Crystalfontz

INTELLIGENT DISPLAY MODULE SPECIFICATIONS

CFA631-TMF-KU & CFA631-RMF-KU

CFA631P-TMF-KU

Data Sheet Release 2014-11-17
for CFA631:
CFA631-TMF-KU
CFA631-RMF-KU
CFA631P-TMF-KU

Hardware Revision: 2h4
Firmware Revision: u3v1

Crystalfontz America, Incorporated

12412 East Saltese Avenue
Spokane Valley, WA 99216-0357

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

CONTENTS

CONTENTS, CONTINUED

CONTENTS, CONTINUED

FIGURES

FORWARD

REVISION INFORMATION

Revision History For CFA631 Data Sheet
Data Sheet Release: 2014-11-17 The following changes were made: ● Information for the new CFA631P-TMF-KU display module was added. ● Reference to the demonstration software “631_WinTest” was replaced by the more versatile demonstration software CF_Test. ● In the Buy Cables Separately (Pg. 25) table, approximate length of cables were changed to more accurate measurements. Also, the new long WR-PWR-Y44 ATX power cable was added. ● Wherever listed, references to USB driver download was updated. ● Added a new section, How to Set ATX Functionality Using cfTest (Pg. 33) that describes the required steps. ● In Reliability (Pg. 63), specification for CFA631-TMF-KU changed from “90%” to “70%”. ● In CARE AND HANDLING PRECAUTIONS (Pg. 64), a caution was added about handling shipping trays. The “How To Clean” description now provides more details. ● Removed “APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTERS (SCAB REQUIRED)” and “APPENDIX B: CONNECTING A DS1963S-F5+ SHA IBUTTON”. ● The Data Sheet was updated to meet current template standards. Changes include standardizing terms. For example, in Command Codes (Pg. 38), “display” replaced “LCD” and “CGROM” replaced “CGRAM”. Command descriptions have been clarified.
Data Sheet Release: 2012-10-19 Complete Data Sheet rewrite.
Data Sheet Release: 2008-10-06, v2.0a (version number did not change) Note added to correct specification of GPIO pull-up/pull-down mode resistance values from “approximately 5Ω” to “approximately 5kΩ”.

Revision History For CFA631 Data Sheet (Continued)

Data Sheet Release: 2005-12-20, v2.0a
The following changes were made to the datasheet:

- 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

Data Sheet Release: 2005-08-01, v2.0
The following changes were made to the datasheet:

- Start Public Version Tracking.
- 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.

CFA631 Hardware And Firmware Revisions

For information about firmware and hardware revisions for the CFA631, see Part Change Notifications under News on our website.

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

NOTICES

About Variations

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

About Volatility

The CFA631 has nonvolatile memory.

Additional Fine Print

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

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

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

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

INTRODUCTION

DESCRIPTION OF THE DIFFERENT CFA631 VARIANTS

The CFA631 has three variants: *CFA631-RMF-KU*, *CFA631-TMF-KU*, and *CFA631P-TMF-KU*. All variants have the same version of firmware and hardware. Except for the two dimensions that are affected by the mounting hardware (the overall depth and width), the dimensions for the *CFA631-TMF-KU* and *CFA631P-TMF-KU* are identical.

	2 Display Color Choices	
Part Number	 **CFA631-TMF-KU and CFA631P-TMF-KU**	 **CFA631-RMF-KU**
LED Backlight	*Display:* white, 4 on 1 edge *Keypad:* blue	*Display:* red, 11 on top and bottom *Keypad:* red
Fluid	STN	
Glass Color	blue	
Image	negative	
Polarizer Film	transmissive	
Viewing Angle	12 o'clock	
Negative Image: Display can be read in typical office lighting and in dark areas. May be difficult to read in direct sunlight. *Viewing Angle:* See Optical Characteristics (Pg. 22).		

2 Stainless Steel Built-In Bracket Choices	
 **CFA631-TMF-KU and CFA631-RMF-KU**	Easily slides into a 3.5" floppy drive bay.
 **CFA631P-TMF-KU**	Use this version to mount to a panel.

When the information in this Data Sheet applies to all three variants, the shorter term "CFA631" is used. When the information applies to the *CFA631-RMF-KU* and *CFA631-TMF-KU* variants but not the *CFA631P-TMF-KU*, the term "CFA631-***-KU" is used.

MAIN FEATURES OF ALL CFA631 VARIANTS

- ❑ Large easy-to-read display in a compact size can display 20 characters x 2 lines.
- ❑ Active Area is 63.55 (W) x 10.35 (H) millimeters.
- ❑ Display modules have a 12 o'clock viewing direction. See Optical Characteristics (Pg. 22).
- ❑ Temperature operating range is 0°C minimum to +50°C maximum.
- ❑ USB interface (factory default 115200 baud equivalent throughput).
- ❑ If your embedded controller or host system has a "real" RS232 serial port (-10v to +10v "full swing" serial interface, typically through a UART), contact Technical Support at support@crystalfontz.com to place a special order. For an additional fee, we will mount a CFA-RS232 Serial Converter Board.
- ❑ Integrated LED backlit 4-button translucent silicone keypad allows assignment of keys to be shown easily on the display. Fully decoded keypad: any key combination is valid and unique. See command 32: Key Legends (Pg. 57).
- ❑ Select from four colors of overlays.
- ❑ Backlight is fully voltage regulated over the power supply range. Adjustments to the backlight brightness can be made, although it is not necessary is most situations.
- ❑ The CFA631 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.
- ❑ 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 may be used with our optional SCAB (System Cooling Accessory Board) to add fan and temperature sensor and fan functions. See Additional Features When Used With Optional SCAB (System Cooling Accessory Board) (Pg. 10) below.
- ❑ Free downloadable sample code. See APPENDIX A: FREE DEMONSTRATION AND OTHER SOFTWARE) (Pg. 66).
- ❑ To download the most current Certificate of Compliance for ISO, RoHS, and REACH, go to the module's Datasheets & Files tab on the part number's website page.

ADDITIONAL FEATURES WHEN USED WITH OPTIONAL SCAB (SYSTEM COOLING ACCESSORY BOARD)

Figure 1. Optional SCAB Connected To CFA631 With WR-EXT-Y19 Extension Cable

To use all of the commands described in Command Codes (Pg. 38) for temperature monitoring and fan control, the optional SCAB (System Cooling Accessory Board) is required. You can add a SCAB to your CFA631 order using the "Customize and Add to Cart" feature on our website. If you add a SCAB, you will be prompted to add one WR-EXT-Y19 extension cable to your order, as well as the SCAB accessories described below.

As shown in the photo above, a SCAB can be conveniently mounted on the built-in drive bay bracket of the *CFA631-TMF-KU* and *CFA631-RMF-KU*. Or set up your own configuration to add a SCAB to the *CFA631P-TMF-KU*.

The combination of the CFA631 with the SCAB (written as "CFA631+SCAB" in this Data Sheet) allows:

- Add up to four functional fan connectors for tachometer speed 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. 52). Buy one 3-pin fan extension cable WR-FAN-X01 to connect each fan.
- Add up to 32 WR-DOW-Y17 temperature sensor cables that have Maxim DS18B20 Programmable Resolution 1-Wire temperature sensors. The DS18B20 has an operating temperature range of -55°C to +125°C and is accurate to ±0.5°C over the range of -10°C to +85°C.
- Instead of ATX power supply control functionality directly from the CFA631, buy the WR-PWR-Y14 ATX power cable for ATX power supply control functionality from the SCAB.

For more information, download the Data Sheet on the SCAB website page.

OTHER ACCESSORIES: KITS

To add an overlay to the built-in bracket's front plate, order a CFA631-***-KU through our Kit Configurator instead of using the "Customize and Add to Cart" tool. The Kit Configurator also offers various combinations of the optional SCAB and useful cables.

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. 25) or see a list of all cables on our website.

DB	631	**	-	***	-	K	U	#
❶		❷		❸				❹

❶	[type] **DB** – Built-in 3.5-inch floppy drive bay mounting bracket.
❷	[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
❸	[variant] Choice of two colors (variants): **RMF** – red characters on dark background **TMF** – light (near-white) characters on blue background
❹	[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

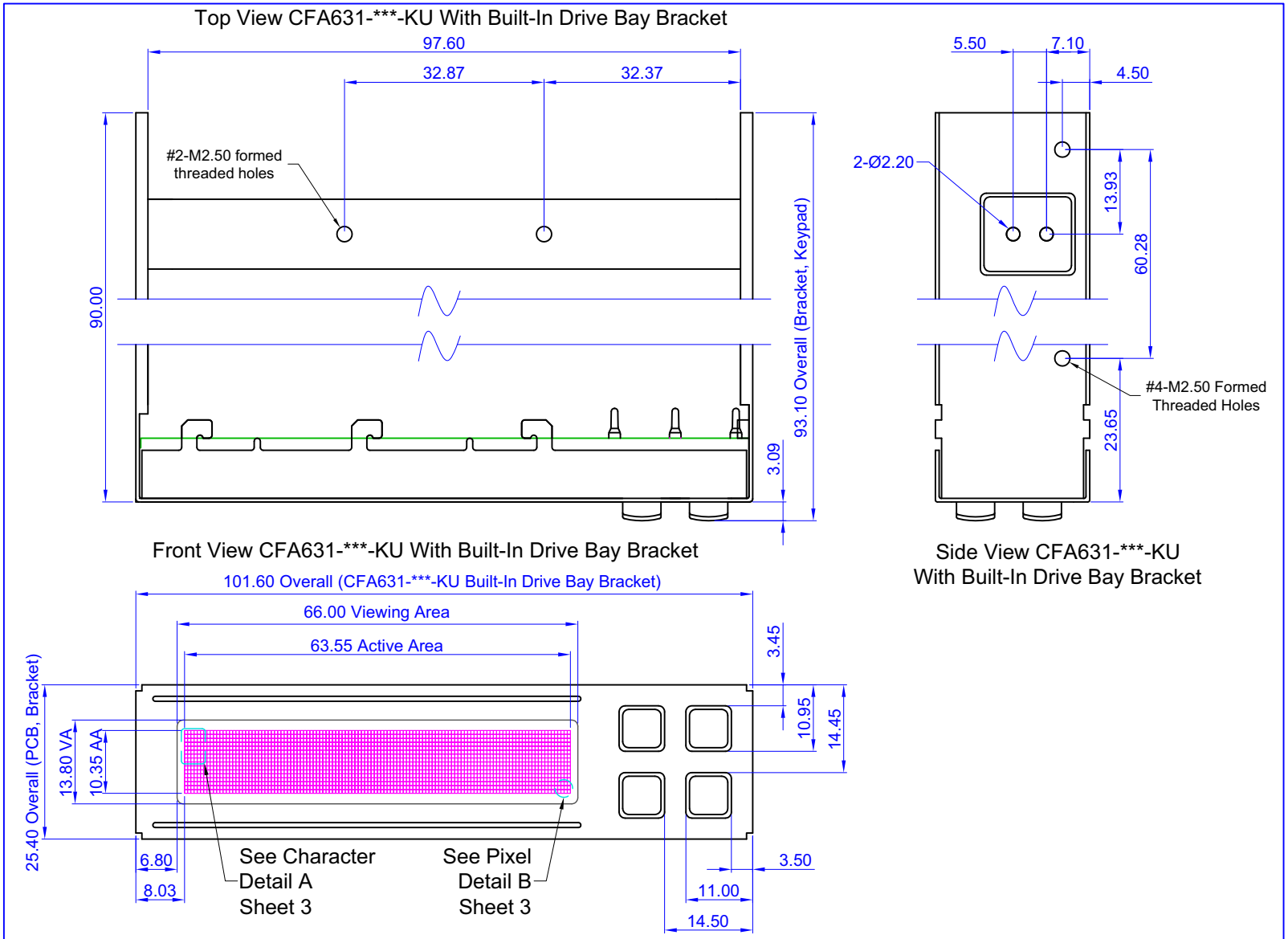
MECHANICAL CHARACTERISTICS

PHYSICAL CHARACTERISTICS

ITEM	SPECIFICATION
Display Module Overall Dimensions (includes built-in bracket)	
Width *CFA631-***-KU* *CFA631P-TMF-KU*	 101.60 (W) mm 120.40 (W) mm
Height	25.40 (H) mm
Depth *CFA631-***-KU* *CFA631P-TMF-KU*	 93.10 (D) mm (includes keypad) <18.00 (D) mm (excludes keypad)
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 *CFA631-***-KU* *CFA631P-TMF-KU*	 80 grams (typical) 53 grams (typical)

