



Crystalfontz America, Incorporated

INTELLIGENT LCD MODULE SPECIFICATIONS



Data Sheet Release 2012/10/19

for

the CFA631 Family:

[CFA631-TMF-KU](#)

[CFA631-RMF-KU](#)

Hardware Version: 2h4

Firmware Version: u3v1

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



Note on Firmware and Hardware Revisions

For information about firmware and hardware revisions, see the Part Change Notifications (PCNs) under "News" in our website's navigation bar. To see the most recent PCNs for the CFA631 family at the time of this Data Sheet release date, see PCN [#10428](#) and [#10427](#).

Data Sheet Revision History

Data Sheet Release: 2012/10/19
Complete Data Sheet rewrite.

2008/10/06	Data Sheet version: v2.0a (not changed) Changes since last revision: Note added to correct specification of GPIO pull-up/pull-down mode resistance values from "approximately 5Ω" to "approximately 5kΩ".
2005/12/20	Data Sheet version: v2.0a Changes since last revision (v2.0): <ul style="list-style-type: none">● Corrected "Character Size" and added "Character Pitch".● Corrected specification for supply voltage maximum.● Corrected return "type" for command 26: Set Fan Tachometer Glitch Filter (SCAB required).● Corrected return "type" for command 27: Query Fan Power & Fail-Safe Mask (SCAB required).● Corrected "type" for command 33: Set Baud Rate.● Corrected length returned by reply for command 35: Read GPIO Pin Levels and Configuration State.● Formatting, content organization, and minor rewording to improve readability.
2005/08/01	Start Public Version Tracking. Data Sheet version: v2.0 Changes since last released version (v1.9): <ul style="list-style-type: none">● Added Revision History (this page).● Added GPIO Current Limits.● Added APPENDIX C: CALCULATING THE CRC.● Added note on operating system delays.● Added note on length of command 30 reply.● Added documentation for commands requiring the Crystalfontz SCAB accessory.● Corrected length returned by command 30.



About Variations

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

The Fine Print

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

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

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

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

Copyright © 2012 by Crystalfontz America, Inc., 12412 East Saltese Avenue, Spokane Valley, WA 99216-0357 U.S.A.



CONTENTS

INTRODUCTION	7
Comparison to Previous Revisions of the CFA631 Family	7
Main Features	7
Explanation of Part Number Codes in This Data Sheet	9
ACCESSORIES	10
Customize Your Module	10
Kits (Modules, Overlays for Built-In Bracket, SCAB, and Cables)	11
MECHANICAL CHARACTERISTICS	12
Physical Characteristics	12
Module Outline Drawings	13
Jumpers That Can Be Modified	15
ELECTRICAL SPECIFICATIONS	16
System Block Diagram	16
Lcd Duty and Bias	17
Absolute Maximum Ratings	18
DC Characteristics	18
Current Consumption	19
GPIO Current Limits	19
ESD (Electro-Static Discharge) Specifications	19
Backlight Fan and Criteria	20
OPTICAL SPECIFICATIONS	20
Viewing Angle	20
Definition of 6 O'clock and 12:00 O'clock Viewing Angles	20
Optical Characteristics	21
Test Conditions and Definitions for Optical Characteristics	21
LED BACKLIGHT INFORMATION	24
CONNECTION INFORMATION	24
Buy Cables Separately	24
USB Connection to Host	26
H1 Connector Pin Assignments - Includes Five GPIOs	27
ATX Power Supply	28
ATX Power and Control Connections	28
ATX Connection With Optional SCAB Using WR-PWR-Y14 ATX Cable	29
ATX Connection Without SCAB Using WR-PWR-Y25 ATX Cable	31
How to Connect the Optional SCAB	31
HOST COMMUNICATIONS	33
Packet Structure	33
About Handshaking	34
Report Codes	34
0x80: Key Activity	35
0x81: Fan Speed Report (SCAB Required)	35
0x82: Temperature Sensor Report (SCAB Required)	36
Command Codes	37
0 (0x00): Ping Command	37



CONTENTS, CONTINUED

1 (0x01): Get Hardware & Firmware Version	37
2 (0x02): Write User Flash Area	37
3 (0x03): Read User Flash Area	38
4 (0x04): Store Current State As Boot State	38
5 (0x05): Reboot CFA631, Reset Host, or Power Off Host Using ATX	39
6 (0x06): Clear LCD Screen	42
7 (0x07): Set LCD Contents, Line 1 (CFA633 Compatible)	42
8 (0x08): Set LCD Contents, Line 2 (CFA633 Compatible)	42
9 (0x09): Set LCD Special Character Data	43
10 (0x0A): Read 8 Bytes of LCD Memory	43
11 (0x0B): Set LCD Cursor Position	43
12 (0x0C): Set LCD Cursor Style	44
13 (0x0D): Set LCD Contrast	44
14 (0x0E): Set LCD & Keypad Backlights	44
15 (0x0F): (Not Accessible)	45
16 (0x10): Set Up Fan Reporting (SCAB Required)	45
17 (0x11): Set Fan Power (SCAB Required)	45
18 (0x12): Read WR-DOW-Y17 Temperature Sensors (SCAB Required)	46
19 (0x13): Set Up WR-DOW-Y17 Temperature Reporting (SCAB Required)	46
20 (0x14): Arbitrary DOW Transaction (SCAB Required)	47
21 (0x15): Set Up Live Fan or Temperature Display (SCAB Required)	48
22 (0x16): Send Command Directly to the LCD Controller	49
23 (0x17): Configure Key Reporting	50
24 (0x18): Read Keypad, Polled Mode	50
25 (0x19): Set Fan Power Fail-Safe (SCAB Required)	51
26 (0x1A): Set Fan Tachometer Glitch Delay (SCAB Required)	51
27 (0x1B): Query Fan Power & Fail-Safe Mask (SCAB Required)	52
28 (0x1C): Set ATX Power Switch Functionality	53
29 (0x1D): Enable/Disable and Reset the Watchdog	56
30: (0x1E) Read Reporting & Status	56
31 (0x1F): Send Data to LCD	57
32: Key Legends	57
33 (0x21): Set Baud Rate	58
34 (0x22): Set or Set and Configure GPIO Pins (SCAB Required)	58
35 (0x23): Read GPIO Pin Levels and Configuration State (SCAB Required)	60
CHARACTER GENERATOR ROM (CGROM)	62
LCD MODULE RELIABILITY AND LONGEVITY	63
Module Reliability	63
Module Longevity (EOL / Replacement Policy)	63
CARE AND HANDLING PRECAUTIONS	64
APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTERS	66
APPENDIX B: CONNECTING A DS1963S SHA IBUTTON	68
APPENDIX C: SAMPLE CODE (Demonstration Software and Sample Code)	71
Drivers	71



Demonstration Software ----- 71

Algorithms to Calculate the CRC ----- 71

 Algorithm 1: "C" Table Implementation ----- 71

 Algorithm 2: "C" Bit Shift Implementation ----- 72

 Algorithm 2B: "C" Improved Bit Shift Implementation ----- 74

 Algorithm 3: "PIC Assembly" Bit Shift Implementation ----- 74

 Algorithm 4: "Visual Basic" Table Implementation ----- 76

 Algorithm 5: "Java" Table Implementation ----- 77

 Algorithm 6: "Perl" Table Implementation ----- 79

 Algorithm 7: For PIC18F8722 or PIC18F2685 ----- 80

APPENDIX D: QUALITY ASSURANCE STANDARDS----- 83

LIST OF FIGURES

Figure 1. CFA631-xxx-KU with optional SCAB and cable WR-EXT-Y19----- 10

Figure 2. Black Aluminum Overlay, 1 of 4 Overlay Choices----- 11

Figure 3. CFA631-xxx-KU Module Outline Drawings (2 pages)----- 13

Figure 4. Location of Jumpers That Can Be Modified----- 15

Figure 5. System Block Diagram ----- 16

Figure 6. Definition of 6:00 O'clock and 12:00 O'clock Viewing Angles----- 20

Figure 7. Definition of Optimal Contrast Setting (Negative Image)----- 22

Figure 8. Definition of Response Time (Tr, Tf) (Negative Image)----- 22

Figure 9. Definition of 6:00 O'clock and 12:00 O'clock Viewing Angles----- 23

Figure 10. Definition of Horizontal and Vertical Viewing Angles (CR>2)----- 23

Figure 11. USB Connector Pins Labeled ----- 26

Figure 12. Location of GPIO Pins on H1 Connector----- 27

Figure 13. ATX Connection with Optional SCAB using WR-PWR-Y14 ATX Cable----- 30

Figure 14. ATX Power Supply and Control Connections using WR-PWR-Y25 ATX Cable ----- 31

Figure 15. CFA631-xxx-KU Connected to Optional SCAB using WR-EXT-Y19 cable ----- 32

Figure 16. Character Generated ROM ----- 62

Appendix A Figure 1. Test Circuit Schematic-----66

Appendix A Figure 1. Connecting DS1963S SHA iButton using DS9094 iButton Clip -----68



INTRODUCTION

The CFA631 family of modules has two color choices: *CFA631-RMF-KU* and *CFA631-TMF-KU*. Both have USB interface. When the information in this Data Sheet applies to both color choices, the term “CFA631-xxx-KU” or the shorter term “CFA631” is used.

COMPARISON TO PREVIOUS REVISIONS OF THE CFA631 FAMILY

For information about firmware and hardware revisions, see the Part Change Notifications (PCNs) under “News” in our website’s navigation bar. To see the most recent PCNs for the CFA631 family at the time of this Data Sheet release, see PCN [#10428](#) and [#10427](#).

MAIN FEATURES

- Large easy-to-read LCD in a compact size can show 20 characters x 2 lines.
- LCD with keypad in a stainless steel bracket with an attractive front plate.
- 3.5-inch floppy drive form factor easily fits in a 1U rack mount case (25.0 millimeters typical overall height).
- Choice of two colors. Edge-lit display is backlit with 4 LEDs, 2 LEDs per side.
- Modules have a 12 o’clock viewing direction. See [Test Conditions and Definitions for Optical Characteristics \(Pg. 21\)](#).
- USB interface (factory default 115200 baud equivalent throughput).
- Integrated LED backlit 4-button translucent silicone keypad allows assignment of keys to be shown easily on the LCD. Fully decoded keypad: any key combination is valid and unique. See command [32: Key Legends \(Pg. 57\)](#).
- Choice of two colors.
 - *CFA631-RMF-KU*: Red LED backlight with negative STN blue transmissive mode LCD. Display is backlit with an array of 22 LEDs (11 on top and 11 on bottom). Displays red characters on dark background. The display can be read in normal office lighting and in dark areas. Not recommended for use in sunlight; may be washed out.
 - *CFA631-TMF-KU*: Edge-lit white LED backlight with negative STN blue transmissive mode LCD. Display is backlit with a 4 LEDs on one edge. Displays light (near-white) characters on blue background. The display can be read in normal office lighting and in dark areas. Not recommended for use in sunlight; may be washed out.
- Backlight is fully voltage regulated over the power supply range. Adjustments to the backlight brightness can be made, although it is not necessary in most situations.
- The CFA631-xxx-KU has a RockWorks RW1067 controller.
- Robust packet based communications protocol with 16-bit CRC.
- ATX power supply control functionality allows the keypad buttons to replace the Power and Reset switches on your system, simplifying front panel design. For using ATX, the optional [SCAB+WR-PWR-Y14](#) ATX power cable or [WR-PWR-Y25](#) ATX power cable is required.
- Nonvolatile memory capability (EEPROM):
 - Customize the “power-on” display settings.
 - 16-byte “scratch” register for storing IP address, netmask, system serial number . . .
- Hardware watchdog can reset host system on host software failure.
- The CFA631-xxx-KU may be used with our optional [SCAB](#) (System Cooling Accessory Board). The combination of the CFA631-xxx-KU with the SCAB (written as “CFA631+SCAB” in this Data Sheet) allows:
 - Four functional fan connectors with RPM monitoring and variable PWM (Pulse Width Modulation) fan power control. Fail-safe fan power settings allows host to safely control four fans based on temperature. Commonly available PC cooling fans may be used. (Fans are not sold by Crystalfontz.) See Command [25 \(0x19\): Set Fan Power Fail-Safe \(SCAB Required\) \(Pg. 51\)](#).
 - Add up to 32 Crystalfontz [WR-DOW-Y17](#) cables with DOW (Dallas 1-Wire) DS18B20 temperature sensors. Sensors monitor temperatures at up to 0.5°C absolute accuracy.



- Instead of ATX power supply control functionality directly from the CFA631-xxx-KU using the [WR-PWR-Y25](#) ATX power cable, you can have ATX power supply control functionality from the optional [SCAB](#) using the [WR-PWR-Y14](#) ATX power cable.
- To download the most current Certificate of Compliance for ISO, product specifications, RoHS, and REACH, go to the module's Doc/Files tab on the module's website page.


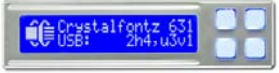


EXPLANATION OF PART NUMBER CODES IN THIS DATA SHEET

<u>CFA</u>	<u>631</u>	-	<u>X</u>	<u>X</u>	<u>X</u>	-	<u>K</u>	<u>U</u>	<u>X</u>
①	②		③	④	⑤		⑥	⑦	⑧

①	Brand	CrystalFontz America, Inc.
②	Model Identifier	631
③	Backlight Type & Color	R – LED, red T – LED, white
④	Fluid Type, Image (positive or negative), & LCD Glass Color	M – STN, negative blue
⑤	Polarizer Film Type, Operating Temperature Range, & View Angle (O 'Clock)	F – Transmissive, NT ¹ , 12:00
¹ Normal Temperature Operating Range is 0°C minimum to +50°C maximum.		
⑥	Special Code 1	K – Manufacturer's code
⑦	Special Code 2	U – USB interface
⑧	Configuration Codes	=1 or more characters ²

²When you order a CFA631-xxx-KU through our website, you may be offered a choice of configurations (including accessories) to add to your order through our "Customize and Add to Cart" feature. Additional choices are available through our [Kit Configurator](#).

Part Number	 CFA631-RMF-KU	 CFA631-TMF-KU
Fluid	STN	
LCD Glass Color	blue	
Image	negative	
Polarizer Film	transmissive	
LEDs	Backlight: red Keypad: red	Backlight: white Keypad: blue
Note	Negative Image = Not recommended for use in sunlight; may be washed out.	



