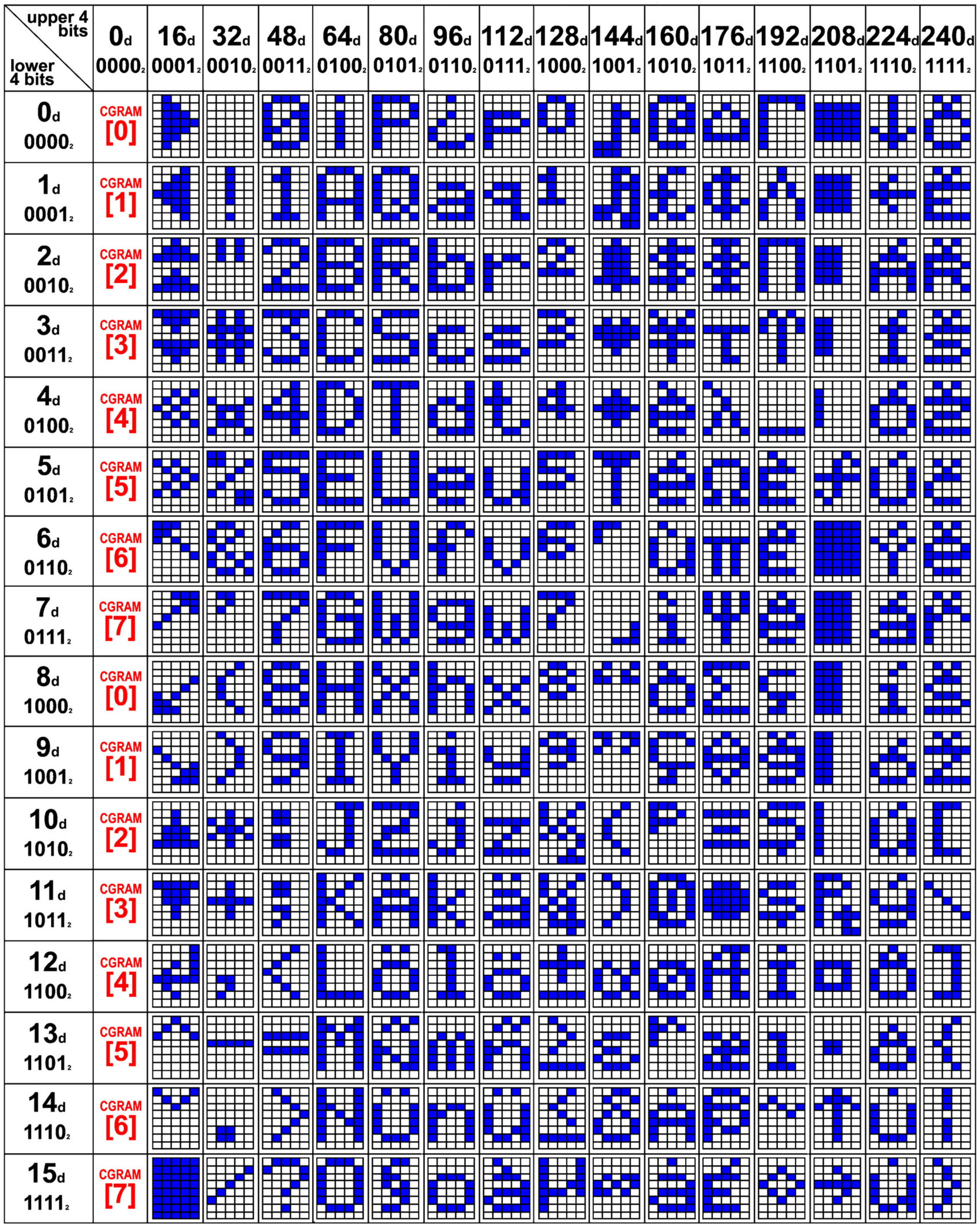

upper 4 bits
lower 4 bits
0d 0000₂
16d 0001₂
32d 0010₂
48d 0011₂
64d 0100₂
80d 0101₂
96d 0110₂
112d 0111₂
128d 1000₂
144d 1001₂
160d 1010₂
176d 1011₂
192d 1100₂
208d 1101₂
224d 1110₂
240d 1111₂
0d 0000₂
1d 0001₂
2d 0010₂
3d 0011₂
4d 0100₂
5d 0101₂
6d 0110₂
7d 0111₂
8d 1000₂
9d 1001₂
10d 1010₂
11d 1011₂
12d 1100₂
13d 1101₂
14d 1110₂
15d 1111₂
CGRAM [0]
CGRAM [1]
CGRAM [2]
CGRAM [3]
CGRAM [4]
CGRAM [5]
CGRAM [6]
CGRAM [7]
CGRAM [0]
CGRAM [1]
CGRAM [2]
CGRAM [3]
CGRAM [4]
CGRAM [5]
CGRAM [6]
CGRAM [7]

Figure 20. Character Generator ROM (CGROM)

DISPLAY MODULE RELIABILITY AND LONGEVITY

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

DISPLAY MODULE RELIABILITY

ITEM		RELIABILITY SPECIFICATION	
Display portion (excluding keypad, status LEDs, and backlights)		50,000 to 100,000 hours	
Keypad		1,000,000 keystrokes	
Bicolor LED status lights		50,000 to 100,000 hours	
	CFA835-TFK (white LED display backlight and white LED keypad backlight)	***Power-On Hours***	***% of Initial Brightness (New Module)***
		10,000 hours	>70%
		<50,000 hours	>50%
	CFA835-TML (white LED display backlight and blue LED keypad backlight)	***Power-On Hours***	***% of Initial Brightness (New Module)***
		<10,000	>70%
		<50,000	>50%
	CFA835-YYK (yellow-green LED display backlight and yellow-green LED keypad backlight)	50,000 to 100,000 hours	
Note: For display modules with white LED backlights (CFA835-TFK and CFA835-TML), 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. *Note: Values listed above are approximate and represent typical lifetime under operating and storage temperature specification limitations, humidity noncondensing RH up to 65%, and no exposure to direct sunlight.*			

DISPLAY MODULE LONGEVITY (EOL / REPLACEMENT POLICY)

Crystalfontz is committed to making all of our display modules available for as long as possible. Occasionally, a supplier discontinues a component, or a process used to make the display module becomes obsolete, or the process moves to a more modern manufacturing line. In order to continue making the display 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 display 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 display module. For example, we must occasionally discontinue a display module when a supplier discontinues a component or a manufacturing process becomes obsolete. When we discontinue a display module, we will do our best to find an acceptable replacement display 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 display module to the discontinued display module it replaces. However, sometimes a change in component or process for the replacement display module results in a slight variation, perhaps an improvement, over the previous design.

Although the replacement display 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:

- *Backlights with 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.* Display 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 display module whenever possible; we only discontinue a display module if we have no other option. We post Part Change Notices (PCN) on the product's website page as soon as possible. If interested, you can subscribe to future part change notifications.

CARE AND HANDLING PRECAUTIONS

For optimum operation of the CFA835 and to prolong its life, please follow the precautions described below.

Caution
When not in use, always verify that the microSD card socket is in the closed and LOCKED position.

Caution
Excessive voltage will shorten the life of the module. You must drive the display within the specified voltage limit. See Absolute Maximum Ratings (Pg. 23).

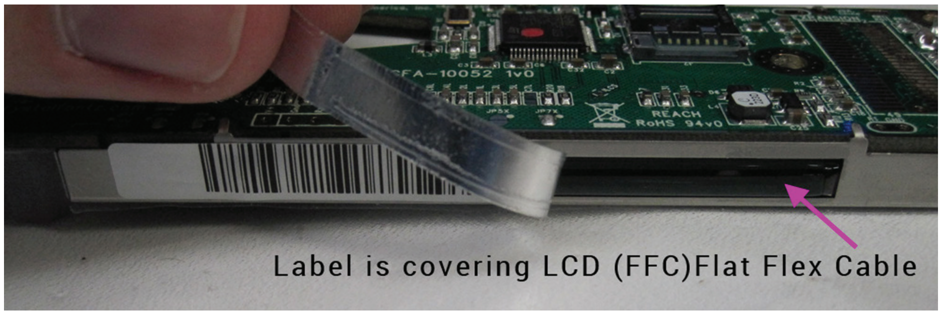
HANDLING CAUTIONS

Display Modules Shipped In Trays

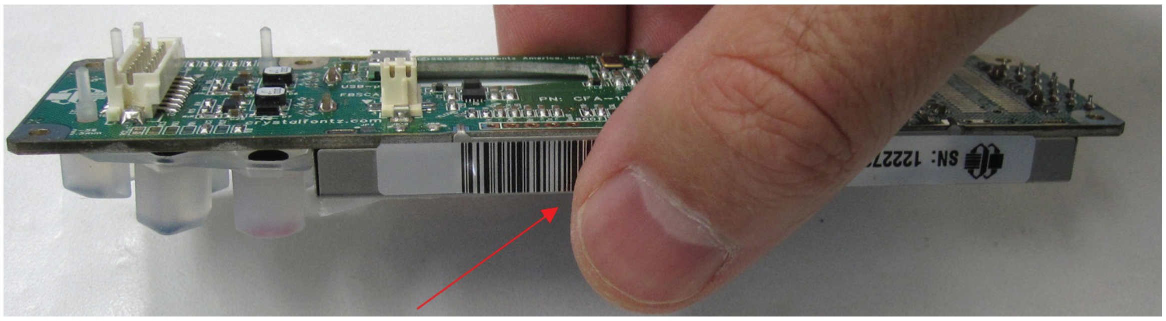
If you receive display modules packed in trays, handle trays carefully by supporting the entire tray. Trays were made to immobilize the display modules inside their packing carton. Trays are not designed to be rigid. Do not carry trays by their edges; trays and display modules may be damaged.

Avoid Damaging Flat Flex Cable

To avoid damaging the CFA835, do not press on the FFC (Flat Flex Cable) which is under the label. Place your fingers on either side of the label.



Bent FFC (Flat Flex Cable) is under the label.



If you press here, you may damage the connection.