DISPLAY MODULE OUTLINE DRAWINGS

Figure 3. CFA631 With CFA631-***-KU Built-In 3.5-Inch Floppy Drive Bay Bracket

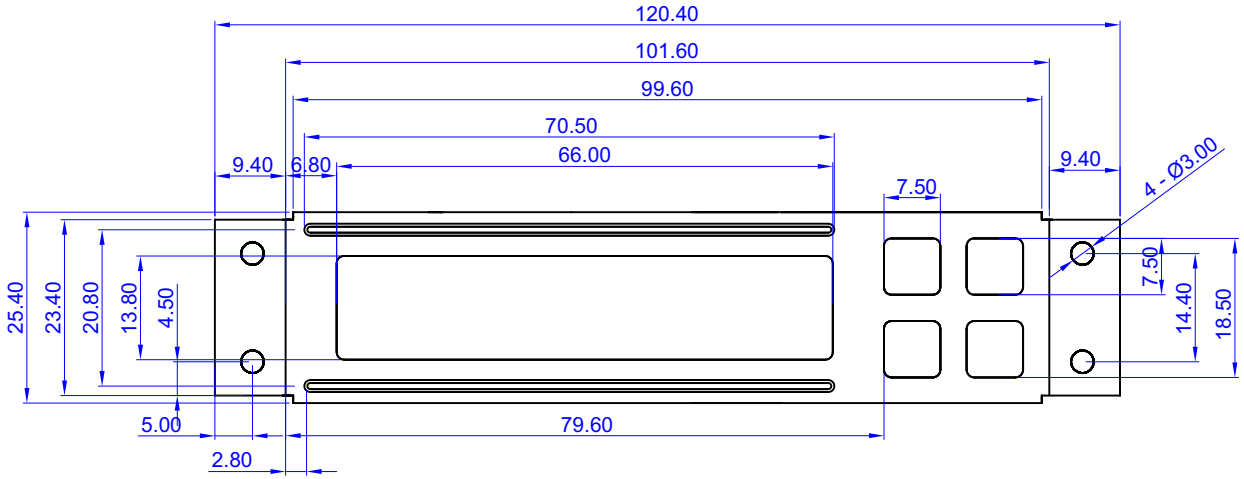


Note: CFA631-***-KU with built-in drive bay bracket dimensions are shown above. Bracket dimensions for the CFA631P-TMF-KU are on the sheet 2 illustration. All other dimensions are identical.

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

Part No.(s):	Scale:	Drawing Number:	Hardware Rev.:
CFA631	Not to scale	CFA631	2h4
	Units: Millimeters	Date: 2014-11-11	Sheet: 1 of 3

Front View Of CFA631P With Built-In Panel Mount Bracket



Side View Of CFA631P With Built-In Panel Mount Bracket

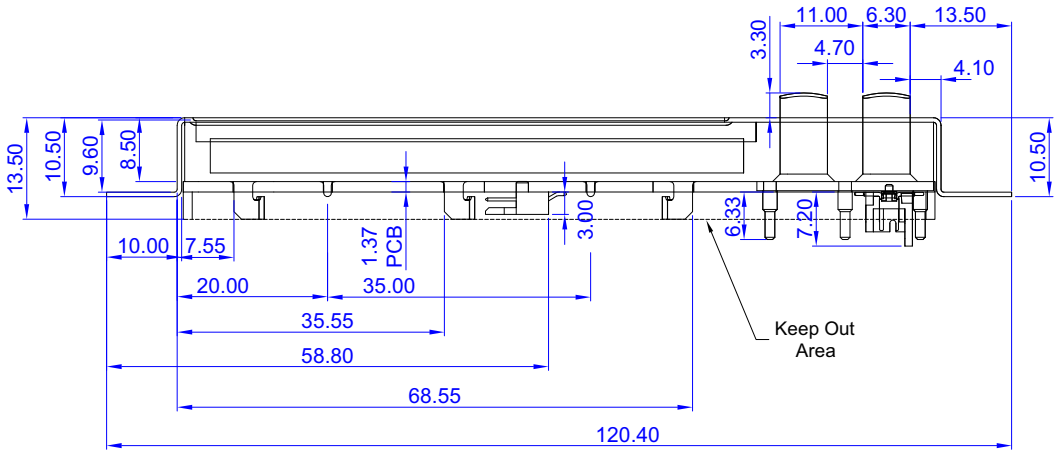
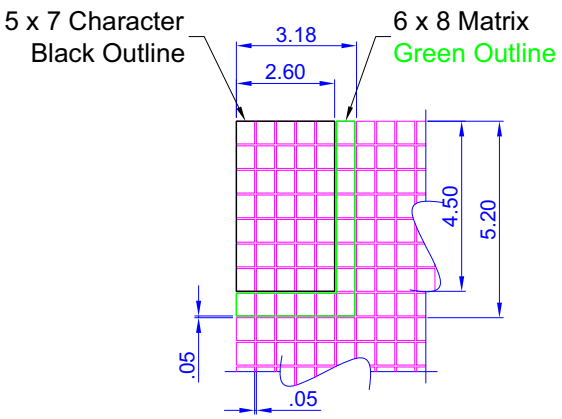
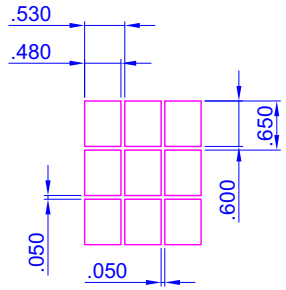


Figure 4. CFA631P-TMF-KU With Built-In Panel Mount Bracket

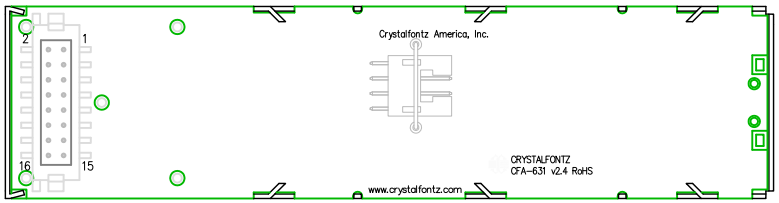
Part No.(s):	Scale:	Drawing Number:	Hardware Rev.:
CFA631P	Not to scale	CFA631	2h4
	Units: Millimeters	Date: 2014-11-17	Sheet: 2 of 3



Character Detail A



Pixel Detail B



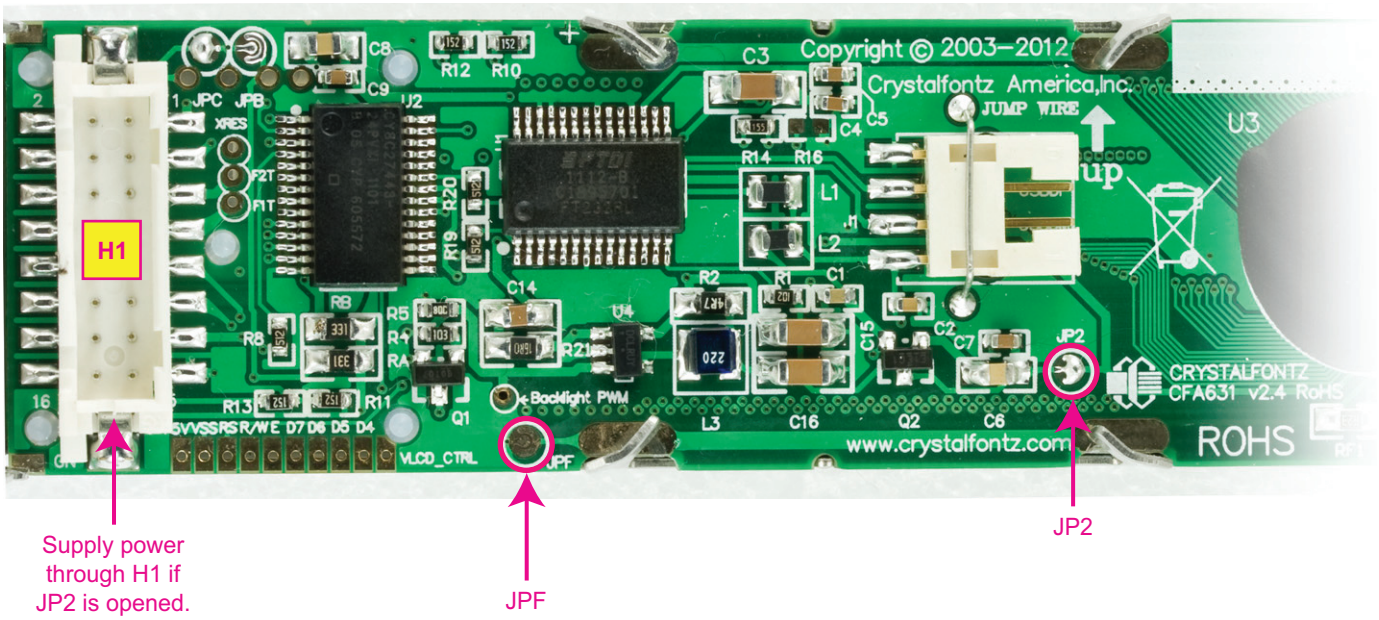
Back View

Figure 5. CFA631 Back View, Character Details, And Pixel Details

Part No.(s):	Scale:	Drawing Number:	Hardware Rev.:
CFA631 Back View. Character Details, & Pixel Detail	Not to scale	CFA631	2h4
	Units: Millimeters	Date: 2014-10-30	Sheet: 3 of 3

JUMPERS THAT CAN BE MODIFIED

The CFA631 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 13. on Pg. 28.) Open JP2 for ATX or when power is supplied over H1. We can do this for you when you order “Make Module ATX” or “Make module and SCAB ATX”. Click using the “Customize and Add to Cart” feature on the display module’s website page.

