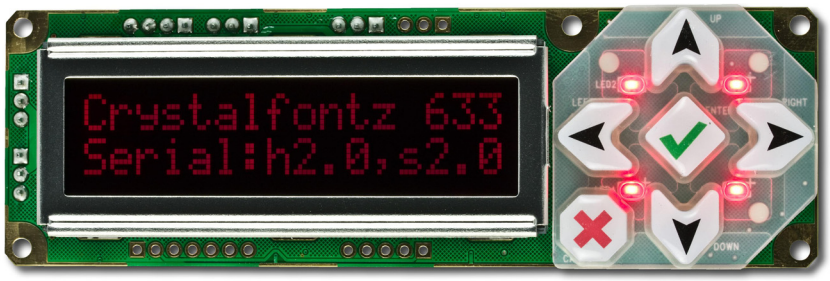
Crystalfontz America, Incorporated

INTELLIGENT LCD MODULE SPECIFICATIONS



Data Sheet Release 2012/02/21

for

CFA633-RDI-KS

Hardware Version: v2.0
Firmware Version: s2v1

Crystalfontz America, Incorporated

12412 East Saltese Avenue
Spokane Valley, WA 99216-0357

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

DATA SHEET REVISION HISTORY

For information about firmware and hardware revisions, see the Part Change Notifications (PCNs) under "News" in our website's navigation bar. To see the most recent PCN for the CFA633 family at the time of this Data Sheet release, see PCN #10402.

Data Sheet version: **2012/02/21**

- Added section on Accessories (Pg. 12).This section includes some useful Crystalfontz cables in Buy Cables Separately (Pg. 13).
- Illustration for Keypad Detail Drawing (Pg. 18) was improved to include more dimensions.
- Added section on DC Characteristics (Pg. 24).
- Added section on LED BACKLIGHT INFORMATION (Pg. 24). Please read this important information.
- In command 1 (0x01): Get Hardware & Firmware Version (Pg. 38), changed revision numbers from

```
data[] = "CFA633:hX.X,yYvY"

XvX is the hardware revision, "2.0" for example
YvY is the firmware version, "s2.0" for example
```

to

```
data[] = "CFA633:h#.#,y#v#"

X.X is the hardware revision, "h2.0" for example
Y.Y is the firmware version, "s2v1" for example
```

- In command 4 (0x04): Store Current State As Boot State (Pg. 39), added the following note: *"If the current state and the boot state do not match after saving, the module will return an error instead of an ACK. In this unlikely error case, the boot state will be undefined."*
- Command 10 (0x0A): Read 8 Bytes of LCD Memory (Pg. 43), corrected "`data[0]` is the address code native to the LCD controller"

from

```
0x80 (\128) to 0x93 (\147) for DDRAM, line 1
0xC0 (\192) to 0xD3 (\211) for DDRAM, line 2
```

to

```
0x80 (\128) to 0x8F (\143) for DDRAM, line 1
0xC0 (\192) to 0xCF (\207) for DDRAM, line 2
```

- In command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54), changed description to reflect enhancement in firmware. From PCN #10402, *"Added auto polarity switch. Improved pulse length functionality, and added feature to hold until system off during power down."* Also corrected "`data[1]: (optional) length of power on & off pulses in 1/32 second`" *from*

```
255 = 8 sec
```

to

```
254 = pulse length
255 = pin held until power drops (length of time pin is held in active state)
```

- In Module Reliability (Pg. 63), please read important note added to the top of this section. Also, we increased backlight specification from "70%" to "90%".
- In Algorithm 2B: "C" Improved Bit Shift Implementation (Pg. 74), added closing } to final line.
- Additional minor text changes were made to make text easier to understand.

Data Sheet Revision History (Continued)	
2011/06/13	Data Sheet Version: v3.2 Changes since last revision: ● Wherever listed, changed part numbers to reflect improvement from “standard” to “wide temperature”. ● Operating temperature range from a “minimum 0°C to maximum +50°C” to “minimum -20°C to +70°C maximum”. - Storage temperature range from a “minimum -10°C to maximum +60°C” to “minimum -30°C to +80°C maximum”. ● Wherever listed, changed classification code in red variant part numbers from “CFA633-*RMC*-KS” to “CFA633-*RDI-KS*”. The change from “M” to “C” reflects an improvement in the glass from “STN” to “FFSTN”. ● Added new variants CFA633-TFH-KS and CFA633-TFH-KU to the Module Classification Information table. ● In Physical Characteristics, decreased weight from 45 grams (typical) to 42 grams (typical). ● Revised illustrations in Module Outline Drawings to show layout changes on the back of the PCB. ● Revised Jumper Locations and Functions to reflect new PCB layout. ● Illustrations throughout CONNECTION INFORMATION section have been revised to reflect changes in PCB layout. ● In addition to the list above, wherever needed, made minor modifications in text and illustrations to improve clarity.

Data Sheet Revision History (Continued)	

2010/09/28	Data Sheet version: v3.1
	Changes since last revision (v3.0):
	● Wherever listed or shown, changed keypad dimensions from “10.5 mm” to “12 mm” height and module overall depth with keypad from “24.10 mm” to nominal “25.6 mm” nominal and “25.9 mm” maximum. We started this gradual transition June 2010. See PCN 10282.
	● In Main Features, added “Factories have ISO certification”. Factories have had certification for several years.
	● In Jumper Locations and Functions, updated information on closing and opening jumpers.
	● In CHARACTER GENERATOR ROM (CGROM), corrected explanation of how to find a code for a given character.
	● Expanded and improved information on power and control connections. See CONNECTION INFORMATION, particularly ATX Power Supply Power and Control Connections.
	● In command 5 (0x05): Reboot CFA633, Reset Host, or Power Off Host, added important notes about timing of command packets.
	● In command 11 (0x0B): Set LCD Cursor Position, corrected from
	`data[0] = column (0-19 valid)`
	`data[1] = row (0-3 valid)`
	to
	`data[0] = column (0-15 valid)`
	`data[1] = row (0-1 valid)`
	This command was described correctly in Data Sheet versions published 2005-05-19 and earlier. It was described incorrectly in Data Sheet versions published 2005-09-06 and later.
	● In command 12 (0x0C): Set LCD Cursor Style, added note to “`3 = blinking block plus underscore`”. “`Note: This behavior is not the same as the CFA533 series which is: blinking underscore.`”
	● In command 22 (0x16): Send Command Directly to the LCD Controller, corrected from “The LCD controller on the CFA633 is *Samsung S6A0073* compatible” to “The controller on the CFA633 is a *Neotec NT7070B (HD44780 compatible*)”. Controller has not changed.
	● In command 35 (0x23): Read GPIO Pin Levels and Configuration State, corrected "`length`" returned by reply from "`4`" to "`1`".
	● Added more information in CARE AND HANDLING PRECAUTIONS.
	● In Sample Code, added Algorithm 7: For PIC18F8722 or PIC18F2685.
	● In addition to list above, wherever needed, made minor modifications in text and illustrations to improve clarity.

Data Sheet Revision History (Continued)	
2010/03/30	Data Sheet version: v3.0 Changes since last revision (v2.0): ● Wherever listed, changed cable part numbers to include dashes (“-”) to match how they appear on our website. ● In ORDERING INFORMATION, added a list of all variants in the CFA633 series. ● In Physical Characteristics table, corrected Active Area height from “11.2” to “11.5”. Also corrected this dimension on website page under Tech Specs tab. Dimension was correct in all drawings, including Module Outline Drawings. Dimension has not changed. ● Added Module Longevity (EOL / Replacement Policy). ● Added source for connector loaded at “PWR” in Standard (Non-ATX) Power Connection. ● In command 33 (0x21): Set Baud Rate, corrected "`data[1]`" to "`data[0]`". ● Revised CARE AND HANDLING PRECAUTIONS section. ● In APPENDIX C: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC), - Added section with hypertext links to our free downloadable code. - Added Algorithm 2B: “C” Improved Bit Shift Implementation. This is a simplified algorithm that implements the CRC. - In sample code for Algorithm 1: “C” Table Implementation and Algorithm 2: “C” Bit Shift Implementation, added typedefs for "`ubyte`" and "`word`". - In Algorithm 6: “Perl” Table Implementation, corrected code from "`my $packet = $type . $length . $data ;`" to "`my $packet = chr(hex $type) .chr(hex $length) .chr(hex $data);`". ● In APPENDIX D: QUALITY ASSURANCE STANDARDS, added information on Color Definitions.
2009/02/02	Data Sheet version: v2.0 Changes since last revision (vk1.9c): ● Added APPENDIX E: VISUAL INSPECTION OF SOLDER BRIDGES. ● Corrected specification of GPIO pull-up/pull-down mode resistance values from "approximately 5Ω" to "approximately 5kΩ". See 34 (0x22): Set or Set and Configure GPIO Pins. ● Deleted *“Built-in reprogrammable microcontroller (factory operation)”* and *“Expandable firmware and configurable hardware can be customized to add specific features for your system needs (tooling fee and minimum order may apply)*” from MAIN FEATURES. ● Corrected label on Figure 1 Test Circuit Schematic in APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER. ● Minor formatting changes so that Appendix E could be added.

Data Sheet Revision History (Continued)	
2005/12/20	Data Sheet version: vk1.9c Changes since last released version (vk1.9b): ● Added *Character Pitch* dimension to PHYSICAL CHARACTERISTICS. ● Corrected return packet "type" for command 26 (0x1A): Set Fan Tachometer Glitch Filter. ● Corrected return packet "type" for command 27 (0x1B): Query Fan Power & Fail-Safe Mask. ● Corrected "type" for 33 (0x21): Set Baud Rate. ● Corrected length returned by reply for command 35 (0x23): Read GPIO Pin Levels and Configuration State. ● Formatting, content organization, and minor rewording to improve readability. ● For added convenience, a separate data sheet is available for each CFA633 module variant.
2005/08/01	Start Public Version Tracking. Data Sheet version: vk1.9b Changes since last released version (vk1.9a): ● Added note on length of command 30 reply (see 30: Read Reporting & Status.

CONTENTS

CONTENTS, CONTINUED

LIST OF FIGURES

INTRODUCTION

COMPARISON TO CFA533

If your project does not need the fan connections, the CFA533 family is an economical replacement for the CFA633 family. In November 2010, we issued a Technical Bulletin that describes the differences between the CFA533 family and three versions of the CFA633 family. Please see Part Change Notice #10291.

MAIN FEATURES

- ❑ 16 characters x 2 lines LCD with keypad and high-level interface. Will fit nicely in a 1U rack mount case (35 mm overall height).
- ❑ Module has a 6 o'clock viewing direction. See Definition of 6 O'clock and 12:00 O'clock Viewing Angles (Pg. 20).
- ❑ Bidirectional 19200 / 115200 baud ESD-protected RS-232 interface.
- ❑ Edge-lit red LED backlight with negative FFSTN near-black transmissive mode LCD. Displays red characters on dark red background. Integrated red LED backlit 6-button translucent silicone keypad. The display can be read in normal office lighting and in dark areas. Not recommended for use in sunlight; may be washed out.
- ❑ An optional 5.25-inch half-height drive bay kit with mounting bracket is available. See CFA633 Kit Configurator.
- ❑ Fully decoded keypad: any key combination is valid and unique.
- ❑ Robust packet based communications protocol with 16-bit CRC.
- ❑ Nonvolatile memory capability (EEPROM):
  - Customize the "power-on" display settings.
  - 16-byte "scratch" register for storing IP address, netmask, system serial number . . .
- ❑ Firmware support for CFA633-RDI-KS allows:
  - ATX power supply control functionality allows the buttons on the CFA633-RDI-KS to replace the Power and Reset switches on your system, simplifying front panel design.
  - Four fan connectors with RPM monitoring and variable PWM fan power control. Fail-safe fan power settings allows safe host fan control based on temperature. See 25 (0x19): Set Fan Power Fail-Safe.
  - Add up to 32 Crystalfontz WR-DOW-Y17 cables with DOW (Dallas One-Wire) DS18B20 temperature sensors. Monitor temperatures at up to 0.5°C absolute accuracy.
  - Hardware watchdog can reset host system on host software failure.
  - "Live Display" shows up to eight temperature or four fan readings without host intervention, allowing fans and temperatures to be shown immediately at boot, even before the host operating system is loaded.
- ❑ RoHS compliant.
- ❑ Factories have ISO certification.
- ❑ Product materials are in compliance with the regulations related to the EU Directive 2006/121/EC for Registration, Evaluation, Authorization and Restriction of Chemicals (REACH).

MODULE CLASSIFICATION INFORMATION

CFA 633 - R D I - K S
❶ ❷ ❸ ❹ ❺ ❻ ❼

❶	**Brand**	Crystalfontz America, Inc.
❷	**Model Identifier**	633
❸	**Backlight Type & Color**	R – LED, red
❹	**Fluid Type, Image (positive or negative), & LCD Glass Color**	D – FFSTN, negative near-black
❺	**Polarizer Film Type, Temperature Range, & View Angle (O 'Clock)**	I – Transmissive, Wide Temperature Range[1], 6:00
❻	**Special Code 1**	K – Manufacturer's code
❼	**Special Code 2**	S – Serial interface with full swing RS-232
[1] *Wide Temperature Range is -20°C minimum to +70°C maximum.*		

ORDERING INFORMATION

PART NUMBER	FLUID	LCD GLASS COLOR	IMAGE	POLARIZER FILM	BACKLIGHT COLOR/TYPE
CFA633-RDI-KS ("full swing" RS-232)	**FFSTN**	**near-black**	**negative**	**transmissive**	**LCD: red edge LEDs Keypad: red LEDs** 
Additional variants (same form factor, different LCD mode or backlight):					
CFA633-TFH-KS ("full swing" RS-232)	FSTN	neutral	positive	transflective	LCD: white edge LEDs Keypad: white LEDs 
CFA633-TMI-KS ("full swing" RS-232)	STN	blue	negative	transmissive	LCD: white edge LEDs Keypad: blue LEDs 
CFA633-YYH-KS ("full swing" RS-232)	STN	yellow-green	positive	transflective	LCD: yellow-green edge LEDs Keypad: yellow-green LEDs 
CFA633-RDI-KU (USB)	FFSTN	near-black	negative	transmissive	LCD: red edge LEDs Keypad: red LEDs 
CFA633-TFH-KU ("full swing" RS-232)	FSTN	neutral	positive	transflective	LCD: white edge LEDs Keypad: white LEDs 
CFA633-TMI-KU (USB)	STN	blue	negative	transmissive	LCD: white edge LEDs Keypad: blue LEDs 
CFA633-YYH-KU (USB)	STN	yellow-green	positive	transflective	LCD: yellow-green edge LEDs Keypad: yellow-green LEDs 

ACCESSORIES

Customize Your Module (Connectors and Cable Choices)

Go to the web page for CFA633-RDI-KS. After you click on the *"Customize and Add to Cart"* button, you will see a list of questions for different cables and connectors.

Kit Configurations (Brackets with Overlays and Cable Choices)

Kits with modules are available here: https://www.crystalfontz.com/products/select_kit.html.

The kits have these accessories:

- *Bracket*: A 5.25-inch half-height drive bay mounting bracket.
- *Overlay:* An overlay for the front of module with a display window of thick hard-coated polycarbonate. Overlays are sold with the bracket. Overlay choices are silver brushed anodized aluminum, beige plastic, and black plastic.
- *Cables:* Three choices. A kit may have one to five cables.

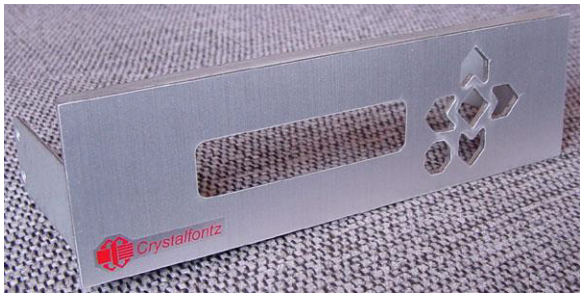


Figure 1. Bracket with Aluminum Overlay

Buy Cables Separately

Cable lengths are approximate. Common configurations are described in the section CONNECTION INFORMATION (Pg. 25). Additional cables are available on our website.

Crystalfontz Cable	Description All cables are RoHS compliant
WR-PWR-Y12 ~13 inches	Cable allows you to plug a 4-pin “hard drive style” Molex power connector into the module's “floppy drive style” power connector, plus provides an additional female 4-pin Molex connector.
WR-232-Y08 ~27 inches	Use this ribbon cable to supply communications. Connect cable’s 10-pin female connector to the module’s J_RS232 male connector. Connect cable’s RS232 DB9 9-pin female connector to host’s DB9 9-pin male serial port.
WR-232-Y13 ~36 inches	DB9 male ribbon cable with PC expansion slot bracket. Connect cable’s 10-pin female connector to the module’s J_RS232 male connector. Connect the cable’s DB9 male connector to host’s DB9 female connector.
WR-232-Y22 ~26 inches	Use this cable to supply communications. Connect one of the 10-pin female connectors to the module’s’ J_RS232 10-pin male connector. Connect cable’s second 10-pin female connector to host’s motherboard 10-pin male connector (standard or alternate pinout).
WR-232-Y23 ~26 inches	Connect cable’s 10-pin female connector to module’s J_RS232 10-pin male connector. Connect cable’s 9-pin female DB9 connector to host’s 9-pin male DB9 connector (standard or alternate pinout).
WR-PWR-Y14 ~24 inches	Use this ATX power cable to turn an ATX power supply on and off, or power cycle the host through the module. Requires optional 7-in female connector at J8 on module. Select J8 connector after you click on the module’s website page button “Customize and Add to Cart”. Connect the cable’s male 7-pin connector to the module’s J8 female connector.
WR-FAN-X01 ~16 inches	Connect up to four cables to connect up to four fans. Connect cable’s 3-pin male connector to module’s connectors labeled FAN1, FAN2, FAN3, or FAN4. Connect cable’s 3-pin female connector to a fan’s connector. (Fans are not sold by Crystalfontz.)
WR-DOW-Y17 ~12 inches + ~12 inches between connectors	Connect (“daisy chain”) up to 32 of these DOW (Dallas One-Wire) DS18B20 temperature sensor cables. Requires optional DOW connector at J_DOW on module. Select DOW connector after you click on the module’s website page button “Customize and Add to Cart”.