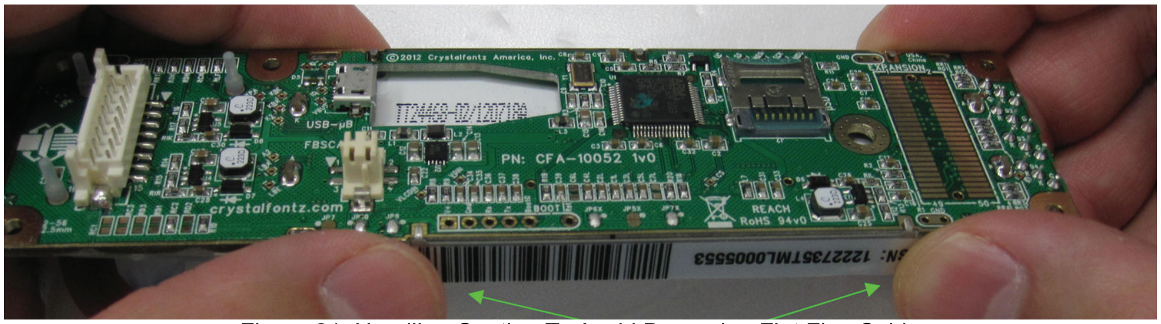


Figure 21. Handling Caution To Avoid Damaging Flat Flex Cable

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

- When handling the CFA835, use care so as to not press on the exposed FFC with excess force. See Location Of CFA835 Connectors (Pg. 34).

- The exposed surface of the display "glass" is actually a polarizer laminated on top of the glass. To protect the polarizer from damage, the display module ships with a protective film over the polarizer. Please peel off the protective film slowly. Peeling off the protective film abruptly may generate static electricity.
- The polarizer is made out of soft plastic and is easily scratched or damaged. When handling the display module, avoid touching the polarizer. Finger oils are difficult to remove.
- To protect the soft plastic polarizer from damage, place a transparent plate (for example, acrylic, polycarbonate, or glass) in front of the display module, leaving a small gap between the plate and the display surface. We recommend Lexan, which is readily available and works well.
- Allow adequate space for the flex at the bottom of the display module. If flex is creased, display module may be permanently damaged.
- For USB interface, keep the micro-B USB cable connector parallel to the CFA835 when plugging or unplugging the cable. Do not lift or pull up on the cable. Too much pressure may permanently damage the CFA835's micro-B USB connector.
- Do not disassemble or modify the display module.
- Do not modify the five tabs of the metal bezel or make connections to them.
- Do not reverse polarity to the power supply connections. Reversing polarity will immediately ruin the display module.

AVOID SHOCK, IMPACT, TORQUE, OR TENSION

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

CAUTION

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 module 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 display 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 display module from ESD and power supply transients.
- Observe the operating temperature limitations: a minimum of -20°C to a maximum of 70°C noncondensing with minimal fluctuation. Operation outside of these limits may shorten life and/or harm display module.
  - 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 displays with white LED backlights (CFA835-TFK and CFA835-TML), 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.

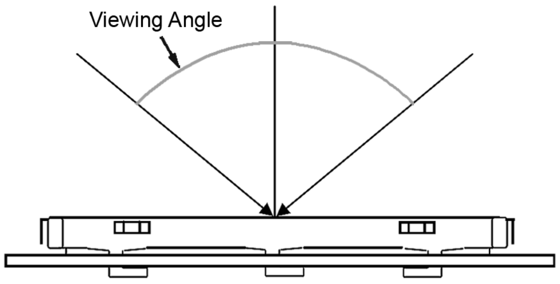
STORAGE AND RECYCLING

- 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 -30°C minimum to +80°C non-condensing maximum with minimal fluctuations. Rapid temperature changes can cause moisture to form, resulting in permanent damage.
- Do not allow weight to be placed on the display modules while they are in storage.
- Please recycle your outdated Crystalfontz display modules at an approved facility.

APPENDIX A: QUALITY ASSURANCE STANDARDS

INSPECTION CONDITIONS

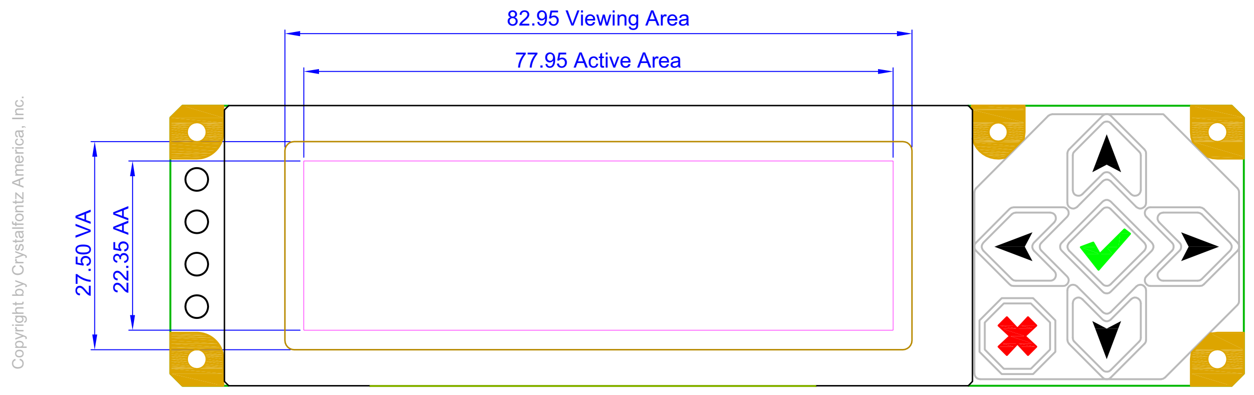
- Environment
  - Temperature: 25±5°C
  - Humidity: 30~85% RH
- For visual inspection of active display area
  - Source lighting: two 20 Watt or one 40 Watt fluorescent light
  - Display adjusted for best contrast
  - Viewing distance: 30±5 cm (about 12 inches)
  - Viewable angle: inspect at 45° angle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



COLOR DEFINITIONS

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

DEFINITION OF ACTIVE AREA AND VIEWING AREA



ACCEPTANCE SAMPLING

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

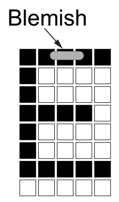
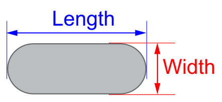
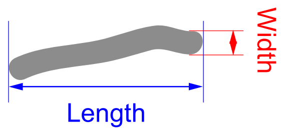
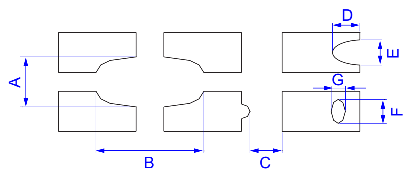
DEFECTS CLASSIFICATION

Defects are defined as:

- Major Defect: results in failure or substantially reduces usability of unit for its intended purpose
- Minor Defect: deviates from standards but is not likely to reduce usability for its intended purpose

ACCEPTANCE STANDARDS

#	DEFECT TYPE	CRITERIA	MAJOR / MINOR
1	Electrical defects	1. No display, display malfunctions, or shorted segments. 2. Current consumption exceeds specifications.	Major
2	Viewing area defect	Viewing area does not meet specifications. (See Inspection Conditions (Pg. 82).	Major
3	Contrast adjustment defect	Contrast adjustment fails or malfunctions.	Major

#	**DEFECT TYPE**	**CRITERIA**			**MAJOR / MINOR**
4	Blemishes or foreign matter on display segments		*Defect Size (mm)*	*Acceptable Qty*	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 	*Defect Size (mm)*	*Acceptable Qty*	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 	*Defect Width (mm)*	*Defect Length (mm)*	*Acceptable Qty*	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		*Defect Size (mm)*	*Acceptable Qty*	Minor
			≤0.20	Ignore	
			0.20 to 0.40	3	
			0.40 to 0.60	2	
			≥0.60	0	
8	Display pattern defect				Minor
		Dot Size (mm)	*Acceptable Qty*		
		((A+B)/2)≤0.2	≤3 total defects ≤2 pinholes per digit		
		C>0			
		((D+E)/2)≤0.25			
		((F+G)/2)≤0.25			

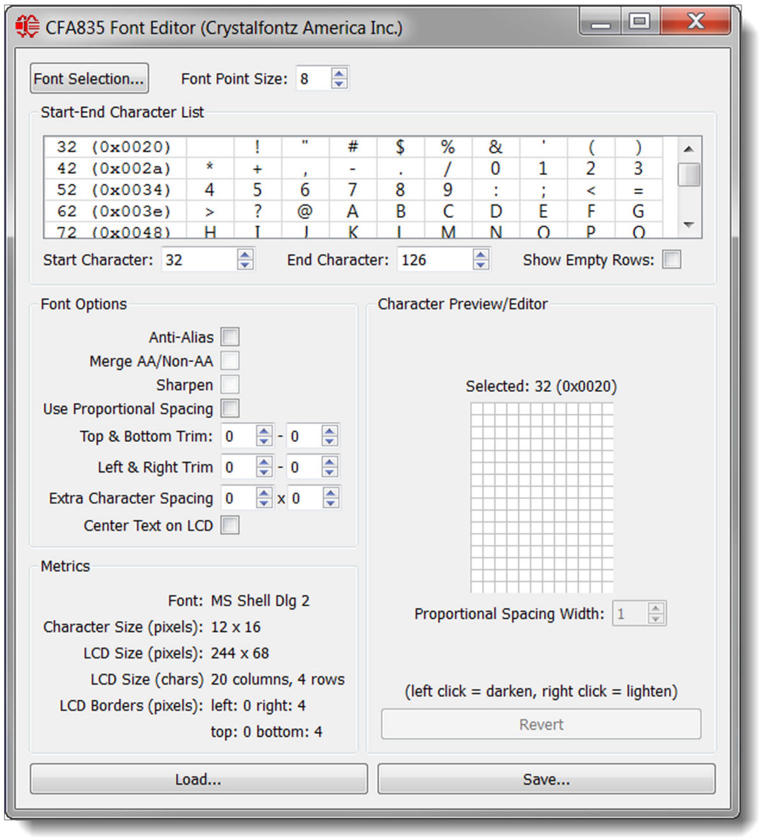
#	**DEFECT TYPE**	**CRITERIA**	**MAJOR / MINOR**
9	Backlight defects	1. Light fails or flickers.* 2. Color and luminance do not correspond to specifications.* 3. Exceeds standards for display's blemishes or foreign matter (see test 5, Pg. 84), and dark lines or scratches (see test 6, Pg. 84). **Minor if display functions correctly. Major if the display fails.*	Minor
10	COB defects	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	1. Oxidation or contamination on connectors.* 2. Wrong parts, missing parts, or parts not in specification.* 3. Jumpers set incorrectly. 4. Solder (if any) on bezel, LED pad, zebra pad, or screw hole pad is not smooth. **Minor if display functions correctly. Major if the display fails.*	Minor
12	Soldering defects	1. Unmelted solder paste. 2. Cold solder joints, missing solder connections, or oxidation.* 3. Solder bridges causing short circuits.* 4. Solder balls. **Minor if display functions correctly. Major if the display fails.*	Minor

APPENDIX B: FREE DEMONSTRATION AND OTHER SOFTWARE

CFA835 UTILITIES

The three CFA835 Window utilities described below are bundled together in a CFA835 utilities package. The utilities are built upon the most recent cfTest command descriptions. cfTest is also included in the package.