Figure 6. Location Of Jumpers That Can Be Modified

ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

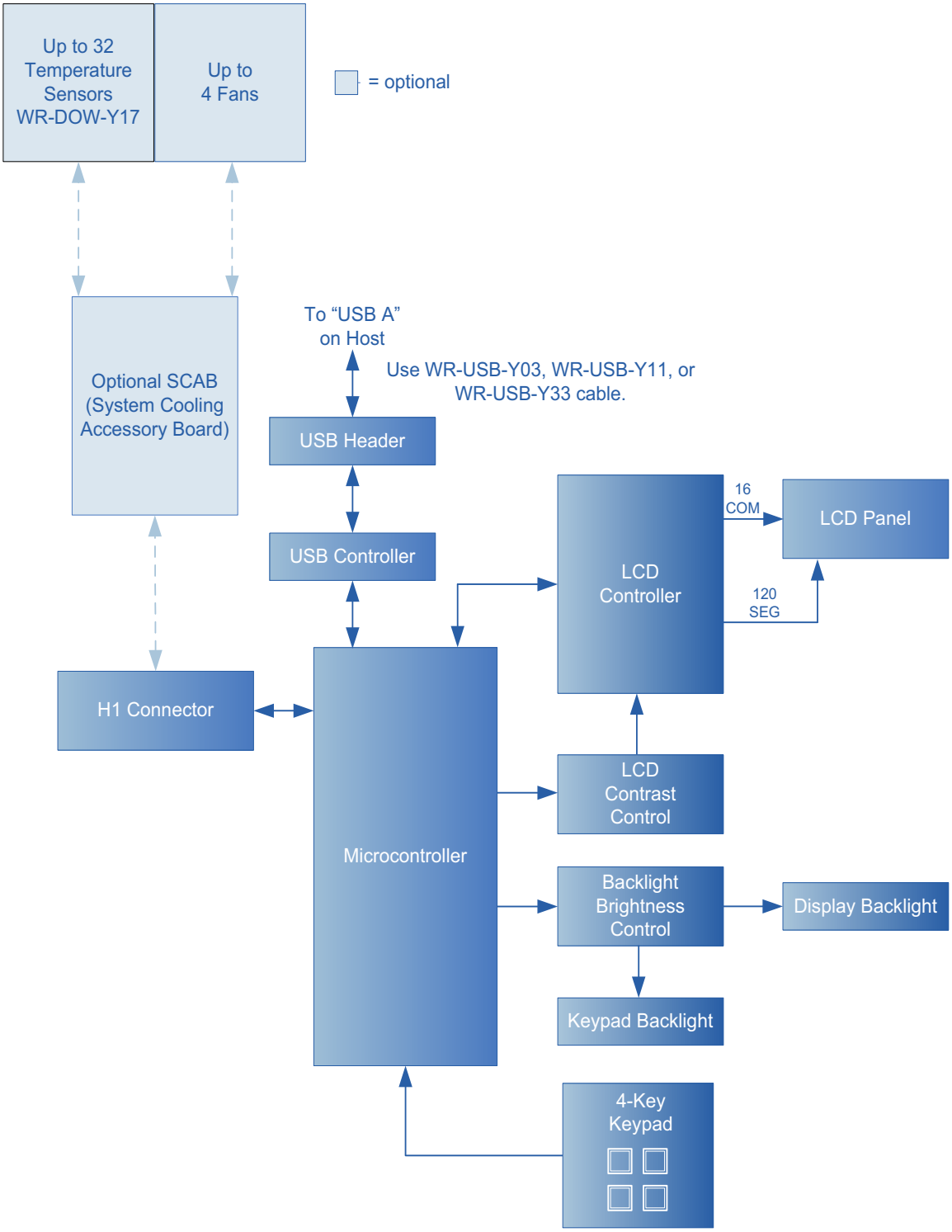


Figure 7. System Block Diagram

DUTY AND BIAS

DRIVING METHOD	SPECIFICATION
Duty[1]	1/32
Bias[2]	6.7

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

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

ABSOLUTE MAXIMUM RATINGS

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

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 display module at these conditions beyond those listed under *Recommended DC Characteristics (Pg. 19)* *is not implied.*

Changes in temperature can result in changes in contrast.

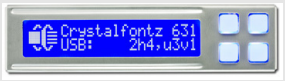
RECOMMENDED DC CHARACTERISTICS

	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}

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

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 display module.

 **CFA631-TMF-KU & CFA631P-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
Luminous Intensity Through Panel CFA631-TMF-KU and CFA631P-TMF-KU CFA631P-TMF-KU	 TBD cd/m^2 TBD cd/m^2
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. See Additional Features When Used With Optional SCAB (System Cooling Accessory Board) (Pg. 10).

[2]*PWM* is *Pulse Width Modulation*. PWM is a way to simulate intermediate levels by switching a level between full on and full off. PWM can be used to control the brightness of LED backlights, relying on the natural averaging done by the human eye, as well as for controlling fan power.

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

OPTICAL SPECIFICATIONS

OPTICAL CHARACTERISTICS

ITEM	SYMBOL	CONDITION	MINIMUM	TYPICAL	MAXIMUM
Test Condition for all: T=25°					
Viewing Angle	Deg θ = 0°	(12 o'clock) CR≥2		40	
	Deg θ = 90°			30	
	Deg θ = 180°			45	
	Deg θ = 270°			30	
Contrast Ratio[1]	CR	θ=ψ= 0		≥5	
LCD Response Time[2,3]	T rise		100 ms	150 ms	200 ms
	T fall		100 ms	150 ms	200 ms

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

TEST CONDITIONS AND DEFINITIONS FOR OPTICAL CHARACTERISTICS

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

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

Definition Of Optimal Contrast Setting

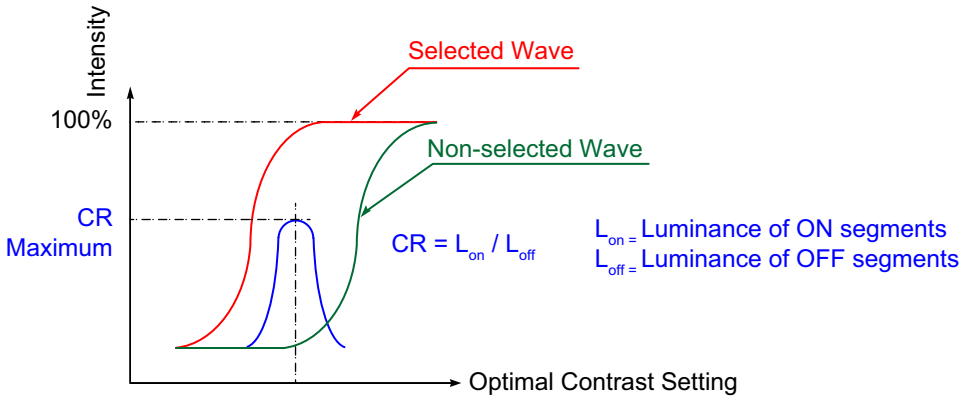


Figure 8. Definition Of Optimal Contrast Setting (Negative Image)

Definition Of Response Time (Tr, Tf)

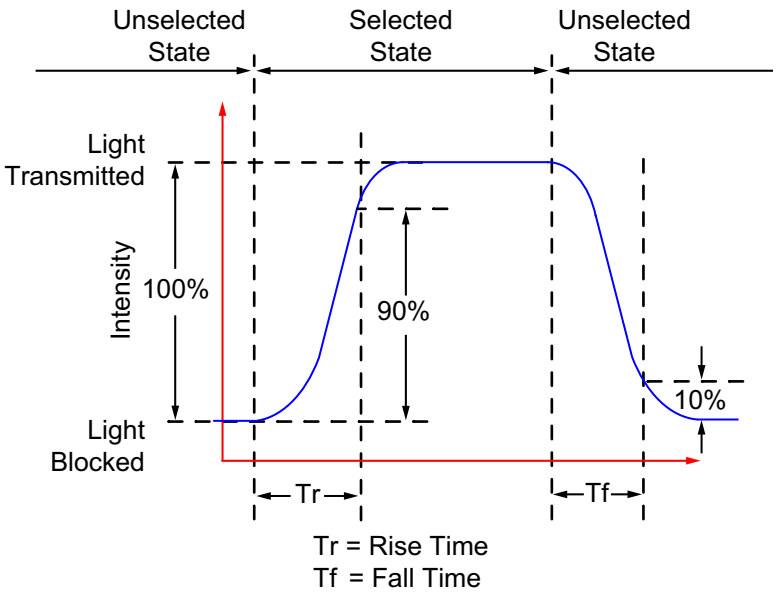


Figure 9. Definition Of Response Time (Tr, Tf) (Negative Image)

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

These modules have a 12:00 o'clock viewing angle.

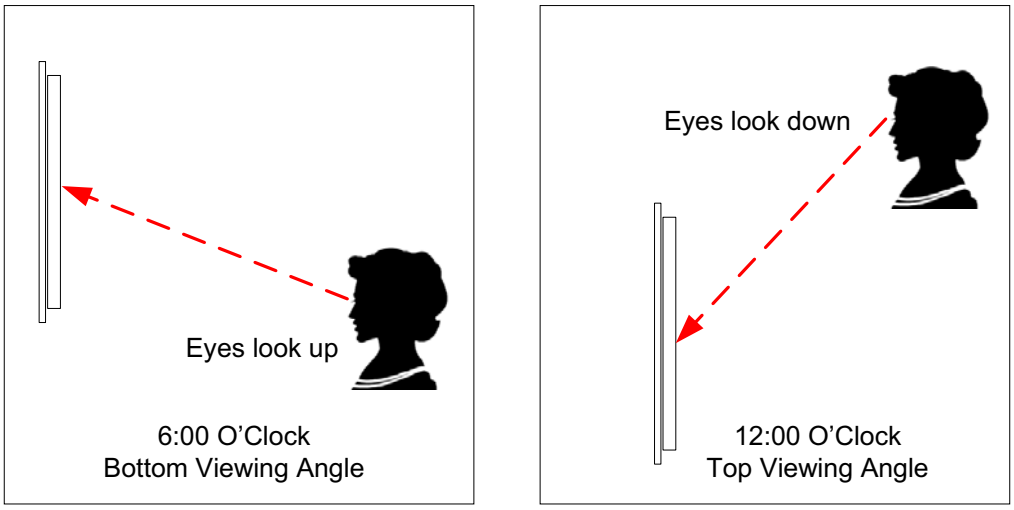


Figure 10. Definition of 6:00 O'clock and 12:00 O'clock Viewing Angles

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

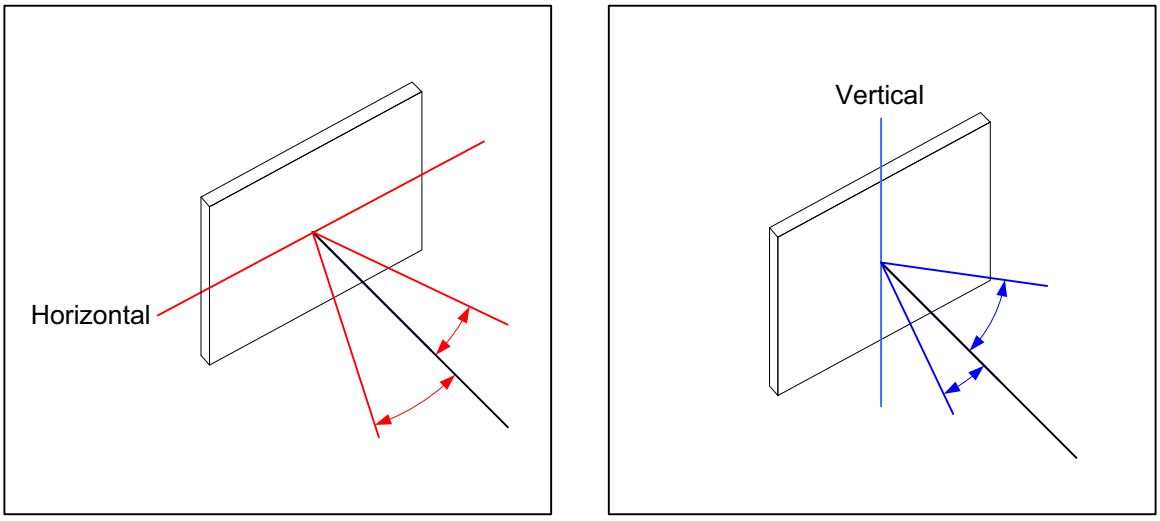


Figure 11. Definition Of Horizontal And Vertical Viewing Angles (CR>2)

LED BACKLIGHT INFORMATION

Note
For *CFA631-TMF-KU* and *CFA631P-TMF-KU* with **white** backlights, we recommend that the display be dimmed or turned off during periods of inactivity to conserve the LEDs' lifetime.

CONNECTION INFORMATION

BUY CABLES SEPARATELY

When you order a CFA631 through our website, you are offered a choice of cables to add to your order through our "Customize and Add to Cart" feature. Additional cables are on our website here. Following the table below are descriptions of common connection configurations. Cable lengths are approximate.