ACCESSORIES



Figure 1. CFA631-xxx-KU with optional SCAB and cable WR-EXT-Y19

CUSTOMIZE YOUR MODULE

Go to the web page for [CFA631-RMF-KU](#) or [CFA631-TMF-KU](#). After you click on the “*Customize and Add to Cart*” button, you will see a list of questions. You can order the module customized to enable ATX power control functionality from the module, add a cable, or order the accessory [SCAB](#) (System Cooling Accessory Board) to use with up to 32 [WR-DOW-Y17](#) temperature sensors and four fans.

To add an overlay to the built-in bracket’s front plate, order a CFA631-xxx-KU through our Kit Configurator instead of using the Customize tool. (See next Data Sheet section.)



KITS (MODULES, OVERLAYS FOR BUILT-IN BRACKET, SCAB, AND CABLES)

See the [Kit Configurator](#) on our website. Below is an explanation of kit part numbers. You can also buy accessories individually. See a detailed description of useful cables in section [Buy Cables Separately \(Pg. 24\)](#) or see all of our cables listed at www.crystalfontz.com/cart/pricing.php?cat=2.

<u>DB</u>	<u>631</u>	<u>xx</u>	-	<u>xxx</u>	-	<u>K</u>	<u>U</u>	<u>#</u>
①		②		③				④

①	[type] DB – Built-in 5.25-inch half-height drive bay mounting bracket.
②	[overlay] An overlay for the front of bracket with a display window of clear thick hard-coated polycarbonate material. Choice of four overlays: AK – Black Aluminum AL – Silver Aluminum BG – Beige Plastic BK – Black Plastic
③	[variant] Choice of two colors (variants): RMF – red characters on dark background TMF – light (near-white) characters on blue background
④	[configuration code: additional parts in kit] # – Kit may include one or more cables, the optional SCAB, and SCAB accessories.



Figure 2. Black Aluminum Overlay, 1 of 4 Overlay Choices



MECHANICAL CHARACTERISTICS

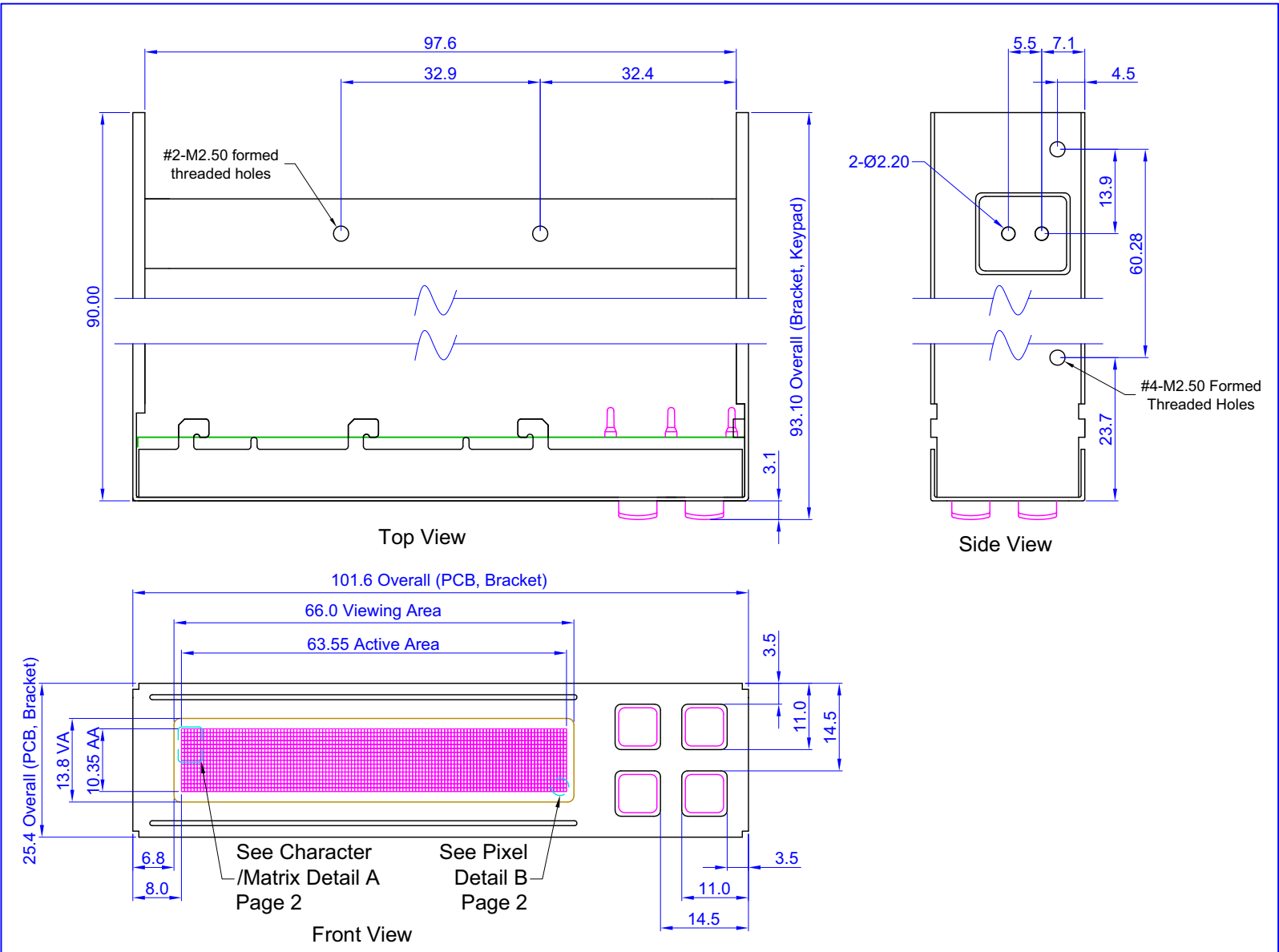
PHYSICAL CHARACTERISTICS

ITEM	SPECIFICATION
Module Overall Dimensions	
Width and Height	101.6 (W) x 25.4 (H) mm
Module Depth (Includes Keypad)	93.1 mm maximum
Viewing Area	66.0 (W) x 13.8 (H) mm
Active Area	63.55 (W) x 10.35 (H) mm
Character Size (5 x 7)	2.60 (W) x 4.50 (H) mm
Character Pitch (6 x 8)	3.18 (W) x 5.20 (H) mm
Pixel Pitch	0.53 (W) x 0.65 (H) mm
Pixel Size	0.48 (W) x 0.60 (H) mm
Keystroke Travel (approximate)	~2.4 mm
Weight	80 grams (typical)




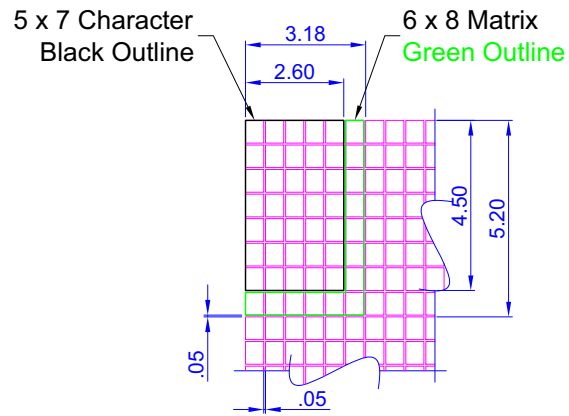
MODULE OUTLINE DRAWINGS

Figure 3. CFA631-xxx-KU Module Outline Drawings (2 pages)

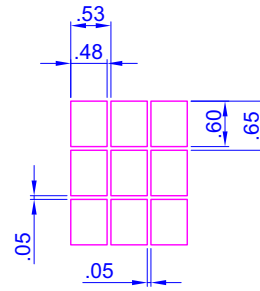


Notes: Tolerance is ±0.5 mm unless specified.

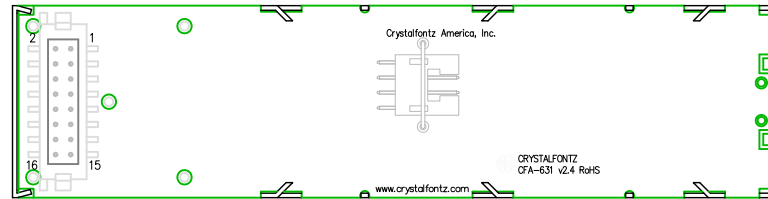
 Crystalfontz America, Inc. copyright © 2012 by www.crystalfontz.com/products/	Part No.(s): CFA631-xxx-KU Family	Scale: Not to scale	Drawing Number: CFA631_Family_master	Hardware Rev.: 2h4
		Units: Millimeters	Date: 2012/10/18	Sheet: 1 of 2



Character Detail A



Pixel Detail B



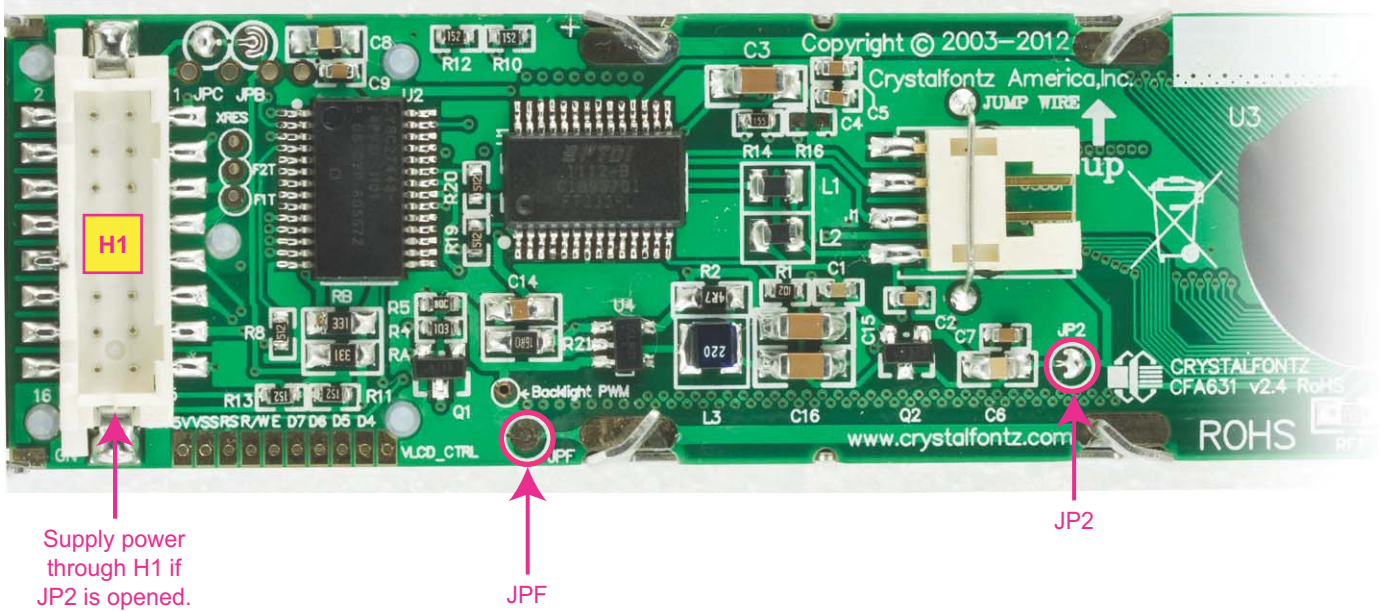
Back View





JUMPERS THAT CAN BE MODIFIED

The CFA631-xxx-KU has four jumpers. Only JPF and JP2 may be changed. To open a jumper, remove the solder. Solder wick works well for this. To close a jumper, melt solder across the gap.



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

JPF	open	Standard configuration: shipped with JPF open. Frame ground is isolated from logic/USB ground.
	closed	You can close JPF to connect frame ground to logic/USB ground.
JP2	closed	Standard configuration: shipped with JP2 closed unless otherwise requested. Power is supplied through USB connector.
JP2	open	Power is not supplied through USB connector. Power must be supplied through pins 15 (Ground) and 16 (+5v) on H1. (See Figure 12. on Pg. 27.) Open JP2 for ATX or when power is supplied over H1. We can do this for you when you order "Make Module ATX". Click on the "Customize and Add to Cart" button on the module's website page.