CFA835 Font Editor



CFA835 Font Editor (Crystalfontz America Inc.)

Font Selection... Font Point Size: 8

Start-End Character List

32 (0x0020)		!	"	#	$	%	&	'	(	)
42 (0x002a)	*	+	,	-	.	/	0	1	2	3
52 (0x0034)	4	5	6	7	8	9	:	;	<	=
62 (0x003e)	>	?	@	A	B	C	D	E	F	G
72 (0x0048)	H	I	J	K	L	M	N	O	P	Q

Start Character: 32 End Character: 126 Show Empty Rows: ☐

Font Options

Anti-Alias ☐
Merge AA/Non-AA ☐
Sharpen ☐
Use Proportional Spacing ☐
Top & Bottom Trim: 0 - 0
Left & Right Trim 0 - 0
Extra Character Spacing 0 x 0
Center Text on LCD ☐

Metrics

Font: MS Shell Dlg 2
Character Size (pixels): 12 x 16
LCD Size (pixels): 244 x 68
LCD Size (chars) 20 columns, 4 rows
LCD Borders (pixels): left: 0 right: 4
top: 0 bottom: 4

Character Preview/Editor

Selected: 32 (0x0020)

Proportional Spacing Width: 1

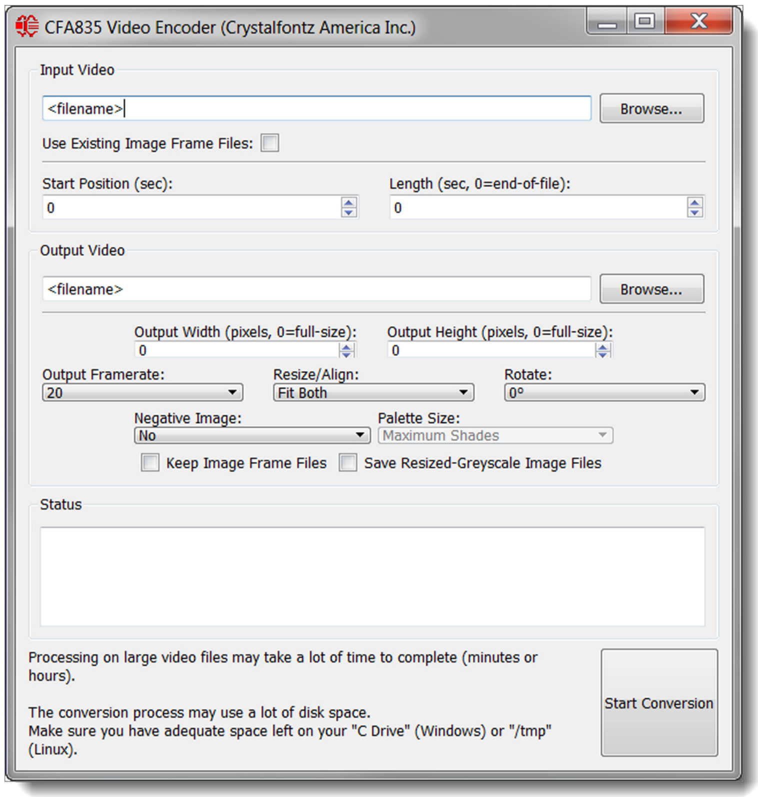
(left click = darken, right click = lighten)

The CFA835 Font Editor converts any font into the CFA835 font format. The editor creates CFA835 compatible custom font files using fonts available on the your PC. When the font file is loaded onto a microSD card inserted into the CFA835 card socket, the module can write custom font text to the display.

The font converter and CFA835 support UTF16 (Unicode) fonts, allowing non-English (for example, Cyrillic, Asian, symbolic, etc.) font files to be created and displayed. Many font size, type, spacing, and other options are available.

See CFA835 commands Subcommand 0: Load Custom Font Files From MicroSD Card and Subcommand 1: Print Custom Font To Display for details on font file use.

CFA835 Video Encoder


CFA835 Video Encoder (Crystalfontz America Inc.)

Input Video
<filename>

Use Existing Image Frame Files: ☐
Start Position (sec): 0
Length (sec, 0=end-of-file): 0

Output Video
<filename>

Output Width (pixels, 0=full-size): 0
Output Height (pixels, 0=full-size): 0
Output Framerate: 20
Resize/Align: Fit Both
Rotate: 0°
Negative Image: No
Palette Size: Maximum Shades
☐ Keep Image Frame Files ☐ Save Resized-Greyscale Image Files

Status

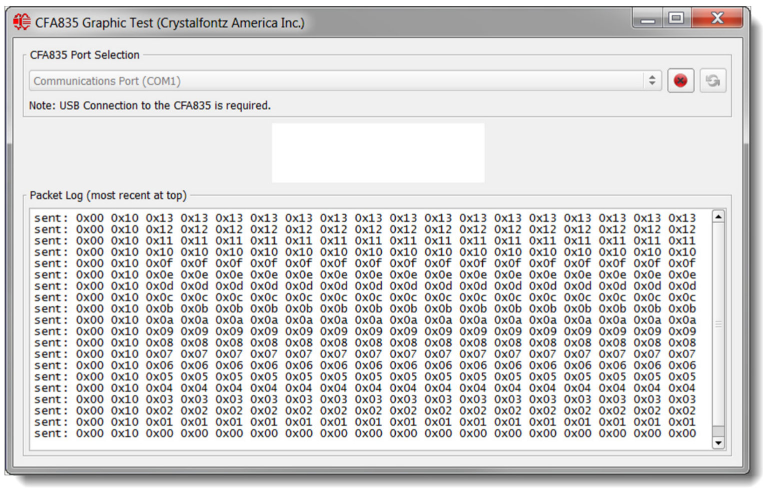
Processing on large video files may take a lot of time to complete (minutes or hours).

The conversion process may use a lot of disk space.
Make sure you have adequate space left on your "C Drive" (Windows) or "/tmp" (Linux).

The Video Encoder converts common video format files into a video file that the CFA835 can play to the display. The video conversion uses MPlayer (a GNU-GPLv2 licensed open-source software) to create many single image files from the source video, and then reassembles the image files into a CFA835 video file. Processing time depends on the source video file.

See CFA835 commands 41 (0x3A): Video Playback Control for details on playing a video on the CFA835.

CFA835 Graphic Test


CFA835 Graphic Test (Crystalfontz America Inc.)
CFA835 Port Selection
Communications Port (COM1)

Note: USB Connection to the CFA835 is required.

Packet Log (most recent at top)

```
sent: 0x00 0x10 0x13 0x13 0x13 0x13 0x13 0x13 0x13 0x13 0x13 0x13 0x13 0x13 0x13 0x13 0x13 0x13
sent: 0x00 0x10 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12
sent: 0x00 0x10 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11 0x11
sent: 0x00 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10 0x10
sent: 0x00 0x10 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f 0x0f
sent: 0x00 0x10 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e 0x0e
sent: 0x00 0x10 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d 0x0d
sent: 0x00 0x10 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c 0x0c
sent: 0x00 0x10 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b 0x0b
sent: 0x00 0x10 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a 0x0a
sent: 0x00 0x10 0x09 0x09 0x09 0x09 0x09 0x09 0x09 0x09 0x09 0x09 0x09 0x09 0x09 0x09 0x09 0x09
sent: 0x00 0x10 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08
sent: 0x00 0x10 0x07 0x07 0x07 0x07 0x07 0x07 0x07 0x07 0x07 0x07 0x07 0x07 0x07 0x07 0x07 0x07
sent: 0x00 0x10 0x06 0x06 0x06 0x06 0x06 0x06 0x06 0x06 0x06 0x06 0x06 0x06 0x06 0x06 0x06 0x06
sent: 0x00 0x10 0x05 0x05 0x05 0x05 0x05 0x05 0x05 0x05 0x05 0x05 0x05 0x05 0x05 0x05 0x05 0x05
sent: 0x00 0x10 0x04 0x04 0x04 0x04 0x04 0x04 0x04 0x04 0x04 0x04 0x04 0x04 0x04 0x04 0x04 0x04
sent: 0x00 0x10 0x03 0x03 0x03 0x03 0x03 0x03 0x03 0x03 0x03 0x03 0x03 0x03 0x03 0x03 0x03 0x03
sent: 0x00 0x10 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02
sent: 0x00 0x10 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01
sent: 0x00 0x10 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

This demonstration shows some of the graphical capabilities of the CFA835 by rendering an animated logo, clock, histogram, and scrolling text. Source code (C++, Qt 4.8 and created in QtCreator 2.5) is included in the utilities package.

DEMONSTRATION SOFTWARE

Demonstration software is available for free download under the ***Related*** tab on the website page for each CFA835 part number. Or click on the links in the software descriptions below. No registration is required for download.

cfTest

cfTest for Windows is testing and configuration software that works on all Crystalfontz Intelligent LCD modules. This software allows you to experiment with the command set for all Crystalfontz Smart LCDs.

Streaming communication based modules (CFA632, CFA634) and packet communication based modules (CFA533, CFA631, CFA633, CFA635, CFA735, CFA835) are supported.

CrystalControl2 (CC2)

CrystalControl2 for Windows displays a great variety of information to a Crystalfontz Intelligent LCD Module in a configurable way. We provide a User Manual and support through our forum.

Note: CC2 does not support any of the CFA835's graphic, custom font, or multiple FBSCAB capabilities.

Linux CLI Examples

CLI Example Software is a Linux compatible command-line demonstration program with C source code. 8K. ***Note:*** It will show as `/dev/ttyACMx` instead of `/dev/ttyUSBx`.

LCDproc is an open source project that supports many of the Crystalfontz displays. The CFA635 configuration should work with the CFA835.

SAMPLE CODE FOR RPM CALCULATION INFORMATION

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

```
bool HandleFanRPMReplyPacket(COMMAND_PACKET *packet, char *output)
{
    uint8_t  fbscab_index;
    uint8_t  fan_index;
    uint8_t  cycles;
    uint8_t  data_offset;
    uint8_t  timer_lsb;
    uint8_t  timer_msb;
    uint8_t  pulses_per_revolution;
    uint16_t timer_ticks;
    uint8_t  output_offset;
    float    fan_rpm;