MECHANICAL CHARACTERISTICS

PHYSICAL CHARACTERISTICS

ITEM	SPECIFICATION
Module Overall Dimensions	
Width and Height	110.5 (W) x 35.0 (H)
Depth Without Keypad	20.1
Nominal Depth With Keypad	25.6
Maximum Depth With Keypad	25.9
Viewing Area	61.0 (W) x 15.8 (H)
Active Area	56.20 (W) x 11.5 0(H)
Character Size	2.95 (W) x 5.55 (H)
Character Pitch	3.55 (W) x 5.95 (H)
Dot Size	0.55 (W) x 0.65 (H)
Dot Pitch	0.60 (W) x 0.70 (H)
Keystroke Travel (approximate)	2.4
Weight	42 grams (typical)

MODULE OUTLINE DRAWINGS

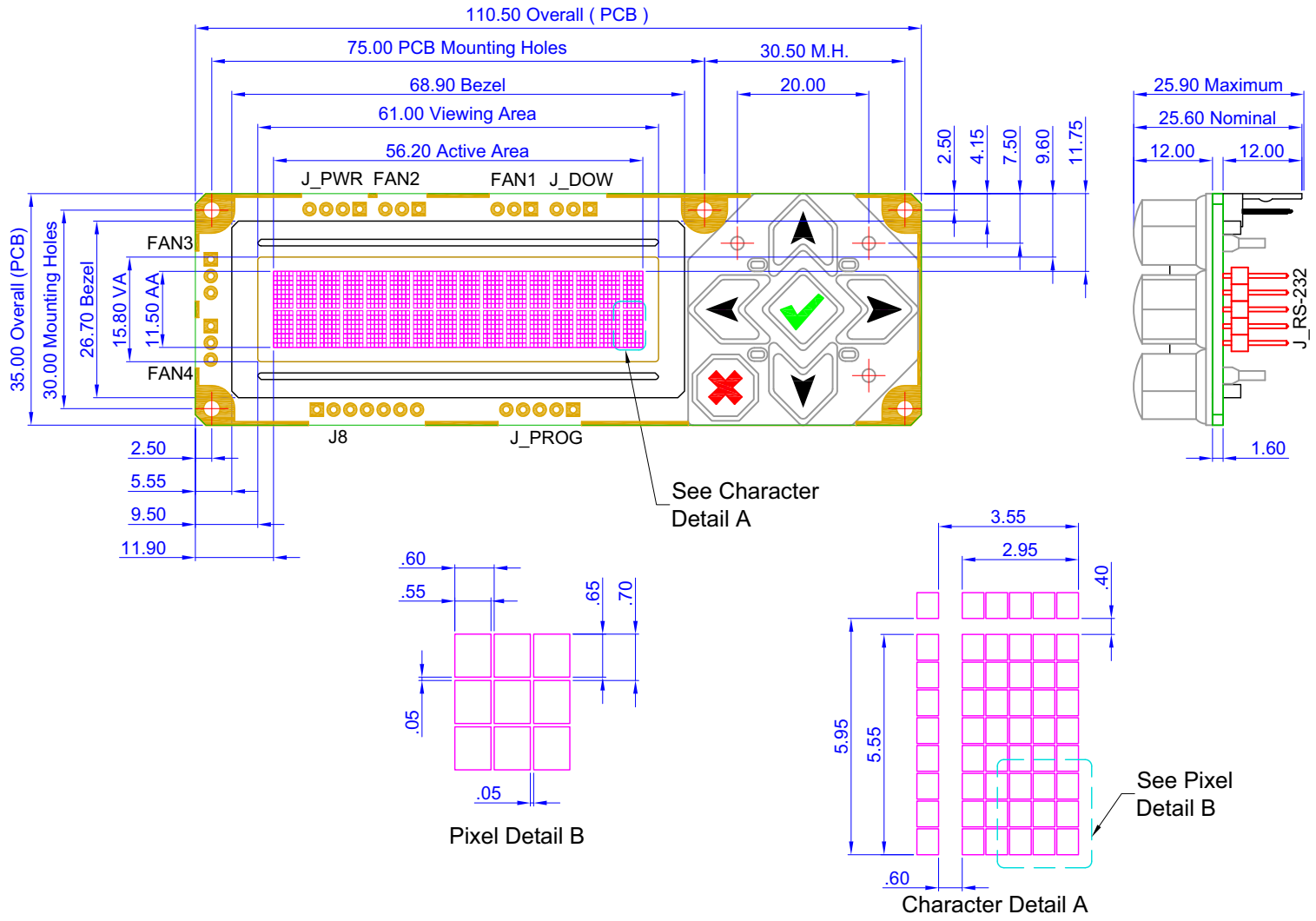
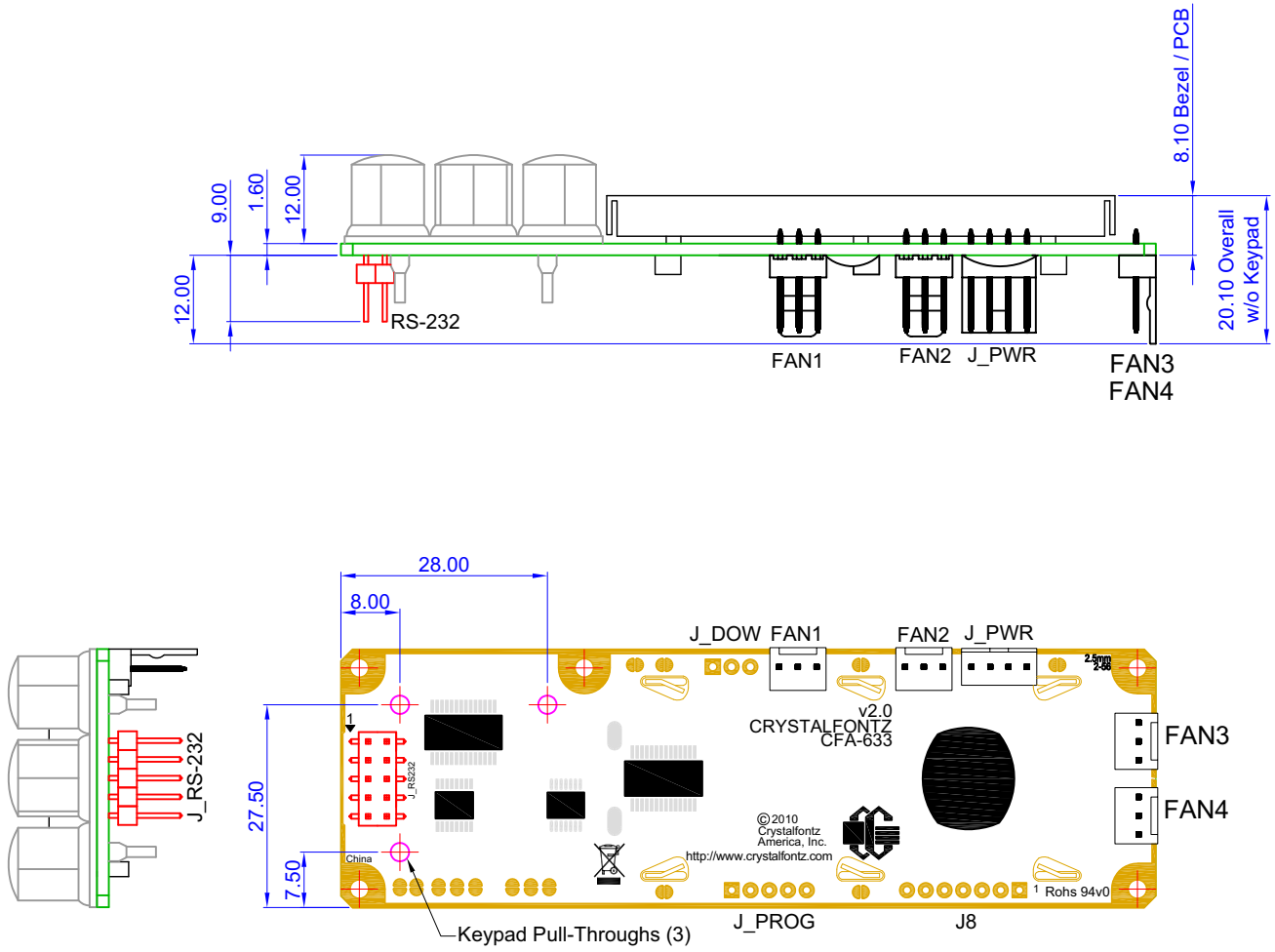


Figure 2. CFA633 Module Outline Drawing (two pages below)

Note: Tolerance is ±0.5 mm unless specified.

Part No.(s):	Scale:	Drawing Number:	
CFA633-xxx-KS	Not to scale	CFA633_master	
	Units: Millimeters	Date: 2012/02/13	Sheet: 1 of 2



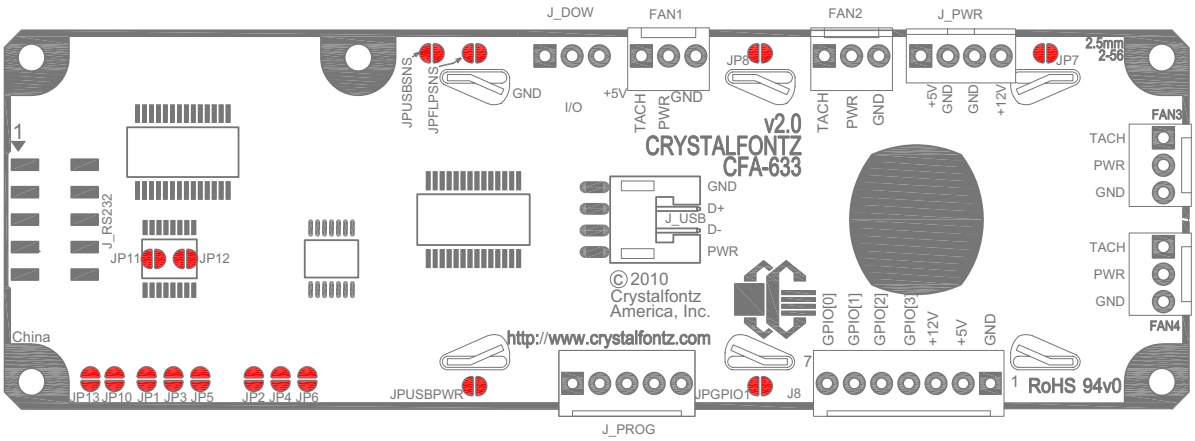
Note: Tolerance is ±0.5 mm unless specified.

Crystalfontz America, Inc.
www.crystalfontz.com/products/

Part No.(s):	Scale:	Drawing Number:	
CFA633-xxx-KS	Not to scale	CFA633_master	
	Units:	Date:	Sheet:
	Millimeters	2012/02/13	2 of 2

JUMPER LOCATIONS AND FUNCTIONS (ALL INTERFACES)

All jumpers are configurable but not all jumpers will affect your interface (serial or USB). Close jumpers by melting a ball of solder across their gap. Reopen the jumpers by removing the solder with solder wick.



CFA633 HW v2.0 Jumper Locations And Functions

JP1	**open** closed	J1 (RS232), Pin 10 is open LCD Tx/Host Rx to J1 (RS232), Pin 10
JP2	open **closed**	J1 (RS232), Pin 2 is open (see also JP3 and JP11) LCD Tx/Host Rx to J1(RS232), Pin 2
JP3	**open** closed	J1 (RS232), Pin 2 is open (see also JP2 and JP11) LCD Tx/Host Rx to J1 (RS232), Pin 2
JP4	open **closed**	J1 (RS232), Pin 3 is open (see also JP5 and JP12) LCD Rx/Host Tx to J1 (RS232), Pin 3
JP5	**open** closed	J1 (RS232), Pin 3 is open (see also JP4 and JP12) Ground to J1 (RS232), Pin 3
JP6	open **closed**	J1 (RS232), Pin 5 is open Ground to J1 (RS232), Pin 5
JP7	open **closed**	+12v fans is isolated from +12v backlight +12v fans is connected to +12v backlight
JP8	open **closed**	+5v pin from PWR connector is open (see also JP9) +5v pin from PWR connector supplies power to module

JP10	**open** closed	J1 (RS232), Pin 1 is open +12v backlight to J1 (RS232), Pin 1
JP11	**open** closed	J1 (RS232), Pin 2 is open (see also JP2 and JP3) Logic-level (0-5v) LCD Tx / Host Rx to J1 (RS232), Pin 2
JP12	**open** closed	J1 (RS232), Pin 3 is open (see also JP4 and JP5) Logic-level (0-5v) LCD Rx / Host Tx to J1 (RS232), Pin 3
JP13	**open** closed	J1 (RS232), Pin 4 is open +5v to J1 (RS232), Pin 4
JPUSBPWR	**open** closed	Module +5v power independent of USB Module +5v power supplied from USB
JPUSBSNS	**open** closed	No function +5v from USB is connected to processor's ATX SENSE
JPFLPSNS	**open** closed	No function +5v from floppy is connected to processor's ATX SENSE
JPGPIO1	open **closed**	Adds 5K Ω series resistor for ATX host power sense GPIO[1] direct connection

Figure 3. Jumper Locations and Functions (All Interfaces)

KEYPAD DETAIL DRAWING

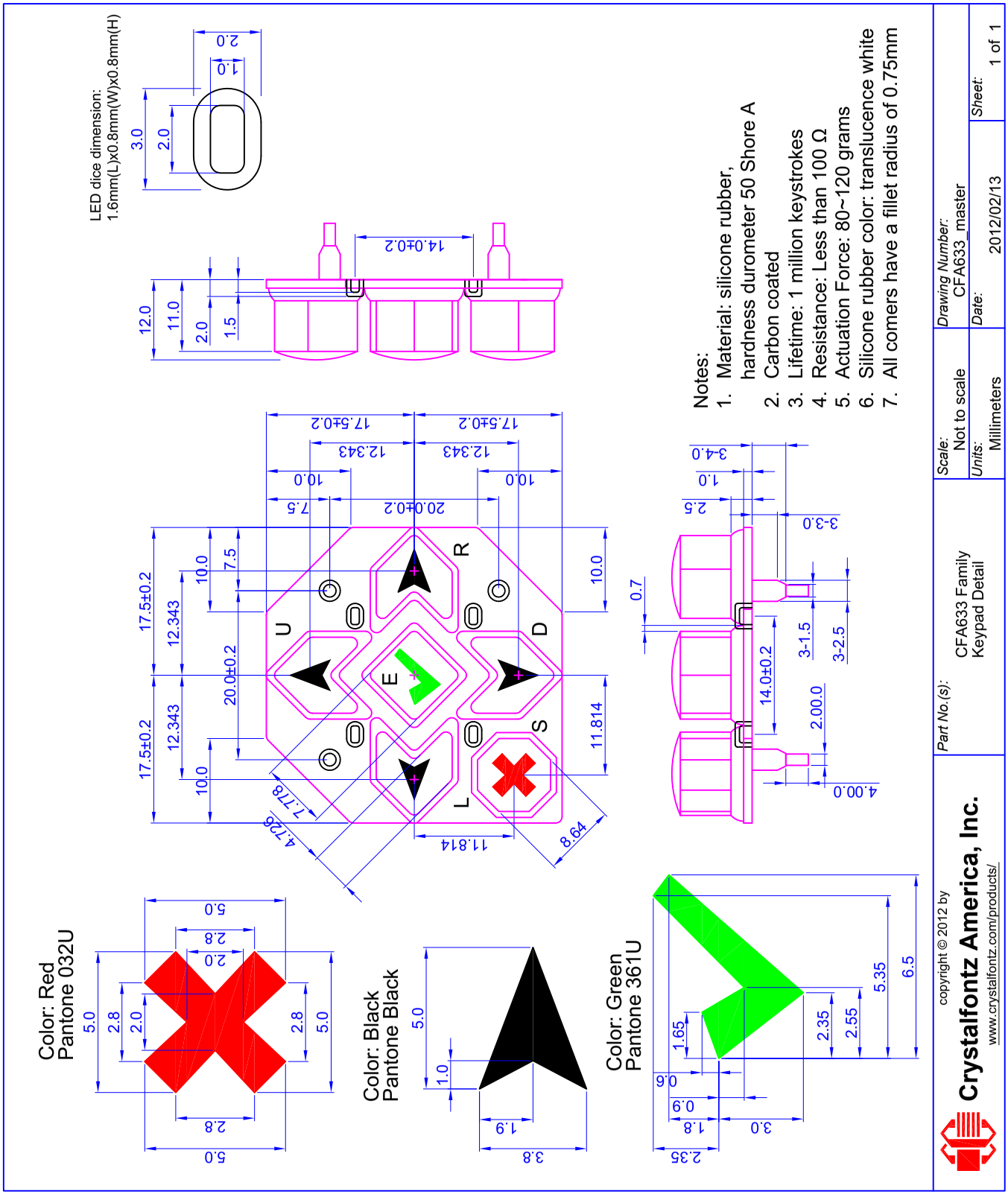


Figure 4. Keypad Detail Drawing

PANEL MOUNTING APPLICATION CUTOUT DRAWING

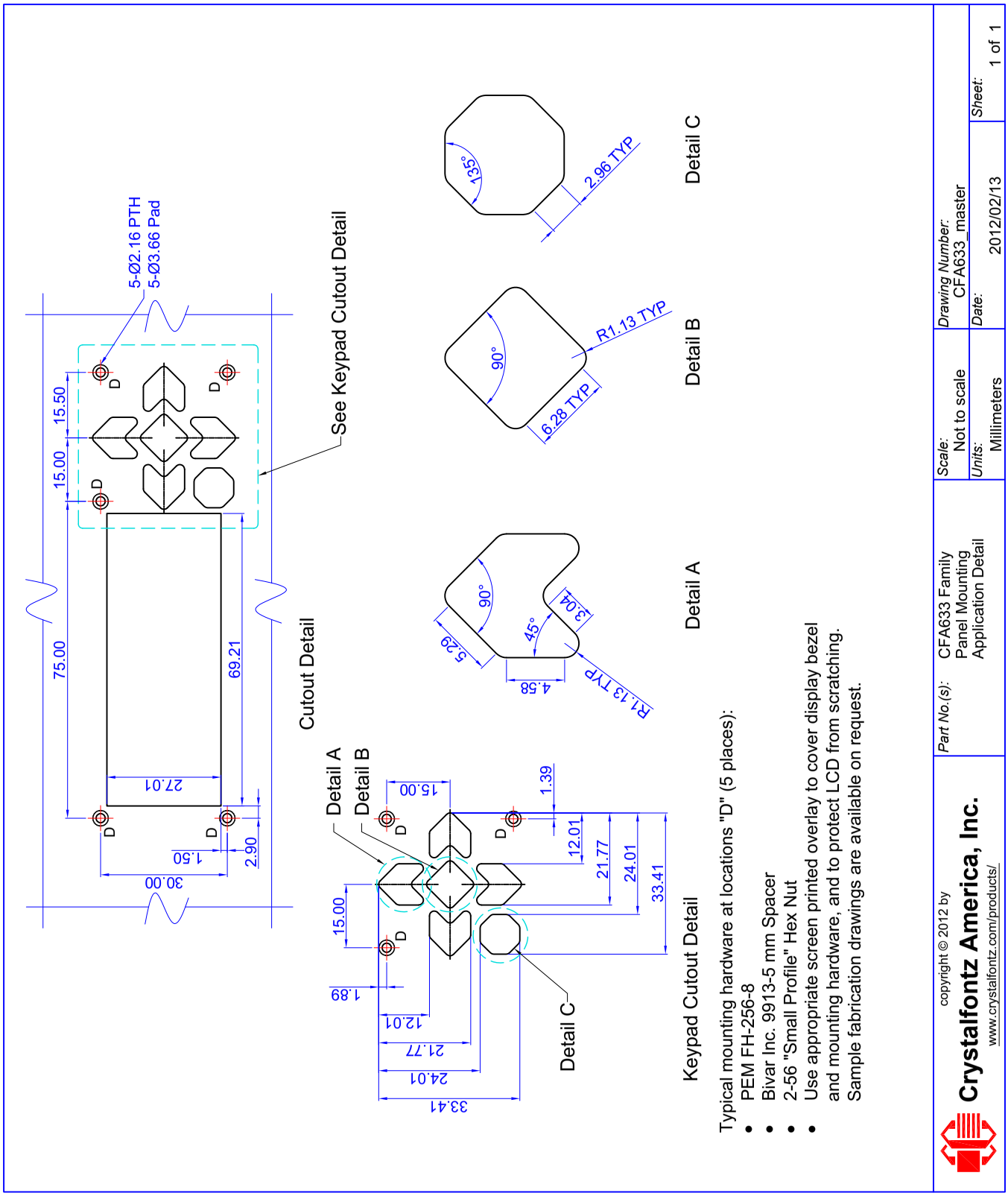


Figure 5. Panel Mounting Application Cutout Drawing for Optional Bracket

OPTICAL CHARACTERISTICS

Viewing Direction	6 o'clock

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

This module has a 6:00 o'clock viewing angle.