Figure 4. Location of Jumpers That Can Be Modified



ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

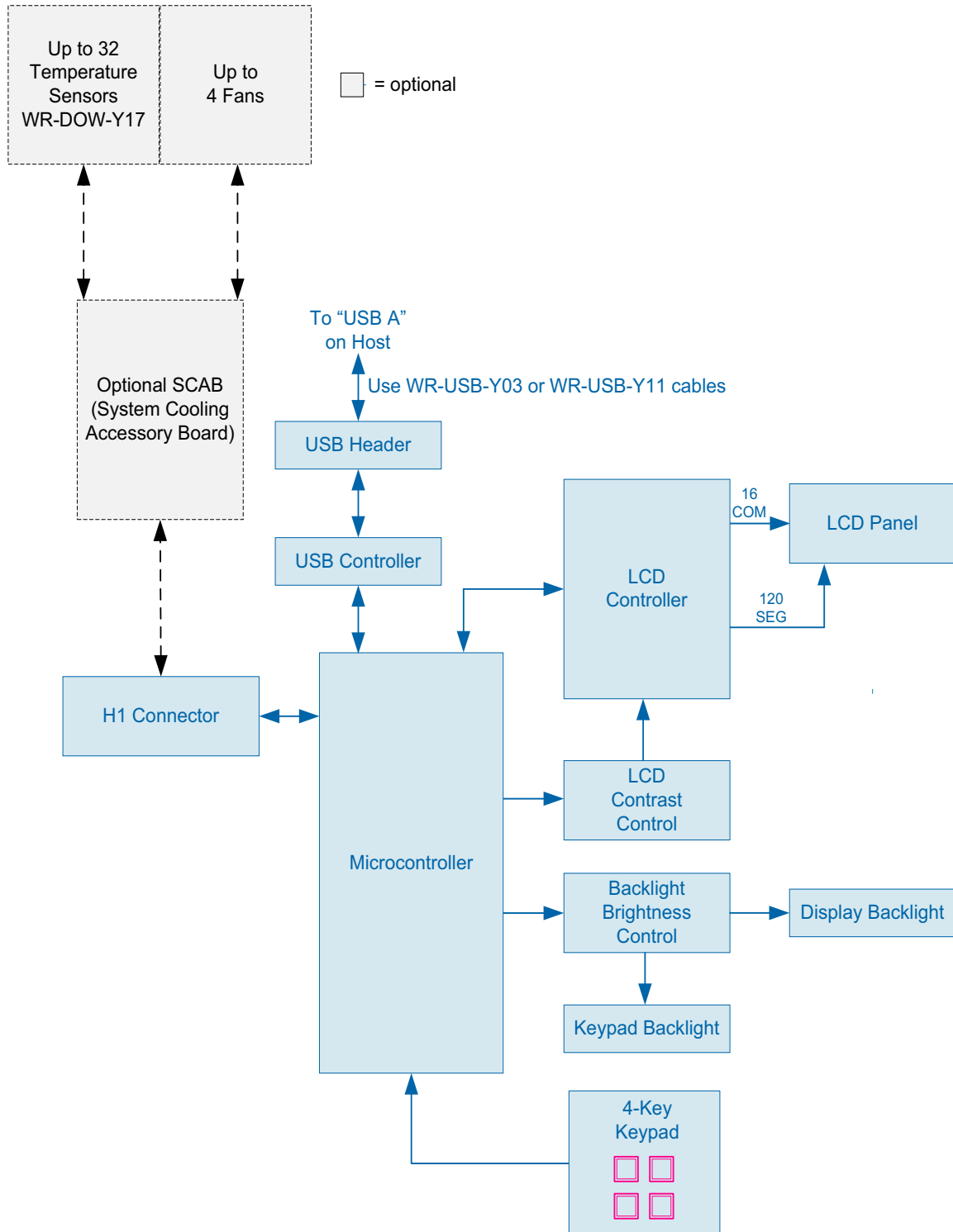


Figure 5. System Block Diagram



LCD DUTY AND BIAS

DRIVING METHOD	SPECIFICATION
Duty ¹	1/32
Bias ²	6.7

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

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



ABSOLUTE MAXIMUM RATINGS

ABSOLUTE MAXIMUM RATINGS	SYMBOL	MINIMUM	MAXIMUM
Operating Temperature	T _{OP}	0°C	+50°C
Storage Temperature	T _{ST}	-10°C	+60°C
Humidity Range (Noncondensing)	RH	10%	90%
Supply Voltage for Logic	V _{DD}	0v	+5.25v
<p>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 (Pg. 18) is not implied. Changes in temperature can result in changes in contrast.</p>			

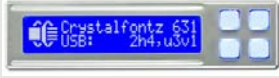
DC CHARACTERISTICS


	DC CHARACTERISTICS	TEST CONDITIONS	SYMBOL	MINIMUM	TYPICAL	MAXIMUM
CONTROLLER AND BOARD	Supply Voltage for Logic	T _{OP} = -0°C to +50°C	V _{DD} - GND	+4.75v	+5.0v	+5.25v ¹
	Input High Voltage	V _{DD} = +5v	V _{IH}	V _{DD} -1.0v		V _{DD}
	Input Low Voltage		V _{IL}	0v (GND)		+0.60v
	Output High Voltage		V _{OH}	+0.V _{DD}		
	Output Low Voltage		V _{OL}	0v (GND)		+0.1V _{DD}
<p>¹Do not exceed +5.25v maximum.</p>						



CURRENT CONSUMPTION

Variables that affect current consumption include the choice of color, brightness of backlights, power supply voltage, and whether or not a [SCAB](#) (System Cooling Accessory Board) is attached to the module.

 CFA631-TMF-KU TYPICAL CURRENT CONSUMPTION (V_{DD} = +5.0v)	BACKLIGHT ONLY	INCLUDING LOGIC
Logic + USB controller, backlight off	30 mA	
Logic + USB controller, backlight at 100%	60 mA	90 mA

 CFA631-RMF-KU TYPICAL CURRENT CONSUMPTION (V_{DD} = +5.0v)	BACKLIGHT ONLY	INCLUDING LOGIC
Logic + USB controller, backlight off	30 mA	
Logic + USB controller, backlight at 100%	150mA	180 mA

GPIO CURRENT LIMITS

TYPICAL GPIO CURRENT LIMITS	
Sink	25 mA
Source	10 mA

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS

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



BACKLIGHT FAN AND CRITERIA

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 Optional SCAB is required to add fans. ² PWM is <i>Pulse Width Modulation</i> . 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 <i>Pulses Per Revolution</i> , can also written as p/r.	

OPTICAL SPECIFICATIONS

VIEWING ANGLE

Viewing Angle	12 o'clock
---------------	------------

DEFINITION OF 6 O'CLOCK AND 12:00 O'CLOCK VIEWING ANGLES

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

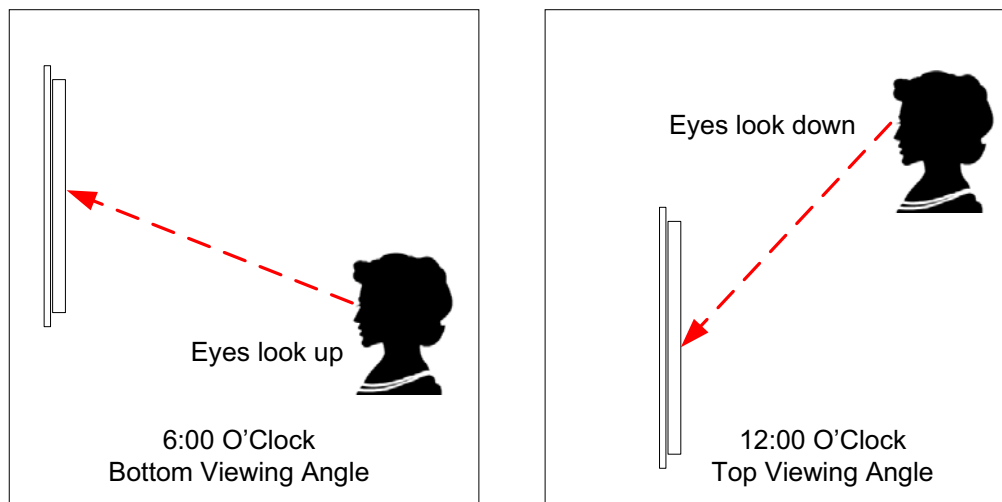


Figure 6. Definition of 6:00 O'clock and 12:00 O'clock Viewing Angles



OPTICAL CHARACTERISTICS

ITEM	SYMBOL	MINIMUM	TYPICAL	MAXIMUM
<i>Test Condition: Ta = 25°C</i>				
Viewing Angle (12 o'clock) CR _{≥2}	Deg θ = 0°		30	
	Deg θ = 90°		30	
	Deg θ = 180°		40	
	Deg θ = 270°		30	
Contrast Ratio ¹	CR		≥5	
LCD Response Time ^{2,3}	T rise	70 ms	100 ms	150 ms
	T fall	70 ms	100 ms	150 ms
¹ Contrast Ratio = (brightness with pixels light)/(brightness with pixels dark). ² Response Time: The amount of time it takes a liquid crystal cell to go from active to inactive or back again. ³ For reference only.				

TEST CONDITIONS AND DEFINITIONS FOR OPTICAL CHARACTERISTICS

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

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



Definition of Optimal Contrast Setting

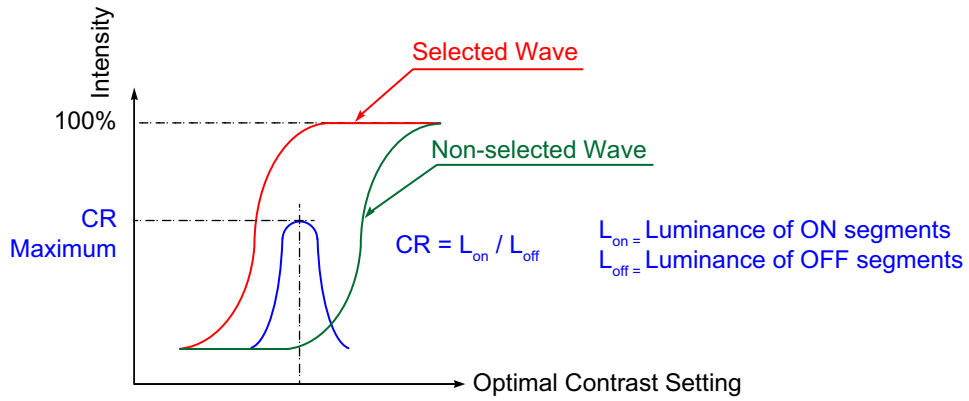


Figure 7. Definition of Optimal Contrast Setting (Negative Image)

Definition of Response Time (Tr, Tf)

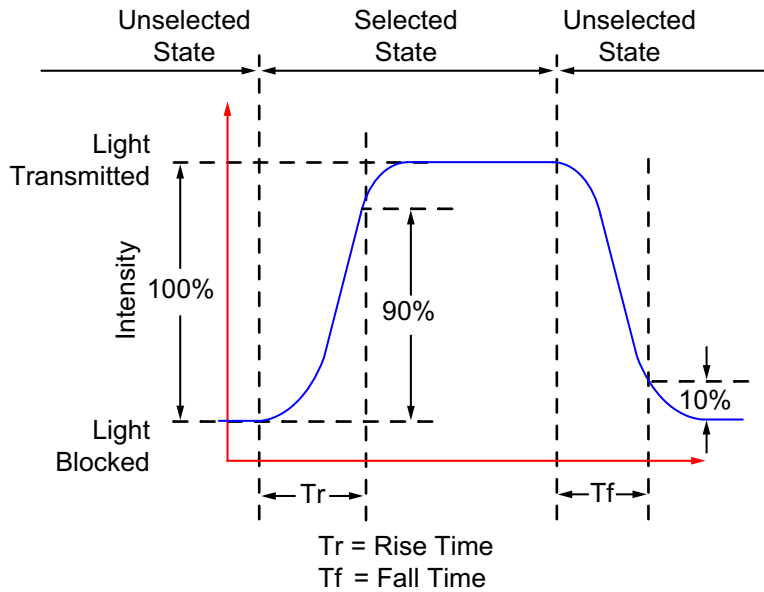


Figure 8. Definition of Response Time (Tr, Tf) (Negative Image)



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

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

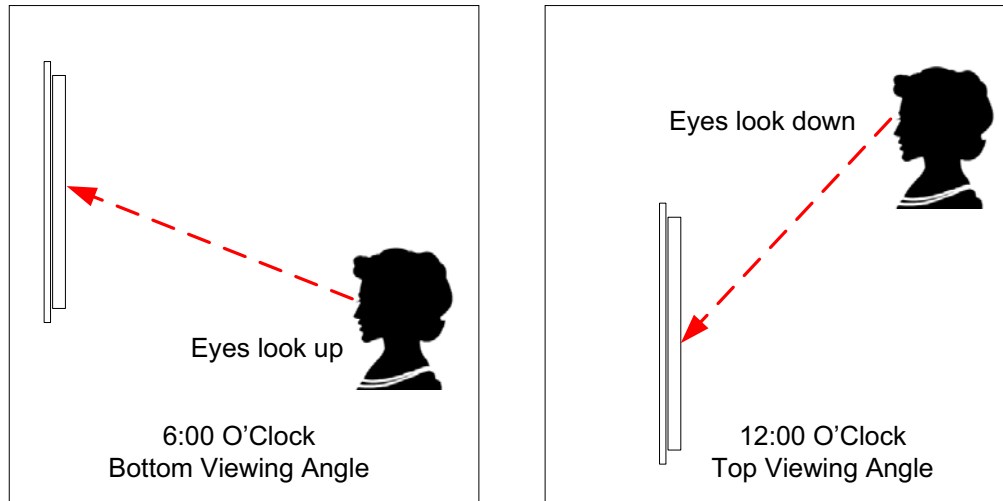


Figure 9. Definition of 6:00 O'clock and 12:00 O'clock Viewing Angles

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

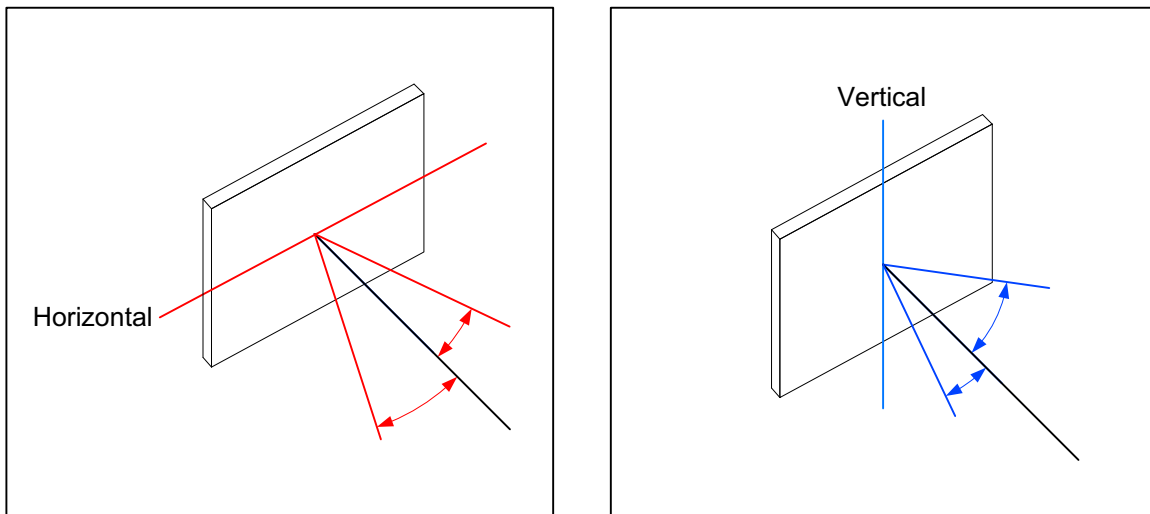


Figure 10. Definition of Horizontal and Vertical Viewing Angles (CR>2)



LED BACKLIGHT INFORMATION

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

Note

For CFA631-TMF-KU with **white** backlight, we recommend that the backlight be dimmed or turned off during periods of inactivity to conserve the LEDs' lifetime.

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

CONNECTION INFORMATION

BUY CABLES SEPARATELY

Cable lengths are approximate. Common configurations using some of these cables are described in the section [CONNECTION INFORMATION \(Pg. 24\)](#). Additional cables are on our [website here](#). Cable lengths are approximate.