    /*
    fan rpm query command reponse packet has the format of:
        type = 0x40 | 0x25 = 0x65 = 101
        data_length: 14
        data[0]: 2 (read fan tachometer speed)
        data[1]: FBSCAB module index
        data[2]: fan 1 number of fan tach cycles
        data[3]: fan 1 LSB of fan timer ticks
        data[4]: fan 1 MSB of fan timer ticks
        data[5]: fan 2 number of fan tach cycles
        data[6]: fan 2 LSB of fan timer ticks
        data[7]: fan 2 MSB of fan timer ticks
        data[8]: fan 3 number of fan tach cycles
        data[9]: fan 3 LSB of fan timer ticks
        data[10]: fan 3 MSB of fan timer ticks
        data[11]: fan 4 number of fan tach cycles
        data[12]: fan 4 LSB of fan timer ticks
        data[13]: fan 4 MSB of fan timer ticks
    */

    //check packet length
    if (packet->length != 14)
    {
        //unexpected packet length, should be 14 bytes
        return false;
    }

    //check the packets command number and type
    //  0x25 | 0x40 = FBSCAB Command Group | Reply Packet
    if (packet->command != (0x25 | 0x40))
    {
        //wrong packet command/type
        return false;
    }

    //check the packets sub-command type
    //  2 = Read fan tachometer speed
    if (packet->data[0] != 2)
    {
        //wrong packet sub-command value
        return false;
    }

    //get fbscab index from the packet
    fbscab_index = packet->data[1];

    //prepare output string
    output_offset = 0;
    output_offset += sprintf(&output[output_offset], "FBSCAB:%d - ", fbscab_index);
    //process packet data for the 4 fans
    for (fan_index = 0; fan_index < 4; fan_index++)
    {
        //data offset for fan_index data in the packet
        data_offset = 2 + (fan_index * 3);

        //prepare output string
        output_offset += sprintf(&output[output_offset], "FAN%d: ", fan_index);

        //get the fan data from the packet
        cycles = packet->data[data_offset];
        timer_lsb = packet->data[data_offset+1];
        timer_msb = packet->data[data_offset+2];
        timer_ticks = timer_lsb | (timer_msb << 8);

        //check fan cycles value
        if (cycles < 3)
        {
            //fan has stopped
            output_offset += sprintf(&output[output_offset], "STOPPED ");
            //next fan
            continue;
        }
        if (cycles < 4)
        {
            //fan is turning too slow to count RPM
            output_offset += sprintf(&output[output_offset], "SLOW ");
            //next fan
            continue;
        }
        if (cycles == 0xFF)
        {
            //unknown value
            output_offset += sprintf(&output[output_offset], "UNKNOWN ");
            //next fan
            continue;
        }

        //if we get to here, we have valid fan tach data
        //calculate fan RPM
        pulses_per_revolution = 2; //specific to each fan, most commonly 2
        fan_rpm = ((27692308L / pulses_per_revolution) * (cycles - 3)) / (float)tim-
er_ticks;

        //add RPM to output string
        output_offset += sprintf(&output[output_offset], "%5.2f ", fan_rpm);

        //done, next fan
    }

    //all done
    return true;
}
```

SAMPLE CODE FOR TEMPERATURE SENSOR REPORT

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

```
bool HandleTempReplyPacket(COMMAND_PACKET *packet, char *output)
{
    uint8_t  fbscab_index;
    uint8_t  sensor_index;
    uint8_t  temp_lsb;
    uint8_t  temp_msb;
    uint16_t temp_raw;
    uint8_t  crc_status;
    float    deg_c;
    float    deg_f;

    /*
    temperature query command reponse packet has the format of:
     type = 0x40 | 0x25 = 0x65 = 101
     data_length: 5
     data[0]: 4 (read WR-DOW-Y17 temperature)
     data[1]: FBSCAB module index
     data[2]: DOW device index (0-15)
     data[3]: LSB of temperature data
     data[4]: MSB of temperature data
    */

    //check the packets command number and type
    //  0x25 | 0x40 = FBSCAB Command Group | Reply Packet
    if (packet->command != (0x25 | 0x40))
    {
        //wrong packet type
        return false;
    }

    //check the packets sub-command type
    //  4 = Read WR-DOW-Y17 temperature
    if (packet->data[0] != 4)
    {
        //wrong packet type
        return false;
    }

    //get fbscab & temp sensor index from the packet
    fbscab_index = packet->data[1];
    sensor_index = packet->data[2];

    //get raw temperature data from the packet
    temp_lsb = packet->data[3];
    temp_msb = packet->data[4];
    temp_raw = temp_lsb | (temp_msb << 8);

    //check temperature data CRC flags
    crc_status = temp_raw << 14;
    if (crc_status == 1)
    {
        //CRC check failed
        return false;
    }
    if (crc_status == 2)
    {
        //no sensor in this location
        //this should never happen
        return false;
    }
    if (crc_status == 3)
    {
        //no valid data from this sensor yet
        return false;
    }

    //if we get to here, crc status==0, so temperature data is valid

    //calculate temperature
    deg_c = temp_raw / (float)16.0;
    deg_f = (deg_c * 9.0) / 5.0 + 32.0;

    //return text
    sprintf(output, "FBSCAB:%d SENSOR:%d TEMP_DEGC:%0.2f TEMP_DEGF:%0.2f", fbscab_index,
sensor_index, deg_c, deg_f);

    //done
    return true;
}
```

SAMPLE CODE FOR FONT FILE FORMAT

The following source code is C pseudo-code. It will need to be modified to fit your application. The structures are little-endian and are byte-aligned packed.

```
//font flags
#define FR_None          0x00
#define FR_AntiAliased   0x01
#define FR_Proportional  0x02
#define FR_MergeAA       0x04
#define FR_Sharpen       0x08
#define FR_CenterScreen  0x10

//char flags
#define FR_NoChar        0x00
#define FR_HasCharacter  0x01
#define FR_IsCustomChar  0x02

//version information
#define FR_FileID        "CFFF"
#define FR_FileVersion   105

typedef struct
{
   char        ID[4];            //FR_FileID
   uint16_t    Version;          //FR_FileVersion

   //rendering data
   uint8_t     DataWidth;        //character width in pixels
   uint8_t     DataHeight;       //character height in pixels
   uint16_t    StartChar;        //UTF16 character number of first character in font file
   uint16_t    EndChar;          //UTF16 character number of last character in font file
   uint8_t     CharSpaceRight;   //extra character spacing on the right
   uint8_t     CharSpaceBelow;   //extra character spacing below
   uint8_t     ScreenSpaceLeft;  //offset character positions to the right by X pixels
   uint8_t     ScreenSpaceTop;   //offset character positions downwards by X pixels
   uint8_t     Flags;            //font flags

   //font editor use only
   //these values can be undefined, CFA835 module disregards these values
   char        OrigFont[128];
   uint8_t     TrimTop;
   uint8_t     TrimBottom;
   uint8_t     TrimLeft;
   uint8_t     TrimRight;
} FR_FileHeader;

typedef struct
{
   uint8_t    CharFlags;       //character flags
   uint8_t    CharWidth;       //character width in pixels (for proportional fonts)
   uint8_t    CharData[FR_FileHeader.DataWidth * FR_FileHeader.DataHeight];
} FR_Character;

typedef struct
{
   FR_FileHeader  Header;
   FR_Character   Characters[FR_FileHeader.EndChar - FR_FileHeader.StartChar];
} FR_FontFile;
```

ALGORITHMS TO CALCULATE THE CRC

Below are eight sample algorithms that will calculate the CRC of a CFA835 packet. Some of the algorithms were contributed by forum members and originally written for the CFA631. The CRC used in the CFA835 is the same one that is used in IrDA, which came from PPP, which seems to be related to a CCITT (ref: Network Working Group Request for Comments: 1171) standard. At that point, the trail was getting a bit cold and diverged into several referenced articles and papers, dating back to 1983.

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

Algorithm 1: “C” Table Implementation

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

```
//This code is from the IRDA LAP documentation, which appears to
//have been copied from PPP:
//
// http://irda.affiniscape.com/associations/2494/files/Specifications/IrLAP11_Plus_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] =
    {0x00000,0x01189,0x02312,0x0329B,0x04624,0x057AD,0x06536,0x074BF,
     0x08C48,0x09DC1,0x0AF5A,0x0BED3,0x0CA6C,0x0DBE5,0x0E97E,0x0F8F7,
     0x01081,0x00108,0x03393,0x0221A,0x056A5,0x0472C,0x075B7,0x0643E,
     0x09CC9,0x08D40,0x0BFDB,0x0AE52,0x0DAED,0x0CB64,0x0F9FF,0x0E876,
     0x02102,0x0308B,0x00210,0x01399,0x06726,0x076AF,0x04434,0x055BD,
     0x0AD4A,0x0BCC3,0x08E58,0x09FD1,0x0EB6E,0x0FAE7,0x0C87C,0x0D9F5,
     0x03183,0x0200A,0x01291,0x00318,0x077A7,0x0662E,0x054B5,0x0453C,
     0x0BDCB,0x0AC42,0x09ED9,0x08F50,0x0FBEF,0x0EA66,0x0D8FD,0x0C974,
     0x04204,0x0538D,0x06116,0x0709F,0x00420,0x015A9,0x02732,0x036BB,
     0x0CE4C,0x0DFC5,0x0ED5E,0x0FCD7,0x08868,0x099E1,0x0AB7A,0x0BAF3,
     0x05285,0x0430C,0x07197,0x0601E,0x014A1,0x00528,0x037B3,0x0263A,
     0x0DECD,0x0CF44,0x0FDDF,0x0EC56,0x098E9,0x08960,0x0BBFB,0x0AA72,
     0x06306,0x0728F,0x04014,0x0519D,0x02522,0x034AB,0x00630,0x017B9,
     0x0EF4E,0x0FEC7,0x0CC5C,0x0DDD5,0x0A96A,0x0B8E3,0x08A78,0x09BF1,
     0x07387,0x0620E,0x05095,0x0411C,0x035A3,0x0242A,0x016B1,0x00738,
     0x0FFCF,0x0EE46,0x0DCDD,0x0CD54,0x0B9EB,0x0A862,0x09AF9,0x08B70,
     0x08408,0x09581,0x0A71A,0x0B693,0x0C22C,0x0D3A5,0x0E13E,0x0F0B7,
     0x00840,0x019C9,0x02B52,0x03ADB,0x04E64,0x05FED,0x06D76,0x07CFF,
     0x09489,0x08500,0x0B79B,0x0A612,0x0D2AD,0x0C324,0x0F1BF,0x0E036,
     0x018C1,0x00948,0x03BD3,0x02A5A,0x05EE5,0x04F6C,0x07DF7,0x06C7E,
     0x0A50A,0x0B483,0x08618,0x09791,0x0E32E,0x0F2A7,0x0C03C,0x0D1B5,
     0x02942,0x038CB,0x00A50,0x01BD9,0x06F66,0x07EEF,0x04C74,0x05DFD,
     0x0B58B,0x0A402,0x09699,0x08710,0x0F3AF,0x0E226,0x0D0BD,0x0C134,
     0x039C3,0x0284A,0x01AD1,0x00B58,0x07FE7,0x06E6E,0x05CF5,0x04D7C,
     0x0C60C,0x0D785,0x0E51E,0x0F497,0x08028,0x091A1,0x0A33A,0x0B2B3,
     0x04A44,0x05BCD,0x06956,0x078DF,0x00C60,0x01DE9,0x02F72,0x03EFB,
     0x0D68D,0x0C704,0x0F59F,0x0E416,0x090A9,0x08120,0x0B3BB,0x0A232,
     0x05AC5,0x04B4C,0x079D7,0x0685E,0x01CE1,0x00D68,0x03FF3,0x02E7A,
     0x0E70E,0x0F687,0x0C41C,0x0D595,0x0A12A,0x0B0A3,0x08238,0x093B1,
     0x06B46,0x07ACF,0x04854,0x059DD,0x02D62,0x03CEB,0x00E70,0x01FF9,
     0x0F78F,0x0E606,0x0D49D,0x0C514,0x0B1AB,0x0A022,0x092B9,0x08330,
     0x07BC7,0x06A4E,0x058D5,0x0495C,0x03DE3,0x02C6A,0x01EF1,0x00F78};