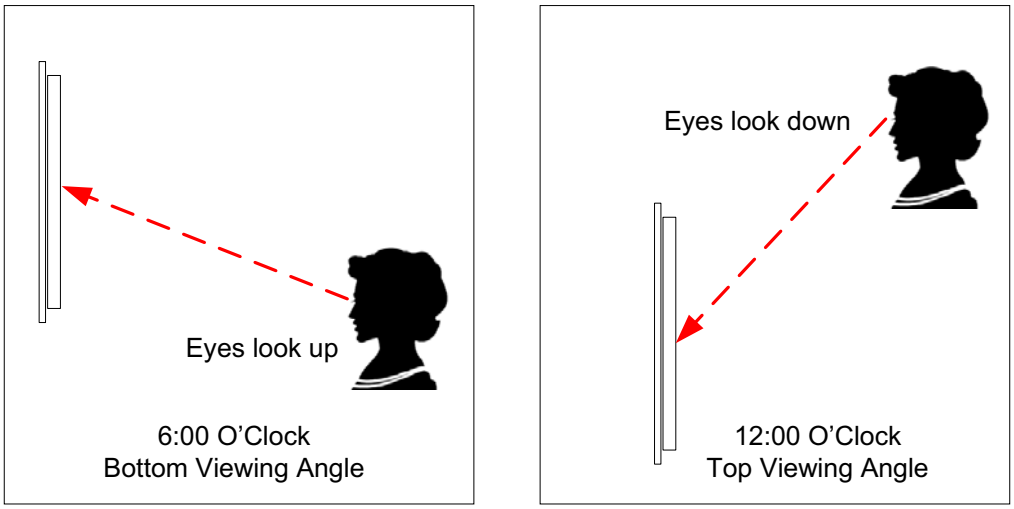


Figure 6. Definition of 6:00 O'clock and 12:00 O'clock Viewing Angles

ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

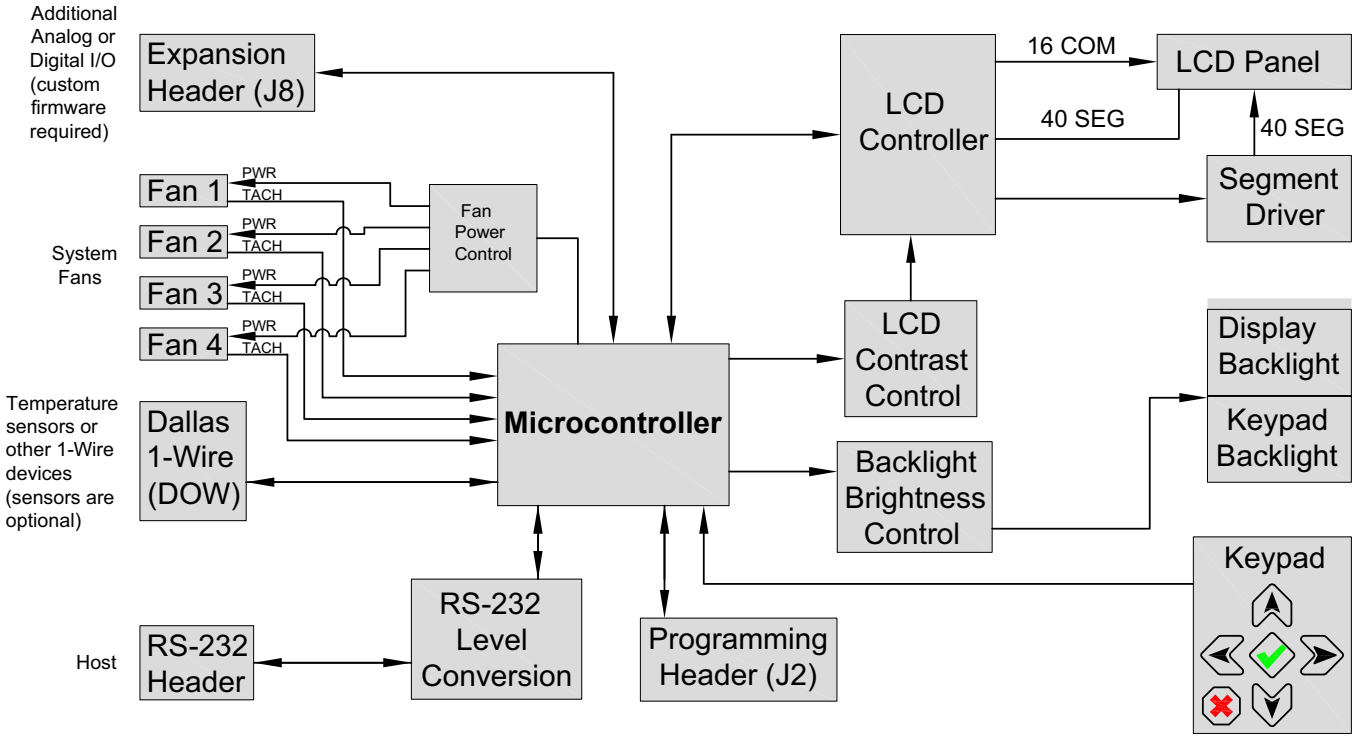


Figure 7. System Block Diagram

SUPPLY VOLTAGES AND CURRENT

DRIVING METHOD	SPECIFICATION
Duty	1/16
Bias	1.5

SUPPLY VOLTAGE	MINIMUM	NOMINAL	MAXIMUM
Supply voltage for driving the LCD module (logic)	+4.75v	+5.0v	+5.25v
Supply voltage for backlight*	+11v	+12v	+13v
Supply voltage to run fans*	+4.75v	+12v	+13v
**JP7 must be opened for the operating range of +12v fans (optional) to be different from +12v backlights (listed in this table). If JP7 is closed (which is the default), then the voltage specification range for backlights must be observed for both backlights AND fans.*			

TYPICAL CURRENT CONSUMPTION	SPECIFICATION
+5v for logic (LCD + microcontroller)	12 mA
+12v for backlight (at 100%)	60 mA

Note

Draw on +12v for optional fans will vary, depending on user equipment connected to FAN1 through FAN4.

- Maximum *continuous* current draw must be <1.5 A per fan connector, no more than 4 A total.
- Maximum *pulsed* current draw may be up to 5 A per connector. The pulse width must be less than 50 mS. (This pulse specification allows for the fan's start-up current spike.)

GPIO CURRENT LIMITS	SPECIFICATION
Sink	25 mA
Source	10 mA

BACKLIGHT AND FAN[1] CRITERIA	SPECIFICATION
Backlight PWM[2] Frequency	320 Hz nominal
Fan Tachometer Speed Range (assuming two PPR[3])	600 RPM to 3,000,000 RPM
Fan Power Control PWM[2] Frequency	18 Hz nominal

[1]Fans are an optional accessory.

[2]*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.

[3]*PPR* is *Pulses Per Revolution*, also written as p/r.

ABSOLUTE MAXIMUM RATINGS

ABSOLUTE MAXIMUM RATINGS	SYMBOL	MINIMUM	MAXIMUM
Operating Temperature	T_{OP}	-20°C	+70°C
Storage Temperature	T_{ST}	-30°C	+80°C
Humidity Range (Noncondensing)	RH	10%	90%
Supply Voltage for Logic	V_{DD}	0v	+5.5v

Notes:
These are stress ratings only. Extended exposure to the absolute maximum ratings listed above may affect device reliability or cause permanent damage. Functional operation of the module at these conditions beyond those listed under DC Characteristics is not implied.

DC CHARACTERISTICS

	DC CHARACTERISTICS	TEST CONDITIONS	SYMBOL	MINIMUM	TYPICAL	MAXIMUM
CONTROLLER AND BOARD	Supply Voltage for Logic	T_{OP} =-30°C to +70°C	V_{DD} - GND	+4.75v	+5.0v	+5.25v[1]
	LCD Driver Voltage	Ta = 25°C	V_{LCD}	+4.8v	+5.0v	+5.2v
	Input High Voltage	V_{DD} = +5v	V_{IH}	V_{DD}-1.0v		V_{DD}
	Input Low Voltage		V_{IL}	0v (GND)		+0.6v
	Output High Voltage		V_{OH}	+0.9V_{DD}		
	Output Low Voltage		V_{OL}	0v (GND)		+0.1V_{DD}

[1] *Do not exceed* +5.25v *maximum.*

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS

Tx and Rx pins of connector RS-232 only:
+15 kV Human Body Model
+15 kV IEC1000-4-2 Air Discharge
+8 kV IEC1000-4-2 Contact Discharge

The remainder of this circuitry is industry standard CMOS logic and 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.

LED BACKLIGHT INFORMATION

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

NOTE

For modules with **white** backlights (CFA633-TFH-Kx and CFA633-TMI-Kx), we recommend that the backlight be dimmed or turned off during periods of inactivity to conserve the LEDs' lifetime.

LEDs are "current" devices. The brightness is controlled by the current flowing through it, not the voltage across it. Ideally, a current source would be used to drive the LEDs. A simple current limiting resistor works well in most applications and is much less complex than a current source.

CONNECTION INFORMATION

STANDARD POWER CONNECTION

The easiest way to provide power from a host to CFA633-RDI-KS is to connect a cable that would normally be used from the host's power supply to a peripheral. If an older PC is the host, this would be the cable for the 3.5-inch floppy drive connection. If a cable is not available, you may buy the Crystalfontz WR-PWR-Y12 cable to connect to the large 4-pin power supply cable connector.

JP7 and JP8 are closed by default. JP7 can be opened to supply power to the fans separate from the power to the backlight. See Jumper Locations and Functions (All Interfaces) (Pg. 17).

The connector loaded at J_PWR is (Tyco Electronics / Amp part number 4-171825-4, Mouser Electronics part number 571-4-171825-4).

NOTE

Do not connect high power fans to the module when +12v is supplied through J8, Pin 3 and JP7 is closed. Total fan current must be less than 500 mA if +12v is supplied through J8, Pin 3 and JP7 is closed. When using high power fans, supply the +12v through the J_PWR connector.

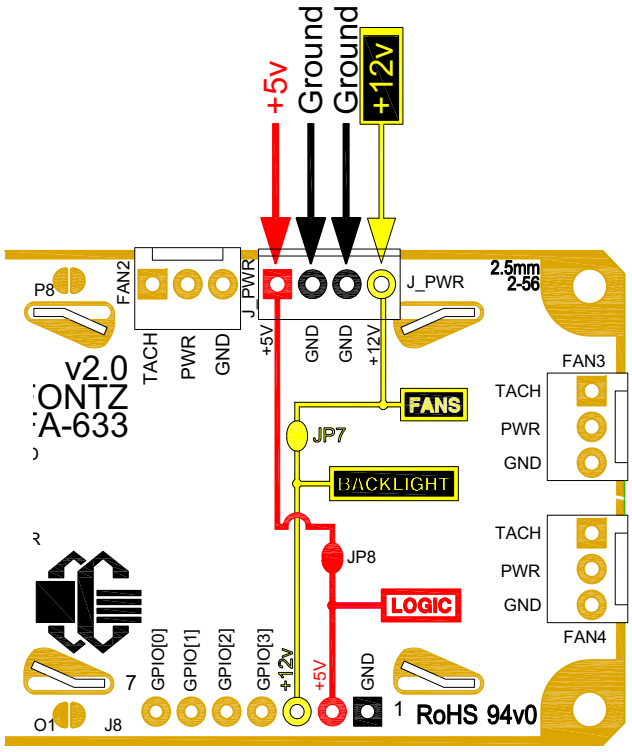


Figure 8. Standard Power Supply Connection through J_PWR

POWER CONNECTION THROUGH J_RS232

The +5v and +12v power can be supplied through connector J_RS232, allowing a single cable to contain both power and data connections. If the "Default RS-232 Pin Assignments" are selected, the five connections needed to operate the module are all on a single column of pins on J_RS232, which allows a single 0.1-inch spacing 5-conductor cable to connect between the CFA633-RDI-KS and your embedded system.

JP10 and JP13 are open by default. To enable +5 v to be supplied through J_RS232, close JP13. To allow +12 v to be supplied through J_RS232, close JP10.

Modification by You

To close JP10 and JP13, see Jumper Locations and Functions (All Interfaces) (Pg. 17) for jumper locations and functions.

Modification by Crystalfontz

Crystalfontz can configure the modules so they will be ready to use in your application without modification by you. We do this by closing JP10 and JP13. For information, please contact technical support (call +1-888-206-9720 or email techinfo@crystalfontz.com). We will provide you with a semi-custom part number. A minimum order quantity may apply.

NOTE

Do not connect high power fans to the module when +12v is supplied through J_RS232/JP10. Total fan current must be less than 500 mA if +12v is supplied through J_RS232/JP10. When using high power fans, supply the +12v through the J_PWR connector.

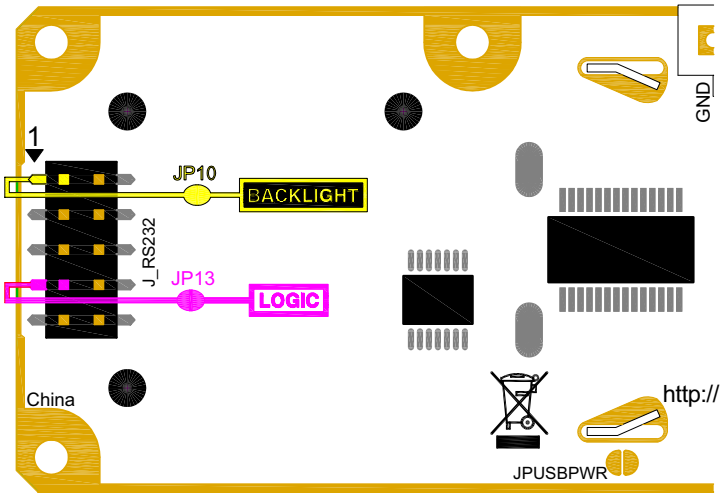


Figure 9. J_RS232 Power Connection

ATX POWER SUPPLY AND CONTROL CONNECTIONS

The CFA633-RDI-KS has the ability to control power on/off and reset functions of a standard ATX PC.

NOTE

The GPIO pins used for ATX control must not be configured as user GPIO, and must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are the default but may be changed by the user. Please see command 34 (0x22): Set or Set and Configure GPIO Pins (Pg. 58).

The Crystalfontz WR-PWR-Y14 cable simplifies ATX power control connections. When using this cable, please open jumper JP8 and JPGPIO1. Close jumper JPFLPSNS in order to ensure correct operation. See Jumper Locations and Functions (All Interfaces) (Pg. 17) for jumper positions and locations.

NOTE

If the Crystalfontz WR-PWR-Y14 cable is ordered at the same time as the module, Crystalfontz will install the WR-PWR-Y14 connector on the module, open jumper per JP8 and JPGPIO1. Close jumper JPFLPSNS, and send the following software configuration commands (unless we are otherwise instructed). Please note that once these changes are made, power must be applied to connector J_PWR, pin +5v and connector J8, pin +5v for the module to power up:

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

For this functionality, this configuration of the CFA633-RDI-KS is powered from the PC's V_{SB} signal (the “stand-by” or “always-on” +5v ATX power supply output).

By default, the pin labeled +5v on the CFA633-RDI-KS’s connector J8 is electrically connected to the +5v pin on the CFA633-RDI-KS’s J_ PWR connector. If you are using CFA633-RDI-KS to do ATX power control, you will want to open jumper JP8, which will disconnect the +5v of the J_PWR connector from the +5v of connector J8.

Since the CFA633-RDI-KS must act differently depending on if the host's power supply is “on” or “off”. By opening JPGPIO1, a 5KΩ resistor is added in series that will connect to the host's “switched +5v” through JPFLPSNS. The switched +5v power can be either the J_PWR for serial or J_USB for USB module.

The motherboard's power switch input is connected to Pin 5 of the CFA633-RDI-KS’s connector J8 (labeled as GPIO[2]). This pin functions as POWER CONTROL. The POWER CONTROL pin is configured as a high-impedance input until the LCD module wants to turn the host on or off, then it will change momentarily to low impedance output, driving either low or high depending on the setting of POWER_INVERT. (See command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54).)

The motherboard's reset switch input is connected to Pin 4 of the CFA633-RDI-KS connector J8 (labeled as GPIO[3]). This pin functions as RESET. The RESET pin is configured as a high-impedance input until the LCD module wants to reset the host. Then it will change momentarily to low impedance output, driving either low or high depending on the

setting of RESET_INVERT. (See command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54).) This connection is also used for the hardware watchdog.

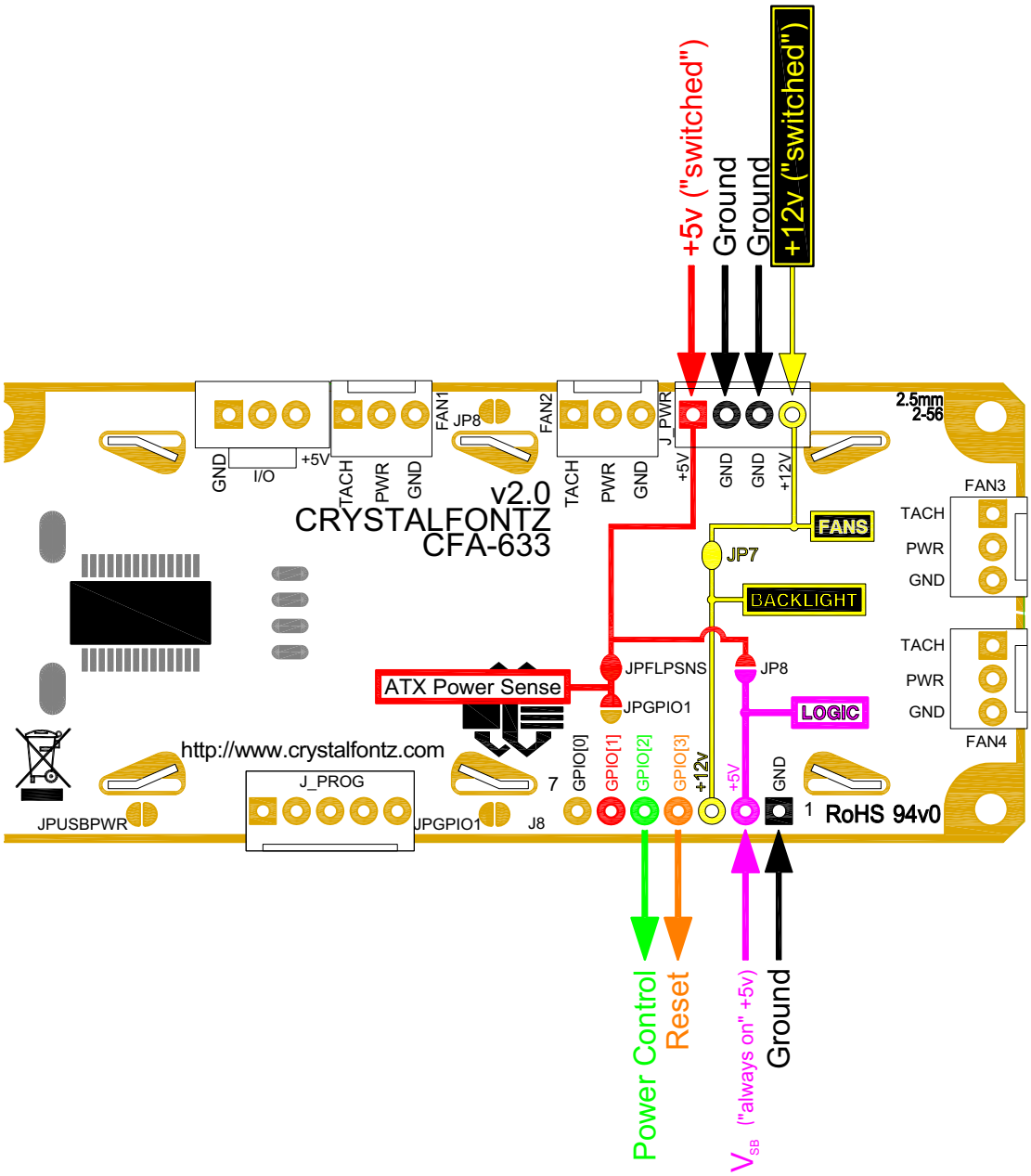


Figure 10. ATX Power Supply and Control Connections

Once configured by the host software (see command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54)), the following functions may be individually enabled:

- **System power on.** If POWER-ON SENSE is low (0v), pressing the green check key for 0.25 seconds will turn the unit on by driving POWER CONTROL line for the pulse width set by command 28: Set ATX Power Switch Functionality (1.0 seconds default).