Part Number	Cable Descriptions All Cables are RoHS Compliant
USB	
<i>Note:</i> The CFA631-xxx-KU uses a nonstandard 2 mm low profile connector. USB cables with this type of connector are not readily available at retail stores.	
WR-USB-Y03 ~6.5 feet	The cable has two different types of USB connectors, one smaller than the other. Connect the cable's smaller 2 mm female USB connector to the module's 2 mm male USB connector. Connect the cable's larger USB-A female connector to host's USB-A connector.
WR-USB-Y11 ~27 inches	Connect the cable's 2 mm female USB connector to the module's USB connector. Connect the four single pin connectors (Ground, +5v, -D, and +D) to the USB pins on your motherboard.
WR-USB-Y33 ~27 inches	Connect the cable's smaller 2 mm female USB connector to the module's 2 mm male USB connector. Connect the cable's larger female 4-pin 0.1" connector to the USB pins on your host's motherboard. <i>For correct orientation, note the +5v location on the 4-pin connector.</i>
WR-PWR-Y24 ~26 inches	Add this cable for powering the module separately from USB. Connect the cable's 16-pin female connector to the module's 16-pin male H1 connector. Connect the cable's 4-pin male connector to the host's power supply. <i>Note: Open JP2 to avoid back-powering USB.</i>
ATX without optional SCAB (System Cooling Accessory Board)	
WR-PWR-Y25 ~26 inches	Use this ATX power cable to turn an ATX power supply on and off, or power cycle the host through the CFA631-xxx-KU. Connect the cable's 16-pin female connector to the CFA631-xxx-KU's 16-pin male H1 connector. Connect the cable's 4-pin ATX connector to the host's ATX power supply. And connect the cable's 4 separate female pins to the appropriate 4 pins on the host's motherboard. (Cable pins are labeled.)
If connecting optional SCAB (System Cooling Accessory Board) to CFA631-xxx-KU	



Part Number	Cable Descriptions (Continued) All Cables are RoHS Compliant
<p><i>Note:</i> The CFA631-xxx-KU does not supply power to the SCAB. The SCAB requires external power, typically supplied by a 4-pin 3.5-inch floppy drive power connector.</p>	
WR-PWR-Y12 ~13.3 inches	<p>4-pin hard drive to floppy connector and splitter power cable. Connect the cable's 4-pin female connector to the SCAB's male J3 connector. Connect the cable's male 4-pin floppy power connector to the host's power supply. Connect the cable's Reset and Power wires, and the WOL connector to the host's motherboard.</p>
WR-PWR-Y14 ~24 inches	<p>This cable allows ATX power control connections through the optional SCAB. Connect the cable's 7-pin female connector to the SCAB's 7-pin male J8 connector. Connect the cable's labeled Reset, Power and 3-pin WOL connector to the host's motherboard. You will need to order either the WR-EXT-Y15 or WR-EXT-Y19 to connect the SCAB to the module's connector H1.</p>
WR-EXT-Y15 ~16 inches	<p>Use this cable to mount the SCAB some distance away from the module. For example, the SCAB could be mounted in a central location within the host's case to the module mounted in a drive bay. Then the connections to the fans and temperature sensors only need to be run to the SCAB, not all the way to the front panel where the LCD module is mounted.</p> <p>Connect one of the cable's two 16-pin female connectors to the module's 16-pin H1 male connector. Connect the cable's other 16-pin female connector to the SCAB's 16-pin male J1 connector.</p>
WR-EXT-Y19 ~3.5 inches	<p>Use this short cable when the SCAB is mounted directly to the module's built-in bracket.</p> <p>Connect one of the cable's two 16-pin female connectors to the module's 16-pin H1 male connector. Connect the cable's other 16-pin female connector to the SCAB's 16-pin male J1 connector.</p>
WR-FAN-X01 ~16 inches	<p>Connect up to four fan extension cables to connect up to four fans. Connect cable's 3-pin male connector to SCAB'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.)</p>
WR-DOW-Y17 ~12 inches + ~12 inches between connectors	<p>Connect ("daisy chain") up to 32 of these DOW (Dallas 1-Wire) DS18B20 temperature sensor cables to one SCAB. Connect the cable's 3-pin female connector to the SCAB's connector labeled J_DOW. If desired, connect the cable's 3-pin male connector to an additional temperature sensor.</p>
<p>UBERSCAB Kit (System Cooling Accessory Board + Cables)</p>	
<p>The module does not supply power to the SCAB. The SCAB requires external power, typically supplied by a 4-pin 3.5-inch floppy drive power connector. The UBERSCAB is a kit that includes one SCAB, four temperature cables (WR-DOW-Y17), four fan extension cables (WR-FAN-X01), one power cable splitter (WR-PWR-Y12), one 3.5-inch cable to connect SCAB to the module (WR-EXT-19), and one 16-inch cable to connect SCAB to the module (WR-EXT-Y15).</p>	



USB CONNECTION TO HOST

The CFA631-xxx-KU is a USB peripheral, requiring only one connection to the host for both data communications and power supply. The CFA631-xxx-KU uses a low profile 2 mm latching polarized connector for USB connection.

Crystalfontz offers three cables to connect between the CFA631-xxx-KU and the host:

- The [WR-USB-Y03](#) (~6 feet) The cable has two different types of USB connectors, one smaller than the other. Connect the cable's smaller 2 mm female USB connector to the CFA631-xxx-KU's 2 mm male USB connector. Connect the cable's larger USB-A female connector to host's USB-A connector.
- The [WR-USB-Y11](#) (27-inch) has a mating 2 mm connector on one end and standard single pin connectors on the opposite end. These single pin connectors are suitable to plug directly onto the USB headers typically found on motherboards.
- The [WR-USB-Y33](#) (~27 inches) Connect the cable's smaller 2 mm female USB connector to the CFA631-xxx-KU's 2 mm male USB connector. Connect the cable's larger female 4-pin 0.1" connector to the USB pins on your host's motherboard.

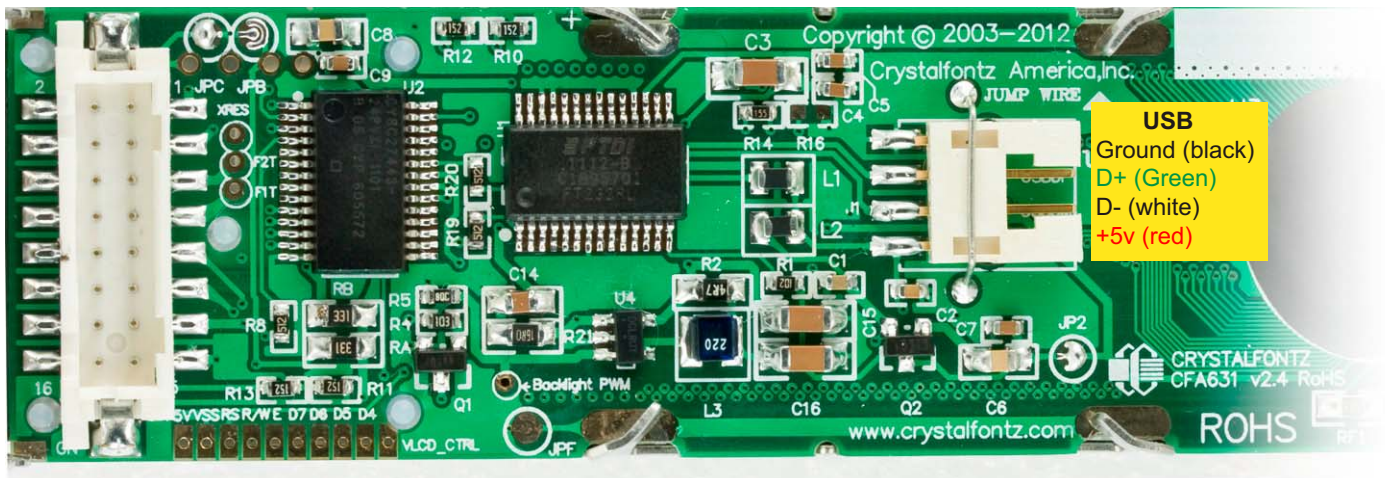


Figure 11. USB Connector Pins Labeled

If you would like to make your own cable, the USB connector on the CFA631-xxx-KU is:

[FCI/Berg 95000-004](#): SMT 2 mm connector, 4-position, polarized

The mating housing and crimping contact for the cable are:

[FCI/Berg 90312-004](#): Housing, 2 mm connector, 4-position, polarized

[FCI/Berg 77138-001](#): Crimping Contact (4 pieces required)

Several versions of Microsoft signed drivers and Macintosh drivers can be downloaded here: www.crystalfontz.com/software/USB_LCD_Driver#docs. Follow the three-step download and installation instructions here: www.crystalfontz.com/software/usb/index.php.



H1 CONNECTOR PIN ASSIGNMENTS - INCLUDES FIVE GPIOs

CFA631-xxx-KU has five GPIOs available on connector H1. These GPIOs can be accessed directly through H1 or through the optional [SCAB](#) (System Cooling Accessory Board) connected to H1.

Note: F1P through F4P and F1T through F4T are reserved for fans with optional SCAB.

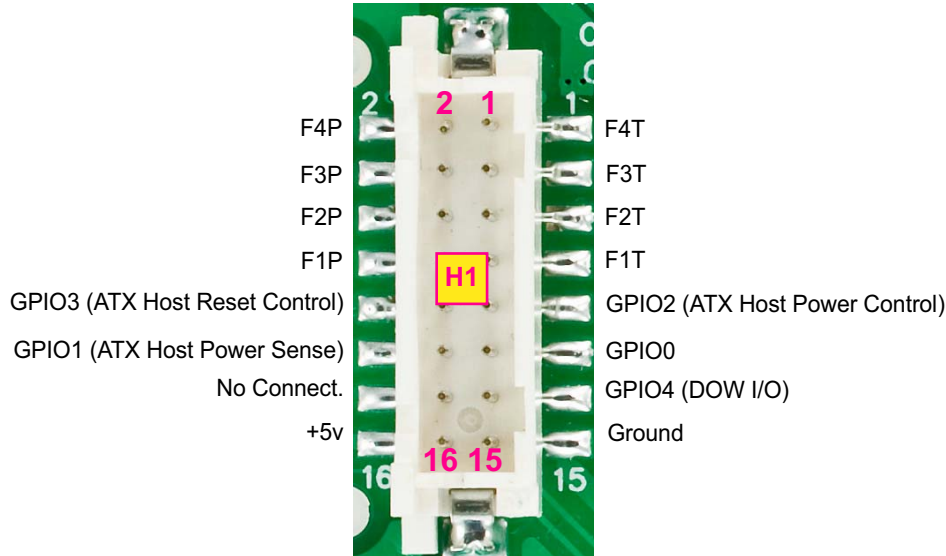


Figure 12. Location of GPIO Pins on H1 Connector

Please see the commands [34 \(0x22\): Set or Set and Configure GPIO Pins \(SCAB Required\) \(Pg. 58\)](#), and [35 \(0x23\): Read GPIO Pin Levels and Configuration State \(SCAB Required\) \(Pg. 60\)](#) below for details on how to control the GPIOs.

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

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

For descriptions of cables that connect to H1, see table descriptions in [Buy Cables Separately \(Pg. 24\)](#).



ATX POWER SUPPLY

ATX Power and Control Connections

- ATX power supply control functionality allows the buttons on the CFA631-xxx-KU to replace the power and reset button on your system, simplifying front panel design. This ATX power supply control functionality can be accomplished with the optional [SCAB+WR-PWR-Y14](#) ATX power cable or use the [WR-PWR-Y25](#) power cable without the SCAB. The SCAB provides fan monitoring and control as well as DOW temperature probe monitoring.

Note

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

When configuring the CFA631-xxx-KU for ATX functionality, **open jumper JP2** in order to ensure correct operation. See [Jumpers That Can Be Modified \(Pg. 15\)](#). This is required whether the optional SCAB is or is not used.

ATX configuration for the CFA631-xxx-KU is powered from the PC's V_{SB} signal, the “stand-by” or “always-on” +5v ATX power supply output, on pins 15 and 16 of the H1 connector. When using the optional SCAB, the +5 standby voltage is supplied on the 7-pin header pins labeled GND and +5v.

GPIO[1] ATX Host Power Sense

Since the CFA631-xxx-KU must act differently depending on whether the host's power supply is on or off, you must also connect the host's “switched +5v” to GPIO[1]. This GPIO line functions as POWER SENSE. The POWER SENSE pin is configured as an input with a pull-down, 5k Ω nominal.

GPIO[2] ATX Host Power Control

The motherboard's power switch input is connected to GPIO[2]. This GPIO line functions as POWER CONTROL. The POWER CONTROL pin is configured as a high impedance input until the LCD module instructs the host to turn on or off. Then it will change momentarily to low impedance output, driving either low or high depending on the setting of POWER INVERT. See command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 53\)](#).

GPIO[3] ATX Host Reset Control

The motherboard's reset switch input is connected to GPIO[3]. This GPIO line functions as RESET. The RESET pin is configured as a high-impedance input until the LCD module wants to RESET the host. Then it will change momentarily



to low impedance output, driving either low or high depending on the setting of RESET_INVERT. See command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 53\)](#). This connection is also used for the hardware watchdog.

ATX Power Supply & Control Connections	With Optional SCAB*	Without Optional SCAB Pins on Connector H1
V _{SB} , +5v	SCAB's 7-pin header, +5v	Pin 16
V _{SB} , Ground	SCAB's 7-pin header, GND	Pin 15
GPIO[1] ATX Host Power Sense	SCAB's 4-pin power header, +5v	Pin 12
GPIO[2] ATX Host Power Control	SCAB's 7-pin power header, GPIO[2]	Pin 9
GPIO[3] ATX Host Reset Control	SCAB's's 7-pin power header, GPIO[3]	Pin 10

*SCAB's JP8 must be open and JP9 must be closed. For details, see the SCAB Data Sheet on www.crystalfontz.com/product/SCAB.html#docs.

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

The Crystalfontz [WR-PWR-Y14](#) cable allows ATX power control connections through the optional [SCAB](#). This allows additional flexibility in cabling and overall functionality of the CFA631-xxx-KU in system control and monitoring. Buy the [WR-EXT-Y15](#) or [WR-EXT-Y19](#) to connect the SCAB to the CFA631-xxx-KU's connector H1.

Note

If the Crystalfontz [WR-PWR-Y14](#) cable and SCAB are ordered at the same time as the module through our Customize and Add to Cart button on the module's website page, Crystalfontz will open JP2 on the CFA631-xxx-KU, open JP8 and close JP9 on the SCAB, and send the following software configuration commands unless we are otherwise instructed.