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

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

Algorithm 2: “C” Bit Shift Implementation

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

```
typedef unsigned char ubyte;
typedef unsigned short word;

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

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

      //If the low bit of the current CRC accumulator was set
      //before the shift, then we need to XOR the accumulator
      //with the polynomial (center 16 bits of 0x00840800)
      if(newCRC&0x00000080)
```

```
          newCRC^=0x00840800;
        //Shift the data byte to put the next bit of the stream
        //into position 0.
        data>>=1;
        }
      }

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

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

Algorithm 2B: “C” Improved Bit Shift Implementation

This is a simplified algorithm that implements the CRC.

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

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

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

Algorithm 3: “PIC Assembly” Bit Shift Implementation

This routine was graciously donated by one of our customers, originally for the CFA635.

```
;====================================================================
; Crystalfontz CFA835 PIC CRC Calculation Example
;
; This example calculates the CRC for the hard coded example provided
; in the documentation.
;
; It uses "This is a test. " as input and calculates the proper CRC
; of 0x93FA.
;=====================================================================
#include "p16f877.inc"
;=====================================================================
; CRC16 equates and storage
;---------------------------------------------------------------------
accuml      equ     40h         ; BYTE - CRC result register high byte
accumh      equ     41h         ; BYTE - CRC result register high low byte
datareg     equ     42h         ; BYTE - data register for shift
j           equ     43h         ; BYTE - bit counter for CRC 16 routine
Zero        equ     44h         ; BYTE - storage for string memory read
index       equ     45h         ; BYTE - index for string memory read
savchr      equ     46h         ; BYTE - temp storage for CRC routine
;
seedlo      equ     021h        ; initial seed for CRC reg lo byte
seedhi      equ     0F3h        ; initial seed for CRC reg hi byte
;
polyL       equ     008h        ; polynomial low byte
polyH       equ     084h        ; polynomial high byte
;=====================================================================
;  CRC Test Program
;---------------------------------------------------------------------
        org         0           ; reset vector = 0000H
;
        clrf        PCLATH      ; ensure upper bits of PC are cleared
        clrf        STATUS      ; ensure page bits are cleared
        goto        main        ; jump to start of program
;
; ISR Vector
;
        org         4           ; start of ISR
        goto        $           ; jump to ISR when coded
;
        org         20          ; start of main program
main
        movlw       seedhi      ; setup intial CRC seed value.
        movwf       accumh      ; This must be done prior to
        movlw       seedlo      ; sending string to CRC routine.
        movwf       accuml      ;
        clrf        index       ; clear string read variables
;
main1
        movlw       HIGH InputStr   ; point to LCD test string
        movwf       PCLATH      ; latch into PCL
        movfw       index       ; get index
        call        InputStr    ; get character
        movwf       Zero        ; setup for terminator test
        movf        Zero,f      ; see if terminator
        btfsc       STATUS,Z    ; skip if not terminator
         goto         main2        ; else terminator reached, jump out of loop
        call        CRC16       ; calculate new crc
        call        SENDUART    ; send data to LCD
        incf        index,f     ; bump index
        goto        main1       ; loop
;
main2
        movlw       00h         ; shift accumulator 16 more bits.
        call        CRC16       ; This must be done after sending
        movlw       00h         ; string to CRC routine.
        call        CRC16       ;
;
        comf        accumh,f    ; invert result
        comf        accuml,f    ;
;
        movfw       accuml      ; get CRC low byte
        call        SENDUART    ; send to LCD
        movfw       accumh      ; get CRC hi byte
        call        SENDUART    ; send to LCD
;
stop    goto        stop           ; word result of 0x93FA is in accumh/accuml
;=====================================================================
; calculate CRC of input byte
;---------------------------------------------------------------------
CRC16
        movwf       savchr      ; save the input character
        movwf       datareg     ; load data register
        movlw       .8          ; setup number of bits to test
        movwf       j           ; save to incrementor
_loop
        clrc                    ; clear carry for CRC register shift
         rrf        datareg,f   ; perform shift of data into CRC register
        rrf         accumh,f    ;
        rrf         accuml,f    ;
        btfss       STATUS,C    ; skip jump if if carry
        goto        _notset     ; otherwise goto next bit
        movlw       polyL       ; XOR poly mask with CRC register
        xorwf       accuml,F    ;
        movlw       polyH       ;
        xorwf       accumh,F    ;
_notset
        decfsz      j,F         ; decrement bit counter
        goto        _loop       ; loop if not complete
        movfw       savchr      ; restore the input character
        return                  ; return to calling routine
;=====================================================================
; USER SUPPLIED Serial port transmit routine
;---------------------------------------------------------------------
SENDUART
        return                  ; put serial xmit routine here
;=====================================================================
; test string storage
;---------------------------------------------------------------------
        org     0100h
;
InputStr
        addwf   PCL,f
        dt      7h,10h,"This is a test. ",0
;
;=====================================================================
        end
```

Algorithm 4: “Visual Basic” Table Implementation

Visual BASIC has its own challenges as a language (such as initializing static arrays), and it is also challenging to use Visual BASIC to work with “binary” (arbitrary length character data possibly containing nulls—such as the “data” portion of the CFA835 packet) data. This routine was adapted from the C table implementation. The complete project can be found in our forums.

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

Private Type WORD
   Lo As Byte
   Hi As Byte
```

```
End Type

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

Dim crcLookupTable(256) As WORD

Private Sub MSComm_OnComm()
'Leave this here
End Sub

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

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

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

Algorithm 5: “Java” Table Implementation

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

```
public class CRC16 extends Object
  {
  public static void main(String[] args)
    {
    byte[] data = new byte[2];
    // hw - fw
    data[0] = 0x01;
    data[1] = 0x00;
    System.out.println("hw -fw req");
    System.out.println(Integer.toHexString(compute(data)));

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

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

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

    // set line 1
    data = new byte[18];
    data[0] = 0x07;
    data[1] = 0x10;
    String text = "Test Test Test  ";
    byte[] textByte = text.getBytes();
    for (int i=0; i < text.length(); i++) data[i+2] = textByte[i];
    System.out.println("text 1");
    System.out.println(Integer.toHexString(compute(data)));
    }
  private CRC16()
    {
    }
  private static final int[] crcLookupTable =
    {
    0x00000,0x01189,0x02312,0x0329B,0x04624,0x057AD,0x06536,0x074BF,
    0x08C48,0x09DC1,0x0AF5A,0x0BED3,0x0CA6C,0x0DBE5,0x0E97E,0x0F8F7,
    0x01081,0x00108,0x03393,0x0221A,0x056A5,0x0472C,0x075B7,0x0643E,
    0x09CC9,0x08D40,0x0BFDB,0x0AE52,0x0DAED,0x0CB64,0x0F9FF,0x0E876,
    0x02102,0x0308B,0x00210,0x01399,0x06726,0x076AF,0x04434,0x055BD,
    0x0AD4A,0x0BCC3,0x08E58,0x09FD1,0x0EB6E,0x0FAE7,0x0C87C,0x0D9F5,
    0x03183,0x0200A,0x01291,0x00318,0x077A7,0x0662E,0x054B5,0x0453C,
    0x0BDCB,0x0AC42,0x09ED9,0x08F50,0x0FBEF,0x0EA66,0x0D8FD,0x0C974,
    0x04204,0x0538D,0x06116,0x0709F,0x00420,0x015A9,0x02732,0x036BB,
    0x0CE4C,0x0DFC5,0x0ED5E,0x0FCD7,0x08868,0x099E1,0x0AB7A,0x0BAF3,
    0x05285,0x0430C,0x07197,0x0601E,0x014A1,0x00528,0x037B3,0x0263A,
    0x0DECD,0x0CF44,0x0FDDF,0x0EC56,0x098E9,0x08960,0x0BBFB,0x0AA72,
    0x06306,0x0728F,0x04014,0x0519D,0x02522,0x034AB,0x00630,0x017B9,
    0x0EF4E,0x0FEC7,0x0CC5C,0x0DDD5,0x0A96A,0x0B8E3,0x08A78,0x09BF1,
    0x07387,0x0620E,0x05095,0x0411C,0x035A3,0x0242A,0x016B1,0x00738,
    0x0FFCF,0x0EE46,0x0DCDD,0x0CD54,0x0B9EB,0x0A862,0x09AF9,0x08B70,
    0x08408,0x09581,0x0A71A,0x0B693,0x0C22C,0x0D3A5,0x0E13E,0x0F0B7,
    0x00840,0x019C9,0x02B52,0x03ADB,0x04E64,0x05FED,0x06D76,0x07CFF,
    0x09489,0x08500,0x0B79B,0x0A612,0x0D2AD,0x0C324,0x0F1BF,0x0E036,
    0x018C1,0x00948,0x03BD3,0x02A5A,0x05EE5,0x04F6C,0x07DF7,0x06C7E,
    0x0A50A,0x0B483,0x08618,0x09791,0x0E32E,0x0F2A7,0x0C03C,0x0D1B5,
    0x02942,0x038CB,0x00A50,0x01BD9,0x06F66,0x07EEF,0x04C74,0x05DFD,
    0x0B58B,0x0A402,0x09699,0x08710,0x0F3AF,0x0E226,0x0D0BD,0x0C134,
    0x039C3,0x0284A,0x01AD1,0x00B58,0x07FE7,0x06E6E,0x05CF5,0x04D7C,
    0x0C60C,0x0D785,0x0E51E,0x0F497,0x08028,0x091A1,0x0A33A,0x0B2B3,
    0x04A44,0x05BCD,0x06956,0x078DF,0x00C60,0x01DE9,0x02F72,0x03EFB,
    0x0D68D,0x0C704,0x0F59F,0x0E416,0x090A9,0x08120,0x0B3BB,0x0A232,
    0x05AC5,0x04B4C,0x079D7,0x0685E,0x01CE1,0x00D68,0x03FF3,0x02E7A,
    0x0E70E,0x0F687,0x0C41C,0x0D595,0x0A12A,0x0B0A3,0x08238,0x093B1,
    0x06B46,0x07ACF,0x04854,0x059DD,0x02D62,0x03CEB,0x00E70,0x01FF9,
    0x0F78F,0x0E606,0x0D49D,0x0C514,0x0B1AB,0x0A022,0x092B9,0x08330,
    0x07BC7,0x06A4E,0x058D5,0x0495C,0x03DE3,0x02C6A,0x01EF1,0x00F78
    };
  public static int compute(byte[] data)
    {
    int newCrc = 0x0FFFF;
    for (int i = 0; i < data.length; i++)
      {
      int lookup = crcLookupTable[(newCrc ^ data[i]) & 0xFF];
      newCrc = (newCrc >> 8) ^ lookup;
      }
    return(~newCrc);
    }
  }
```

Algorithm 6: “Perl” Table Implementation

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