- **System hard power off.** If POWER-ON SENSE is high (+5v) pressing the red X key for 4 seconds will turn the system off by driving the POWER CONTROL line. The line will be driven for a minimum of the pulse width set by command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54) (1.0 seconds default). If the user continues to press the key, the CFA633-RDI-KS will continue to drive the line for up to an additional 5 seconds.
- **System hard reset.** If POWER-ON SENSE is high (+5v) pressing the green check key for 4 seconds will reset the system off by driving the RESET line for 1 second. The CFA633-RDI-KS will reboot itself immediately after resetting the host.

Since the computer and LCD module must look off if the computer's power is off, the CFA633-RDI-KS can be configured to monitor the POWER-ON SENSE line and blank its display any time the POWER-ON SENSE line is low. If +12v remains active (which would not be expected, since the host is "off"), the fans will remain on at their previous settings.

Here is an illustration of how the optional Crystalfontz WR-PWR-Y14 cable connects to the CFA633 connector J8 and your ATX power supply:

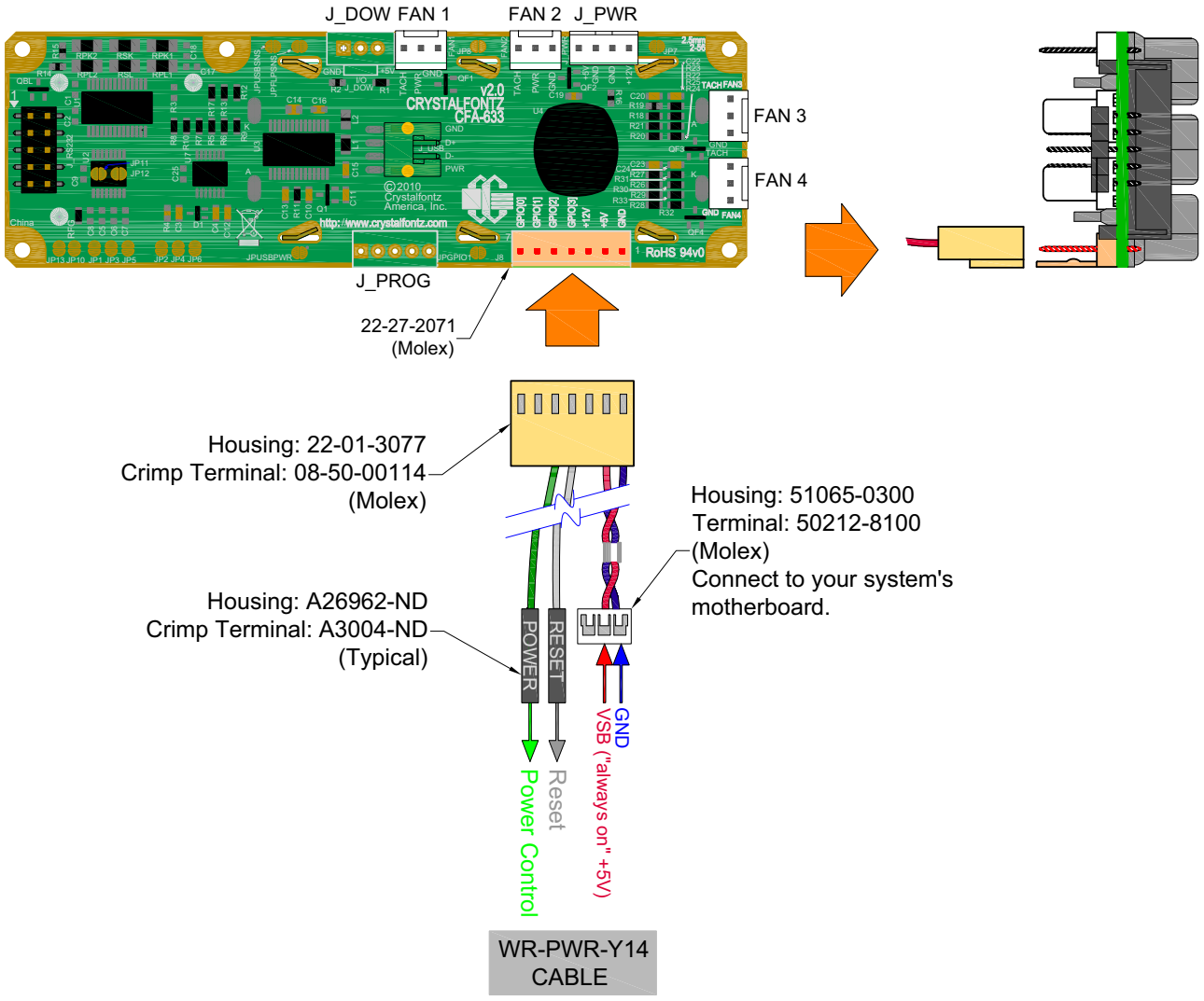


Figure 11. ATX Power Supply and Control Using Crystalfontz WR-PWR-Y14 Cable

RS-232 CONNECTIONS

JP2, JP4, and JP6 are closed by default, selecting the "Default RS-232 Pin Assignments". This connection allows a low-cost ribbon cable (Crystalfontz WR-232-Y08) to connect the CFA633-RDI-KS DB-9 COM port. By opening JP2, JP4 and JP6 and closing JP1, JP3, and JP5 you can select the "Alternate RS-232 Pin Assignments". See Jumper Locations and Functions (All Interfaces) (Pg. 17) for jumper positions.

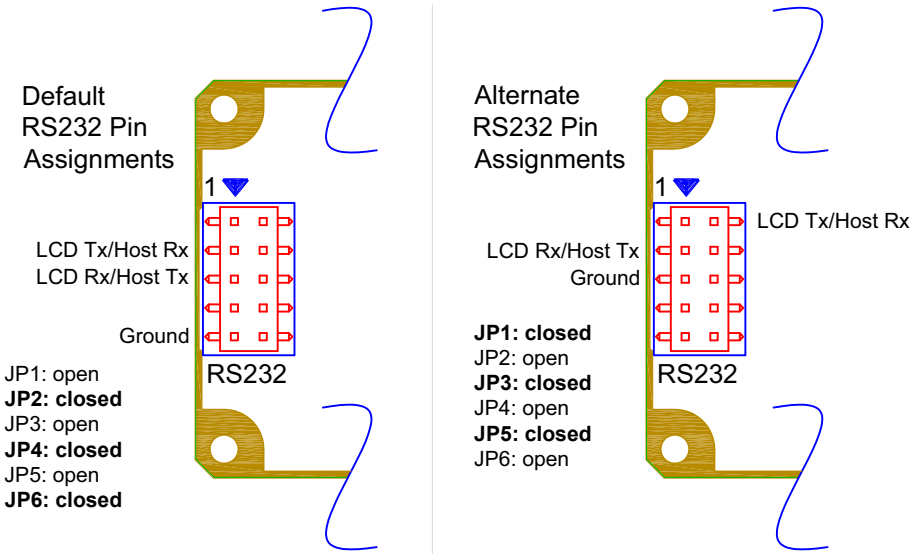


Figure 12. Default and Alternate RS232 Pin Assignments

If there is a matching 0.1-inch center, 10-pin RS-232 connector on your system's motherboard, then in most cases a simple straight-through ribbon cable (such as CW Industries' C3AAG-1018G-ND cable available from Digi-Key) can be used to connect from the CFA633-RDI-KS to the motherboard's header. The pin order of your motherboard's header will determine if the CFA633-RDI-KS's pin assignments need to be "Default" or "Alternate".

FAN CONNECTIONS

The CFA633-RDI-KS supports up to 4 standard “3-pin” cooling fans. The fan connectors are compatible with industry standard “3-pin” fans.

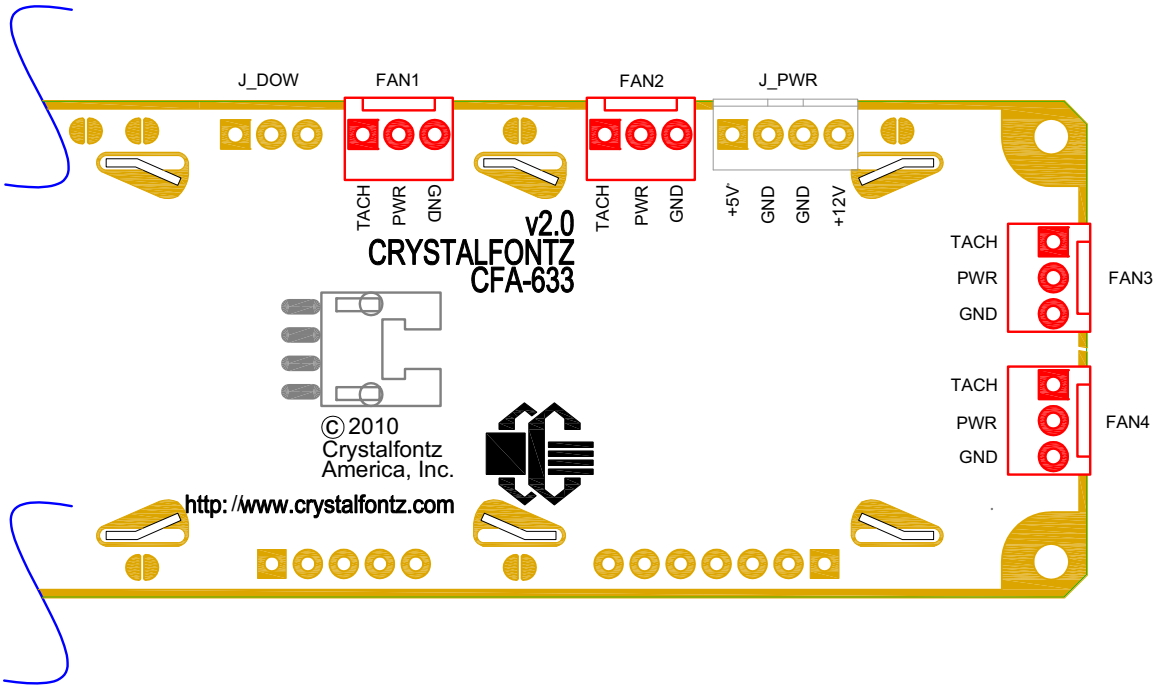


Figure 13. Four Fan Connectors

The average power delivered to each fan may be set to any level between 0% and 100% through command 17 (0x11): Set Fan Power (Pg. 45). The power setting controls the PWM duty cycle of a high-performance open-drain FET connected between the system ground and the GND pin of each fan connector. The PWM frequency is nominally 18 Hz.

The CFA633-RDI-KS can measure the frequency of the fan's tachometer signal, and given the pulses-per-revolution, calculate the RPM and display it on the LCD or report the information needed to calculate the RPM to the host. If a fan's power is set to 100%, then the average frequency of each fan's tachometer signal is measured over a 1/8 second (125 ms) period of time. Each fan is measured in sequence, so updated fan speed information is available every 1/2 second (500 ms) for each fan.

The power to a fan must be on in order for the fan's tachometer signal to be valid. If a fan is configured to report its speed to the host, the power of the fan will be unconditionally set to 100% at the start of the 1/8 second period of time when the CFA633-RDI-KS is measuring the frequency of the tachometer signal, overriding the PWM. The CFA633-RDI-KS will leave the power to the fan on until the glitch delay (see command 26 (0x1A): Set Fan Tachometer Glitch Filter (Pg. 52) has expired and two tachometer edges have been detected. The normal PWM cycle will then resume.

This technique allows the fan speed to be measured with a very minimal effect on the speed of the fan. If the fan power is set to 100% or if the speed of the fan and length of the PWM on time are such that the speed can be measured without stretching the PWM, then this override will not change the speed of the fan at all. If the fan power is set to some level other than 100% and the PWM on time is short compared to the tachometer signal frequency, then the fan speed will pulse slightly every 1/2 second due to the stretching of the PWM on time. During tachometer measuring, the maximum width of a stretched on pulse is 1/8 second. For some fans, the result is not very noticeable, and this technique will allow you to monitor the average speed of the fan while controlling the average power of the fan. For other fans (particularly high torque, high RPM models) the pulsing effect may be undesirable.

Since the on-time is dynamically stretched by the CFA633-RDI-KS to force the fan to produce two tachometer edges, the result is that the fan will resist stalling as power is reduced towards 0% and the RPM is being measured.

Here is a graph of fan RPM vs. the fan power setting for a typical high-performance 80 mm fan (for example, Delta FFB0812SHE):

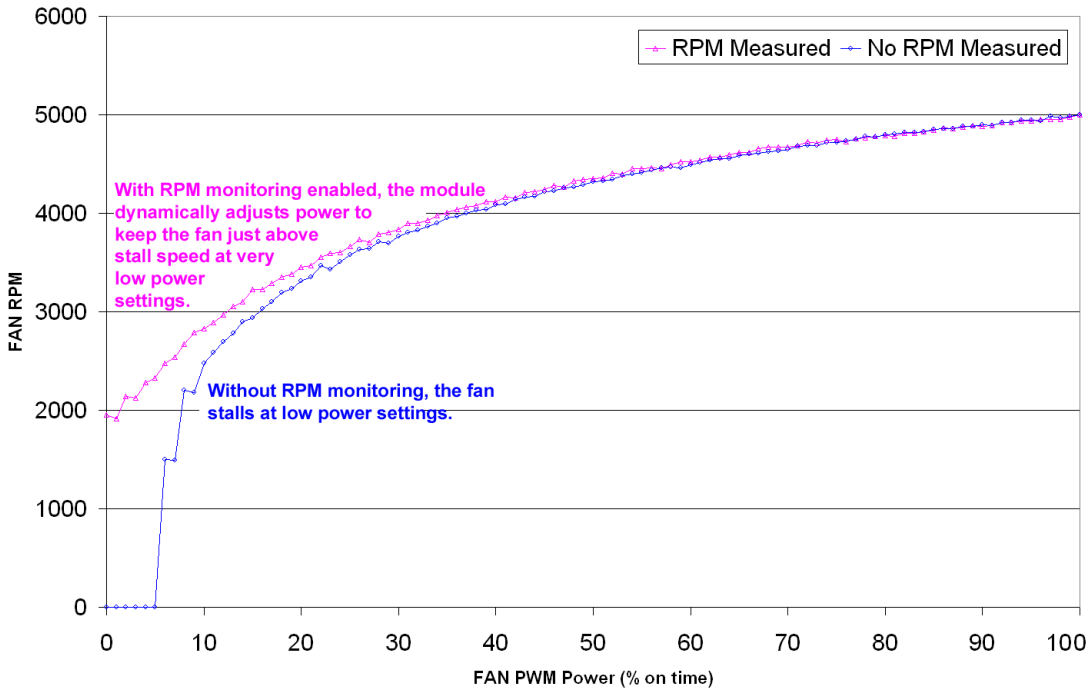


Figure 14. Graph showing fan PWM power (% on time) for various fan RPM

Typically if the fan speed is not at 100% then it is being controlled by the host software to drive a temperature sensor to a given reading in a closed-loop arrangement. In this case, the temperature, rather than the fan speed would be monitored for out-of-range conditions. If the temperature is within specification, you really do not care how fast the fan is turning. In an unattended system, it may be a good idea to set each fan to 100% for a few seconds during a test cycle — perhaps once a day or once a week — and log the steady-state RPM attained by the fan. If that steady state RPM were higher (this can be caused by a blocked airflow) or lower than expected (perhaps the fan's bearings are failing), a maintenance warning could be generated by the host software.

NOTE

For safety, enable the fan power fail-safe (see command 25 (0x19): Set Fan Power Fail-Safe (Pg. 51)) on any fans involved in host-based speed control. By enabling the fail-safe on a fan that is being used in closed-loop control through host software, the CFA633-RDI-KS will turn that fan to 100% if the host fails to update the power of the fans within a given time interval. For instance, if the communications cable is dislodged, the host operating system hangs, or cooling control process is terminated, the CFA633-RDI-KS will automatically force those fans to 100%, preventing potential equipment damage due to lack of cooling.

We tested the CFA633-RDI-KS with a large range of fans and had good results. However, you are responsible for determining if the control and monitoring methods employed by the LCD module are acceptable for your application. In particular, if a fan's power is set too low, it may stall or fail to start, providing no cooling. Using a PWM to control fan speed is generally accepted; however we make no claims that it is compatible with any particular fan or that it does not affect the lifetime of the fans. Some higher torque fans (especially the ball-bearing models) may click, buzz, or growl at low power settings due to the torque in the fan going from positive to negative in each PWM cycle. If you limit the power

setting to 0% or 100% there should be no compatibility issues. We do not recommend operating a fan below 20% PWM duty cycle for an extended period of time. Limiting the minimum PWM duty cycle to 30% or 40% should reduce the mechanical and electrical stresses in the fan, avoiding premature failure.

When power is applied to the CFA633-RDI-KS, it will set each fan's power to the default value of 100% or to the value that is stored in the boot state. To minimize peak current loading on the +12v supply during start-up, the fans are started in sequence with a 0.5 second delay between any fans that are on.

CONNECT OPTIONAL CRYSTALFONTZ WR-DOW-Y17 TEMPERATURE SENSORS

The Crystalfontz WR-DOW-Y17 cable has a DS18B20 Dallas Programmable Resolution One-Wire (DOW) temperature sensor attached to a “daisy chainable” cable. (“Daisy chain” means several devices connected in a linear series.) Connect one WR-DOW-Y17 to the connector labeled J_DOW.