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

```

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

```



The illustration below shows how:

- ❑ Optional [SCAB](#) connects to the module using a WR-EXT-Y19 cable (or WR-EXT-Y15 can be used).
- ❑ How the optional SCAB connects to your host's motherboard using a Crystalfontz WR-PWR-Y14 cable.

Note: For ATX functionality,
 - JP8 must be open.
 - JP9 must be closed.

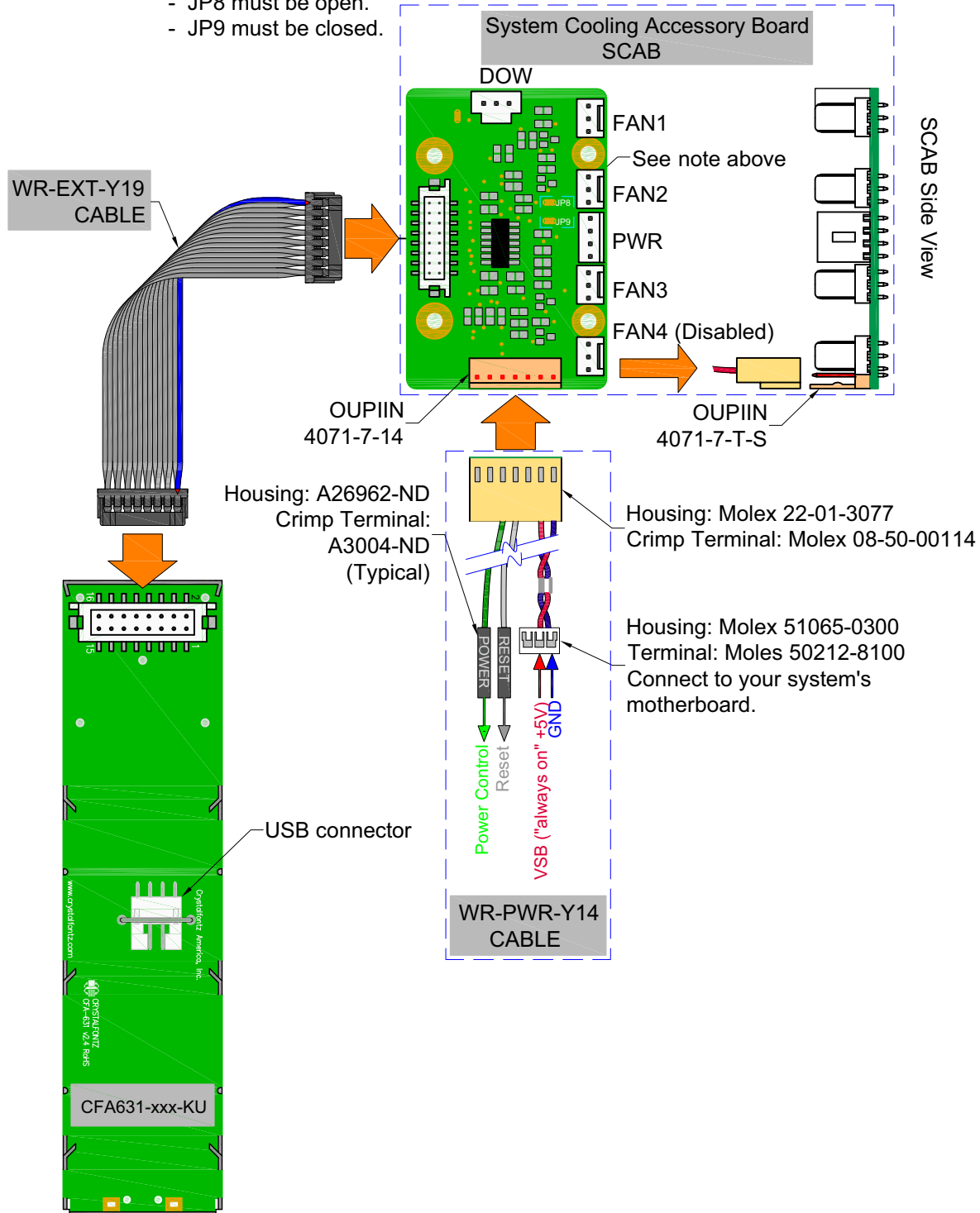


Figure 13. ATX Connection with Optional SCAB using WR-PWR-Y14 ATX Cable



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

The optional Crystalfontz [WR-PWR-Y25](#) cable simplifies ATX power control connections, allowing all ATX power supply control functionality through the CFA631-xxx-KU 's H1 connector.

Note

If the Crystalfontz WR-PWR-Y25 cable is ordered at the same time as the module through our Customize and Add to Cart button on the module's website page, we will open jumper JP2 and send the following software configuration commands unless we are otherwise instructed. Please note that once these changes are made, for the module to power up, power must be applied to connector H1 with +5v applied to pin 15 and ground to pin 16.

```

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

Below is an illustration of how the optional Crystalfontz WR-PWR-Y25 cable connects to the CFA631-xxx-KU's H1 connector and your system's motherboard and ATX power supply:

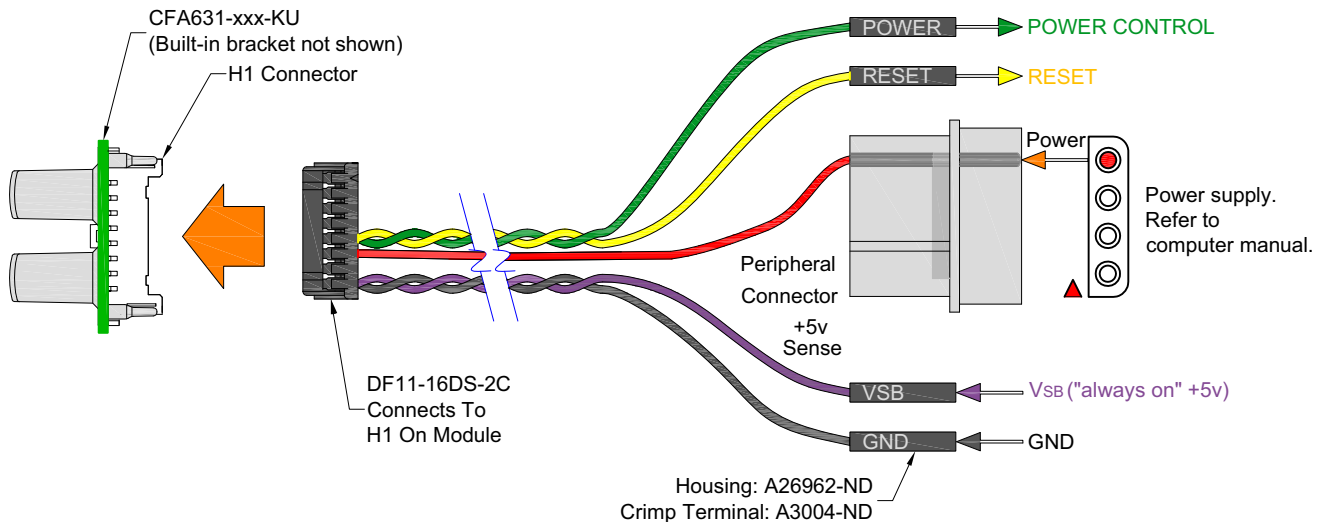


Figure 14. ATX Power Supply and Control Connections using WR-PWR-Y25 ATX Cable

HOW TO CONNECT THE OPTIONAL SCAB

The optional [SCAB](#) is designed to connect to a CFA631-xxx-KU's H1 connector. The SCAB will receive the correct signals to operate from the module.



Here is a photo showing the CFA631-xxx-KU connected to the optional SCAB using the [WR-EXT-Y19](#) cable:

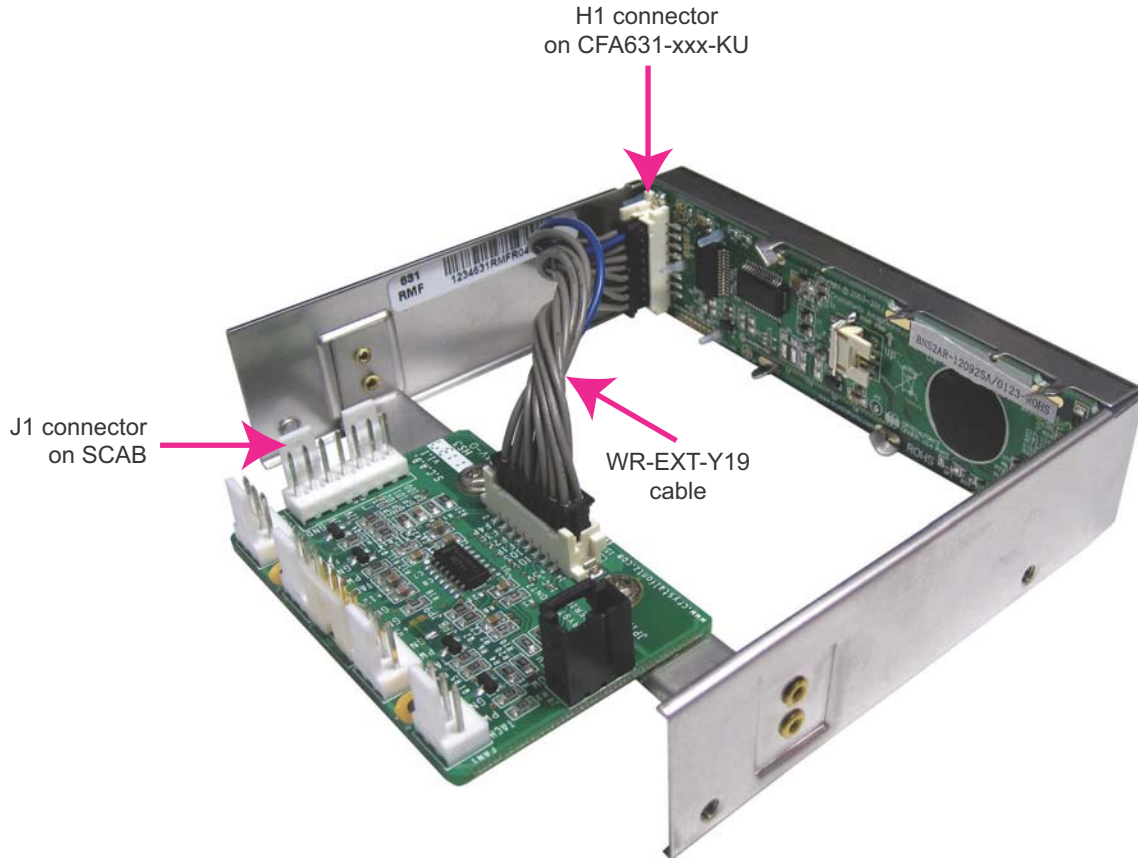


Figure 15. CFA631-xxx-KU Connected to Optional SCAB using WR-EXT-Y19 cable

Two cables are available from Crystalfontz to make the connection between the SCAB and the CFA631-xxx-KU:

1. [WR-EXT-Y15](#) SCAB cable (~16-inch)
This cable allows the SCAB to be mounted some distance away from the CFA631-xxx-KU. For instance, the SCAB could be mounted in a central location within a PC's case. The WR-EXT-Y15 would connect from this central location to the LCD module that is mounted in a drive bay. Then the connections to the fans and temperature sensors only need to be run to the SCAB, not all the way to the front panel where the CFA631-xxx-KU is mounted.
2. [WR-EXT-Y19](#) SCAB cable (~3.5-inch)
This cable can be used when the SCAB is mounted close to the CFA631-xxx-KU, as is the case when the SCAB is fastened directly to the LCD module's mounting bracket. (See the photo above.)



`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: SAMPLE CODE \(Demonstration Software and Sample Code\) \(Pg. 71\)](#) 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, [631_WinTest](#) along with its C source code. Included in the `631_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 CFA631's packets makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for a corresponding acknowledge packet from the CFA631 before sending the next command packet. The CFA631 will respond to all packets within 250 mS. The host software should stop waiting and retry the packet if the CFA631 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 CFA631 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 CFA631 can be configured to send several types of report packets along with regular acknowledge packets. The host should be able to buffer several incoming packets and must guarantee that it can process and remove packets from its input buffer faster than the packets can arrive given the 115200 baud rate of the VCP and the reporting configuration of the CFA631. 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 CFA631 can be configured to report three items. The CFA631 sends reports automatically when the data becomes available. Reports are not sent in response to a particular packet received from the host. The three report types are (1) 0x80: Key Activity, (2) 0x81: Fan Speed Report (SCAB Required), and (3) 0x82: Temperature Sensor Report (SCAB Required). Details are below.



0x80: Key Activity

If a key is pressed or released, the CFA631 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_UL_PRESS    13
    KEY_UR_PRESS    14
    KEY_LL_PRESS    15
    KEY_LR_PRESS    16
    KEY_UL_RELEASE  17
    KEY_UR_RELEASE  18
    KEY_LL_RELEASE  19
    KEY_LR_RELEASE  20
```

0x81: Fan Speed Report (SCAB Required)

If any of up to four fans connected to CFA631+[SCAB](#) is configured to report its speed information to the host, the CFA631 will send Fan Speed Reports for each selected fan every 1/2 second. See command [16 \(0x10\): Set Up Fan Reporting \(SCAB Required\) \(Pg. 45\)](#).

```
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 (SCAB Required)

If any of the up to 32 temperature sensors is configured to report to the host, the CFA631 will send Temperature Sensor Reports for each selected sensor every second. See the command [19 \(0x13\): Set Up WR-DOW-Y17 Temperature Reporting \(SCAB Required\) \(Pg. 46\)](#).

```
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 CFA631
    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 CFA631. 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 CFA631 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 CFA631 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[] = "CFA631: XhX, uYvY"
```

XhX is the hardware revision.
uYvY is the firmware version.

2 (0x02): Write User Flash Area

The CFA631 reserves 16 bytes of nonvolatile memory for arbitrary use by the host. This memory can be used to store a serial number, IP address, gateway address, netmask, or any other data required. All 16 bytes must be supplied.