```
#!/usr/bin/perl

use strict;

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

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

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

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

my $valid_crc = '5584' ;

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

my $crc = 0xFFFF ;

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

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

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

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

Algorithm 7: For PIC18F8722 or PIC18F2685

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

```
; CRC Algorithm for CrystalFontz CFA-635 display (DB535)
; This code written for PIC18F8722 or PIC18F2685
;
; Your main focus here should be the ComputeCRC2 and
; CRC16_ routines
;
;====================================================================
ComputeCRC2:
        movlb   RAM8
        movwf   dsplyLPCNT      ;w has the byte count
nxt1_dsply:
        movf    POSTINC1,w
        call    CRC16_
        decfsz  dsplyLPCNT
        goto    nxt1_dsply
        movlw   .0              ; shift accumulator 16 more bits
        call    CRC16_
        movlw   .0
        call    CRC16_
        comf    dsplyCRC,F        ; invert result
        comf    dsplyCRC+1,F
        return
;====================================================================
CRC16_ movwf:
        dsplyCRCData              ; w has byte to crc
        movlw   .8
        movwf   dsplyCRCCount
_cloop:
        bcf     STATUS,C          ; clear carry for CRC register shift
        rrcf    dsplyCRCData,f    ; perform shift of data into CRC
                                  ;register
        rrcf    dsplyCRC,F
        rrcf    dsplyCRC+1,F
        btfss   STATUS,C          ; skip jump if carry
        goto    _notset           ; otherwise goto next bit
        movlw   0x84
        xorwf   dsplyCRC,F
        movlw   0x08              ; XOR poly mask with CRC register
        xorwf   dsplyCRC+1,F
_notset:
        decfsz  dsplyCRCCount,F ; decrement bit counter
        bra _cloop                ; loop if not complete
        return
;====================================================================
; example to clear screen
dsplyFSR1_TEMP  equu    0x83A    ; 16-bit save for FSR1 for display
                                  ; message handler
dsplyCRC        equ     0x83C    ; 16-bit CRC (H/L)
dsplyLPCNT      equ     0x83E    ; 8-bit save for display message
                                  ; length - CRC
dsplyCRCData    equ     0x83F    ; 8-bit CRC data for display use
dsplyCRCCount   equ     0x840    ; 8-bit CRC count for display use
SendCount       equ     0x841    ; 8-bit byte count for sending to
                                  ; display
RXBUF2          equ     0x8C0    ; 32-byte receive buffer for
                                  ; Display
TXBUF2          equ     0x8E0    ; 32-byte transmit buffer for
                                  ; Display
;--------------------------------------------------------------------
ClearScreen:
        movlb   RAM8
        movlw   .0
        movwf   SendCount
        movlw   0xF3
        movwf   dsplyCRC          ; seed ho for CRC calculation
        movlw   0x21
        movwf   dsplyCRC+1        ; seen lo for CRC calculation
        call    ClaimFSR1
        movlw   0x06
        movwf   TXBUF2
        LFSR    FSR1,TXBUF2
        movf    SendCount,w
        movwf   TXBUF2+1          ; message data length
        call    BMD1
        goto    SendMsg
;====================================================================
; send message via interrupt routine. The code is made complex due
; to the limited FSR registers and extended memory space used
;
; example of sending a string to column 0, row 0
;--------------------------------------------------------------------
SignOnL1:
        call    ClaimFSR1
        lfsr    FSR1,TXBUF2+4     ; set data string position
        SHOW    C0R0,BusName      ; move string to TXBUF2
        movlw   .2                ;
        addwf   SendCount       ;
        movff   SendCount,TXBUF2+1
                                ; insert message data length
        call    BuildMsgDSPLY
        call    SendMsg
        return
;===================================================================
; BuildMsgDSPLY used to send a string to LCD
;-------------------------------------------------------------------
BuildMsgDSPLY:
        movlw   0xF3
        movwf   dsplyCRC        ; seed hi for CRC calculation
        movlw   0x21
        movwf   dsplyCRC+1      ; seed lo for CRC calculation
        LFSR    FSR1,TXBUF2     ; point at transmit buffer
        movlw   0x1F            ; command to send data to LCD
        movwf   TXBUF2          ; insert command byte from us to
                                ; CFA-635
        BMD1    movlw .2
        ddwf    SendCount,w     ; + overhead
        call    ComputeCRC2     ; compute CRC of transmit message
        movf    dsplyCRC+1,w
        movwf   POSTINC1        ; append CRC byte
        movf    dsplyCRC,w
        movwf   POSTINC1        ; append CRC byte
        return
;===================================================================
SendMsg:
        call    ReleaseFSR1
        LFSR    FSR0,TXBUF2
        movff   FSR0H,irptFSR0
        movff   FSR0L,irptFSR0+1
                                ; save interrupt use of FSR0
        movff   SendCount,TXBUSY2
        bsf     PIE2,TX2IE
                                ; set transmit interrupt enable
                                ; (bit 4)
        return
;===================================================================
; macro to move string to transmit buffer
SHOW macro src, stringname
        call    src
        MOVLF   upper stringname, TBLPTRU
        MOVLF   high stringname, TBLPTRH
        MOVLF   low stringname, TBLPTRL
        call    MOVE_STR
        endm
;===================================================================
MOVE_STR:
        tblrd   *+
        movf    TABLAT,w
        bz      ms1b
        movwf   POSTINC1
        incf    SendCount
        goto    MOVE_STR
ms1b:
        return
;===================================================================
```

APPENDIX C: VIBRATION TEST REPORT

KeyTronicEMS
Trust. Commitment. Results.

Test: Sine & Random Vibration **Reliability Laboratory**

Originator: Brent Crosby – Crystalfontz America
Test Coordinator: Larry Bettinger - lbetting@keytronic.com 509-927-5577
Test Started: April 9, 2013
Test Completed: April 10, 2013

Summary:

The following Crystalfontz America samples were submitted for operational vibration testing:

Sample Description	S/N
533 Yellow	1148533YYHD063605
533 Blue	1234533TMITD075774
633 Yellow	1037633YYH297069
633 White	1217633TFHD356000
735 Yellow	1212735TFK0002778
735 White	1212735TFK0002778
2x CFA-10036 ver. 1.0	Pilot run samples, no S/N assigned.
2x CFA-10037 ver. 1.0	Pilot run samples, no S/N assigned.

CFA835 shares hardware with CFA735.

Test Conditions:

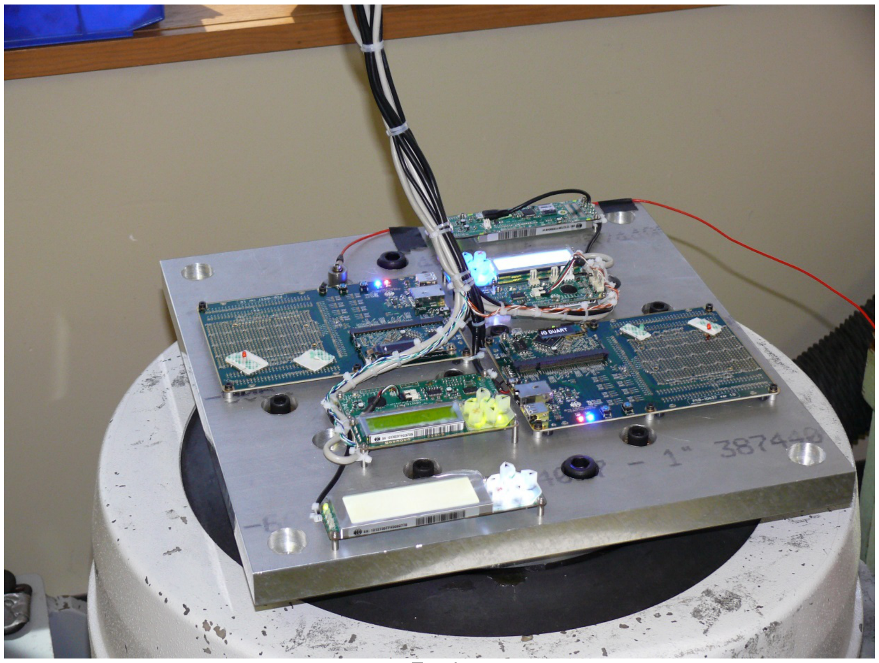
The samples were mounted to a customer's fixture plate which was bolted directly to the slip table for the X and Y-axes. For the Z-axis the fixture was bolted to the tester with a small aluminum coupling plate. The vibration testing was performed on a Ling Dynamic Systems V730 vibrator with a Data Physics SignalStar Scalar vibration control system version 2.2.923. The samples were subjected to following profiles:

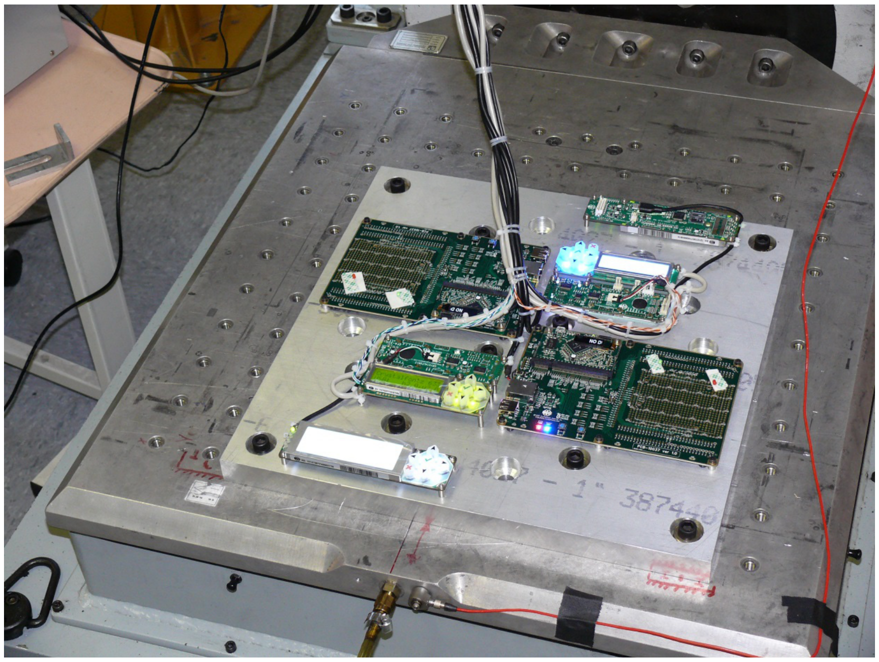
- GR-63-CORE 5.4.2, Office Vibration, Alternative Test: 5-100-5 Hz at 1.0 g with a sweep rate of .25 octave/minute, 35 minutes per axis.
- MIL-STD 810F, Figure 514C-17, Random: 1 hour per axis.
- MIL-STD 810F, Figure 514C-18, Sine: 1 hour per axis.

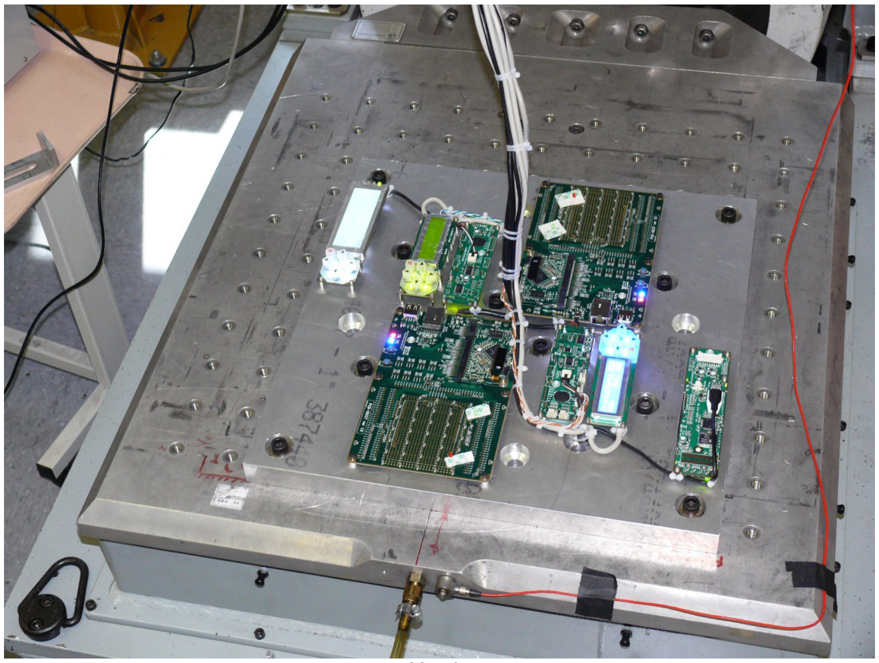
Equipment used:

Equipment	Model	S/N	Calibration Due Date
Endevco Control Accelerometer	7221	AM67	12-03-13
Endevco Charge amplifier	2721B	ER01	12-03-13
Data Physics Vibration controller	SignalStar Scalar	74244	05-29-13

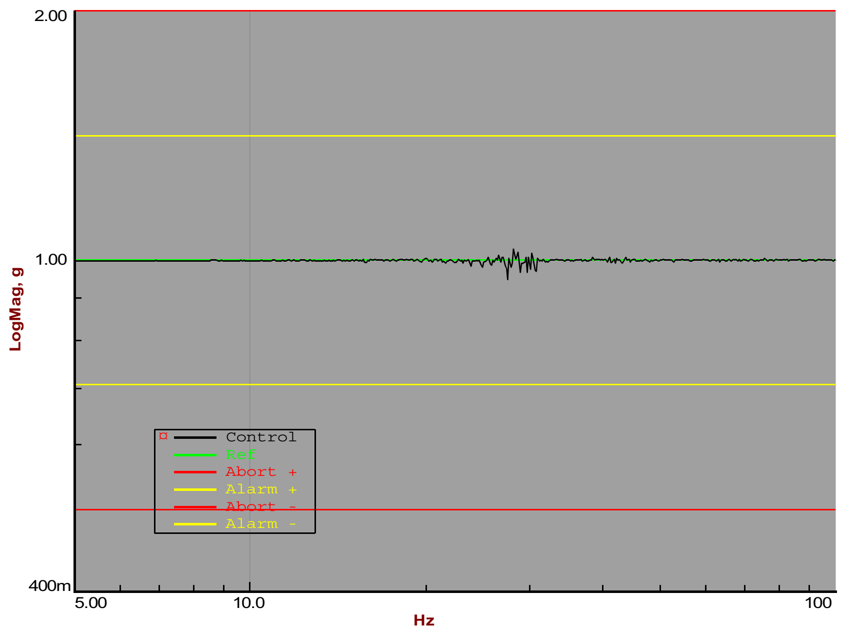
Test Setups:


Z-axis

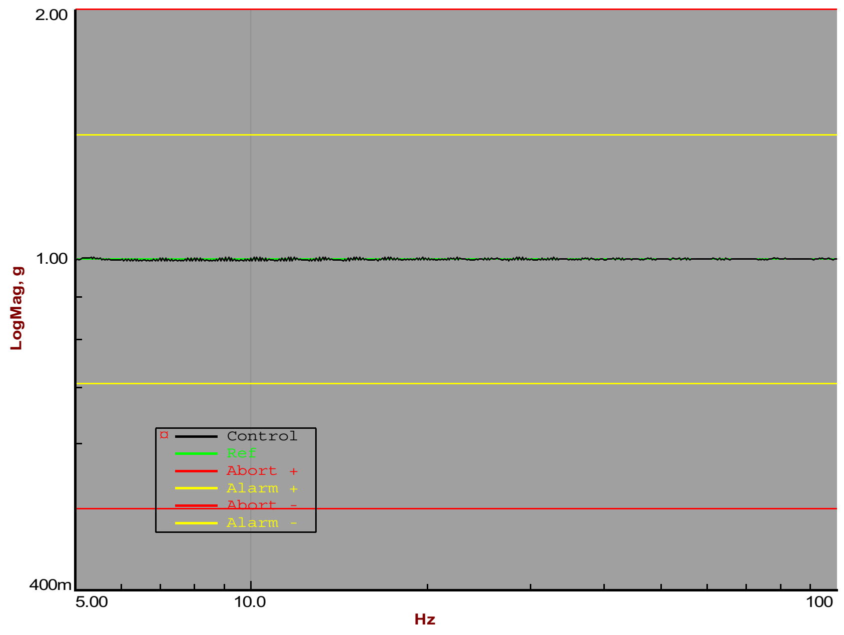

Y-axis


X-axis

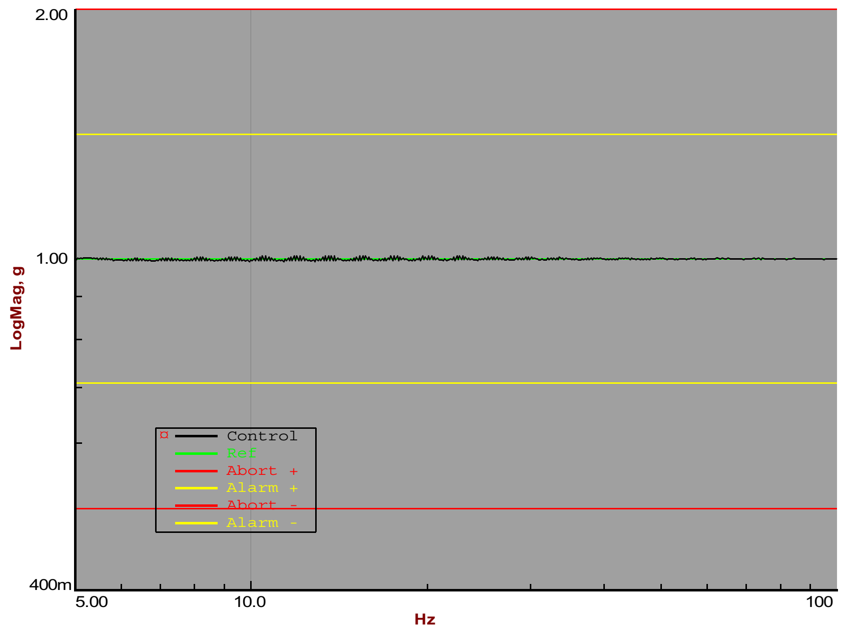
Control accelerometer vibration level graphs:



Z-axis (GR-63-CORE 5.4.2, Office Vibration, Alternative Test)