Part Number	Cable Descriptions All Cables are RoHS Compliant
USB Cables	
Note: The CFA631 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 ft. 4.35 inches	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 display 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 ~2 ft. 6 inches	Connect the cable's 2 mm female USB connector to the display 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 ~2 ft. 3.15 inches	Connect the cable's smaller 2 mm female USB connector to the display 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. *For correct orientation, note the +5v location on the 4-pin connector.*
WR-PWR-Y24 ~2 ft. 1.95 inches	Add this cable for powering the display module separately from USB. Connect the cable's 16-pin female connector to the display module's 16-pin male H1 connector. Connect the cable's 4-pin male connector to the host's power supply. Note: Open JP2 to avoid back-powering USB.
Cables for ATX Functionality (Power Off, Power On, & Reset) Without Optional SCAB (System Cooling Accessory Board)	
WR-PWR-Y25 ~11 inches	Use this ATX power cable to turn an ATX power supply on and off, or power cycle the host through the CFA631. Connect the cable's 16-pin female connector to the CFA631'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.)
Cables For Optional SCAB (System Cooling Accessory Board)	
Note: The CFA631 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.	

Part Number	Cable Descriptions (Continued) All Cables are RoHS Compliant
WR-PWR-Y12 ~1 ft. 0.55 in inches	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.
WR-PWR-Y14 ~1 ft. 11 inches	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 display module's connector H1.
WR-PWR-Y44 ~3 ft. 3 inches	This cable has the same connectors as the WR-PWR-Y14 **ATX** cable listed immediately above. It can be used with a rack mount chassis where additional length is needed.
WR-EXT-Y15 ~1 ft. 5.70 inches	Use this cable to mount the SCAB some distance away from the display module. For example, the SCAB could be mounted in a central location within the host's case to the display module mounted in a drive bay or on the panel. 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 display module is mounted. Connect one of the cable's two 16-pin female connectors to the display 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.
WR-EXT-Y19 ~3.5 inches	Use this short cable when the SCAB is mounted directly to the CFA631-***-KU built-in bracket. Connect one of the cable's two 16-pin female connectors to the display 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.
WR-FAN-X01 ~1 ft. 4.30 inches	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.)
WR-DOW-Y17 ~12 inches + ~12 inches between connectors	Connect ("daisy chain") up to 32 of these DOW 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.
UBERSCAB Kit (System Cooling Accessory Board + Cables)	
The SCAB requires external power,. 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 display module (WR-EXT-19), and one 16-inch cable to connect SCAB to the display module (WR-EXT-Y15).	

USB CONNECTION TO HOST

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

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

- The WR-USB-Y03 (~6 ft. 4.35 inches) 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'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 (~2 ft. 6 inches) 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 (~2 ft. 3.15 inches) Connect the cable's smaller 2 mm female USB connector to the CFA631'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.

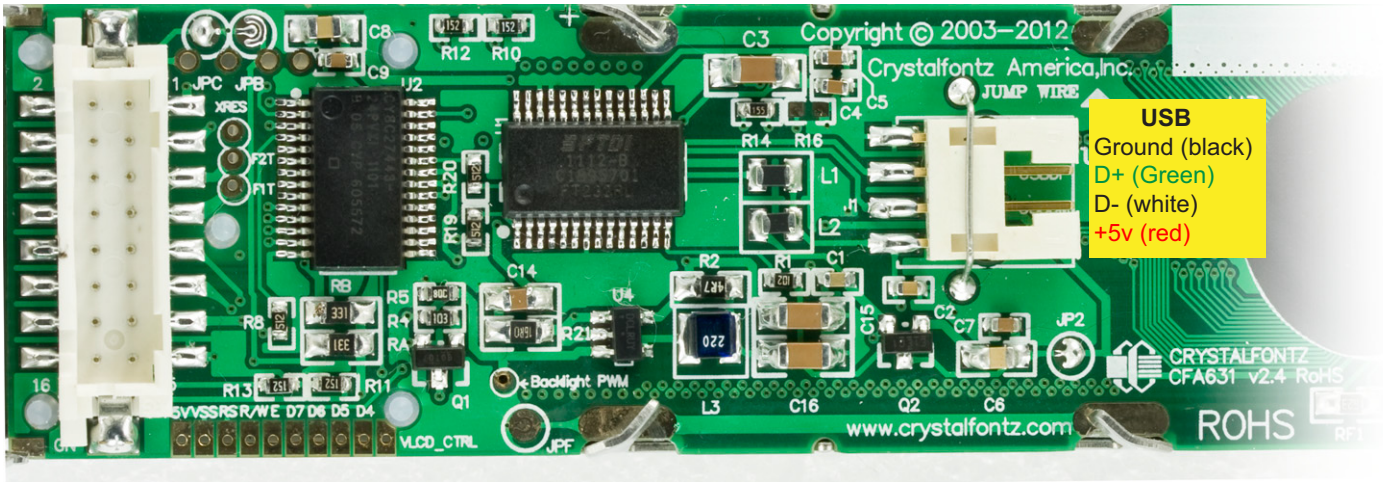


Figure 12. USB Connector Pins Labeled

If you would like to make your own cable, the USB connector on the CFA631 is:
FCI/Berg 95000-004LF: 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/product/USBLCDDRIVER. If you do Windows updates on your PC, Windows USB drivers are automatically included.

H1 CONNECTOR PIN ASSIGNMENTS - INCLUDES FIVE GPIOS

CFA631 has five GPIOs available on connector H1. These GPIOs can be accessed directly through H1 or through the optional SCAB (System Cooling Accessory Board) when it is connected to H1.

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

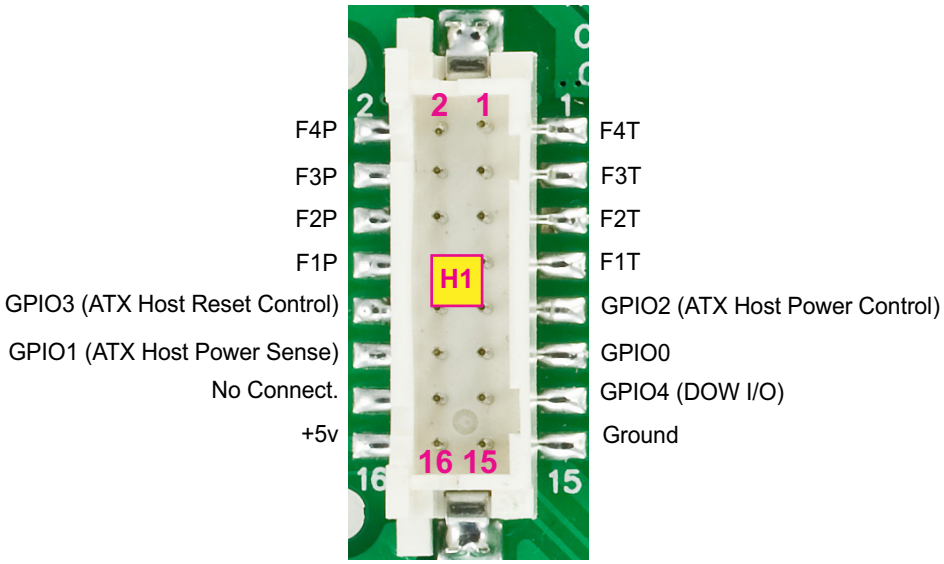


Figure 13. Location Of GPIO Pins On H1 Connector

Please see the commands 34 (0x22): GPIO Settings (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. 25).

ATX POWER SUPPLY

ATX Power And Control Connections

- ATX power supply control functionality allows the buttons on the CFA631 to replace the power and reset button on your system, simplifying front panel design. This ATX power supply control functionality can be accomplished with the optional SCAB+WR-PWR-Y14 ATX power cable or use the WR-PWR-Y25 or WR-PWR-Y38 ATX power cable without the SCAB. The SCAB provides fan monitoring and control as well as DOW temperature sensor monitoring.

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

When configuring the CFA631 for ATX functionality, **open jumper JP2** in order to ensure correct operation. See Jumpers That Can Be Modified (Pg. 16). This is required whether the optional SCAB is or is not used.

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

GPIO[2] ATX Host Power Control
The motherboard's power switch input is connected to GPIO[2]. This GPIO line functions as POWER CONTROL. The POWER CONTROL pin is configured as a high impedance input until the display 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. 54).

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 display module wants to RESET the host. Then it will change momentarily

to low impedance output, driving either low or high depending on the setting of RESET_INVERT. See command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54). This connection is also used for the hardware watchdog.

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 in system control and monitoring. Buy the WR-EXT-Y15 or WR-EXT-Y19 to connect the SCAB to the CFA631's connector H1.

Note

If the Crystalfontz WR-PWR-Y14 cable and SCAB are ordered at the same time as the CFA631 through "Customize and Add to Cart" feature on the display module's website page, Crystalfontz will open JP2 on the CFA631, open JP8 and close JP9 on the SCAB, and send the following software configuration commands.