```
type: 0x02 = 210  
valid data length is 16  
data[] = 16 bytes of arbitrary user data to be stored in the CFA631's nonvolatile memory
```

The return packet will be:

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

3 (0x03): Read User Flash Area

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

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

The return packet will be:

```
type: 0x40 | 0x03 = 0x43 = 6710  
data_length = 16  
data[] = 16 bytes user data recalled from the CFA631's nonvolatile memory
```

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

The CFA631 loads its power-up configuration from nonvolatile memory when power is applied. The CFA631 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 \(CFA633 Compatible\) \(Pg. 42\)](#).
 - command [8 \(0x08\): Set LCD Contents, Line 2 \(CFA633 Compatible\) \(Pg. 42\)](#).
 - command [31 \(0x1F\): Send Data to LCD \(Pg. 57\)](#).
- Special character font definitions (command [9 \(0x09\): Set LCD Special Character Data \(Pg. 43\)](#)).
- Cursor position (command [11 \(0x0B\): Set LCD Cursor Position \(Pg. 43\)](#)).
- 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 Backlights \(Pg. 44\)](#)).
- Fan power settings (command [17 \(0x11\): Set Fan Power \(SCAB Required\) \(Pg. 45\)](#)).
- Settings of any “live” displays (command [21 \(0x15\): Set Up Live Fan or Temperature Display \(SCAB Required\) \(Pg. 48\)](#)).
- Key press and release masks (command [23 \(0x17\): Configure Key Reporting \(Pg. 50\)](#)).
- Fan glitch delay settings (command [26 \(0x1A\): Set Fan Tachometer Glitch Delay \(SCAB Required\) \(Pg. 51\)](#)).
- ATX function enable and pulse length settings (command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 53\)](#)).
- Key legends (command [32: Key Legends \(Pg. 57\)](#)).
- Baud rate (command [33 \(0x21\): Set Baud Rate \(Pg. 58\)](#)).
- GPIO settings (command [34 \(0x22\): Set or Set and Configure GPIO Pins \(SCAB Required\) \(Pg. 58\)](#)).

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 = 410  
valid data_length is 0
```



The return packet will be:

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

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

5 (0x05): Reboot CFA631, Reset Host, or Power Off Host Using ATX

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

This command instructs the CFA631 with ATX to simulate a power-on restart of itself, reset the host, or turn the host's power off. The ability to reset the host may be useful to allow certain host operating system configuration changes to complete. The ability to turn the host's power off under software control may be useful in systems that do not have ACPI* compatible BIOS.

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

Note

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

Rebooting the CFA631 may be useful when testing the boot configuration. It may also be useful to re-enumerate the optional [WR-DOW-Y17](#) temperature sensors on the 1-Wire bus (optional SCAB required).

To reboot the CFA631, send the following packet:

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



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

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

At bootup, there is up to a 500ms (1/2 second) delay between turning on fans. By default, all fans are set to "on" at 100%. If you are not using a fan, set power to 0% (command [17 \(0x11\): Set Fan Power \(SCAB Required\) \(Pg. 45\)](#)) and save this setting as the default boot state (command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#)). This will reduce the boot time.

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

To reset the host using CFA631with ATX, assuming the host's reset line is connected to GPIO[3] as described in command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 53\)](#), send the following packet:

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

Note

The CFA631 will return the acknowledge packet immediately, then reset the host. After resetting the host (~1.5 seconds), the module will reboot itself. The module will not respond to new command packets for up to 3 seconds (~4.5 seconds overall) after its reboot. Part of this delay is the intentional staggered sequencing of turning on power to the fans. If you are not using fans, you can speed the boot process by setting the fan power to 0 (command [17 \(0x11\): Set Fan Power \(SCAB Required\)](#)) 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* using CFA631with ATX, assuming the host's power control line is connected to GPIO[2] as described in command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 53\)](#), send the following packet:

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




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

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

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 \(Pg. 53\)](#), send the following packet:

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

Note

The CFA631 will return the acknowledge packet immediately, then reset the host. After resetting the host (~1.5 seconds), the module will reboot itself. The module will not respond to new command packets for up to 3 seconds (~4.5 seconds overall) after its reboot. Part of this delay is the intentional staggered sequencing of turning on power to the fans. If you are not using fans, you can speed the boot process by setting the fan power to 0 (command [17 \(0x11\): Set Fan Power \(SCAB Required\)](#)) 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 \(Pg. 53\)](#), send the following packet:

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

Note

The CFA631 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 \(SCAB Required\)](#)) 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 = 6910  
data_length = 0
```



6 (0x06): Clear LCD Screen

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

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

The return packet will be:

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

Clear LCD Screen changes the LCD. The LCD contents is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

7 (0x07): Set LCD Contents, Line 1 (CFA633 Compatible)

Sets the center 16 characters displayed for the top line of LCD screen. The first two and last two characters are blanked.

Note

This command allows legacy software that displays data on older CFA633 modules to work unchanged on the CFA631. For new applications, please use the more flexible command [31 \(0x1F\): Send Data to LCD](#).

```
type: 0x7 = 710  
valid data_length is 16  
data[] = top line's display content (must supply 16 bytes)
```

The return packet will be:

```
type: 0x40 | 0x07 = 0x47 = 7110  
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 \(Pg. 38\)](#).

8 (0x08): Set LCD Contents, Line 2 (CFA633 Compatible)

Sets the center 16 characters displayed for the top line of LCD screen. The first two and last two characters are blanked.

Note

This command allows legacy software that displays data on older CFA633 modules to work unchanged on the CFA631. For new applications, please use the more flexible command [31 \(0x1F\): Send Data to LCD](#).

```
type: 0x8 = 810  
valid data_length is 16  
data[] = bottom line's display content (must supply 16 bytes)
```

The return packet will be:

```
type: 0x40 | 0x08 = 0x48 = 7210  
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 \(Pg. 38\)](#).

9 (0x09): Set LCD Special Character Data

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

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

data [1-8] are the bitmap information for this character. Any value is valid between 0 and 63, the msb is at the left of the character cell of the row, and the lsb is at the right of the character cell. Additionally, if you set bit 7 of any of the data bytes, the entire line of pixels within this character will blink.

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

The return packet will be:

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

Set LCD Special Character Data is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

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

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

```
type: 0x0A = 1010
valid data_length is 1
data[0] = address code of desired data
```

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

```
0x40 (\064) to 0x7F (\127) for CGRAM
0x80 (\128) to 0x93 (\147) for DDRAM, line 1
0xC0 (\192) to 0xD3 (\211) for DDRAM, line 2
```

The return packet will be:

```
type: 0x40 | 0x0A = 0x4A = 7410
data_length = 9
```

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

11 (0x0B): Set LCD Cursor Position

This command allows the cursor to be placed at the desired location on the CFA631'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 = 1110
valid data_length is 2
data[0] = column (0-19 valid)
data[1] = row (0-1 valid)
```

The return packet will be:

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



Set LCD Cursor Position is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

12 (0x0C): Set LCD Cursor Style

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

```
type: 0x0C = 1210
valid data_length is 1
data[0] = cursor style (0-4 valid)
0 = no cursor.
1 = blinking block cursor.
2 = static underscore cursor.
3 = blinking block plus underscore.
4 = blinking underscore (Behavior is different from previous CFA631 versions (firmware v2.0 and earlier.)
```

The return packet will be:

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

Set LCD Cursor Style is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

13 (0x0D): Set LCD Contrast

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

```
type: 0x0D = 1310
valid data_length is 1
data[0] = contrast setting (0-254 valid)
60 = light
105 = about right
129 = dark
130-254 = very dark (may be useful at cold temperatures)
```

The return packet will be:

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

Set LCD Contrast is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

14 (0x0E): Set LCD & Keypad Backlights

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

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

The return packet will be:

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

Set LCD & Keypad Backlight is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).



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

When power is applied to the CFA631+ [SCAB](#)+[WR-DOW-Y17](#) temperature sensors, it detects any devices (WR-DOW-Y17) connected to the Dallas Semiconductor 1-Wire (DOW) bus and stores the device's information. This command will allow the host to read the device's information.

Note

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

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

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 CFA631+SCAB reboots, it will detect the WR-DOW-Y17 temperature sensors.

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

The return packet will be:

```
type: 0x40 | 0x12 = 0x52 = 8210  
data_length = 9  
data[0] = device index (0-31 valid)  
data[1-8] = ROM ID of the device
```

If data[1] is 0x22 (WR-DOW-Y17 temperature sensor), then that device can be set up to automatically convert and report the temperature every second. See the command [19 \(0x13\): Set Up WR-DOW-Y17 Temperature Reporting \(SCAB Required\) \(Pg. 46\)](#).

19 (0x13): Set Up WR-DOW-Y17 Temperature Reporting (SCAB Required)

This command will configure the CFA631+[SCAB](#)+[WR-DOW-Y17](#) to report the temperature information to the host every second.



type: 0x13 = 19₁₀
valid data_length is 4
data[0-3] = 32-bit bitmask indicating which temperature sensors 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 = disabilities
|   |   |   |   |   |   |   |   |--- 30: 1 = enable, 0 = disable
|   |   |   |   |   |   |   |   |--- 31: 1 = enable, 0 = disable

```

Sensor enabled must have been detected as 0x28 (WR-DOW-Y17 temperature sensor) during DOW enumeration. This can be verified by using the command [18 \(0x12\): Read WR-DOW-Y17 Temperature Sensors \(SCAB Required\) \(Pg. 46\)](#).

The return packet will be:

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

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

The CFA631+SCAB can function as an RS-232 to Dallas1-Wire bridge. CFA631+SCAB can send up to 15 bytes and receive up to 14 bytes. This will be sufficient for many devices, but some devices require larger transactions cannot be fully used with the CFA631+SCAB. This command allows you to specify arbitrary transactions on the 1-Wire bus. The 1-Wire commands follow this basic layout:



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

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

```
type: 0x14 = 2010
valid data_length is 2 to 16
data[0] = device_index (0-32 valid)
data[1] = number_of_bytes_to_read (0-14 valid) 0
data[2-15] = data_to_be_written[data_length-2]
```

If `device_index` is 32, then no address phase will be executed. If `device_index` is in the range of 0 to 31, and a 1-Wire device was detected for that `device_index` at power on, then the write cycle will be prefixed with a "Match ROM" command and the address information for that device.

If `data_length` is 2, then no specific write phase will be executed (although address information may be written independently of `data_length` depending on the value of `device_index`).

If `data_length` is greater than 2, then `data_length-2` bytes of `data_to_be_written` will be written to the 1-Wire bus immediately after the address phase.

If `number_of_bytes_to_read` is 0, then no read phase will be executed. If `number_of_bytes_to_read` is not 0, then `number_of_bytes_to_read` will be read from the bus and loaded into the response packet.

The return packet will be:

```
type: 0x40 | 0x14 = 0x54 = 8410
data_length = 2 to 16
data[0] = device_index (0-31 valid)
data[data_length-2] = Data read from the 1-Wire bus. This is the same
                    as number_of_bytes_to_read from the command.
data[data_length-1] = 1-Wire CRC
```

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

You can configure the CFA631+[SCAB](#) to automatically update a portion of the LCD with a "live" RPM or temperature reading. Once the display is configured using this command, the CFA631+SCAB 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 \(Pg. 38\)](#). You can configure the CFA631+SCAB 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 4 slots. Each of the 4 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 = 2110
valid data length is 7 or 2 (for turning a slot off)
data[0]: display slot (0-3)
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 sensor 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 1-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 tachometer will override the fan power setting to 100% for up to 1/8 of a second every 1/2 second. Please see "Fan Connections" section in the [SCAB](#) Data Sheet for details.

The return packet will be:

```
type: 0x40 | 0x15 = 0x55 = 8510
data_length = 0
```

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

This command allows you to access the CFA631's LCD controller directly. Note: It is possible to corrupt the CFA631 display using this command.

Note

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



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

The return packet will be:

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

23 (0x17): Configure Key Reporting

By default, the CFA631 reports any key event to the host. This command allows the key events to be enabled or disabled on an individual basis.

```
#define KP_UL      0x01 //(upper-left)
#define KP_UR      0x02 //(upper-right)
#define KP_LL      0x04 //(lower-left)
#define KP_LR      0x08 //(lower-right)
```

```
type: 0x17 = 2310
data_length = 2
data[0]: press mask
data[1]: release mask
```

Valid values of the mask are \000-\015.

The return packet will be:

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

Configure Key Reporting is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the CFA631+[SCAB](#) 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_UL      0x01 //(upper-left)
#define KP_UR      0x02 //(upper-right)
#define KP_LL      0x04 //(lower-left)
#define KP_LR      0x08 //(lower-right)
```

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



The return packet will be:

```
type: 0x40 | 0x18 = 0x58 = 8810
data_length = 3
data[0] = bit mask showing the keys currently pressed
data[1] = bit mask showing the keys that have been pressed since the last poll
data[2] = bit mask showing the keys that have been released since the last poll
```

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

The CFA631+[SCAB](#) can be used as part of an active cooling system. The fans in a system can be slowed down to reduce noise when a system is idle or when the ambient temperature is low. The fans can be 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 CFA631's firmware that would be an optimal fit for your application.