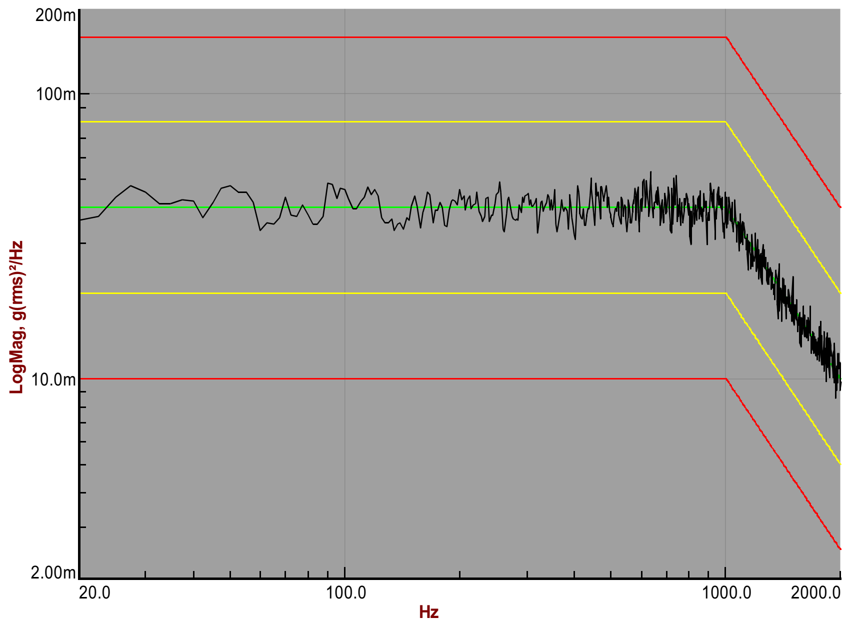


Y-axis (GR-63-CORE 5.4.2, Office Vibration, Alternative Test)

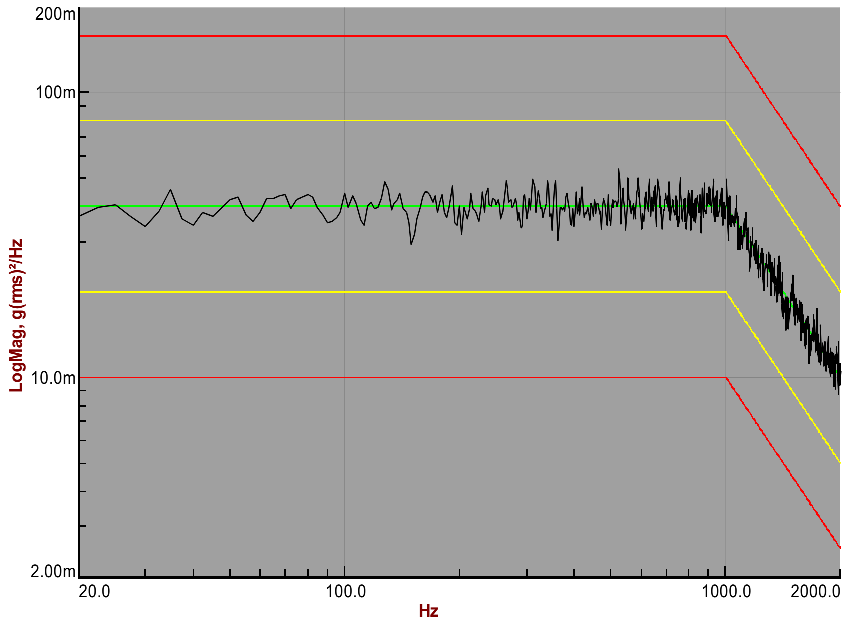


X-axis (GR-63-CORE 5.4.2, Office Vibration, Alternative Test)

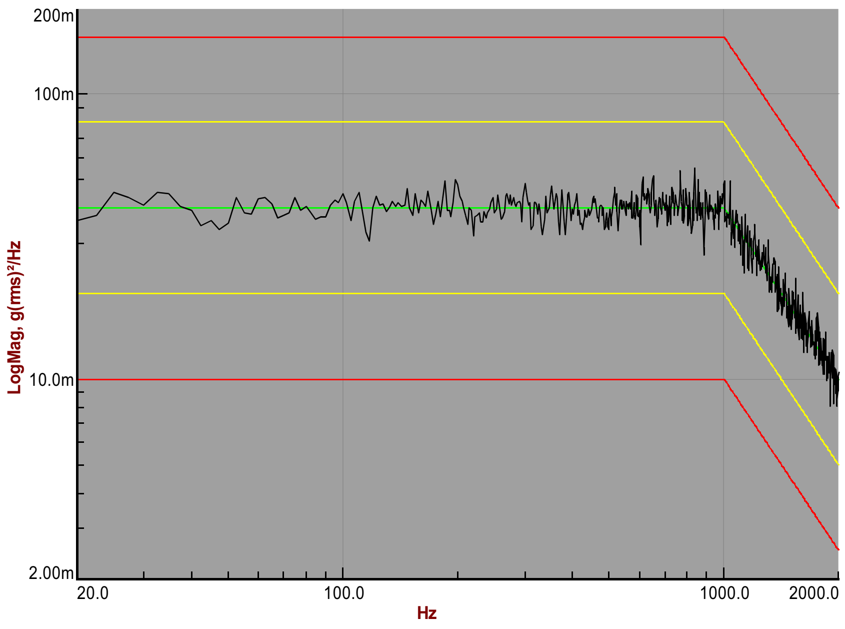
.



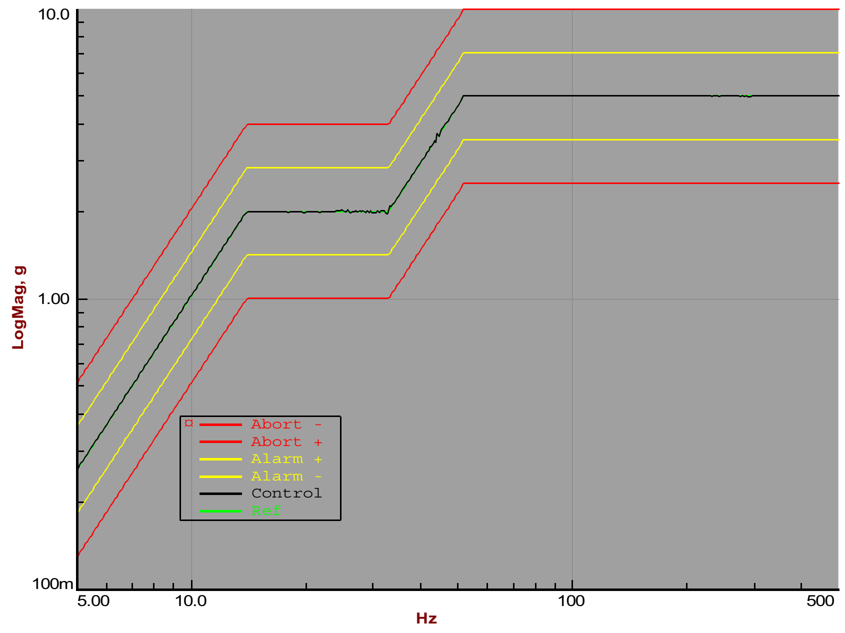
X-axis (MIL-STD 810F, Figure 514C-17, Random)



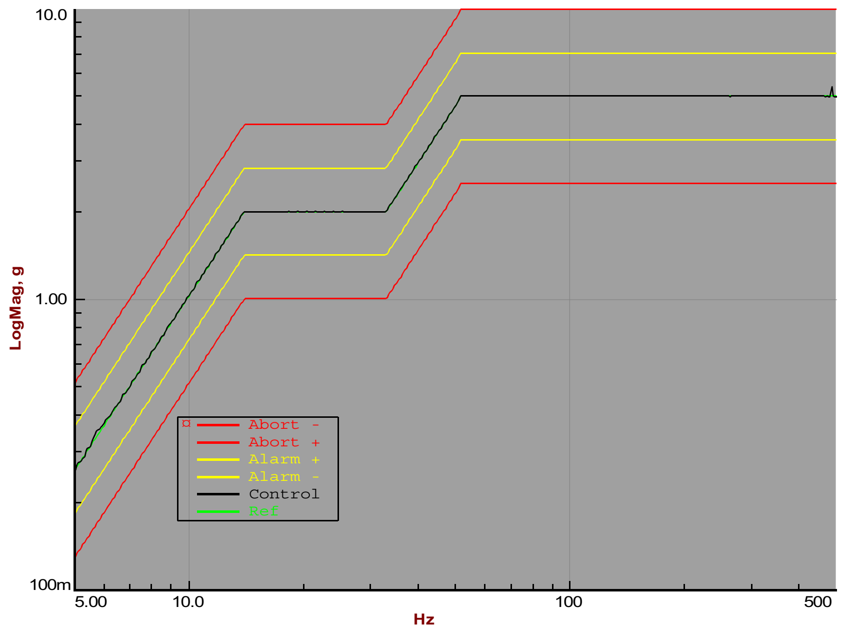
Y-axis (MIL-STD 810F, Figure 514C-17, Random)



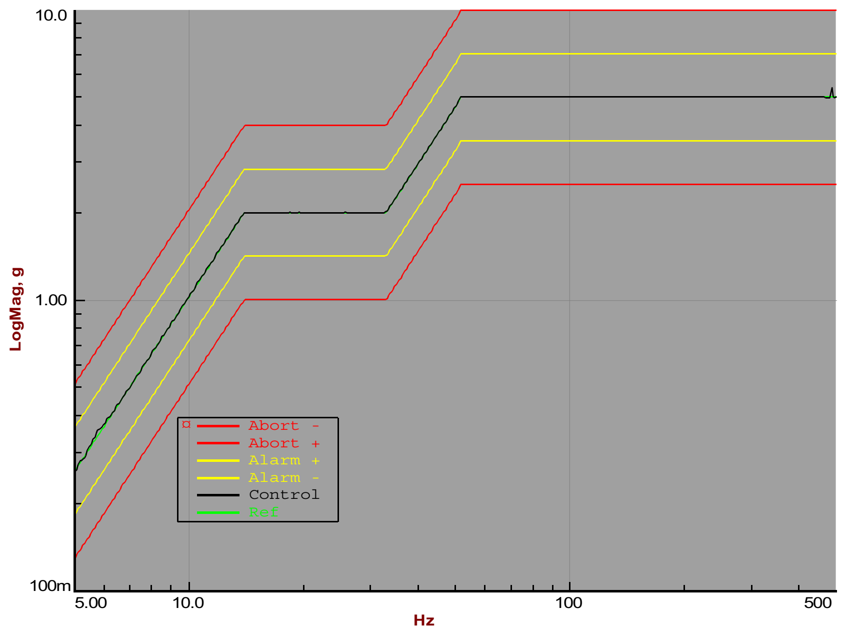
Z-axis (MIL-STD 810F, Figure 514C-17, Random)



Z-axis (MIL-STD 810F, Figure 514C-18, Sine)



Y-axis (MIL-STD 810F, Figure 514C-18, Sine)



X-axis (MIL-STD 810F, Figure 514C-18, Sine)