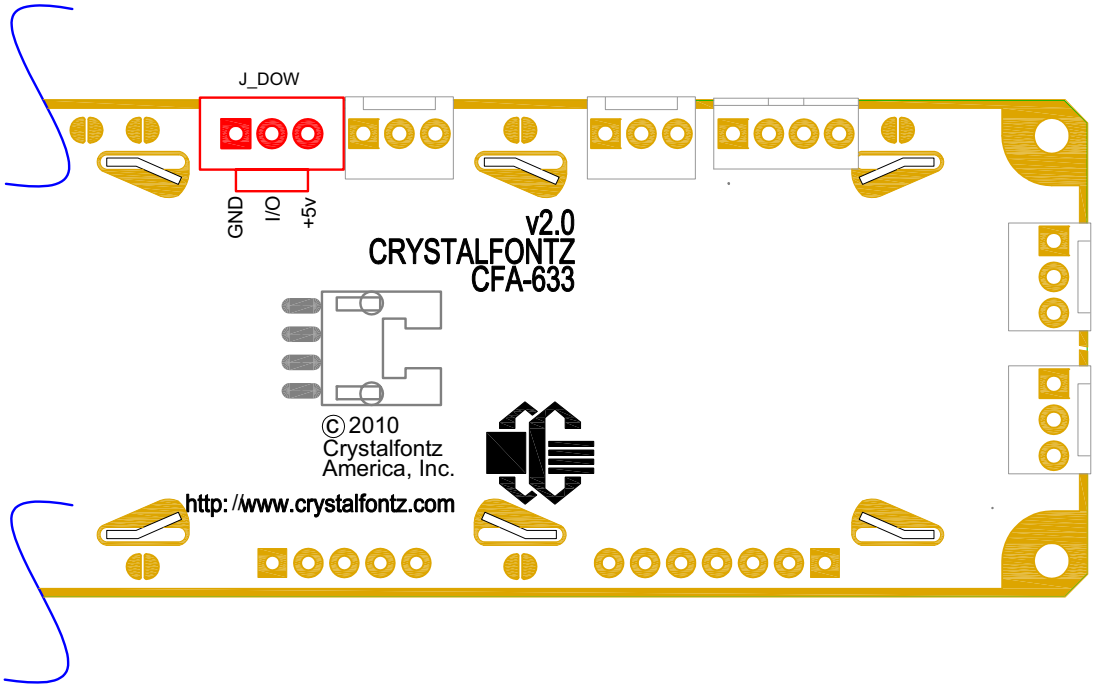


Figure 15. J_DOW Connection for WR-DOW-Y17 Temperature Sensor

If desired, connect the WR-DOW-Y17’s 3-pin male connector to an additional WR-DOW-Y17 temperature sensor. Up to 32 WR-DOW-Y17 temperature sensors can be connected. (“Daisy chained”.)

The DS18B20 on the WR-DOW-Y17 has 0.5°C absolute accuracy. Or you make a temperature sensor cable using a DS1822 Dallas Econo One-Wire Digital Thermometer with ±2°C accuracy.

Any temperature sensor can be configured to be automatically read and display to the CFA633-RDI-KS's LCD in °C or °F (see command 21 (0x15): Set Up Live Fan or Temperature Display (Pg. 49)). Independently, any temperature sensor can be configured to report to the host (see 19 (0x13): Set Up Temperature Reporting (Pg. 48)). Any sensors configured to be reported are updated once each second.

HOST COMMUNICATIONS

CFA633 communicates with its host using an RS-232 interface. The port settings are 19200 baud, 8 data bits, no parity, 1 stop bit by factory default. The speed can be set to 115200 baud under software control (see command 33 (0x21): Set Baud Rate (Pg. 58)).

PACKET STRUCTURE

All communication between the CFA633 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 CFA633 and the host without the traditional problems that occur in a stream-based serial communication (such as having to send data in inefficient ASCII format, to “escape” certain “control characters”, or losing sync if a character is corrupted, missing, or inserted).

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

All packets have the following structure:

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

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

```
TTcc cccc
|||| ||||--Command, response, error or report code 0-63
||---------Type:
             00 = normal command from host to CFA633
             01 = normal response from CFA633 to host
             10 = normal report from CFA633 to host (not in
                  direct response to a command from the host)
             11 = error response from CFA633 to host (a packet
                  with valid structure but illegal content
                  was received by the CFA633)
```

`data_length` specifies the number of bytes that will follow in the data field. The valid range of `data_length` is 0 to 18.

`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 C: DEMONSTRATION SOFTWARE AND SAMPLE CODE (Pg. 71) for details.

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

```
typedef struct
  {
  unsigned char
    command;
  unsigned char
    data_length;
  unsigned char
    data[MAX_DATA_LENGTH];
  unsigned short
    CRC;
  }COMMAND_PACKET;
```

On our website, Crystalfontz supplies a demonstration and test program, 633_WinTest along with its C source code. Included in the 633_WinTest source is a CRC algorithm and an algorithm that detects packets. The algorithm will automatically re-synchronize to the next valid packet in the event of any communications errors. Please follow the algorithm in the sample code closely in order to realize the benefits of using the packet communications.

ABOUT HANDSHAKING

The nature of CFA633's packets makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for a corresponding acknowledge packet from the CFA633 before sending the next command packet. The CFA633 will respond to all packets within 250 mS. The host software should stop waiting and retry the packet if the CFA633 fails to respond within 250 mS. The host software should report an error if a packet is not acknowledged after several retries. This situation indicates a hardware problem — for example, a disconnected cable.

Please note that some operating systems may introduce delays between when the data arrives at the physical port from the CFA633 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 CFA633 can be configured to send several types of report packets along with regular acknowledge packets. The host should be able to buffer several incoming packets and must guarantee that it can process and remove packets from its input buffer faster than the packets can arrive given the baud rate and the reporting configuration of the CFA633. For any modern PC using reasonably efficient software, this requirement will not pose a challenge.

The report packets are sent asynchronously with respect to the command packets received from the host. The host should not assume that the first packet received after it sends a command is the acknowledge packet for that command. The host should inspect the `type` field of incoming packets and process them accordingly.

REPORT CODES

The CFA633 can be configured to report three items. The CFA633 sends reports automatically when the data becomes available. Reports are not sent in response to a particular packet received from the host. The three report types are (1) 0x80: Key Activity, (2) 0x81: Fan Speed Report, and (3) 0x82: Temperature Sensor Report. Details are below.

0x80: Key Activity

If a key is pressed or released, the CFA633 sends a Key Activity report packet to the host. Key event reporting may be individually enabled or disabled by command 23 (0x17): Configure Key Reporting (Pg. 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
       KEY_EXIT_RELEASE         12
```

0x81: Fan Speed Report

If any of up to four fans connected to CFA633 is configured to report its speed information to the host, the CFA633 will send Fan Speed Reports for each selected fan every 1/2 second. See command 16 (0x10): Set Up Fan Reporting (Pg. 45).

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

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

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

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

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

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

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

0x82: Temperature Sensor Report

If any of the up to 32 temperature sensors is configured to report to the host, the CFA633 will send Temperature Sensor Reports for each selected sensor every second. See the command 19 (0x13): Set Up Temperature Reporting (Pg. 48).

```
type = 0x82
data_length = 4
data[0] is the index of the temperature sensor being reported:
       0 = temperature sensor 1
       1 = temperature sensor 2
       . . .
       31 = temperature sensor 32
data[1] is the MSB of Temperature_Sensor_Counts
data[2] is the LSB of Temperature_Sensor_Counts
data[3] is DOW_crc_status
```

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

```
void OnReceivedTempReport(COMMAND_PACKET *packet, char *output)
  {
  //First check the DOW CRC return code from the CFA633
  if(packet->data[3]==0)
    strcpy(output,"BAD CRC");
  else
    {
    double
      degc;
    degc=(*(short *)&(packet->data[1]))/16.0;

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

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

COMMAND CODES

Below is a list of valid commands for the CFA633. Each command packet is answered by either a response packet or an error packet.The low 6 bits of the `type` field of the response or error packet is the same as the low 6 bits of the `type` field of the command packet being acknowledged.

0 (0x00): Ping Command

The CFA633 will return the Ping Command to the host.

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

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

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

1 (0x01): Get Hardware & Firmware Version

The CFA633 will return the hardware and firmware version information to the host.

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

The return packet will be:

```
type: 0x40 | 0x01 = 0x41 = 65₁₀
data_length = 16
data[] = "CFA633:h#.#,y#v#"

X.X is the hardware revision, "h2.0" for example
Y.Y is the firmware version, "s2v1" for example
```

2 (0x02): Write User Flash Area

The CFA633 reserves 16 bytes of nonvolatile memory for arbitrary use by the host. This memory can be used to store a serial number, IP address, gateway address, netmask, or any other data required. All 16 bytes must be supplied.

```
type: 0x02 = 2(10)
valid data_length is 16
data[] = 16 bytes of arbitrary user data to be stored in
         the CFA633's non-volatile memory
```

The return packet will be:

```
type: 0x40 | 0x02 = 0x42 = 66(10)
data_length = 0
```

3 (0x03): Read User Flash Area

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

```
type: 0x03 = 3(10)
valid data_length is 0
```

The return packet will be:

```
type: 0x40 | 0x03 = 0x43 = 67(10)
data_length = 16
data[] = 16 bytes user data recalled from the CFA633's
         non-volatile memory
```

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

The CFA633 loads its power-up configuration from nonvolatile memory when power is applied. The CFA633 is configured at the factory to display a “welcome” screen when power is applied. This command can be used to customize the “welcome” screen, as well as the following items:

- Characters shown on LCD, which are affected by:
  - command 6 (0x06): Clear LCD Screen (Pg. 42).
  - command 7 (0x07): Set LCD Contents, Line 1 (Pg. 42).
  - command 8 (0x08): Set LCD Contents, Line 2 (Pg. 43).
  - command 31 (0x1F): Send Data to LCD (Pg. 58).
- Special character font definitions (command 9 (0x09): Set LCD Special Character Data (Pg. 43)).
- Cursor position (command 11 (0x0B): Set LCD Cursor Position (Pg. 44)).
- Cursor style (command 12 (0x0C): Set LCD Cursor Style (Pg. 44)).
- Contrast setting (command 13 (0x0D): Set LCD Contrast (Pg. 44)).
- Backlight setting (command 14 (0x0E): Set LCD & Keypad Backlight (Pg. 45)).
- Fan power settings (command 17 (0x11): Set Fan Power (Pg. 45)).
- Settings of any “live” displays (command 21 (0x15): Set Up Live Fan or Temperature Display (Pg. 49)).
- Key press and release masks (command 23 (0x17): Configure Key Reporting (Pg. 50)).
- Fan glitch delay settings (command 26 (0x1A): Set Fan Tachometer Glitch Filter (Pg. 52)).
- ATX function enable and pulse length settings (command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54)).
- Baud rate (command 33 (0x21): Set Baud Rate (Pg. 58)).
- GPIO settings (command 34 (0x22): Set or Set and Configure GPIO Pins (Pg. 58)).

You cannot store the fan or temperature reporting (although the live display of fans or temperatures can be saved). You cannot store the fan fail-safe or host watchdog.The host software should enable these items once the system is initialized and it is ready to receive the data.

```
type: 0x04 = 4(10)
valid data_length is 0
```

The return packet will be:

```
type: 0x40 | 0x04 = 0x44 = 68(10)
data_length = 0
```

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

5 (0x05): Reboot CFA633, Reset Host, or Power Off Host

This command instructs the CFA633 to simulate a power-on restart of itself, reset the host, or turn the host's power off. The ability to reset the host may be useful to allow certain host operating system configuration changes to complete. The ability to turn the host's power off under software control may be useful in systems that do not have ACPI compatible BIOS.

NOTE

The GPIO pins used for ATX control must not be configured as user GPIO, and must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are factory default, but may be changed by the user. Please see command 34 (0x22): Set or Set and Configure GPIO Pins (Pg. 58).

Rebooting the CFA633 may be useful when testing the boot configuration. It may also be useful to re-enumerate the devices (WR-DOW-Y17 temperature sensors) on the One-Wire bus. To reboot the CFA633, send the following packet:

```
type: 0x05 = 5(10)
valid data_length is 3
data[0] = 8
data[1] = 18
data[2] = 99
```

NOTE

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

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

If only one fan is on, there is no additional delay. If two fans are on, there is up to an additional 500ms delay. If three fans are on, there is a up to an additional 1,000ms delay. If all four fans are on, there is up to an additional 1,500ms delay.

# of Fans Powered On	Expected Boot Time
0 to 1	300ms - 500ms
2	800ms - 1,000ms
3	1.3s - 1.5s
4	1.8s - 2.0s

To reset the host, assuming the host's reset line is connected to GPIO[3] as described in command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54), send the following packet:

```
type: 0x05 = 5(10)
valid data_length is 3
data[0] = 12
data[1] = 28
data[2] = 97
```

NOTE

The CFA633 will return the acknowledge packet immediately, then reset the host. After resetting the host (~1.5 seconds), the module will reboot itself. The module will not respond to new command packets for up to 3 seconds (~4.5 seconds overall) after its reboot. Part of this delay is the intentional staggered sequencing of turning on power to the fans. If you are not using fans, you can speed the boot process by setting the fan power to 0 (command 17 (0x11): Set Fan Power and saving this as the default boot state (command 4 (0x04): Store Current State As Boot State). Normally, the host will be recovering from its own reset, so the boot delay of the module will not be of consequence.

To turn the host's power off, assuming the host's power control line is connected to GPIO[2] as described in command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54), send the following packet:

```
type: 0x05 = 5(10)
valid data_length is 3
data[0] = 3
data[1] = 11
data[2] = 95
```

NOTE

The CFA633 will return the acknowledge packet immediately, then power cycle the host. The power cycle length is dependent on the length of the power pulse (command 28 (0x1C): Set ATX Power Switch Functionality). After power cycling the host, the module will reboot itself. The module will not respond to new command packets for up to 3 seconds after its reboot. Part of this delay is the intentional staggered sequencing of turning on power to the fans. If you are not using fans, you can speed the boot process by setting the fan power to 0 (command 17 (0x11): Set Fan Power and saving this as the default boot state (command 4 (0x04): Store Current State As Boot State). Normally the host will be off or recovering from its own power cycle, so the boot delay of the module will not be of consequence.

In any of the above cases, the return packet will be:

```
type: 0x40 | 0x05 = 0x45 = 69(10)
data_length = 0
```

6 (0x06): Clear LCD Screen

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

```
type: 0x06 = 6(10)
valid data_length is 0
```

The return packet will be:

```
type: 0x40 | 0x06 = 0x46 = 70(10)
data_length = 0
```

Clear LCD Screen changes the LCD. The LCD contents is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

7 (0x07): Set LCD Contents, Line 1

Sets the center 16 characters displayed for the top line of LCD screen.

NOTE

Please use this command only if you need backwards compatibility with older CFA633 units. For new applications, please use the more flexible command 31 (0x1F): Send Data to LCD (Pg. 58).

```
type: 0x7 = 7(10)
valid data_length is 16
data[] = top line's display content (must supply 16 bytes)
```

The return packet will be:

```
type: 0x40 | 0x07 = 0x47 = 71(10)
data_length = 0
```

Set LCD Contents, Line 1 is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

8 (0x08): Set LCD Contents, Line 2

Sets the center 16 characters displayed for the bottom line of LCD screen.

> **NOTE**
> Please use this command only if you need backwards compatibility with older CFA633 units. For new applications, please use the more flexible command 31 (0x1F): Send Data to LCD (Pg. 58).

```
type: 0x8 = 8[10]
valid data_length is 16
data[] = bottom line's display content (must supply 16 bytes)
```

The return packet will be:

```
type: 0x40 | 0x08 = 0x48 = 72[10]
data_length = 0
```

Set LCD Contents, Line 2 is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

9 (0x09): Set LCD Special Character Data

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

```
type: 0x09 = 9[10]
valid data_length is 9
data[0] = index of special character that you would like to modify, 0-7 are valid
data[1-8] = bitmap of the new font for this character
```

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

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

The return packet will be:

```
type: 0x40 | 0x09 = 0x49 = 73[10]
data_length = 0
```

Set LCD Special Character Data is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

10 (0x0A): Read 8 Bytes of LCD Memory

This command will return the contents of the LCD's DDRAM or CGRAM. This command is intended for debugging.

Note: firmware version prior to v1.9 did not return the address code.

```
type: 0x0A = 10[10]
valid data_length is 1
data[0] = address code of desired data
```

`data[0]` is the address code native to the LCD controller:

```
0x40 (\064) to 0x7F (\127) for CGRAM
0x80 (\128) to 0x93 (\143) for DDRAM, line 1
0xC0 (\192) to 0xD3 (\207) for DDRAM, line 2
```

The return packet will be:

```
type: 0x40 | 0x0A = 0x4A = 74(10)
data_length = 9
```

`data[0]` of the return packet will be the address code.
`data[1-8]` of the return packet will be the data read from the LCD controller's memory.

11 (0x0B): Set LCD Cursor Position

This command allows the cursor to be placed at the desired location on the CFA633's LCD screen. If you want the cursor to be visible, you may also need to send a command 12 (0x0C): Set LCD Cursor Style (Pg. 44).

```
type: 0x0B = 11(10)
valid data_length is 2
data[0] = column (0-15 valid)
data[1] = row (0-1 valid)
```

The return packet will be:

```
type: 0x40 | 0x0B = 0x4B = 75(10)
data_length = 0
```

Set LCD Cursor Position is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

12 (0x0C): Set LCD Cursor Style

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

```
type: 0x0C = 12(10)
valid data_length is 1
data[0] = cursor style (0-3 valid)
        0 = no cursor.
        1 = blinking block cursor.
        2 = static underscore cursor.
        3 = blinking underscore cursor.
        (Note for 3: This behavior is different from the previous CFA633 HW v1.x family
        which had a blinking block with a static underscore cursor.)
```

The return packet will be:

```
type: 0x40 | 0x0C = 0x4C = 76(10)
data_length = 0
```

Set LCD Cursor Style is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

13 (0x0D): Set LCD Contrast

This command sets the contrast or vertical viewing angle of the display. Initiated by the host, responded to by the CFA633.

```
type: 0x0D = 13(10)
valid data_length is 1
data[0] = contrast setting (0-50 valid)
 0 = light         (Vlcd=4v)
16 = about right   (Vlcd=5v)
29 = dark          (Vlcd=6v)
30-50 = very dark  (Vlcd=6.0 to 6.7v)
                   (may be useful at cold temperatures)
```

The return packet will be:

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

Set LCD Contrast is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

14 (0x0E): Set LCD & Keypad Backlight

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

```
type: 0x0E = 14(10)
valid data_length is 1
data[0] = backlight power setting (0-100 valid)
        0 = off
     1-99 = variable brightness
      100 = on
```

The return packet will be:

```
type: 0x40 | 0x0E = 0x4E = 78(10)
data_length = 0
```

Set LCD & Keypad Backlight is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

15 (0x0F): (Deprecated)

16 (0x10): Set Up Fan Reporting

This command will configure the CFA633 to report the fan speed information to the host every 500 mS.

```
type = 0x10 = 16(10)
valid data_length is 1
data[0] = bitmask indicating which fans are enabled to
          report (0-15 valid)
---- 8421 Enable Reporting of this Fan's Tach Input
|||| ||||-- Fan 1: 1 = enable, 0 = disable
|||| |||--- Fan 2: 1 = enable, 0 = disable
|||| ||---- Fan 3: 1 = enable, 0 = disable
|||| |----- Fan 4: 1 = enable, 0 = disable
```

The return packet will be:

```
type = 0x40 | 0x10 = 0x50 = 80(10)
data_length = 0
```

If `data[0]` is not 0, then the CFA633 will start sending 0x81: Fan Speed Report packets for each enabled fan every 500 mS. (See 0x81: Fan Speed Report (Pg. 36).) Each of the report packets is staggered by 1/8 of a second.