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

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

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

type = 0x19 = 2510
data_length = 2
data[0] = bit mask of fans set to fail-safe (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 = 8910
data_length = 0
```

26 (0x1A): Set Fan Tachometer Glitch Delay (SCAB Required)

The CFA631+[SCAB](#) 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 CFA631 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 CFA631+SCAB will recognize transitions on the tachometer line. The delay is specified in counts, each count being nominally 552.5 μ S long (1/100 of one period of the 18 Hz PWM repetition rate).

In practice, most fans will not need the delay to be changed from the default length of 1 count. If a fan's tachometer output is not stable when its PWM setting is other than 100%, simply increase the delay until the reading is stable. Typically you



would (1) start at a delay count of 50 or 100, (2) reduce it until the problem reappears, and then (3) slightly increase the delay count to give it some margin.

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

The Fan Glitch Delay is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

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

The return packet will be:

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

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

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

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

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

The return packet will be:

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



28 (0x1C): Set ATX Power Switch Functionality

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

The combination of the CFA631 with ATX can be used to replace the function of the power and reset switches in a standard ATX-compatible system. The ATX power switch functionality is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

Note

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

These settings are factory default but may be changed by the user. Please see command [34 \(0x22\): Set or Set and Configure GPIO Pins \(SCAB Required\) \(Pg. 58\)](#). 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 CFA631 with ATX are normally high-impedance. Electrically, they appear to be disconnected or floating. When the CFA631 with ATX 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 CFA631 with ATX 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, pressing the upper right key for 4 seconds will pulse RESET (GPIO[3]) pin for 1 second. During the 1-second pulse, the CFA631 with ATX will show "RESET", and then reset itself, showing its boot state as if it had just powered on. Once the pulse has finished, the CFA631 with ATX 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 upper right 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 CFA631 with ATX will show POWER ON, then the CFA631 with ATX will reset itself.

Function 3: KEYPAD_POWER_OFF

If POWER-ON SENSE (GPIO[1]) is high, holding the lower right 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 CFA631 with ATX will continue to drive the line for a maximum of 5 additional seconds. During this time the CFA631 with ATX will show "POWER OFF".

Function 4: LCD_OFF_IF_HOST_IS_OFF

If LCD_OFF_IF_HOST_IS_OFF is set, the CFA631 with ATX will blank its screen and turn off its backlight to simulate its power being off any time POWER-ON SENSE is low. The CFA631 with ATX will still be active (since it is powered by V_{SB}), monitoring the keypad for a power-on keystroke. If +12v remains active (which would not be expected since the host is "off"), the fans will remain on at their previous settings. Once POWER-ON SENSE (GPIO[1]) goes high, the CFA631 with ATX will reboot as if power had just been applied to it.

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

#define LCD_OFF_IF_HOST_IS_OFF 0x10
#define KEYPAD_RESET          0x20
#define KEYPAD_POWER_ON       0x40
#define KEYPAD_POWER_OFF      0x80

type: 0x1C = 2810
data_length: 1 or 2
data[0]: 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 = 9210
data_length: 0
```



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

Some systems use hardware watchdog timers to ensure that a software or hardware failure does not result in an extended system outage. Once the host system has booted, a system monitor program is started. The system monitor program would enable the watchdog timer on the CFA631 with ATX. If the system monitor program fails to reset the watchdog timer, the CFA631 with ATX will reset the host system

Note

The GPIO pins used for ATX control must not be configured as user GPIO. They must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are factory default, but may be changed by the user. Please see the note under command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 53\)](#) or command [34 \(0x22\): Set or Set and Configure GPIO Pins \(SCAB Required\) \(Pg. 58\)](#).

```
type: 0x1D = 2910  
data_length = 1  
data[0] = enable/timeout
```

If timeout is 0, the watchdog is disabled.

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

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

If the command is not re-issued within timeout seconds, then the CFA631 with ATX will reset the host (see command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 53\)](#) for details). Since the watchdog is off by default when the CFA631 powers up, the CFA631 with ATX will not issue another host reset until the host has once again enabled the watchdog.

The return packet will be:

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

30: (0x1E) Read Reporting & Status

This command can be used to verify the current items configured to report to the host, as well as some other miscellaneous status information.

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




The return packet will be:

```

type = 0x40 | 0x1E = 0x5E = 9410
data_length = 15
data[0] = fan 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 = 3110
data_length = 3 to 22
data[0]: col = x = 0 to 19
data[1]: row = y = 0 to 1
data[2-21]: text to place on the LCD, variable from 1 to 20 characters
  
```

The return packet will be:










```

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

Send Data to LCD is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

32: Key Legends

The CFA631 offers firmware support for “soft keys”. Eight predefined icons correspond to common key functions:

#define_KEY_LEGEND_BLANK	0	
#define_KEY_LEGEND_CANCEL	1	
#define_KEY_LEGEND_CHECK	2	
#define_KEY_LEGEND_UP	3	
#define_KEY_LEGEND_DOWN	4	
#define_KEY_LEGEND_RIGHT	5	
#define_KEY_LEGEND_LEFT	6	
#define_KEY_LEGEND_PLUS	7	
#define_KEY_LEGEND_MINUS	8	
#define_KEY_LEGEND_NONE	9	// no key or symbol



The host simply enables key legends—specifying the icon to display corresponding to each key—and then the CFA631 firmware draws the legends. Each soft-key legend “inverts” when the corresponding hard key is pressed, providing instant feedback that the key has been actuated.

The key legends use special characters 2,3,4,5,6 and 7. Special characters 0 and 1 are available for other functions.

The key legends act as a second layer of the display over the 6 right-most characters. Text written to the key legends area are overwritten instantly by the key legends.

```
type: 0x20 = 3210
data_length = 1 (to disable) or 5 (to enable and specify)
data[0]: enable = 1, disable = 0
data[1] = code for icon to be displayed for upper-left key
data[2] = code for icon to be displayed for upper-right key
data[3] = code for icon to be displayed for lower-left key
data[4] = code for icon to be displayed for lower-right key
```

The return packet will be:

```
type = 0x40 | 32
data_length = 0
```

The key reports are not affected by the key legend settings. The host should make the appropriate action based on the key legend settings and the keys reported.

By using special character definitions and key reports, the functionality of the key legends can be emulated in host software, allowing unlimited icon definitions.

Key Legends is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 38\)](#).

33 (0x21): Set Baud Rate

This command will change the CFA631’s baud rate. The CFA631 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 CFA631 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 \(Pg. 38\)](#) if you want the CFA631 to power up at the new baud rate.

The factory default baud rate is 115200.

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

The return packet will be:

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

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

The CFA631 (hardware versions v2.0 and up, firmware versions 2.0 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 CFA631 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.

Note

GPIO[1] has R8 (5.6k) in series by default. If you need GPIO[1] to be a low impedance output, please replace R8 with a 0Ω resistor.

The CFA631 continuously polls the GPIOs as inputs at 32 Hz. The present level can be queried by the host software at a lower rate. The CFA631 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 CFA631 to read the inputs is inherently “debounced”.

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

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

Note

The GPIO pins may also be used for ATX control through the optional [SCAB](#)'s 7-pin connector and [WR-DOW-Y17](#) temperature sensing through the SCAB's DOW header. By factory default, the GPIO output setting, function, and drive mode are set correctly to enable operation of the ATX and DOW functions. **The GPIO output setting, function, and drive mode must be set to the correct values in order for the ATX and DOW functions to work. Improper use of this command can disable the ATX and DOW functions.** The [631_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 \(Pg. 38\)](#).

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

data[0]: index of GPIO to modify on optional SCAB's connector when using CFA631+SCAB+WR-
PWR-Y14
  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--has 1kΩ hardware pull-up)
  5-255 = not accessible

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

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



```

data[2] = Pin function select and drive mode (optional, 0-15 valid)
---- FDDD
| | | | -- DDD = Drive Mode (based on output state of 1 or 0)
| | | | =====
| | | | 000: 1=Fast, Strong Drive Up, 0=Resistive Pull Down
| | | | 001: 1=Fast, Strong Drive Up, 0=Fast, Strong Drive Down
| | | | 010: Hi-Z, use for input
| | | | 011: 1=Resistive Pull Up, 0=Fast, Strong Drive Down
| | | | 100: 1=Slow, Strong Drive Up, 0=Hi-Z
| | | | 101: 1=Slow, Strong Drive Up, 0=Slow, Strong Drive Down
| | | | 110: reserved, do not use -- error returned
| | | | 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 = 9810

data_length = 0

```

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

Please see command [34 \(0x22\): Set or Set and Configure GPIO Pins \(SCAB Required\) \(Pg. 58\)](#) for details on the GPIO architecture.

```

type: 0x23 = 3510
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--has series R8 of 5.6kΩ)
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--has 1kΩ hardware pull-up on SCAB.)
5-255 = not accessible

```

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

The return packet will be:

```

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

```

(Continues on the next page.)



CHARACTER GENERATOR ROM (CGROM)

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

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

Figure 16. Character Generated ROM



LCD MODULE RELIABILITY AND LONGEVITY

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

MODULE RELIABILITY

ITEM	SPECIFICATION	
LCD module excluding Keypad, status LEDs, and Backlights	50,000 to 100,000 hours	
Keypad	1,000,000 keystrokes	
Backlights for Red LED Display and Red LED Keypad	50,000 to 100,000 hours	
Backlights for White LED Display and Blue LED Keypad	<i>Power-On Hours</i>	<i>% of Initial Brightness (New Module)</i>
	<10,000 hours	>90%
	<50,000 hours	>50%
<i>Note: For modules with white LED backlights (CFA631-TMF-KU), adjust backlight brightness so the display is readable but not too bright. Dim or turn off the backlight during periods of inactivity to conserve the white LED backlight lifetime.</i>		
<i>Note: Values listed above are approximate and represent typical lifetime.</i>		

MODULE LONGEVITY (EOL / REPLACEMENT POLICY)

Crystalfontz is committed to making all of our LCD modules available for as long as possible. 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.

CARE AND HANDLING PRECAUTIONS

For optimum operation of the module and to prolong its life, please follow the precautions below. Excessive voltage will shorten the life of the module. You must drive the display within the specified voltage limit. See [Absolute Maximum Ratings \(Pg. 18\)](#).

ESD (ELECTROSTATIC DISCHARGE)

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

AVOID SHOCK, IMPACT, TORQUE, OR TENSION

- Do not expose the CFA631 to strong mechanical shock, impact, torque, or tension.
- Do not drop, toss, bend, or twist the module.
- Do not place weight or pressure on the module.
- If 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.



HOW TO CLEAN

- The polarizer (laminated to the glass) is soft plastic. The soft plastic is easily scratched or damaged. Damage is especially obvious on “negative” modules such as the CFA631-RMF-KU and CFA631-TMF-KU (modules that appear dark when power is “off”). Be careful when you clean the polarizer.
- 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. 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.

OPERATION

- Your circuit should be designed to protect the CFA631 from ESD and power supply transients.
- Observe the operating temperature limitations: a minimum of 0°C to a maximum of +50°C with minimal fluctuation. Operation outside of these limits may shorten life and/or harm display. Changes in temperature can result in changes in contrast.
 - At lower temperatures of this range, response time is delayed.
 - At higher temperatures of this range, display becomes dark. (You may need to adjust the contrast.)
- Operate away from dust, moisture, and direct sunlight.
- For the CFA631-TMF-KU with white LEDs, adjust backlight brightness so the display is readable but not too bright. Dim or turn off the backlight during periods of inactivity to conserve the white LED backlight lifetime.

STORAGE AND RECYCLING

- Store in an ESD-approved container away from dust, moisture, and direct sunlight, fluorescent lamps, or any ultraviolet ray with humidity less than 90% noncondensing.
- Observe the storage temperature limitations: -10°C minimum, +60°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 modules while they are in storage.
- To discard, please recycle your modules at an approved facility.

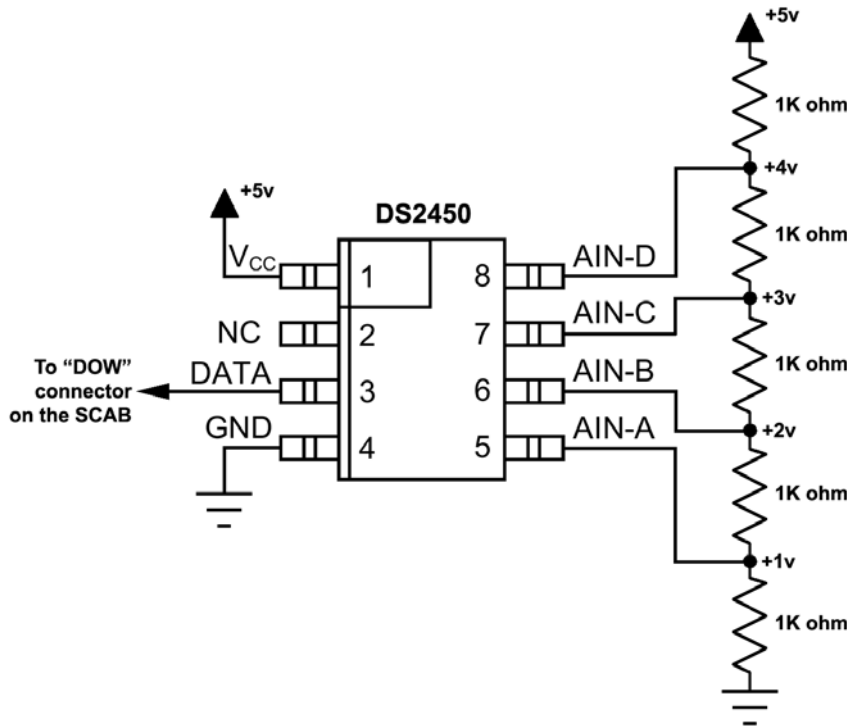


APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTERS

This appendix describes a simple test circuit that demonstrates how to connect a Dallas Semiconductor DS2450 4-channel ADC to the CFA631'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 CFA631. In this example the DS2450 appears at device index 0. Your software should query the connected devices using command [18 \(0x12\): Read WR-DOW-Y17 Temperature Sensors \(SCAB Required\) \(Pg. 46\)](#) to verify the locations and types of DOW devices connected in your application.

Please refer to the [DS2450 Data Sheet](#) and the description for command [20 \(0x14\): Arbitrary DOW Transaction \(SCAB Required\) \(Pg. 47\)](#) more information.



Appendix A Figure 1. Test Circuit Schematic

Start [631_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 = 130561.016015625 volts (channel A)
0x64DF = 258232.009541321 volts (channel B)
0x9684 = 385322.998553467 volts (channel C)
0xC86A = 513063.992623901 volts (channel D)

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

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

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



APPENDIX B: CONNECTING A DS1963S SHA IBUTTON

This appendix describes connecting a Dallas Semiconductor DS1963S Monetary iButton with SHA-1 Challenge Response Algorithm and 4KB of nonvolatile RAM to the CFA631+ SCAB's DOW (Dallas One Wire) connector. It also gives a sample command sequence to read and write the DS1963S's scratch memory.

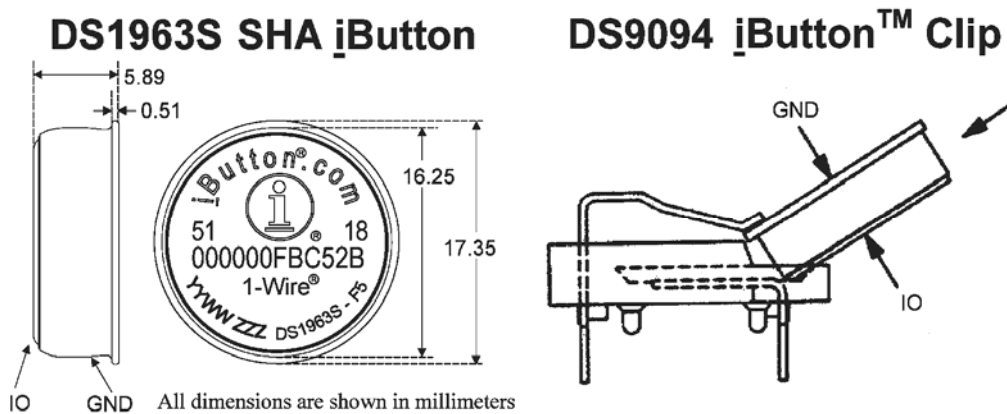
The DS1963S can be used as a secure dongle to protect your system's application software from being copied. Even if the communication channel is compromised or the host is not authentic, the SHA algorithm ensures that the data is still secure. Please see the following Maxim/Dallas white papers and application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

- [White Paper 1: SHA Devices Used in Small Cash Systems](#)
- [White Paper 2: Using the 1-Wire Public-Domain Kit](#)
- [White Paper 3: Why are 1-Wire SHA-1 Devices Secure?](#)
- [White Paper 4: Glossary of 1-Wire SHA-1 Terms](#)
- [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 CFA631+SCAB. In this example the DS1963S appears at device index 0. Your software should query the connected devices using command [19 \(0x13\): Set Up WR-DOW-Y17 Temperature Reporting \(SCAB Required\) \(Pg. 46\)](#) to verify the locations and types of DOW devices connected in your application.

Please refer to the [DS1963S Data Sheet](#) and the description for command [20 \(0x14\): Arbitrary DOW Transaction \(SCAB Required\) \(Pg. 47\)](#) for more information.

To connect the DS1963S to the CFA631+SCAB, simply make one connection between the DS1963S's "GND" terminal and the CFA631 DOW connector's GND pin, and a second connection between the DS1963S's "IO" pin and the CFA631 DOW connector's I/O pin. By using a DS9094 iButton Clip, the connection is easy.



Appendix A Figure 1. Connecting DS1963S SHA iButton using DS9094 iButton Clip



To demonstrate reading and writing the scratch memory on DS1963S, open the [631_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 631_WinTest Packet Debugger dialog.

First read the address of the DS1963S as detected by the CFA631 at boot. Since only one device is connected, you only need to query index 0. In a production situation, query all 32 indices to get a complete picture of the devices available on the DOW bus.

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

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

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

Erase Scratchpad [C3h]

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

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

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

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

Read Scratchpad Command [AAh]

HIDE = 0:

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

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

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



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

Write Scratchpad Command [0Fh]

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

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

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

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

Command:

20 = Arbitrary DOW transaction

Data sent:

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

Data received:

C=84 (d=0) :00

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

Command:

20 = Arbitrary DOW transaction

Data sent:

\000\013\xAA

Data received:

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

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

Command:

20 = Arbitrary DOW transaction

Data sent:

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

Data received:

C=84 (d=0) :00

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

Command:

20 = Arbitrary DOW transaction

Data sent:

\000\013\xAA

Data received:

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

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



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

DRIVERS

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

DEMONSTRATION SOFTWARE

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

- ❑ Windows compatible test/demonstration program and source.
www.crystalfontz.com/product/631WinTest
- ❑ Linux compatible command-line demonstration program with C source code. 8K.
www.crystalfontz.com/product/linux_cli_examples.html
- ❑ Supported by CrystalControl freeware.
www.crystalfontz.com/product/CrystalControl2.html

ALGORITHMS TO CALCULATE THE CRC

Below are eight sample algorithms that will calculate the CRC of a CFA631 packet. The CRC used in the CFA631 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 CFA631 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