Once these changes are made, for the CFA631 to power up, power must be applied to the 7-pin header on the SCAB as well as the 4-pin power header. If you do not want these jumper changes when you order a CFA631 and SCAB, please write a note in the Special Instructions box.

```
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 display 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.

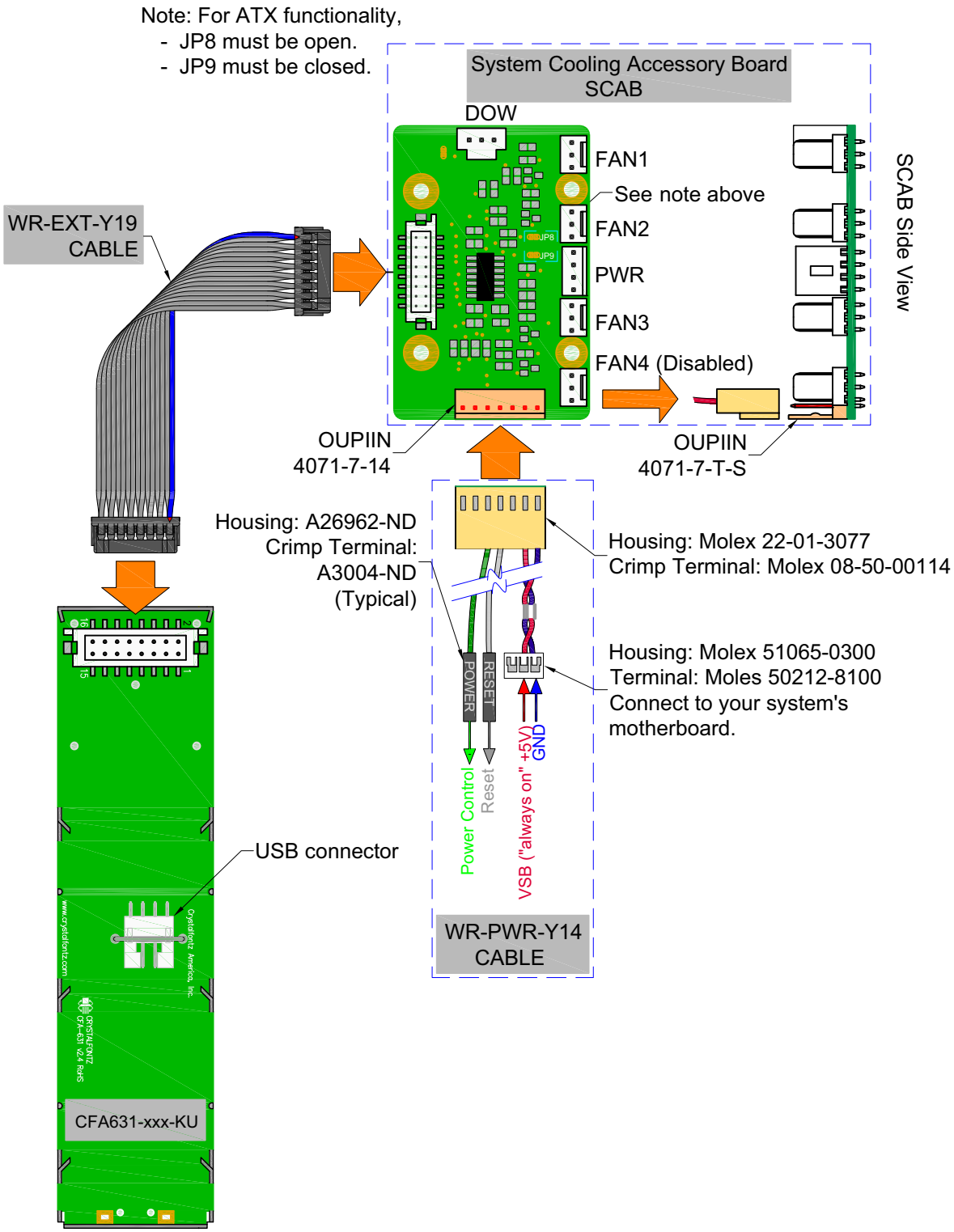


Figure 14. ATX Connection With Optional SCAB Using WR-PWR-Y14 ATX Cable

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 's H1 connector.

Note

If the Crystalfontz WR-PWR-Y25 cable is ordered at the same time as the display module through our Customize and Add to Cart button on the display 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 display 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 WR-PWR-Y25 cable connects to the CFA631's H1 connector and your system's motherboard and ATX power supply:

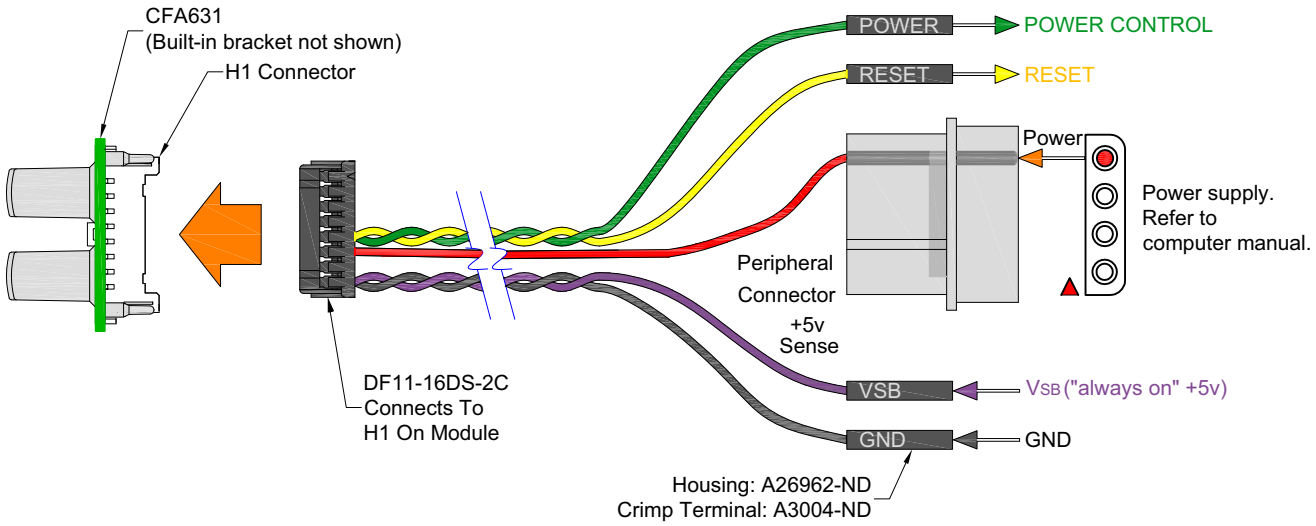


Figure 15. ATX Power Supply And Control Connections Using WR-PWR-Y25 ATX Cable

HOW TO SET ATX FUNCTIONALITY USING CFTEST

1. Download the cfTest application here: http://www.crystalfontz.com/software/CFTEST.html.
2. Connect the CFA631 to a Windows' based PC. You may want to connect the +5VSB and +5VSENSE so you will be able to see the CFA631 when it powers up.
3. Disable any applications that communicate with the CFA631 to free up the virtual COM port.
4. Launch cfTest. The application should automatically recognize the CFA631 and display it in the *Communications Port* dropdown list. If not, select your CFA631 from the dropdown list.
5. In the *Send Packet* section, select command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54) from the dropdown list.
6. Type in the following value: "\240" into the *Data* field. The '\240" represents the bitmask value for data[0].
7. Click *Send Packet*.
8. Select command 4 (0x04): Store Current State As Boot State (Pg. 39) from *The PacketType* dropdown list.
9. Clear the *Data* text box.
10. Click *Send Packet*. This saves the current state set with ATX.

HOW TO CONNECT THE OPTIONAL SCAB

The optional SCAB is designed to connect to a CFA631's H1 connector. The SCAB will receive the correct signals to operate from the display module.

Here is a photo showing the CFA631-***-KU connected to the optional SCAB using the WR-EXT-Y19 cable:

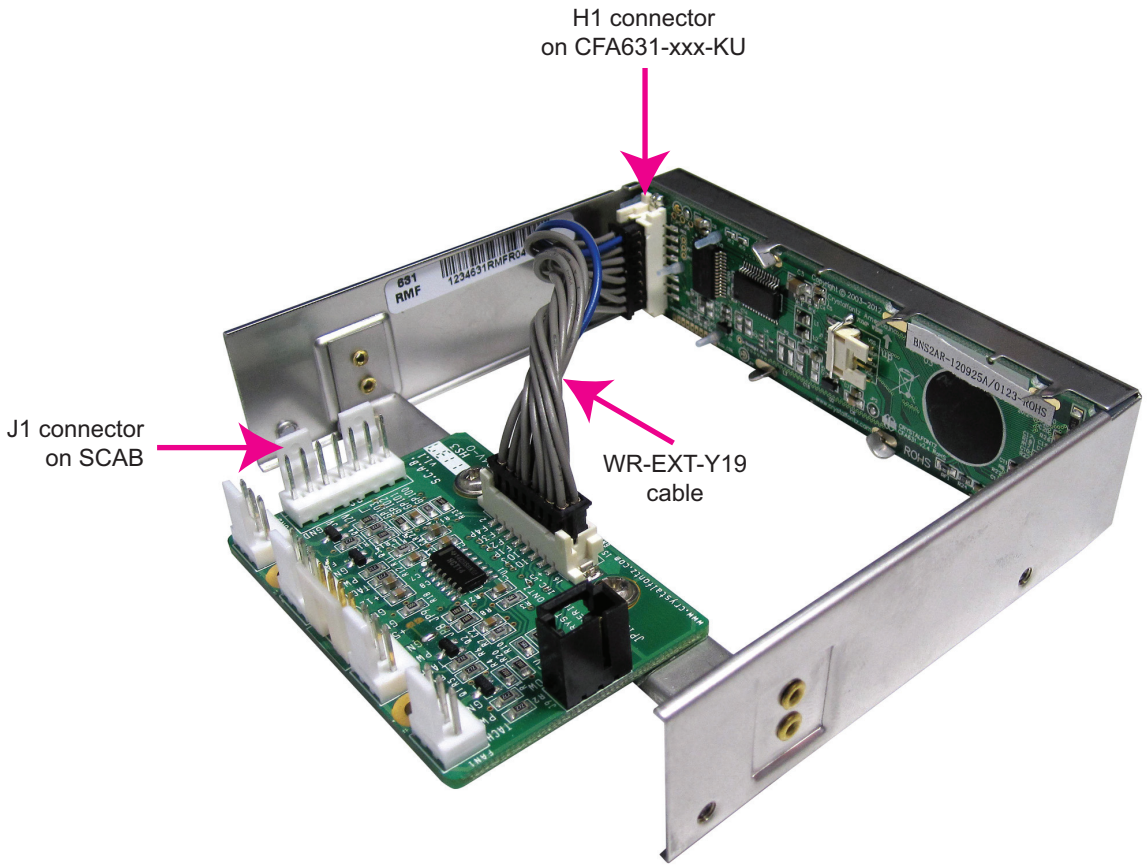


Figure 16. CFA631-***-KU Connected To Optional SCAB Using WR-EXT-Y19 Cable

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

1. WR-EXT-Y15 SCAB cable (~16-inch)
   This cable allows the SCAB to be mounted some distance away from the CFA631. 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 display 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 is mounted.
2. WR-EXT-Y19 SCAB cable (~3.5-inch)
   This cable can used when the SCAB is mounted close to the CFA631, as is the case when the SCAB is fastened directly to the CFA631-***-KU's built-in drive bay bracket. (See the photo above.)

HOST COMMUNICATIONS

Note:
Where "*CFA631 with ATX"* is described, you can use any of these: WR-PWR-Y25 ATX power cable, WR-PWR-Y38 power cable, or SCAB+WR-PWR-Y14 ATX power cable

CFA631 communicates with its host using the USB interface.The easiest and most common way for the host software to access the USB is through the Crystalfontz virtual COM port (VCP) drivers. Several versions of Microsoft signed drivers and MacIntosh drivers can be downloaded here: www.crystalfontz.com/product/USBLCDDRIVER. If you do Windows updates on your PC, Windows USB drivers are automatically included. Using these drivers makes it appear to the host software as if there is an additional serial port (the VCP) on the host system when the CFA631 is connected. This VCP should be opened at 115200 baud, 8 data bits, no parity, 1 stop bit.

PACKET STRUCTURE

All communication between the CFA631 and the host takes place in the form of a simple and robust CRC checked packet. The packet format allows for very reliable communications between the CFA631 and the host without the traditional problems that occur in a stream-based serial communication (such as having to send data in inefficient ASCII format, to "escape" certain "control characters", or losing sync if a character is corrupted, missing, or inserted).

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

All packets have the following structure:

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

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

```
TTcc cccc
|||| ||||--Command, response, error or report code 0-63
||---------Type:
             00 = normal command from host to CFA631
             01 = normal response from CFA631 to host
             10 = normal report from CFA631 to host (not in
                  direct response to a command from the host)
             11 = error response from CFA631 to host (a packet
                  with valid structure but illegal content
                  was received by the CFA631)
```

data_length specifies the number of bytes that will follow in the data field. The valid range of **data_length** is 0 to 22.

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 Sample Algorithms To Calculate The CRC (Pg. 66) 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, cfTest for Windows. cfTest allows you to experiment with the command set described below.

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 be a challenge.

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

REPORT CODES

The 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. 51).

```
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. 46).

```
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+SCAB will send Temperature Sensor Reports for each selected sensor every second. See the command 19 (0x13): Set Up WR-DOW-Y17 Temperature Reporting (SCAB Required) (Pg. 47).

```
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. The commands are in numerical order, with command 15 intentionally left out.