Reporting a fan will override the fan power setting to 100% for up to 1/8 of a second every 1/2 second. Please see Fan Connections in Fan Connections (Pg. 31) for a detailed description.

17 (0x11): Set Fan Power

This command will configure the power for the fan connectors. The fan power setting is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

```
type = 0x11 = 17(10)
valid data_length is 4
data[0] = power level for FAN 1 (0-100 valid)
data[1] = power level for FAN 2 (0-100 valid)
data[2] = power level for FAN 3 (0-100 valid)
data[3] = power level for FAN 4 (0-100 valid)
```

The return packet will be:

```
type = 0x40 | 0x11 = 0x51 =81(10)
data_length = 0
```

Set Fan Power is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

18 (0x12): Read DOW Device Information

When power is applied to the CFA633, it detects any devices (WR-DOW-Y17 temperature sensors) connected to the Dallas Semiconductor One-Wire (DOW) bus and stores the device's information. This command will allow the host to read the device's information.

The first byte returned is the Family Code of the Dallas One-Wire / iButton device. There is a list of the possible Dallas One-Wire / iButton device family codes available in App Note 155: 1-Wire Software Resource Guide on the Maxim/Dallas website.

NOTE ON COMMAND 18: READ DOW DEVICE INFORMATION

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

These settings are factory default but may be changed by the user. Please see command 34 (0x22): Set or Set and Configure GPIO Pins (Pg. 58).

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

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

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

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

This setting must be saved as the boot state, so when the CFA633 reboots it will detect the DOW devices.

```
type: 0x12 = 18(10)
valid data_length is 1
data[0] = device index (0-31 valid)
```

The return packet will be:

```
type: 0x40 | 0x12 = 0x52 = 82(10)
data_length = 9
data[0] = device index (0-31 valid)
data[1-8] = ROM ID of the device
```

If data[1] is 0x22 (DS1822 Econo One-Wire Digital Thermometer temperature sensor) or 0x28 (DS18B20 High Precision One-Wire Digital Thermometer temperature sensor used on our WR-DOW-Y17), then that device can be set up to automatically convert and report the temperature every second. See the command 19 (0x13): Set Up Temperature Reporting (Pg. 48).

19 (0x13): Set Up Temperature Reporting

This command will configure the CFA633 to report the temperature information to the host every second.

```
type: 0x13 = 19(10)
valid data_length is 4
data[0-3] = 32-bit bitmask indicating which temperature
            sensors fans are enabled to report (0-255 valid in each location)

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

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

data[2]
24 23 22 21   20 19   18 17  Enable Reporting of sensor with
|  |  |  |    |  |    |  |   device index of:
|  |  |  |    |  |    |  |--  16: 1 = enable, 0 = disable
|  |  |  |    |  |    |-----  17: 1 = enable, 0 = disable
|  |  |  |    |  |---------  18: 1 = enable, 0 = disable
|  |  |  |    |------------  19: 1 = enable, 0 = disable
|  |  |  |-----------------  20: 1 = enable, 0 = disable
|  |  |--------------------  21: 1 = enable, 0 = disable
|  |-----------------------  22: 1 = enable, 0 = disable
|--------------------------  23: 1 = enable, 0 = disable

data[3]
32 31 30 29   28 27   26 25  Enable Reporting of sensor with
|  |  |  |    |  |    |  |   device index of:
|  |  |  |    |  |    |  |--  24: 1 = enable, 0 = disable
|  |  |  |    |  |    |-----  25: 1 = enable, 0 = disable
|  |  |  |    |  |---------  26: 1 = enable, 0 = disable
|  |  |  |    |------------  27: 1 = enable, 0 = disable
|  |  |  |-----------------  28: 1 = enable, 0 = disable
|  |  |--------------------  29: 1 = enable, 0 = disables
|  |-----------------------  30: 1 = enable, 0 = disable
|--------------------------  31: 1 = enable, 0 = disable
```

Any sensor enabled must have been detected as a 0x22 (DS1822 temperature sensor) or 0x28 (DS18B20 temperature sensor) during DOW enumeration. This can be verified by using the command 18 (0x12): Read DOW Device Information (Pg. 47).

The return packet will be:

```
type: 0x40 | 0x13 = 0x53 = 83(10)
data_length = 0
```

20 (0x14): Arbitrary DOW Transaction

The CFA633 can function as an RS-232 to Dallas One-Wire bridge. This command allows you to specify arbitrary transactions on the One-Wire bus. One-Wire commands follow this basic layout:

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

Please see APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER (Pg. 66) for an example of using this command.

```
type: 0x14 = 20(10)
valid data_length is 2 to 16
data[0] = device_index (0-32 valid)
data[1] = number_of_bytes_to_read (0-14 valid)
data[2-15] = data_to_be_written[data_length-2]
```

If **device_index** is 32, then no address phase will be executed. If **device_index** is in the range of 0 to 31, and a One-Wire device was detected for that **device_index** at power on, then the write cycle will be prefixed with a "Match ROM" command and the address information for that device.

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

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

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

The return packet will be:

```
type: 0x40 | 0x14 = 0x54 = 84(10)
data_length = 2 to 16
data[0] = device index (0-31 valid)
data[data_length-2] = Data read from the One-Wire bus. This is the same
                      as number_of_bytes_to_read from the command.
data[data_length-1] = One-Wire CRC
```

21 (0x15): Set Up Live Fan or Temperature Display

You can configure the CFA633 to automatically update a portion of the LCD with a “live” RPM or temperature reading. Once the display is configured using this command, the CFA633 will continue to display the live reading on the LCD without host intervention. The Set Up Live Fan or Temperature Display is one of the items stored by command 4 (0x04): Store Current State As Boot State (Pg. 39), so you can configure the CFA633 to immediately display fan speeds or system temperatures as soon as power is applied.

The live display is based on a concept of display slots. There are 8 slots, and each of the 8 slots may be enabled or disabled independently.

Any slot may be requested to display any data that is available. For instance, slot 0 could display temperature sensor 3 in °C, while slot 1 could simultaneously display temperature sensor 3 in °F.

Any slot may be positioned at any location on the LCD, as long as all the digits of that slot fall fully within the display area. It is legal to have the display area of one slot overlap the display area of another slot, but senseless. This situation should be avoided in order to have meaningful information displayed.

```
type: 0x15 = 21(10)
valid data_length is 7 or 2 (for turning a slot off)
data[0]: display slot (0-7)
data[1]: type of item to display in this slot
          0 = nothing (data_length then must be 2)
1 = fan tachometer RPM (data_length then must be 7)
2 = temperature (data_length then must be 7)
data[2]: index of the sensor to display in this slot:
          0-3  are valid for fans
          0-31 are valid for temperatures (and the temperature
               device must be attached)
data[3]: number of digits
          for a fan: 4 digits (0 to  9999) valid fan speed range
          for a fan: 5 digits (0 to 50000) valid fan speed range
          for a temperature: 3 digits (  -XX or   XXX)
          for a temperature: 5 digits (-XX.X or XXX.X)
data[4]: display column
          0-13 valid for a 3-digit temperature
          0-12 valid for a 4-digit fan
          0-11 valid for a 5-digit fan or temperature
data[5]: display row (0-1 valid)
data[6]: pulses_per_revolution or temperature units
          for a fan: pulses per revolution for this fan (1 to 32)
          for a temperature: units (0 = deg C, 1 = deg F)
```

If a One-Wire CRC error is detected, the temperature will be displayed as "ERR" or "ERROR".

If the frequency of the tachometer signal is below the detectable range, the speed will be displayed as "SLOW" or "STOP".

Displaying a fan will override the fan power setting to 100% for up to 1/8 of a second every 1/2 second. Please see Fan Connections (Pg. 31) for a detailed description.

The return packet will be:

```
type: 0x40 | 0x15 = 0x55 = 85(10)
data_length = 0
```

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

The controller on the CFA633 is HD44780 compatible. Generally you won't need low-level access to the LCD controller but some arcane functions of the HD44780 are not exposed by the CFA633's command set. This command allows you to access the CFA633's LCD controller directly. Note: It is possible to corrupt the CFA633 display using this command.

```
type: 0x16 = 22(10)
data_length: 2
data[0]: location code
       0 = "Data" register
       1 = "Control" register
data[1]: data to write to the selected register
```

The return packet will be:

```
type: 0x40 | 0x16 = 0x56 = 86(10)
data_length = 0
```

23 (0x17): Configure Key Reporting

By default, the CFA633 reports any key event to the host. This command allows the key events to be enabled or disabled on an individual basis. The key events set to report are one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

```
#define KP_UP     0x01
#define KP_ENTER  0x02
#define KP_CANCEL 0x04
#define KP_LEFT   0x08
#define KP_RIGHT  0x10
#define KP_DOWN   0x20

type: 0x17 = 23(10)
data_length = 2
data[0]: press mask
data[1]: release mask
```

The return packet will be:

```
type: 0x40 | 0x17 = 0x57 = 87(10)
data_length = 0
```

Configure Key Reporting is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the CFA633 for key activity. This command allows the host to detect which keys are currently pressed, which keys have been pressed since the last poll, and which keys have been released since the last poll.

This command is independent of the key reporting masks set by command 23 (0x17): Configure Key Reporting (Pg. 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.

```
#define KP_UP     0x01
#define KP_ENTER  0x02
#define KP_CANCEL 0x04
#define KP_LEFT   0x08
#define KP_RIGHT  0x10
#define KP_DOWN   0x20

type: 0x18 = 24(10)
data_length = 0
```

The return packet will be:

```
type: 0x40 | 0x18 = 0x58 = 88(10)
data_length = 3
data[0] = bit mask showing the keys currently pressed
data[1] = bit mask showing the keys that have been pressed since
          the last poll
data[2] = bit mask showing the keys that have been released since
          the last poll
```

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

The CFA633 can be used as part of an active cooling system. For instance, the fans in a system can be slowed down to reduce noise when a system is idle or when the ambient temperature is low, and sped up when the system is under heavy load or the ambient temperature is high.

Since there are a very large number of ways to control the speed of the fans (thresholds, thermostat, proportional, PID, multiple temperature sensors contributing to the speed of several fans . . .) there was no way to foresee the particular requirements of your system and include an algorithm in the CFA633's firmware that would be an optimal fit for your application.

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

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

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

type = 0x19 = 25[10]
data_length = 2
data[0] = bit mask of fans set to fail-safe (1-15 valid)
data[1] = timeout value in 1/8 second ticks:
       1 = 1/8 second
       2 = 1/4 second
     255 = 31 7/8 seconds
```

The return packet will be:

```
type = 0x40 | 0x19 = 0x59 = 89[10]
data_length = 0
```

26 (0x1A): Set Fan Tachometer Glitch Filter

The CFA633 controls fan speed by using PWM. Using PWM turns the power to a fan on and off quickly to change the average power delivered to the fan. The CFA633 uses approximately 18 Hz for the PWM repetition rate. The fan's tachometer output is only valid if power is applied to the fan. Most fans produce a valid tachometer output very quickly after the fan has been turned back on but some fans take time after being turned on before their tachometer output is valid.

This command allows you to set a variable-length delay after the fan has been turned on before the CFA633 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 RPM monitoring slightly more intrusive at low power settings. Also, the higher values will increase the lowest speed that a fan with RPM reporting enabled will seek at 0% power setting.

The Fan Glitch Delay is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

```
type = 0x1A = 26[10]
data_length = 4
data[0] = delay count of fan 1
data[1] = delay count of fan 2
data[2] = delay count of fan 3
data[3] = delay count of fan 4
```

The return packet will be:

```
type = 0x40 | 0x1A = 0x5A = 90[10]
data_length = 0
```

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

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

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

type = 0x1B = 27[10]
data_length = 0
```

The return packet will be:

```
type = 0x40 | 0x1B = 0x5B = 91[10]
data_length = 5
data[0] = fan 1 power
data[1] = fan 2 power
data[2] = fan 3 power
data[3] = fan 4 power
data[4] = bit mask of fans with fail-safe set
```

28 (0x1C): Set ATX Power Switch Functionality

The combination of the CFA633 with the Crystalfontz WR-PWR-Y14 cable can be used to replace the function of the power and reset switches in a standard ATX-compatible system. The ATX Power Switch Functionality is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

NOTE ON COMMAND 28: SET ATX SWITCH FUNCTIONALITY

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

These settings are factory default but may be changed by the user. Please see command 34 (0x22): Set or Set and Configure GPIO Pins (Pg. 58). These settings must be saved as the boot state.

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

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

This configuration can be assured by sending the following command:

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

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

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

This configuration can be assured by sending the following command:

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

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

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

This configuration can be assured by sending the following command:

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

These settings must be saved as the boot state.

The RESET (GPIO[3]) and POWER CONTROL (GPIO[2]) lines on the CFA633 are normally high-impedance. Electrically, they appear to be disconnected or floating. When the CFA633 asserts the RESET or POWER CONTROL lines, they are momentarily driven high or low (as determined by the AUTO_POLARITY, RESET_INVERT or POWER_INVERT bits, detailed below). To end the power or reset pulse, the CFA633 changes the lines back to high-impedance.

FOUR FUNCTIONS MAY BE ENABLED BY COMMAND 28

Function 1: KEYPAD_RESET

If POWER-ON SENSE (GPIO[1]) is high, holding the green check key for 4 seconds will pulse RESET (GPIO[3]) pin for 1 second. During the 1-second pulse, the CFA633 will show "RESET", and then the CFA633 will reset itself, showing its boot state as if it had just powered on. Once the pulse has finished, the CFA633 will not respond to any commands until after it has reset the host and itself.

Function 2: KEYPAD_POWER_ON

If POWER-ON SENSE (GPIO[1]) is low, pressing the green check key for 0.25 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified by in data[1] or the default of 1 second. During this time the CFA633 will show "POWER ON", then the CFA633 will reset itself.

Function 3: KEYPAD_POWER_OFF

If POWER-ON SENSE (GPIO[1]) is high, holding the red X key for 4 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified by in data[1] or the default of 1 second. If the user continues to hold the power key down, then the CFA633 will continue to drive the line for a maximum of 5 additional seconds. During this time the CFA633 will show "POWER OFF".

Function 4: LCD_OFF_IF_HOST_IS_OFF

If LCD_OFF_IF_HOST_IS_OFF is set, the CFA633 will blank its screen and turn off its backlight to simulate its power being off any time POWER-ON SENSE is low.

NOTE

By default there is an internal POWER-ON-SENSE connected to the +5v pin of J_PWR, selected by setting data[2] to 1. Alternatively, GPIO[1] may be configured to act as POWER-ON-SENSE through R21 of 5K, and specifying data[2] as 0. The CFA633 will still be active (since it is powered by V_{SB}, standby power which is always-on), monitoring the keypad for a power-on keystroke. Once POWER-ON SENSE goes high, the CFA633 will reboot as if power had just been applied to it.

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

type: 0x1C = 28(10)
data_length: 1 or 2
data[0]: bit mask of enabled functions
data[1]: (optional) length of power on & off pulses in 1/32 second
       1 = 1/32 sec
       2 = 1/16 sec
      16 = 1/2 sec
     254 = 7.9 seconds
     255 = Assert power control line until host power state changes
```

The return packet will be:

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

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

Some high-availability systems use hardware watchdog timers to ensure that a software or hardware failure does not result in an extended system outage. Once the host system has booted, a system monitor program is started. The system monitor program would enable the watchdog timer on the CFA633. If the system monitor program fails to reset the CFA633's watchdog timer, the CFA633 will reset the host system.

NOTE

The GPIO pins used for ATX control must not be configured as user GPIO. They must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are factory default, but may be changed by the user. Please see the note under command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54) or command 34 (0x22): Set or Set and Configure GPIO Pins (Pg. 58).

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

If timeout is 0, the watchdog is disabled.

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

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

If the command is not re-issued within timeout seconds, then the CFA633 will reset the
host (see command 28 for details). Since the watchdog is off by default when the CFA633
powers up, the CFA633 will not issue another host reset until the host has once again
enabled the watchdog.
```

The return packet will be:

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

30: Read Reporting & Status

This command can be used to verify the current items configured to report to the host, as well as some other miscellaneous status information. Please note that the information returned by other modules is not identical to this.

```
type = 0x1E = 30(10)
data_length = 0
```

The return packet will be:

```
type = 0x40 | 0x1E = 0x5E = 94(10)
data_length = 15
data[0] = fan 1-4 reporting status (as set by command 16)
data[1] = temperatures 1-8 reporting status (as set by command 19)
data[2] = temperatures 9-15 reporting status (as set by command 19)
data[3] = temperatures 16-23 reporting status (as set by command 19)
data[4] = temperatures 24-32 reporting status (as set by command 19)
data[5] = key presses (as set by command 23)
data[6] = key releases (as set by command 23)
data[7] = ATX Power Switch Functionality (as set by command 28),
data[8] = current watchdog counter (as set by command 29)
data[9] = fan RPM glitch delay[0] (as set by command 26)
data[10] = fan RPM glitch delay[1] (as set by command 26)
data[11] = fan RPM glitch delay[2] (as set by command 26)
data[12] = fan RPM glitch delay[3] (as set by command 26)
data[13] = contrast setting (as set by command 13)
data[14] = backlight setting (as set by command 14)
```

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

31 (0x1F): Send Data to LCD

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

```
type: 0x1F = 31(10)
data_length = 3 to 18
data[0]: col = x = 0 to 15
data[1]: row = y = 0 to 1
data[2-21]: text to place on the LCD, variable from 1 to 16 characters
```

The return packet will be:

```
type: 0x40 | 0x1F = 0x5F = 95(10)
data_length = 0
```

Send Data to LCD is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

32: Reserved for CFA631 Key Legends

33 (0x21): Set Baud Rate

This command will change the CFA633's baud rate. The CFA633 will send the acknowledge packet for this command and change its baud rate to the new value. The host should send the baud rate command, wait for a positive acknowledge from the CFA633 at the old baud rate, and then switch itself to the new baud rate. The baud rate must be saved by the command 4 (0x04): Store Current State As Boot State (Pg. 39) if you want the CFA633 to power up at the new baud rate.

The factory default baud rate is 19200.

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

The return packet will be:

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

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

The CFA633 (hardware versions v1.4 and up, firmware versions 1.9 and up) has five pins for user-definable general purpose input / output (GPIO). These pins are shared with the DOW and ATX functions. Be careful when you configure the GPIO if you want to use the ATX or DOW at the same time.

The architecture of the CFA633 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 CFA633 continuously polls the GPIOs as inputs at 32 Hz. The present level can be queried by the host software at a lower rate. The CFA633 also keeps track of whether there were rising or falling edges since the last host query (subject

to the resolution of the 32 Hz sampling). This means that the host is not forced to poll quickly in order to detect short events. The algorithm used by the CFA633 to read the inputs is inherently “bounce-free”.

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

Pull-up/pull-down resistance values are approximately 5kΩ. Do not exceed current of 25 mA per GPIO.

NOTE ON SETTING AND CONFIGURING GPIO PINS

The GPIO pins may also be used for ATX control through header J8 and temperature sensing through the CFA633’s DOW header. By factory default, the GPIO output setting, function, and drive mode are set correctly to enable operation of the ATX and DOW functions. **The GPIO output setting, function, and drive mode must be set to the correct values in order for the ATX and DOW functions to work. Improper use of this command can disable the ATX and DOW functions.**The 633_WinTest may be used to easily check and reset the GPIO configuration to the default state so the ATX and DOW functions will work.