```




```

        rrf      datareg,f    ; perform shift of data into CRC register
        rrf      accumh,f    ;
        rrf      accuml,f    ;
        btfsz   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 CFA631 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 631 WinTest:
'http://www.crystalfontz.com/product/635WinTest.html
'Full zip of the project is available in our forum:
'http://www.crystalfontz.com/forum/showthread.php?postid=9921#post9921

```

```

Private Type WORD
    Lo As Byte
    Hi As Byte
End Type

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

Dim crcLookupTable(256) As WORD

Private Sub MSComm_OnComm()
'Leave this here
End Sub

```



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

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

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

Algorithm 5: “Java” Table Implementation

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

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



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

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

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

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

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



```

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

```

Algorithm 6: “Perl” Table Implementation

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

```

#!/usr/bin/perl

use strict;

my @CRC_LOOKUP =
(0x00000, 0x01189, 0x02312, 0x0329B, 0x04624, 0x057AD, 0x06536, 0x074BF,
0x08C48, 0x09DC1, 0x0AF5A, 0x0BED3, 0x0CA6C, 0x0DBE5, 0x0E97E, 0x0F8F7,
0x01081, 0x00108, 0x03393, 0x0221A, 0x056A5, 0x0472C, 0x075B7, 0x0643E,
0x09CC9, 0x08D40, 0x0BFDB, 0x0AE52, 0x0DAED, 0x0CB64, 0x0F9FF, 0x0E876,
0x02102, 0x0308B, 0x00210, 0x01399, 0x06726, 0x076AF, 0x04434, 0x055BD,
0x0AD4A, 0x0BCC3, 0x08E58, 0x09FD1, 0x0EB6E, 0x0FAE7, 0x0C87C, 0x0D9F5,
0x03183, 0x0200A, 0x01291, 0x00318, 0x077A7, 0x0662E, 0x054B5, 0x0453C,
0x0BDCB, 0x0AC42, 0x09ED9, 0x08F50, 0x0FBEEF, 0x0EA66, 0x0D8FD, 0x0C974,
0x04204, 0x0538D, 0x06116, 0x0709F, 0x00420, 0x015A9, 0x02732, 0x036BB,
0x0CE4C, 0x0DFC5, 0x0ED5E, 0x0FCD7, 0x08868, 0x099E1, 0x0AB7A, 0x0BAF3,
0x05285, 0x0430C, 0x07197, 0x0601E, 0x014A1, 0x00528, 0x037B3, 0x0263A,
0x0DECD, 0x0CF44, 0x0FDDF, 0x0EC56, 0x098E9, 0x08960, 0x0BBFB, 0x0AA72,
0x06306, 0x0728F, 0x04014, 0x0519D, 0x02522, 0x034AB, 0x00630, 0x017B9,
0x0EF4E, 0x0FEC7, 0x0CC5C, 0x0DDD5, 0x0A96A, 0x0B8E3, 0x08A78, 0x09BF1,
0x07387, 0x0620E, 0x05095, 0x0411C, 0x035A3, 0x0242A, 0x016B1, 0x00738,
0x0FFCF, 0x0EE46, 0x0DCDD, 0x0CD54, 0x0B9EB, 0x0A862, 0x09AF9, 0x08B70,
0x08408, 0x09581, 0x0A71A, 0x0B693, 0x0C22C, 0x0D3A5, 0x0E13E, 0x0F0B7,
0x00840, 0x019C9, 0x02B52, 0x03ADB, 0x04E64, 0x05FED, 0x06D76, 0x07CFF,
0x09489, 0x08500, 0x0B79B, 0x0A612, 0x0D2AD, 0x0C324, 0x0F1BF, 0x0E036,
0x018C1, 0x00948, 0x03BD3, 0x02A5A, 0x05EE5, 0x04F6C, 0x07DF7, 0x06C7E,
0x0A50A, 0x0B483, 0x08618, 0x09791, 0x0E32E, 0x0F2A7, 0x0C03C, 0x0D1B5,
0x02942, 0x038CB, 0x00A50, 0x01BD9, 0x06F66, 0x07EEF, 0x04C74, 0x05DFD,
0x0B58B, 0x0A402, 0x09699, 0x08710, 0x0F3AF, 0x0E226, 0x0D0BD, 0x0C134,
0x039C3, 0x0284A, 0x01AD1, 0x00B58, 0x07FE7, 0x06E6E, 0x05CF5, 0x04D7C,
0x0C60C, 0x0D785, 0x0E51E, 0x0F497, 0x08028, 0x091A1, 0x0A33A, 0x0B2B3,
0x04A44, 0x05BCD, 0x06956, 0x078DF, 0x00C60, 0x01DE9, 0x02F72, 0x03EFB,
0x0D68D, 0x0C704, 0x0F59F, 0x0E416, 0x090A9, 0x08120, 0x0B3BB, 0x0A232,
0x05AC5, 0x04B4C, 0x079D7, 0x0685E, 0x01CE1, 0x00D68, 0x03FF3, 0x02E7A,
0x0E70E, 0x0F687, 0x0C41C, 0x0D595, 0x0A12A, 0x0B0A3, 0x08238, 0x093B1,
0x06B46, 0x07ACF, 0x04854, 0x059DD, 0x02D62, 0x03CEB, 0x00E70, 0x01FF9,
0x0F78F, 0x0E606, 0x0D49D, 0x0C514, 0x0B1AB, 0x0A022, 0x092B9, 0x08330,
0x07BC7, 0x06A4E, 0x058D5, 0x0495C, 0x03DE3, 0x02C6A, 0x01EF1, 0x00F78);

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

```



```

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

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

my $valid_crc = '5584' ;

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

my $crc = 0xFFFF ;

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

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

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

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

```

Algorithm 7: For PIC18F8722 or PIC18F2685

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

```

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

```




```

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

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

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

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

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

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

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

```



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

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

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

    return

;=====
; macro to move string to transmit buffer
SHOW macro src, stringname
    call     src
    MOVLFS  upper stringname, TBLPTRU
    MOVLFS  high stringname, TBLPTRH
    MOVLFS  low stringname, TBLPTRL
    call     MOVE_STR
endm

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

ms1b:
    return

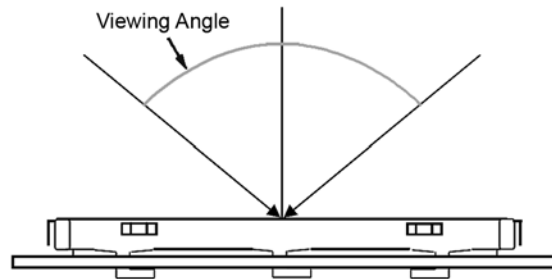
;=====
```



APPENDIX D: QUALITY ASSURANCE STANDARDS

INSPECTION CONDITIONS

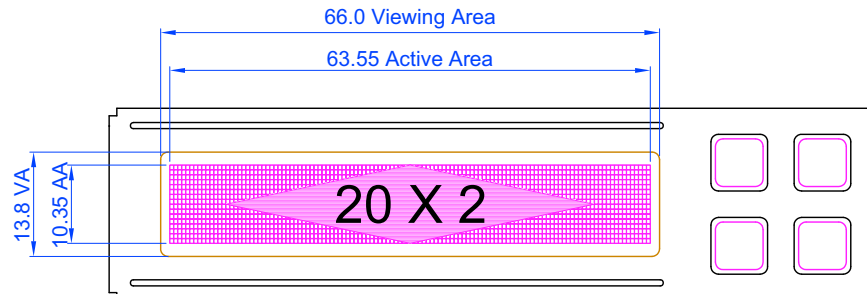
- Environment
 - Temperature: $25\pm 5^{\circ}\text{C}$
 - Humidity: 30~85% RH
- For visual inspection of active display area
 - Source lighting: two 20 watt or one 40 watt fluorescent light
 - Display adjusted for best contrast
 - Viewing distance: 30 ± 5 cm (about 12 inches)
 - Viewing angle: inspect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



COLOR DEFINITIONS

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

DEFINITION OF ACTIVE AREA AND VIEWING AREA





ACCEPTANCE SAMPLING

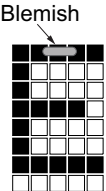
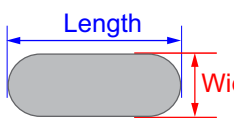
DEFECT TYPE	AQL*
Major	≤.65%
Minor	<1.0%
* Acceptable Quality Level: maximum allowable error rate or variation from standard	

DEFECTS CLASSIFICATION

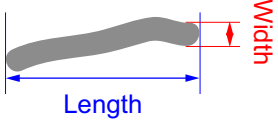
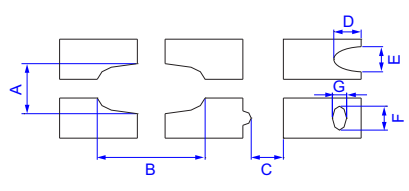
Defects are defined as:

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

ACCEPTANCE STANDARDS

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



#	DEFECT TYPE	ACCEPTANCE STANDARDS CRITERIA (Continued)			MAJOR/ MINOR
6	Dark lines or scratches in display area 	<i>Defect Width (mm)</i>	<i>Defect Length (mm)</i>	<i>Acceptable Qty</i>	Minor
		≤0.03	≤3.0	3	
		0.03 to 0.05	≤2.0	2	
		0.05 to 0.08	≤2.0	1	
		0.08 to 0.10	≤3.0	0	
		≥0.10	>3.0	0	
7	Bubbles between polarizer film and glass	<i>Defect Size (mm)</i>	<i>Acceptable Qty</i>	Minor	
		≤0.20	Ignore		
		0.20 to 0.40	3		
		0.40 to 0.60	2		
		≥0.60	0		
8	Display pattern defect 	<i>Dot Size (mm)</i>	<i>Acceptable Qty</i>		Minor
		$((A+B)/2) \leq 0.2$	≤ 3 total defects ≤ 2 pinholes per digit		
		C > 0			
		$((D+E)/2) \leq 0.25$			
		$((F+G)/2) \leq 0.25$			
9	Backlight defects	1. Light fails or flickers.* 2. Color and luminance do not correspond to specifications.* 3. Exceeds standards for display's blemishes or foreign matter (see test 5, Pg. 84), and dark lines or scratches (see test 6, Pg. 85). <i>*Minor if display functions correctly. Major if the display fails.</i>			Minor
10	COB defects	1. Pinholes >0.2 mm. 2. Seal surface has pinholes through to the IC. 3. More than 3 locations of sealant beyond 2 mm of the sealed areas.			Minor



#	DEFECT TYPE	ACCEPTANCE STANDARDS CRITERIA (Continued)	MAJOR/ MINOR
11	PCB defects	<ol style="list-style-type: none">1. Oxidation or contamination on connectors.*2. Wrong parts, missing parts, or parts not in specification.*3. Jumpers set incorrectly.4. Solder (if any) on bezel, LED pad, zebra pad, or screw hole pad is not smooth. <p><i>*Minor if display functions correctly. Major if the display fails.</i></p>	Minor
12	Soldering defects	<ol style="list-style-type: none">1. Unmelted solder paste.2. Cold solder joints, missing solder connections, or oxidation.*3. Solder bridges causing short circuits.*4. Solder balls. <p><i>*Minor if display functions correctly. Major if the display fails.</i></p>	Minor