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

Used to verify communication with the CFA631. The CFA631 will echo the Ping Command to the host.

```
type: 0x00 = 0(10)
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 = 64(10)
data_length = (identical to received packet)
data[0-(data_length-1)] = (identical to received packet)
```

1 (0x01): Get Hardware And Firmware Version

The CFA631 will return the hardware and firmware version information to the host.

```
type: 0x01 = 1(10)
valid data_length is 0
```

The return packet will be:

```
type: 0x40 | 0x01 = 0x41 = 65(10)
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 = 2(10)
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 = 66(10)
data_length = 0
```

3 (0x03): Read User Flash Area

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

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

The return packet will be:

```
type: 0x40 | 0x03 = 0x43 = 67(10)
data_length = 16
data[] = 16 bytes user data recalled from the 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” bootscreen when power is applied. This command can be used to customize the “welcome” screen, as well as the following items:

- Characters shown on display, which are affected by:
  - Command 6 (0x06): Clear Display (Pg. 43).
  - Command 7 (0x07): Set Display Contents, Line 1 (CFA633 Compatible) (Pg. 43).
  - Command 8 (0x08): Set Display Contents, Line 2 (CFA633 Compatible) (Pg. 43).
  - Command 31 (0x1F): Send Data To Display (Pg. 57).
- Special character font definitions (command 9 (0x09): Set Display Special Character Data (Pg. 44)).
- Cursor position (command 11 (0x0B): Set Display Cursor Position (Pg. 44)).
- Cursor style (command 12 (0x0C): Set Display Cursor Style (Pg. 45)).
- Contrast setting (command 13 (0x0D): Set Display Contrast (Pg. 45)).
- Backlight setting (command 14 (0x0E): Set Display And Keypad Backlights (Pg. 45)).
- Fan power settings (command 17 (0x11): Set Fan Power (SCAB Required) (Pg. 46)).
- Settings of any “live” displays (command 21 (0x15): Set Up Live Fan Or Temperature Display (SCAB Required) (Pg. 49)).
- Key press and release masks (command 23 (0x17): Configure Key Reporting (Pg. 51)).
- Fan glitch delay settings (command 26 (0x1A): Set Fan Tachometer Glitch Delay (SCAB Required) (Pg. 52)).
- ATX function enable and pulse length settings (command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54)).
- Key legends (command) 32: Key Legends (Pg. 57).
- Baud rate (command 33 (0x21): Set Baud Rate (Pg. 58)).
- GPIO settings (command 34 (0x22): GPIO Settings (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 = 4(10)
valid data_length is 0
```

The return packet will be:

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

If the current state and the boot state do not match after saving, the display 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 ATX, WR-PWR-Y25, WR-PWR-Y38 ATX power cable or the optional SCAB+WR-PWR-Y14 ATX power cable is required.

This command instructs the 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 **P**ower **I**nterface) 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): GPIO Settings (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 = 5(10)
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. 46) and save this setting as the default boot state (command 4 (0x04): Store Current State As Boot State (Pg. 39)). 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. 54), 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 display module will reboot itself. The display 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 display 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. 54), 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 = 69(10)
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. 54), send the following packet:

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

Note

The CFA631 will return the acknowledge packet immediately, then reset the host. After resetting the host (~1.5 seconds), the display module will reboot itself. The display 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 display module will not be of consequence.

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

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

Note

The 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 display module will reboot itself. The display 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 display module will not be of consequence.

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

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

6 (0x06): Clear Display

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

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

The return packet will be:

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

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

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

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

> Note
> This command allows legacy software that displays data on older CFA633 display modules to work unchanged on the CFA631. For new applications, please use the more flexible command 31 (0x1F): Send Data To Display.

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

The return packet will be:

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

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

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

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

> Note
> This command allows legacy software that displays data on older CFA633 display modules to work unchanged on the CFA631. For new applications, please use the more flexible command 31 (0x1F): Send Data To Display.

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

The return packet will be:

```
type: 0x40 | 0x08 = 0x48 = 72(10)
data_length = 0
```

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

9 (0x09): Set Display Special Character Data

Sets the font definition for one of the special characters (CGROM). (See CHARACTER GENERATOR ROM (CGROM) (Pg. 62)

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

`data[1-8]` are the bitmap information for this character. Any value is valid between 0 and 63, the msb is at the left of the character cell of the row, and the lsb is at the right of the character cell. 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 = 73(10)
data_length = 0
```

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

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

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

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

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

```
0x40 (\064) to 0x7F (\127) for CGROM
 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 = 74(10)
data_length = 9
```

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

11 (0x0B): Set Display Cursor Position

This command allows the cursor to be placed at the desired location on the CFA631's display. If you want the cursor to be visible, you may also need to send a command 12 (0x0C): Set Display Cursor Style (Pg. 45).

```
type: 0x0B = 11(10)
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 = 75(10)
data_length = 0
```

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

12 (0x0C): Set Display Cursor Style

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

```
type: 0x0C = 12(10)
valid data_length is 1
data[0] = cursor style (0-4 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 = 76(10)
data_length = 0
```

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

13 (0x0D): Set Display Contrast

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

```
type: 0x0D = 13(10)
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 = 77(10)
data_length = 0
```

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

14 (0x0E): Set Display And Keypad Backlights

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

```
type: 0x0E = 14(10)
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 = 78(10)
data_length = 0
```

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

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

This command will configure the CFA631+SCAB to report the fan speed information to the host every 500 mS.

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

The return packet will be:

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

If `data[0]` is not 0, then the CFA631 will start sending 0x81: Fan Speed Report packets for each enabled fan every 500 mS. (See 0x81: Fan Speed Report (SCAB Required) (Pg. 36).) Each of the report packets is staggered by 1/8 of a second.

Reporting a fan will override the fan power setting to 100% for up to 1/8 of a second every 1/2 second. Please see "Fan Connections" in the SCAB Data Sheet for a detailed description.

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

This command will configure the power settings for the fan connectors on the SCAB.

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

The return packet will be:

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

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

18 (0x12): Read 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 DOW bus and stores the device's information. This command will allow the host to read the device's information.

Note

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

These settings are factory default but may be changed by the user. Please see command 34 (0x22): GPIO Settings (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 = 18(10)
valid data_length is 1
data[0] = device index (0-31 valid)
```

The return packet will be:

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

If data[1] is 0x22 (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. 47).

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 = 19(10)
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 = disables
|  |-----------------------  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. 47).

The return packet will be:

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

```
type: 0x14 = 20(10)
valid data_length is 2 to 16
data[0] = device_index (0-32 valid)
data[1] = number_of_bytes_to_read (0-14 valid) 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 = 84(10)
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 display 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 display without host intervention. The Set Up Live Fan or Temperature Display is one of the items stored by command 4 (0x04): Store Current State As Boot State (Pg. 39). 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 display, as long as all the digits of that slot fall fully within the display area. It is legal to have the display area of one slot overlap the display area of another slot, but senseless. This situation should be avoided in order to have meaningful information displayed.

```
type: 0x15 = 21(10)
valid data_length is 7 or 2 (for turning a slot off)
data[0]: display slot (0-3)
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 = 85(10)
data_length = 0
```

22 (0x16): Send Command Directly To The Display Controller

This command allows you to access the CFA631’s display’s 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 = 22(10)
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 = 86(10)
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 = 23(10)
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 = 87(10)
data_length = 0
```

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

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the CFA631 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. 51). 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 = 24(10)
data_length = 0
```

The return packet will be:

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

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

The CFA631+SCAB can be used as part of an active cooling system. The fans can be slowed down to reduce noise when a system is idle or when the ambient temperature is low. The fans can be sped up when the system is under heavy load or the ambient temperature is high.

Since there are a large number of ways to control the speed of the fans (thresholds, thermostat, proportional, PID, multiple temperature sensors contributing to the speed of several fans . . .) there was no way to foresee the particular requirements of your system and include an algorithm in the 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 = 25(10)
data_length = 2
data[0] = bitmask of fans set to fail-safe (1-15 valid)data[1] = timeout value in 1/8 sec-
ond ticks:
       1 = 1/8 second
       2 = 1/4 second
     255 = 31 7/8 seconds
```

The return packet will be:

```
type = 0x40 | 0x19 = 0x59 = 89(10)
data_length = 0
```

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

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 µS long (1/100 of one period of the 18 Hz PWM repetition rate).

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

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

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

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

The return packet will be:

```
type = 0x40 | 0x1A = 0x5A = 90(10)
data_length = 0
```

27 (0x1B): Query Fan Power And 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 = 27(10)
data_length = 0
```

The return packet will be:

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

28 (0x1C): Set ATX Power Switch Functionality

For ATX, WR-PWR-Y25, WR-PWR-Y38 ATX power cable or the optional SCAB+WR-PWR-Y14 ATX power cable is required.

The combination of the 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. 39)

See How to Set ATX Functionality Using cfTest (Pg. 33) for detailed steps.

Note

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

These settings are factory default but may be changed by the user. Please see command 34 (0x22): GPIO Settings (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, holding 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 = 28[10]
data_length: 1 or 2
data[0]: bitmask of enabled functions
data[1]: (optional) length of power on & off pulses in 1/32 second
       1 = 1/32 sec
       2 = 1/16 sec
      16 = 1/2 sec
     254 = 7.9 seconds
     255 = Assert power control line until host power state changes
```

The return packet will be:

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

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

Some 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. 54) or command 34 (0x22): GPIO Settings (SCAB Required) (Pg. 58).

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

If timeout is 0, the watchdog is disabled.

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

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

If the command is not re-issued within timeout seconds, then the CFA631 with ATX will
reset the host (see command 28 (0x1C): Set ATX Power Switch Functionality (Pg. 54) 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 = 93(10)
data_length = 0
```

30: (0x1E) Read Reporting And 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 = 30(10)
data_length = 0
```

The return packet will be:

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

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

31 (0x1F): Send Data To Display

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

```
type = 0x1F = 31(10)
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 display, variable from 1 to 20 characters
```

The return packet will be:

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

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

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 = 32(10)
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. 39).

33 (0x21): Set Baud Rate

After sending this command, the host should wait for a positive acknowledgment from the CFA631 at the old baud rate. The host can then begin communicating at the new baud rate.

The baud rate must be saved by command 4 (0x04): Store Current State As Boot State (Pg. 39) if you want the CFA631 to power-up/restart using the new baud rate. The factory default baud rate is 115200.

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

The return packet will be:

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

34 (0x22): GPIO Settings (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.

The default GPIO mode uses PWM and a suitable current limiting resistor to control the LEDs on the front of the display module. They can be turned on and off and even dimmed under host software control. With suitable external circuitry, the GPIOs can also be used to drive external logic or power transistors.

> 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 function to function properly,** Our free demonstration software cfTest may be used to easily check and reset the GPIO configuration to the default state so the ATX and DOW functions will work.

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

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

data[0]: index of GPIO to modify on optional SCAB's connector when using CFA631+SCAB+WR-
PWR-Y14
       0 = GPIO[0] = J8, Pin 11
       1 = GPIO[1] = J8, Pin 12 (default is ATX Host Power Sense)
       2 = GPIO[2] = J8, Pin 9 (default is ATX Host Power Control)
       3 = GPIO[3] = J8, Pin 10 (default is ATX Host Reset Control)
       4 = GPIO[4] = J9, Pin 13 (default is DOW I/O--has 1kΩ hardware pull-up)
   5-255 = not accessible

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

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

```
data[2] = Pin function select and drive mode (optional, 0-15 valid)
  ---- 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 = 98(10)

data_length = 0
```

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

Please see command 34 (0x22): GPIO Settings (SCAB Required) (Pg. 58) for details on the GPIO architecture.

```
type: 0x23 = 35(10)
data_length: 1
data[0]: index of GPIO to query
      0 = GPIO[0] = J8, Pin 11
      1 = GPIO[1] = J8, Pin 12 (default is ATX Host Power Sense--has series R8 of 5.6kΩ)
      2 = GPIO[2] = J8, Pin 9 (default is ATX Host Power Control)
      3 = GPIO[3] = J8, Pin 10 (default is ATX Host Reset Control)
      4 = GPIO[4] = J9, Pin 13 (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 display 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 = 99(10)
data_length = 4


returns:
  data[0] = index of GPIO read

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

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

  data[3] = Pin function select and drive mode
    ---- FDDD
    |||| ||||-- DDD = Drive Mode
    |||| |       ========================================================
    |||| |       000: 1=Fast, Strong Drive Up, 0=Resistive Pull Down
    |||| |       001: 1=Fast, Strong Drive Up, 0=Fast, Strong Drive Down
    |||| |       010: Hi-Z, use for input
    |||| |       011: 1=Resistive Pull Up,     0=Fast, Strong Drive Down
    |||| |       100: 1=Slow, Strong Drive Up, 0=Hi-Z
    |||| |       101: 1=Slow, Strong Drive Up, 0=Slow, Strong Drive Down
    |||| |       110: reserved
    |||| |       111: 1=Hi-Z,                  0=Slow, Strong Drive Down
    |||| |
    |||| |----- F = Function
    ||||         ========================================================
    ||||         0: Port unused for GPIO. It will take on the default
    ||||            function such as ATX, DOW or unused. The user is
    ||||            responsible for setting the drive to the correct
    ||||            value in order for the default function to work
    ||||            correctly.
    ||||         1: Port used for GPIO under user control. The user is
    ||||            responsible for setting the drive to the correct
    ||||            value in order for the desired GPIO mode to work
    ||||            correctly.
    ||||------- reserved, will return 0
```