The GPIO configuration is one of the items stored by the command 4 (0x04): Store Current State As Boot State (Pg. 39).

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

data[0]: index of GPIO to modify
       0 = GPIO[0] = J8, Pin 7
       1 = GPIO[1] = J8, Pin 6 (default is ATX Host Power Sense)
       2 = GPIO[2] = J8, Pin 5 (default is ATX Host Power Control)
       3 = GPIO[3] = J8, Pin 4 (default is ATX Host Reset Control)
       4 = GPIO[4] = J9, Pin 2 (default is DOW I/O--always has 1kΩ hardware pull-up)
   5-255 = reserved

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

(Continues on the next page.)

```
data[1] = Pin output state (actual behavior depends on drive mode):
       0 = Output set to low
    1-99 = Output duty cycle percentage (100 Hz nominal)
     100 = Output set to high
 101-255 = invalid

data[2] = Pin function select and drive mode (optional, 0-15 valid)
  ---- FDDD
  |||| ||||-- DDD = Drive Mode (based on output state of 1 or 0)
  |||| |       =====================================================
  |||| |       000: 1=Fast, Strong Drive Up, 0=Resistive Pull Down
  |||| |       001: 1=Fast, Strong Drive Up, 0=Fast, Strong Drive Down
  |||| |       010: Hi-Z, use for input
  |||| |       011: 1=Resistive Pull Up,     0=Fast, Strong Drive Down
  |||| |       100: 1=Slow, Strong Drive Up, 0=Hi-Z
  |||| |       101: 1=Slow, Strong Drive Up, 0=Slow, Strong Drive Down
  |||| |       110: reserved, do not use
  |||| |       111: 1=Hi-Z,                  0=Slow, Strong Drive Down
  |||| |
  |||| |----- F = Function
  ||||         =====================================================
  ||||         0: Port unused for GPIO. It will take on the default
  ||||            function such as ATX, DOW or unused. The user is
  ||||            responsible for setting the drive to the correct
  ||||            value in order for the default function to work
  ||||            correctly.
  ||||         1: Port used for GPIO under user control. The user is
  ||||            responsible for setting the drive to the correct
  ||||            value in order for the desired GPIO mode to work
  ||||            correctly.
  ||||------- reserved, must be 0
```

The return packet will be:

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

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

Please see command 34 (0x22): Set or Set and Configure GPIO Pins (Pg. 58) for details on the GPIO architecture.

```
type: 0x23 = 35(10)
data_length: 1
data[0]: index of GPIO to query
       0 = GPIO[0] = J8, Pin 7
       1 = GPIO[1] = J8, Pin 6 (default is ATX Host Power Sense)
       2 = GPIO[2] = J8, Pin 5 (default is ATX Host Power Control)
       3 = GPIO[3] = J8, Pin 4 (default is ATX Host Reset Control)
       4 = GPIO[4] = J9, Pin 2 (default is DOW I/O--always has 1KΩ hardware pull-up on
           SCAB.)
   5-255 = reserved

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

The return packet will be:

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

(Continues on the next page.)

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

CHARACTER GENERATOR ROM (CGROM)

To find the code for a given character, add the two numbers that are shown in bold for its row and column. For example, the Greek letter "β" is in the column labeled “224d” and in the row labeled “2d”. Add 224 + 2 to get 226. When you send a byte with the value of 226 to the display, the Greek letter "β" will be shown.

Figure 16. Character Generated ROM

LCD MODULE RELIABILITY AND LONGEVITY

MODULE RELIABILITY

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

ITEM	SPECIFICATION
LCD portion (excluding Keypad and Backlights)	50,000 to 100,000 hours (typical)
Keypad	1,000,000 keystrokes
Red LED Display and Red LED Keypad Backlights	50,000 to 100,000 hours (typical)

MODULE LONGEVITY (EOL / REPLACEMENT POLICY)

Crystalfontz is committed to making all of our LCD modules available for as long as possible. For each module we introduce, we intend to offer it indefinitely. We do not preplan a module's obsolescence. The majority of modules we have introduced are still available.

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

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

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

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

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

CARE AND HANDLING PRECAUTIONS

For optimum operation of the CFA633-RDI-KS and to prolong its life, please follow the precautions described below.

ESD (ELECTROSTATIC DISCHARGE)

Tx and Rx pins of connector RS-232 only:
+15 kV Human Body Model
+15 kV IEC1000-4-2 Air Discharge
+8 kV IEC1000-4-2 Contact Discharge

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

DESIGN AND MOUNTING

- The exposed surface of the “glass” is actually a polarizer laminated on top of the glass. To protect the soft plastic polarizer from damage, the module ships with a protective film over the polarizer. Please peel off the protective film slowly. Peeling off the protective film abruptly may generate static electricity.
- The polarizer is made out of soft plastic and is easily scratched or damaged. When handling the module, avoid touching the polarizer. Finger oils are difficult to remove.
- *CFA633-RDI-KS without Crystalfontz overlay:* To protect the soft plastic polarizer from damage, place a transparent plate (for example, acrylic, polycarbonate or glass) in front of the module, leaving a small gap between the plate and the display surface.
- Do not disassemble or modify the module.
- Do not modify the six 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 module.

AVOID SHOCK, IMPACT, TORQUE, OR TENSION

- Do not expose the CFA633-RDI-KS to strong mechanical shock, impact, torque, or tension.
- Do not drop, toss, bend, or twist the CFA633-RDI-KS.
- Do not place weight or pressure on the CFA633-RDI-KS.
- If the LCD panel breaks, be careful to not get the liquid crystal fluid in your mouth or eyes. If the liquid crystal fluid touches your skin, clothes, or work surface, wash it off immediately using soap and plenty of water.

IF LCD PANEL BREAKS

- If the LCD panel breaks, be careful to not get the liquid crystal fluid in your mouth or eyes.
- If the liquid crystal fluid touches your skin, clothes, or work surface, wash it off immediately using soap and plenty of water.

CLEANING

- The polarizer (laminated to the glass) is soft plastic. The soft plastic is easily scratched or damaged. Be very careful when you clean the polarizer.
- Do not clean the polarizer with liquids. Do not wipe the polarizer with any type of cloth or swab (for example, Q-tips).

- Use the removable protective film to remove smudges (for example, fingerprints) and any foreign matter. If you no longer have the protective film, use standard transparent office tape (for example, Scotch® brand “Crystal Clear Tape”). If the polarizer is dusty, you may carefully blow it off with clean, dry, oil-free compressed air.
- *CFA633-RDI-KS without Crystalfontz overlay:* The exposed surface of the LCD “glass” is actually the front polarizer laminated to the glass. The polarizer is made out of a fairly soft plastic and is easily scratched or damaged. The polarizer will eventually become hazy if you do not take great care when cleaning it. Long contact with moisture (from condensation or cleaning) may permanently spot or stain the polarizer.

OPERATION

- Your circuit should be designed to protect the CFA633-RDI-KS from ESD and power supply transients.
- Observe the operating temperature limitations: a minimum of -20°C to a maximum of +70°C with minimal fluctuation. Operation outside of these limits may shorten life and/or harm display.
  - At lower temperatures of this range, response time is delayed.
  - At higher temperatures of this range, display becomes dark. (You may need to adjust the contrast.)
- Operate away from dust, moisture, and direct sunlight.

STORAGE AND RECYCLING

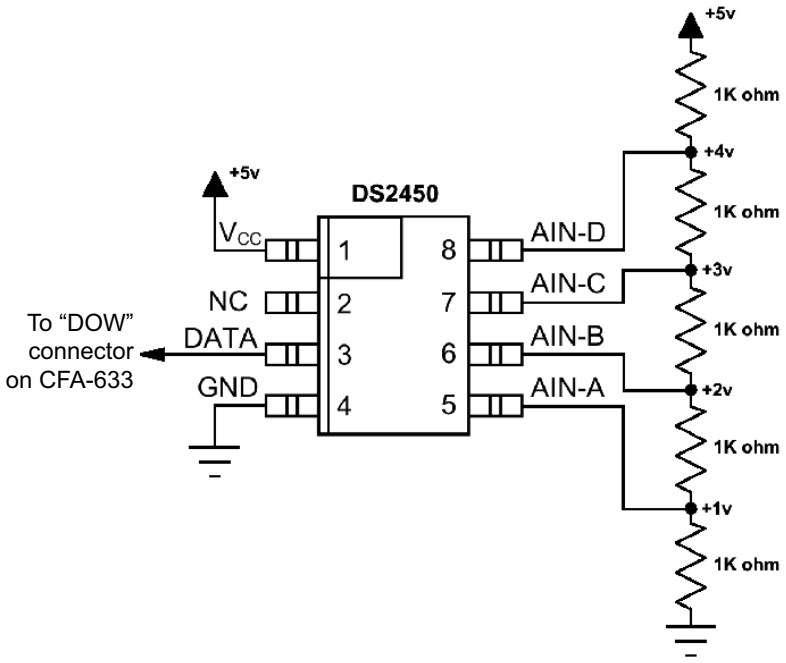
- Store in an ESD-approved container away from dust, moisture, and direct sunlight.
- Observe the storage temperature limitations: -30°C minimum, +80°C maximum with minimal fluctuation. Rapid temperature changes can cause moisture to form, resulting in permanent damage.
- Do not allow weight to be placed on the CFA633-RDI-KS while they are in storage.
- Please recycle your outdated Crystalfontz modules at an approved facility.

APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER

This appendix describes a simple test circuit that demonstrates how to connect a Dallas Semiconductor DS2450 4-channel ADC to the CFA633's DOW (Dallas One Wire) connector. It also gives a sample command sequence to initialize and read the ADC.

Up to 32 DOW devices can be connected to the CFA633. In this example the DS2450 appears at device index 0. Your software should query the connected devices using command 18 (0x12): Read DOW Device Information (Pg. 47) to verify the locations and types of DOW devices connected in your application.

Please refer to the DS2450 Data Sheet and the description for command 20 (0x14): Arbitrary DOW Transaction (Pg. 49) more information.



Appendix A, Figure 1. Test Circuit Schematic

Start 633_WinTest and open the Packet Debugger dialog.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

//Decoded response:
0x3300 = 13056 1.016015625 volts (channel A)
0x64DF = 25823 2.009541321 volts (channel B)
0x9684 = 38532 2.998553467 volts (channel C)
0xC86A = 51306 3.992623901 volts (channel D)

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

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

//Decoded response:
0x336B = 13163 1.024342346 volts (channel A)
0x64B2 = 25778 2.006039429 volts (channel B)
0x9697 = 38551 3.000032043 volts (channel C)
0xC842 = 51266 3.989511108 volts (channel D)
```

APPENDIX B: CONNECTING A DS1963S SHA IBUTTON

This appendix describes connecting a Dallas Semiconductor DS1963S Monetary iButton with SHA-1 Challenge Response Algorithm and 4KB of nonvolatile RAM to the CFA633's DOW (Dallas One Wire) connector. It also gives a sample command sequence to read and write the DS1963S's scratch memory.

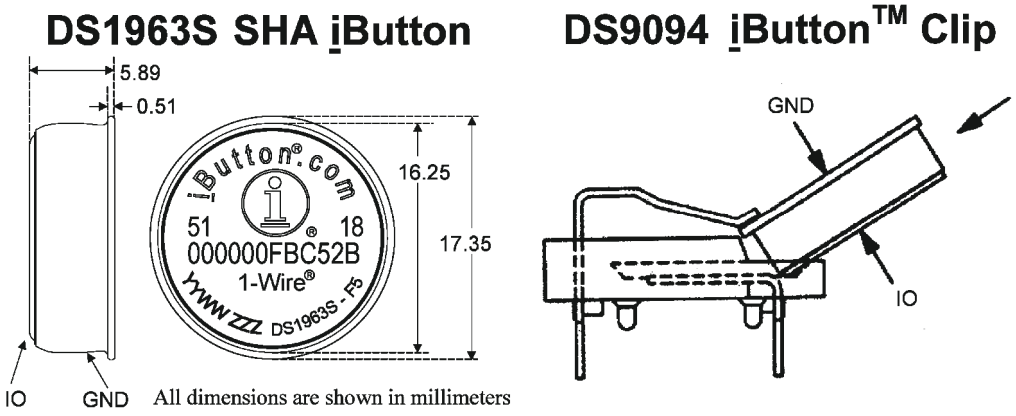
The DS1963S can be used as a secure dongle to protect your system's application software from being copied. Even if the communication channel is compromised or the host is not authentic, the SHA algorithm ensures that the data is still secure. Please see the following Maxim/Dallas white papers and application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

- White Paper 1: SHA Devices Used in Small Cash Systems
- White Paper 2: Using the 1-Wire Public-Domain Kit
- White Paper 3: Why are 1-Wire SHA-1 Devices Secure?
- White Paper 4: Glossary of 1-Wire SHA-1 Terms
- App Note 1201: White Paper 8: 1-Wire SHA-1 Overview
- App Note 150: Small Message Encryption using SHA Devices
- App Note 152: SHA iButton Secrets and Challenges
- App Note 154: Passwords in SHA Authentication
- App Note 156: DS1963S SHA 1-Wire API Users Guide
- App Note 157: SHA iButton API Overview
- App Note 190: Challenge and Response with 1-Wire SHA devices

Up to 32 DOW devices can be connected to the CFA633. In this example the DS1963S appears at device index 0. Your software should query the connected devices using command 19 (0x13): Set Up Temperature Reporting (Pg. 48) to verify the locations and types of DOW devices connected in your application.

Please refer to the DS1963S Data Sheet and the description for command 20 (0x14): Arbitrary DOW Transaction (Pg. 49) for more information.

To connect the DS1963S to the CFA633, simply make one connection between the DS1963S's "GND" terminal and the CFA633 DOW connector's GND pin, and a second connection between the DS1963S's "IO" pin and the CFA633 DOW connector's I/O pin. By using a DS9094 iButton Clip, the connection is easy.



Appendix B, Figure 1. Connecting DS1963S SHA iButton using DS9094 iButton Clip

To demonstrate reading and writing the scratch memory on DS1963S, open the 633_WinTest Packet Debugger dialog and use it to experiment with the following commands: Erase Scratchpad, Read Scratchpad, and Write Scratchpad.

To use the full power of the DS1963S, a program based on the Dallas/Maxim application notes listed above is needed. The challenge/response sequence would be unwieldy to demonstrate using the 633_WinTest Packet Debugger dialog.

First read the address of the DS1963S as detected by the CFA633 at boot. Since only one device is connected, you only need to query index 0. In a production situation, query all 32 indices to get a complete picture of the devices available on the DOW bus.

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

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

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

Erase Scratchpad [C3h]
The purpose of this command is to clear the HIDE flag and to wipe out data that might have been left in the scratchpad from a previous operation. After having issued the command code the bus master transmits a target address, as with the write scratchpad command, but no data. Next the whole scratchpad will be automatically filled with FFh bytes, regardless of the target address. This process takes approximately 32 µs during which the master reads 1's. After this the master reads a pattern of alternating 0's and 1's indicating that the command has completed. The master must read at least 8 bits of this alternating pattern. Otherwise the device might not properly respond to a subsequent Reset Pulse.

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

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

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

Read Scratchpad Command [AAh]
HIDE = 0:
The Read Scratchpad command allows verifying the target address, ending offset and the integrity of the scratchpad data. After issuing the command code the master begins reading. The first 2 bytes will be the target address. The next byte will be the ending offset/data status byte (E/S) followed by the scratchpad data beginning at the byte offset (T4: T0). The master may read data until the end of the scratchpad after which it will receive the inverted CRC generated by the DS1963S. If the master continues reading after the CRC all data will be logic 1's.

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

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

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

Write Scratchpad Command [0Fh]
HIDE = 0, Target Address range 0000h to 01FFh only
After issuing the write scratchpad command, the master must first provide the 2–byte target address, followed by the data to be written to the scratchpad. The data will be written to the scratchpad starting at the byte offset (T4:T0). The ending offset (E4: E0) will be the byte offset at which the master stops writing data. Only full data bytes are accepted. If the last data byte is incomplete its content will be ignored and the partial byte flag PF will be set.

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

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

```
Command:
  20 = Arbitrary DOW transaction
Data sent:
  \000\000\x0F\x00\x00\x11\x22\x33\x44\x55\x66\x77\x88\x99\xAA
Data received:
  C=84(d=0):00
```

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

```
Command:
  20 = Arbitrary DOW transaction
Data sent:
  \000\013\xAA
Data received:
  C=84(d=0):00,00,09,11,22,33,44,55,66,77,88,99,AA,1E
```

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

```
Command:
  20 = Arbitrary DOW transaction
Data sent:
  \000\000\x0F\x0A\x00\x12\x23\x34\x45\x56\x67\x78\x89\x9A\xAB
Data received:
  C=84(d=0):00
```

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

```
Command:
  20 = Arbitrary DOW transaction
Data sent:
  \000\013\xAA
Data received:
  C=84(d=0):00,02,09,12,23,34,45,56,67,78,89,9A,AB,62
```

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

APPENDIX C: DEMONSTRATION SOFTWARE AND SAMPLE CODE

SAMPLE CODE

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

- ❑ Windows compatible test/demonstration program and source.
  http://www.crystalfontz.com/product/633WinTest#docs
- ❑ Linux compatible command-line demonstration program with C source code. 8K.
  http://www.crystalfontz.com/product/linux_cli_examples#docs
- ❑ Supported by CrystalControl freeware.
  http://www.crystalfontz.com/product/CrystalControl2.html

In addition, see http://lcdproc.org/hardware.php3 for Linux LCD drivers. LCDproc is an open source project that supports many of the Crystalfontz displays.

ALGORITHMS TO CALCULATE THE CRC

Below are eight sample algorithms that will calculate the CRC of a CFA633 packet. Some of the algorithms were contributed by forum members and originally written for the CFA631 aand CFA635. The CRC used in the CFA633 is the same one that is used in IrDA, which came from PPP, which seems to be related to a CCITT (ref: Network Working Group Request for Comments: 1171) standard. At that point, the trail was getting a bit cold and diverged into several referenced articles and papers, dating back to 1983.