CHARACTER GENERATOR ROM (CGROM)

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

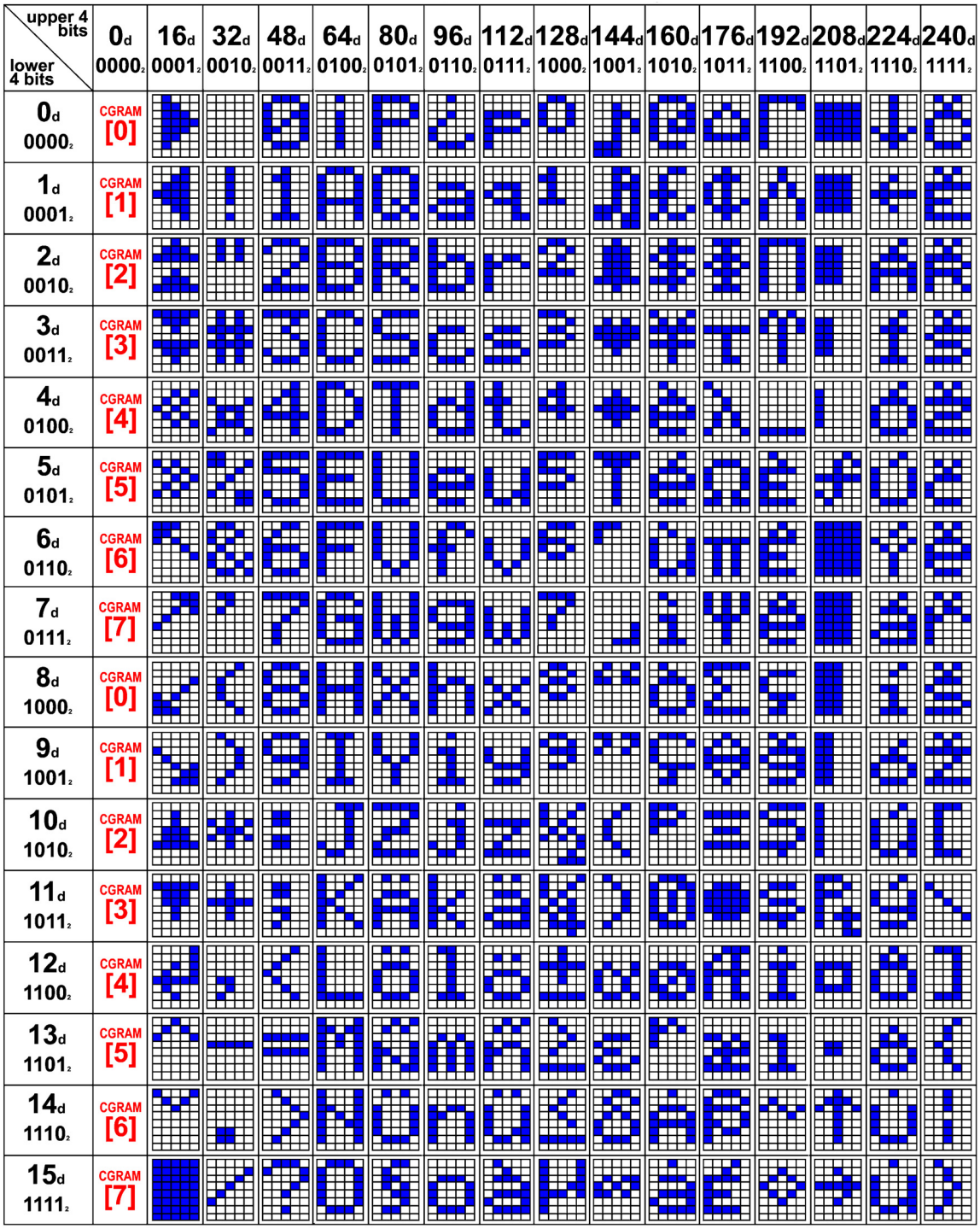


Figure 17. Character Generated ROM

RELIABILITY AND LONGEVITY

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

RELIABILITY

<table>
<tr><th colspan="2">ITEM</th><th colspan="2">RELIABILITY SPECIFICATION</th></tr>
<tr><td colspan="2">Display portion (excluding keypad, status LEDs, and backlights)</td><td colspan="2">50,000 to 100,000 hours</td></tr>
<tr><td colspan="2">Keypad</td><td colspan="2">1,000,000 keystrokes</td></tr>
<tr><td rowspan="3">  </td><td rowspan="3">CFA631-TMF-KU & CFA631P-TMF-KU (white LED display backlight and blue LED keypad backlight)</td><td>Power-On Hours</td><td>% of Initial Brightness (New)</td></tr>
<tr><td><10,000</td><td>>70%</td></tr>
<tr><td><50,000</td><td>>50%</td></tr>
<tr><td>  </td><td>CFA631-RMF-KU (red LED display backlight and keypad backlight)</td><td colspan="2">50,000 to 100,000 hours</td></tr>
<tr><td colspan="4">Note: For display modules with white LED backlights (CFA631-TMF-KU and CFA631P-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.

Under operating and storage temperature specification limitations, humidity noncondensing RH up to 65%, and no exposure to direct sunlight. Value listed above are approximate and represent typical lifetime.</td></tr>
</table>

LONGEVITY (EOL / REPLACEMENT POLICY)

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

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

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

Although the replacement display module is still within the stated Data Sheet specifications and tolerances of the discontinued display 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.* Display module components have manufacturing tolerances. In extreme cases, the tolerance stack can change the visual or operating characteristics.

Please understand that we avoid changing a display module whenever possible; we only discontinue a display module if we have no other option. We post Part Change Notices (PCN) on the product's website page as soon as possible.

CARE AND HANDLING PRECAUTIONS

Caution

Excessive voltage will shorten the life of the display module. You must drive the display within the specified voltage limit. See Absolute Maximum Ratings (Pg. 18).

HANDLING CAUTION: DISPLAY MODULES SHIPPED IN TRAYS

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

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 display module ships with a protective film over the polarizer. Please peel off the protective film slowly. Peeling off the protective film abruptly may generate static electricity.
- The polarizer is made out of soft plastic and is easily scratched or damaged. When handling the display module, avoid touching the polarizer. Finger oils are difficult to remove.
- Place a transparent plate (for example, acrylic, polycarbonate, or glass) in front of the display module, leaving a small gap between the plate and the display surface. We recommend HP-92 Lexan, which is readily available and works well.
- Do not disassemble or modify the display 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 display module.

AVOID SHOCK, IMPACT, TORQUE, OR TENSION

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

CAUTION

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

HOW TO CLEAN

1. Turn display module off.
2. Use the removable protective film to remove smudges (for example, fingerprints) and any foreign matter. If you no longer have the protective film, use standard transparent office tape (for example, Scotch® brand “Crystal Clear Tape”).
3. If the polarizer is dusty, you may carefully blow it off with clean, dry, oil-free compressed air.
4. If you must clean with a liquid, never use glass cleaners, as they may contain ammonia or alcohol that will damage the polarizer over time. Never apply liquids directly on the polarizer. Long contact with moisture may permanently spot or stain the polarizer. Use filtered water to slightly moisten a clean lint-free microfiber cloth designed for cleaning optics. (For example, use a cloth sold for cleaning plastic eyeglasses.)
5. The plastic is easily scratched or damaged. Use a light touch as you clean the polarizer. Wipe gently.
6. Use a dry microfiber cloth to remove any trace of moisture before turning on the display module.
7. Gently wash the microfiber cloths in warm, soapy water and air dry before reuse.

OPERATION

- Your circuit should be designed to protect the display module from ESD and power supply transients.
- Observe the operating temperature limitations: a minimum of 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 display modules with white LEDs (*CFA631-TMF-KU* and *CFA631P-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.

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 display modules while they are in storage.
- To discard, please recycle your display modules at an approved facility.

APPENDIX A: FREE DEMONSTRATION AND OTHER SOFTWARE)

DRIVERS

- Several versions of Microsoft signed drivers and MacIntosh drivers can be downloaded here: www.crystalfontz.com/product/USBLCDDRIVER. If you do Windows updates on your PC, Windows USB drivers are automatically included.
- 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

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

cfTest

cfTest for Windows is testing and configuration software that works on all Crystalfontz Intelligent Display Modules. This software allows you to experiment with the command set for the CFA631.

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

CrystalControl2 (CC2)

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

Linux CLI Examples

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

LCDproc is an open source project that supports many of the Crystalfontz displays.

SAMPLE 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_Er-
rata.zip
//
//I doubt that there are any worries about the legality of this code,
//searching for the first line of the table below, it appears that
//the code is already included in the linux 2.6 kernel "Driver for
//ST5481 USB ISDN modem". This is an "industry standard" algorithm
//and I do not think there are ANY issues with it at all.
typedef unsigned char ubyte;
typedef unsigned short word;
word get_crc(ubyte *bufptr,word len)
  {
  //CRC lookup table to avoid bit-shifting loops.
  static const word crcLookupTable[256] =
    {0x00000,0x01189,0x02312,0x0329B,0x04624,0x057AD,0x06536,0x074BF,
     0x08C48,0x09DC1,0x0AF5A,0x0BED3,0x0CA6C,0x0DBE5,0x0E97E,0x0F8F7,
     0x01081,0x00108,0x03393,0x0221A,0x056A5,0x0472C,0x075B7,0x0643E,
     0x09CC9,0x08D40,0x0BFDB,0x0AE52,0x0DAED,0x0CB64,0x0F9FF,0x0E876,
     0x02102,0x0308B,0x00210,0x01399,0x06726,0x076AF,0x04434,0x055BD,
     0x0AD4A,0x0BCC3,0x08E58,0x09FD1,0x0EB6E,0x0FAE7,0x0C87C,0x0D9F5,
     0x03183,0x0200A,0x01291,0x00318,0x077A7,0x0662E,0x054B5,0x0453C,
     0x0BDCB,0x0AC42,0x09ED9,0x08F50,0x0FBEF,0x0EA66,0x0D8FD,0x0C974,
     0x04204,0x0538D,0x06116,0x0709F,0x00420,0x015A9,0x02732,0x036BB,
     0x0CE4C,0x0DFC5,0x0ED5E,0x0FCD7,0x08868,0x099E1,0x0AB7A,0x0BAF3,
     0x05285,0x0430C,0x07197,0x0601E,0x014A1,0x00528,0x037B3,0x0263A,
     0x0DECD,0x0CF44,0x0FDDF,0x0EC56,0x098E9,0x08960,0x0BBFB,0x0AA72,
     0x06306,0x0728F,0x04014,0x0519D,0x02522,0x034AB,0x00630,0x017B9,
     0x0EF4E,0x0FEC7,0x0CC5C,0x0DDD5,0x0A96A,0x0B8E3,0x08A78,0x09BF1,
     0x07387,0x0620E,0x05095,0x0411C,0x035A3,0x0242A,0x016B1,0x00738,
     0x0FFCF,0x0EE46,0x0DCDD,0x0CD54,0x0B9EB,0x0A862,0x09AF9,0x08B70,
     0x08408,0x09581,0x0A71A,0x0B693,0x0C22C,0x0D3A5,0x0E13E,0x0F0B7,
     0x00840,0x019C9,0x02B52,0x03ADB,0x04E64,0x05FED,0x06D76,0x07CFF,
     0x09489,0x08500,0x0B79B,0x0A612,0x0D2AD,0x0C324,0x0F1BF,0x0E036,
     0x018C1,0x00948,0x03BD3,0x02A5A,0x05EE5,0x04F6C,0x07DF7,0x06C7E,
     0x0A50A,0x0B483,0x08618,0x09791,0x0E32E,0x0F2A7,0x0C03C,0x0D1B5,
     0x02942,0x038CB,0x00A50,0x01BD9,0x06F66,0x07EEF,0x04C74,0x05DFD,
     0x0B58B,0x0A402,0x09699,0x08710,0x0F3AF,0x0E226,0x0D0BD,0x0C134,
     0x039C3,0x0284A,0x01AD1,0x00B58,0x07FE7,0x06E6E,0x05CF5,0x04D7C,
     0x0C60C,0x0D785,0x0E51E,0x0F497,0x08028,0x091A1,0x0A33A,0x0B2B3,
     0x04A44,0x05BCD,0x06956,0x078DF,0x00C60,0x01DE9,0x02F72,0x03EFB,
     0x0D68D,0x0C704,0x0F59F,0x0E416,0x090A9,0x08120,0x0B3BB,0x0A232,
     0x05AC5,0x04B4C,0x079D7,0x0685E,0x01CE1,0x00D68,0x03FF3,0x02E7A,
     0x0E70E,0x0F687,0x0C41C,0x0D595,0x0A12A,0x0B0A3,0x08238,0x093B1,
     0x06B46,0x07ACF,0x04854,0x059DD,0x02D62,0x03CEB,0x00E70,0x01FF9,
     0x0F78F,0x0E606,0x0D49D,0x0C514,0x0B1AB,0x0A022,0x092B9,0x08330,
     0x07BC7,0x06A4E,0x058D5,0x0495C,0x03DE3,0x02C6A,0x01EF1,0x00F78};