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

Algorithm 1: “C” Table Implementation

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

```
//This code is from the IRDA LAP documentation, which appears to
//have been copied from PPP:
//
// http://irda.affiniscape.com/associations/2494/files/Specifications/
IrLAP11_Plus_Errata.zip
//
//I doubt that there are any worries about the legality of this code,
//searching for the first line of the table below, it appears that
//the code is already included in the linux 2.6 kernel "Driver for
//ST5481 USB ISDN modem". This is an "industry standard" algorithm
//and I do not think there are ANY issues with it at all.
typedef unsigned char ubyte;
typedef unsigned short word;
word get_crc(ubyte *bufptr,word len)
  {
  //CRC lookup table to avoid bit-shifting loops.
  static const word crcLookupTable[256] =
    {0x00000,0x01189,0x02312,0x0329B,0x04624,0x057AD,0x06536,0x074BF,
     0x08C48,0x09DC1,0x0AF5A,0x0BED3,0x0CA6C,0x0DBE5,0x0E97E,0x0F8F7,
     0x01081,0x00108,0x03393,0x0221A,0x056A5,0x0472C,0x075B7,0x0643E,
     0x09CC9,0x08D40,0x0BFDB,0x0AE52,0x0DAED,0x0CB64,0x0F9FF,0x0E876,
     0x02102,0x0308B,0x00210,0x01399,0x06726,0x076AF,0x04434,0x055BD,
     0x0AD4A,0x0BCC3,0x08E58,0x09FD1,0x0EB6E,0x0FAE7,0x0C87C,0x0D9F5,
```

```
    0x03183,0x0200A,0x01291,0x00318,0x077A7,0x0662E,0x054B5,0x0453C,
    0x0BDCB,0x0AC42,0x09ED9,0x08F50,0x0FBEF,0x0EA66,0x0D8FD,0x0C974,
    0x04204,0x0538D,0x06116,0x0709F,0x00420,0x015A9,0x02732,0x036BB,
    0x0CE4C,0x0DFC5,0x0ED5E,0x0FCD7,0x08868,0x099E1,0x0AB7A,0x0BAF3,
    0x05285,0x0430C,0x07197,0x0601E,0x014A1,0x00528,0x037B3,0x0263A,
    0x0DECD,0x0CF44,0x0FDDF,0x0EC56,0x098E9,0x08960,0x0BBFB,0x0AA72,
    0x06306,0x0728F,0x04014,0x0519D,0x02522,0x034AB,0x00630,0x017B9,
    0x0EF4E,0x0FEC7,0x0CC5C,0x0DDD5,0x0A96A,0x0B8E3,0x08A78,0x09BF1,
    0x07387,0x0620E,0x05095,0x0411C,0x035A3,0x0242A,0x016B1,0x00738,
    0x0FFCF,0x0EE46,0x0DCDD,0x0CD54,0x0B9EB,0x0A862,0x09AF9,0x08B70,
    0x08408,0x09581,0x0A71A,0x0B693,0x0C22C,0x0D3A5,0x0E13E,0x0F0B7,
    0x00840,0x019C9,0x02B52,0x03ADB,0x04E64,0x05FED,0x06D76,0x07CFF,
    0x09489,0x08500,0x0B79B,0x0A612,0x0D2AD,0x0C324,0x0F1BF,0x0E036,
    0x018C1,0x00948,0x03BD3,0x02A5A,0x05EE5,0x04F6C,0x07DF7,0x06C7E,
    0x0A50A,0x0B483,0x08618,0x09791,0x0E32E,0x0F2A7,0x0C03C,0x0D1B5,
    0x02942,0x038CB,0x00A50,0x01BD9,0x06F66,0x07EEF,0x04C74,0x05DFD,
    0x0B58B,0x0A402,0x09699,0x08710,0x0F3AF,0x0E226,0x0D0BD,0x0C134,
    0x039C3,0x0284A,0x01AD1,0x00B58,0x07FE7,0x06E6E,0x05CF5,0x04D7C,
    0x0C60C,0x0D785,0x0E51E,0x0F497,0x08028,0x091A1,0x0A33A,0x0B2B3,
    0x04A44,0x05BCD,0x06956,0x078DF,0x00C60,0x01DE9,0x02F72,0x03EFB,
    0x0D68D,0x0C704,0x0F59F,0x0E416,0x090A9,0x08120,0x0B3BB,0x0A232,
    0x05AC5,0x04B4C,0x079D7,0x0685E,0x01CE1,0x00D68,0x03FF3,0x02E7A,
    0x0E70E,0x0F687,0x0C41C,0x0D595,0x0A12A,0x0B0A3,0x08238,0x093B1,
    0x06B46,0x07ACF,0x04854,0x059DD,0x02D62,0x03CEB,0x00E70,0x01FF9,
    0x0F78F,0x0E606,0x0D49D,0x0C514,0x0B1AB,0x0A022,0x092B9,0x08330,
    0x07BC7,0x06A4E,0x058D5,0x0495C,0x03DE3,0x02C6A,0x01EF1,0x00F78};

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

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

Algorithm 2: "C" Bit Shift Implementation

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

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

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

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

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

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

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

Algorithm 2B: "C" Improved Bit Shift Implementation

This is a simplified algorithm that implements the CRC.

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

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

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

Algorithm 3: "PIC Assembly" Bit Shift Implementation

This routine was graciously donated by one of our customers.

```
;*****************************************************************
; Crystalfontz CFA633 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 CFA633 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 633_WinTest:
'http://www.crystalfontz.com/products/633/633_WinTest.zip
'Full zip of the project is available in our forum:
'http://www.crystalfontz.com/forum/showthread.php?postid=9921#post9921

Private Type WORD
   Lo As Byte
   Hi As Byte
End Type

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

Dim crcLookupTable(256) As WORD

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

```
End Sub

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

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

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

Algorithm 5: "Java" Table Implementation

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

```
public class CRC16 extends Object
  {
  public static void main(String[] args)
```

```
    {
    byte[] data = new byte[2];
    // hw - fw
    data[0] = 0x01;
    data[1] = 0x00;
    System.out.println("hw -fw req");
    System.out.println(Integer.toHexString(compute(data)));

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

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

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

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

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

Algorithm 6: “Perl” Table Implementation

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

```
#!/usr/bin/perl

use strict;

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

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

```
#   data = 5

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

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

my $valid_crc = '5584' ;

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

my $crc = 0xFFFF ;

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

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

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

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

Algorithm 7: For PIC18F8722 or PIC18F2685

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

```
; CRC Algorithm for CrystalFontz CFA635 display (DB535)
; This code written for PIC18F8722 or PIC18F2685
;
; Your main focus here should be the ComputeCRC2 and
; CRC16_ routines
;
;*****************************************************************
ComputeCRC2:
        movlb   RAM8
        movwf   dsplyLPCNT      ;w has the byte count
nxt1_dsply:
        movf    POSTINC1,w
        call    CRC16_
        decfsz  dsplyLPCNT
        goto    nxt1_dsply
        movlw   .0              ; shift accumulator 16 more bits
        call    CRC16_
        movlw   .0
        call    CRC16_
        comf    dsplyCRC,F      ; invert result
        comf    dsplyCRC+1,F
        return
;*****************************************************************
CRC16_ movwf:
        dsplyCRCData            ; w has byte to crc
        movlw   .8
        movwf   dsplyCRCCount
```

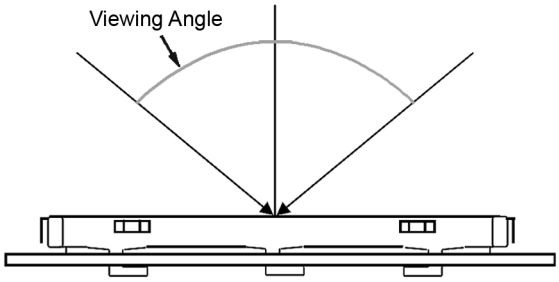
```
_cloop:
        bcf     STATUS,C        ; clear carry for CRC register shift
        rrcf    dsplyCRCData,f  ; perform shift of data into CRC
                                ;register
        rrcf    dsplyCRC,F
        rrcf    dsplyCRC+1,F
        btfss   STATUS,C        ; skip jump if carry
   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

```
BuildMsgDSPLY:
        movlw   0xF3
        movwf   dsplyCRC        ; seed hi for CRC calculation
        movlw   0x21
        movwf   dsplyCRC+1      ; seed lo for CRC calculation
        LFSR    FSR1,TXBUF2     ; point at transmit buffer
        movlw   0x1F            ; command to send data to LCD
        movwf   TXBUF2          ; insert command byte from us to
                                ; CFA635
        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 D: QUALITY ASSURANCE STANDARDS

INSPECTION CONDITIONS

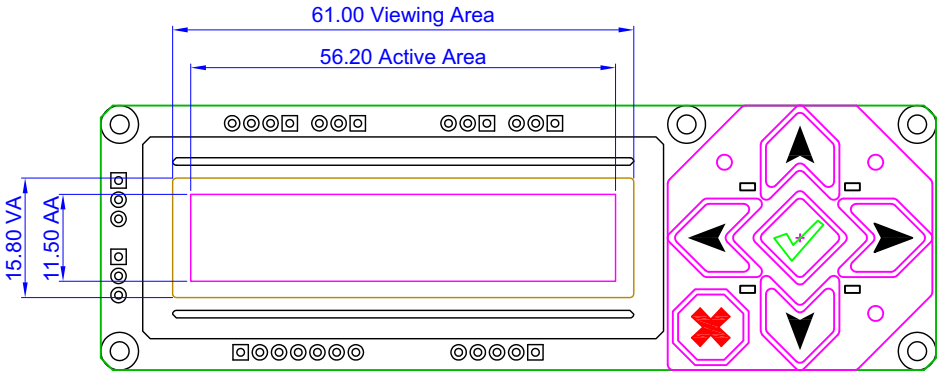
- Environment
  - Temperature: 25±5°C
  - Humidity: 30~85% RH noncondensing
- For visual inspection of active display area
  - Source lighting: two 20 Watt or one 40 Watt fluorescent light
  - Display adjusted for best contrast
  - Viewing distance: 30±5 cm (about 12 inches)
  - Viewable angle: inspect at 45° angle of vertical line right and left, top and bottom0



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

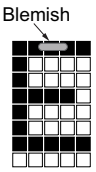
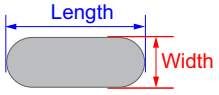
DEFECT TYPE	AQL*
Major	≤0.65%
Minor	≤1.00%
*Acceptable Quality Level: maximum allowable error rate or variation from standard	

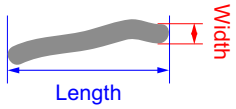
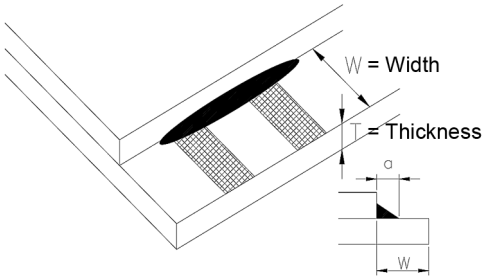
DEFECTS CLASSIFICATION

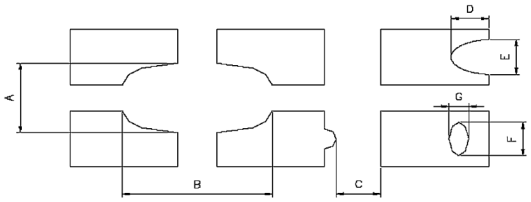
Defects are defined as:

- 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

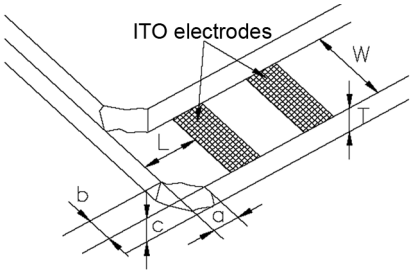
ACCEPTANCE STANDARDS

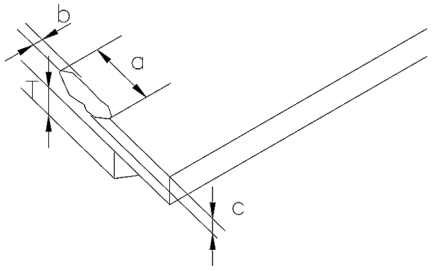
<table>
<tr><th>#</th><th>DEFECT TYPE</th><th colspan="3">CRITERIA</th><th>MAJOR / MINOR</th></tr>
<tr><td>1</td><td>Electrical defects</td><td colspan="3">1. No display, display malfunctions, or shorted segments.
2. Current consumption exceeds specifications.</td><td>Major</td></tr>
<tr><td>2</td><td>Viewing area defect</td><td colspan="3">Viewing area does not meet specifications. (See Inspection Conditions (Pg. 83).</td><td>Major</td></tr>
<tr><td>3</td><td>Contrast adjustment defect</td><td colspan="3">Contrast adjustment fails or malfunctions.</td><td>Major</td></tr>
<tr><td rowspan="3">4</td><td rowspan="3">Blemishes or foreign matter on display segments</td><td rowspan="3"> 
</td><td>Defect Size (mm)</td><td>Acceptable Qty</td><td rowspan="3">Minor</td></tr>
<tr><td>≤0.3</td><td>3</td></tr>
<tr><td colspan="2">≤2 defects within 10 mm of each other</td></tr>
<tr><td rowspan="5">5</td><td rowspan="5">Other blemishes or foreign matter outside of display segments</td><td rowspan="5">Defect size = (A + B)/2
 
</td><td>Defect Size (mm)</td><td>Acceptable Qty</td><td rowspan="5">Minor</td></tr>
<tr><td><0.15</td><td>Ignore</td></tr>
<tr><td>0.15 to 0.20</td><td>3</td></tr>
<tr><td>0.20 to 0.25</td><td>2</td></tr>
<tr><td>0.25 to 0.30</td><td>1</td></tr>
</table>

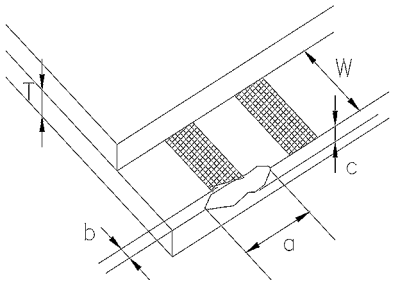
<table>
<tr><th>#</th><th>DEFECT TYPE</th><th colspan="3">CRITERIA</th><th>MAJOR / MINOR</th></tr>
<tr><td rowspan="6">6</td><td rowspan="6">Dark lines or scratches in display area
 
</td><td>Defect Width (mm)</td><td>Defect Length (mm)</td><td>Acceptable Qty</td><td rowspan="6">Minor</td></tr>
<tr><td>≤0.03</td><td>≤3.0</td><td>3</td></tr>
<tr><td>0.03 to 0.05</td><td>≤2.0</td><td>2</td></tr>
<tr><td>0.05 to 0.08</td><td>≤2.0</td><td>1</td></tr>
<tr><td>0.08 to 0.10</td><td>≤3.0</td><td>0</td></tr>
<tr><td>≥0.10</td><td>>3.0</td><td>0</td></tr>
<tr><td rowspan="5">7</td><td rowspan="5">Bubbles between polarizer film and glass</td><td colspan="2">Defect Size (mm)</td><td>Acceptable Qty</td><td rowspan="5">Minor</td></tr>
<tr><td colspan="2"><0.20</td><td>Ignore</td></tr>
<tr><td colspan="2">0.20 to 0.40</td><td>3</td></tr>
<tr><td colspan="2">0.40 to 0.60</td><td>2</td></tr>
<tr><td colspan="2">≥0.60</td><td>0</td></tr>
<tr><td>8</td><td>Glass rest defect</td><td colspan="3"> 

a≤1/4W</td><td>Minor</td></tr>
<tr><td>9</td><td>Display pattern defect</td><td colspan="3"> 

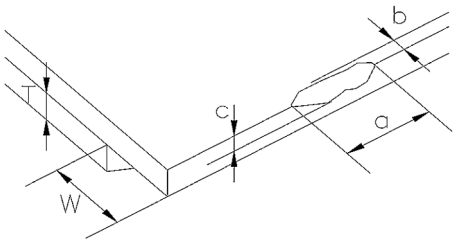
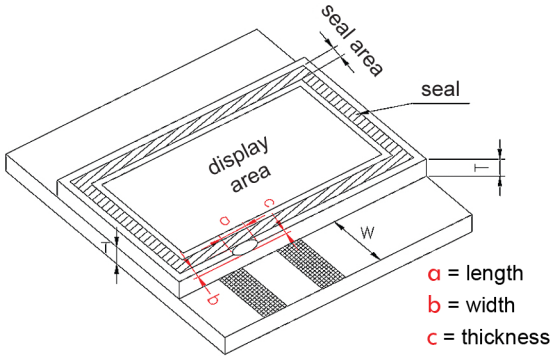
<table>
<tr><th>Dot Size (mm)</th><th>Acceptable Qty</th></tr>
<tr><td>((A+B)/2)≤0.2</td><td rowspan="4">≤3 total defects
≤2 pinholes per digit</td></tr>
<tr><td>C>0</td></tr>
<tr><td>((D+E)/2)≤0.25</td></tr>
<tr><td>((F+G)/2)≤0.25</td></tr>
</table></td><td>Minor</td></tr>
</table>

<table>
<tr><th>#</th><th>DEFECT TYPE</th><th>CRITERIA</th><th>MAJOR / MINOR</th></tr>
<tr><td>10</td><td>Chip in corner</td><td> 

<table>
<tr><th>a</th><th>b</th><th>c</th><th>Acceptable Qty</th></tr>
<tr><td><4 mm</td><td>≤W</td><td>c≤T</td><td>3</td></tr>
</table></td><td>Minor</td></tr>
<tr><td>11</td><td>Chip on “non-contact” edge of LCD</td><td> 

<table>
<tr><th>a</th><th>b</th><th>c</th></tr>
<tr><td>≤3 mm</td><td>≤1 mm</td><td>≤T</td></tr>
<tr><td>≤4 mm</td><td>≤1.5 mm</td><td>≤T</td></tr>
</table></td><td>Minor</td></tr>
<tr><td>12</td><td>Chip on “contact” edge of LCD, on the active side</td><td> 

<table>
<tr><th>a</th><th>b</th><th>c</th><th>Acceptable Qty</th></tr>
<tr><td>≤2 mm</td><td>≤W/4</td><td>≤T</td><td>Ignore</td></tr>
<tr><td>≤3 mm</td><td>≤W/4</td><td>≤T</td><td>3</td></tr>
</table></td><td>Minor</td></tr>
</table>

<table>
<tr><th>#</th><th>DEFECT TYPE</th><th>CRITERIA</th><th>MAJOR / MINOR</th></tr>
<tr><td rowspan="4">13</td><td rowspan="4">Chip on “contact” edge of LCD, on the inactive side</td><td> 
</td><td rowspan="4">Minor</td></tr>
<tr><td><table>
<tr><td>a</td><td>b</td><td>c</td><td>Acceptable Qty</td></tr>
<tr><td>≤3 mm</td><td>≤1 mm</td><td>≤T</td><td>Ignore</td></tr>
<tr><td>≤4 mm</td><td>≤1.5 mm</td><td>≤T</td><td>3</td></tr>
</table></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td rowspan="4">14</td><td rowspan="4">Chip in seal area</td><td> 
</td><td rowspan="4">Minor</td></tr>
<tr><td><table>
<tr><td>a</td><td>b</td><td>c</td><td>Acceptable Qty</td></tr>
<tr><td><3 mm</td><td>≤1.5 mm</td><td>≤1/2 T</td><td>3</td></tr>
</table></td></tr>
<tr><td>Unacceptable if c>50% of glass thickness or if the seal area is damaged.</td><td>Major</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>15</td><td>Backlight defects</td><td>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. 85).
*Minor if display functions correctly. Major if the display fails.</td><td>Minor</td></tr>
<tr><td>16</td><td>COB defects</td><td>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.</td><td>Minor</td></tr>
<tr><td>17</td><td>PCB defects</td><td>1. Oxidation or contamination on connectors.*
2. Wrong parts, missing parts, or parts not in specification.*
3. Jumpers set incorrectly.
4. Solder (if any) on bezel, LED pad, zebra pad, or screw hole pad is not smooth.
*Minor if display functions correctly. Major if the display fails.</td><td>Minor</td></tr>
</table>

#	DEFECT TYPE	CRITERIA	MAJOR / MINOR
18	Soldering defects	1. Unmelted solder paste. 2. Cold solder joints, missing solder connections, or oxidation.* 3. Solder bridges causing short circuits.* 4. Residue or solder balls. 5. Solder flux is black or brown. **Minor if display functions correctly. Major if the display fails.*	Minor