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

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

Algorithm 2: "C" Bit Shift Implementation

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

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

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

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

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

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

Algorithm 2B: "C" Improved Bit Shift Implementation

This is a simplified algorithm that implements the CRC.

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

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

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

Algorithm 3: "PIC Assembly" Bit Shift Implementation

This routine was graciously donated by one of our customers.

```
;=====================================================================
; 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
savchr      equ     46h         ; BYTE - temp storage for CRC routine
```

```
;
seedlo      equ     021h        ; initial seed for CRC reg lo byte
seedhi      equ     0F3h        ; initial seed for CRC reg hi byte
;
polyL       equ     008h        ; polynomial low byte
polyH       equ     084h        ; polynomial high byte
;=====================================================================
;  CRC Test Program
;---------------------------------------------------------------------
        org         0           ; reset vector = 0000H
;
        clrf        PCLATH      ; ensure upper bits of PC are cleared
        clrf        STATUS      ; ensure page bits are cleared
        goto        main        ; jump to start of program
;
; ISR Vector
;
        org         4           ; start of ISR
        goto        $           ; jump to ISR when coded
;
        org         20          ; start of main program
main
        movlw       seedhi      ; setup intial CRC seed value.
        movwf       accumh      ; This must be done prior to
        movlw       seedlo      ; sending string to CRC routine.
        movwf       accuml      ;
        clrf        index       ; clear string read variables
;
main1
        movlw       HIGH InputStr   ; point to LCD test string
        movwf       PCLATH      ; latch into PCL
        movfw       index       ; get index
        call        InputStr    ; get character
        movwf       Zero        ; setup for terminator test
        movf        Zero,f      ; see if terminator
        btfsc       STATUS,Z    ; skip if not terminator
         goto         main2        ; else terminator reached, jump out of loop
        call        CRC16       ; calculate new crc
        call        SENDUART    ; send data to LCD
        incf        index,f     ; bump index
        goto        main1       ; loop
;
main2
        movlw       00h         ; shift accumulator 16 more bits.
        call        CRC16       ; This must be done after sending
        movlw       00h         ; string to CRC routine.
        call        CRC16       ;
;
        comf        accumh,f    ; invert result
        comf        accuml,f    ;
;
        movfw       accuml      ; get CRC low byte
        call        SENDUART    ; send to LCD
        movfw       accumh      ; get CRC hi byte
        call        SENDUART    ; send to LCD
;
stop    goto        stop           ; word result of 0x93FA is in accumh/accuml
;=====================================================================
; calculate CRC of input byte
;---------------------------------------------------------------------
CRC16
        movwf       savchr      ; save the input character
        movwf       datareg     ; load data register
        movlw       .8          ; setup number of bits to test
        movwf       j           ; save to incrementor
_loop
        clrc                    ; clear carry for CRC register shift
         rrf        datareg,f   ; perform shift of data into CRC register
```

```
        rrf         accumh,f    ;
        rrf         accuml,f    ;
        btfss       STATUS,C    ; skip jump if if carry
        goto        _notset     ; otherwise goto next bit
        movlw       polyL       ; XOR poly mask with CRC register
        xorwf       accuml,F    ;
        movlw       polyH       ;
        xorwf       accumh,F    ;
_notset
        decfsz      j,F         ; decrement bit counter
        goto        _loop       ; loop if not complete
        movfw       savchr      ; restore the input character
        return                  ; return to calling routine
;=====================================================================
; USER SUPPLIED Serial port transmit routine
;---------------------------------------------------------------------
SENDUART
        return                  ; put serial xmit routine here
;=====================================================================
; test string storage
;---------------------------------------------------------------------
        org     0100h
;
InputStr
        addwf   PCL,f
        dt      7h,10h,"This is a test. ",0
;
;=====================================================================
        end
```

Algorithm 4: “Visual Basic” Table Implementation

Visual BASIC has its own challenges as a language (such as initializing static arrays), and it is also challenging to use Visual BASIC to work with “binary” (arbitrary length character data possibly containing nulls—such as the “data” portion of the 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 = 0x0FFFF;
    for (int i = 0; i < data.length; i++ )
      {
      int lookup = crcLookupTable[(newCrc ^ data[i]) & 0xFF];
      newCrc = (newCrc >> 8) ^ lookup;
      }
    return(~newCrc);
    }
  }
```

Algorithm 6: “Perl” Table Implementation

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

```
#!/usr/bin/perl

use strict;

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

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

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

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

my $valid_crc = '5584' ;

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

my $crc = 0xFFFF ;

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

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

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

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

Algorithm 7: For PIC18F8722 or PIC18F2685

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

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

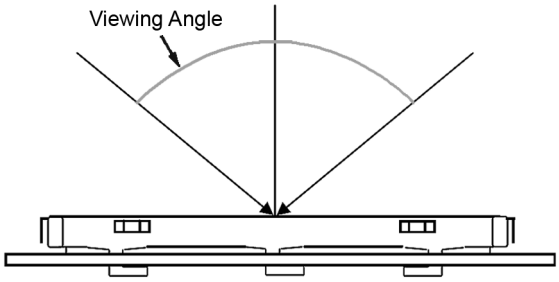
```
        bcf     STATUS,C        ; clear carry for CRC register shift
        rrcf    dsplyCRCData,f  ; perform shift of data into CRC
                                ;register
        rrcf    dsplyCRC,F
        rrcf    dsplyCRC+1,F
        btfss   STATUS,C        ; skip jump if carry
        goto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

```
        movlw   0xF3
        movwf   dsplyCRC        ; seed hi for CRC calculation
        movlw   0x21
        movwf   dsplyCRC+1      ; seed lo for CRC calculation
        LFSR    FSR1,TXBUF2     ; point at transmit buffer
        movlw   0x1F            ; command to send data to LCD
        movwf   TXBUF2          ; insert command byte from us to
                                ; CFA-635
        BMD1    movlw .2
        ddwf    SendCount,w     ; + overhead
        call    ComputeCRC2     ; compute CRC of transmit message
        movf    dsplyCRC+1,w
        movwf   POSTINC1        ; append CRC byte
        movf    dsplyCRC,w
        movwf   POSTINC1        ; append CRC byte
        return
;=====================================================================
SendMsg:
        call    ReleaseFSR1
        LFSR    FSR0,TXBUF2
        movff   FSR0H,irptFSR0
        movff   FSR0L,irptFSR0+1
                                ; save interrupt use of FSR0
        movff   SendCount,TXBUSY2
        bsf     PIE2,TX2IE
                                ; set transmit interrupt enable
                                ; (bit 4)
        return
;=====================================================================
; macro to move string to transmit buffer
SHOW macro src, stringname
        call    src
        MOVLF   upper stringname, TBLPTRU
        MOVLF   high stringname, TBLPTRH
        MOVLF   low stringname, TBLPTRL
        call    MOVE_STR
        endm
;=====================================================================
MOVE_STR:
        tblrd   *+
        movf    TABLAT,w
        bz      ms1b
        movwf   POSTINC1
        incf    SendCount
        goto    MOVE_STR
ms1b:
        return
;=====================================================================
```

APPENDIX B: QUALITY ASSURANCE STANDARDS

INSPECTION CONDITIONS

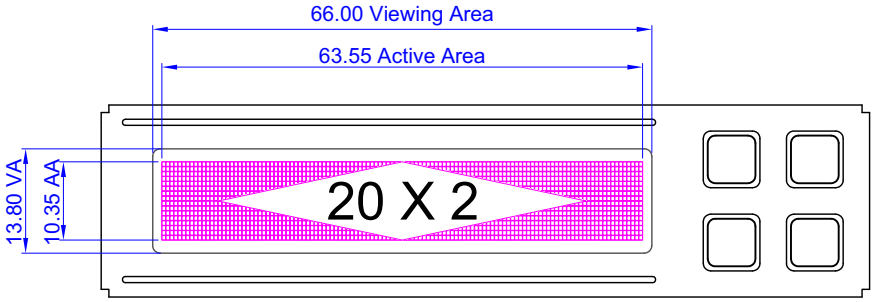
- Environment
  - Temperature: 25±5°C
  - Humidity: 30~85% RH
- For visual inspection of active display area
  - Source lighting: two 20 watt or one 40 watt fluorescent light
  - Display adjusted for best contrast
  - Viewing distance: 30±5 cm (about 12 inches)
  - Viewing angle: inspect at 45° angle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

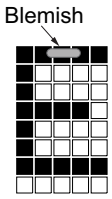
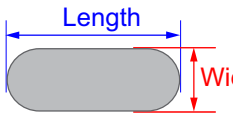
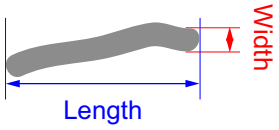
DEFECT TYPE	AQL*
Major	≤.65%
Minor	<1.0%
* \Acceptable Quality Level: maximum allowable error rate or variation from standard	

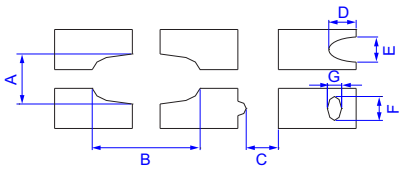
DEFECTS CLASSIFICATION

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. 78).			Major
3	Contrast adjustment defect	Contrast adjustment fails or malfunctions.			Major
4	Blemishes or foreign matter on display segments		*Defect Size (mm)*	*Acceptable Qty*	Minor
			≤0.3	3	
			≤2 defects within 10 mm of each other		
5	Other blemishes or foreign matter outside of display segments	Defect size = (A + B)/2 	*Defect Size (mm)*	*Acceptable Qty*	Minor
			≤0.15	Ignore	
			0.15 to 0.20	3	
			0.20 to 0.25	2	
			0.25 to 0.30	1	
6	Dark lines or scratches in display area 	*Defect Width (mm)*	*Defect Length (mm)*	*Acceptable Qty*	Minor
		≤0.03	≤3.0	3	
		0.03 to 0.05	≤2.0	2	
		0.05 to 0.08	≤2.0	1	
		0.08 to 0.10	≤3.0	0	
		≥0.10	>3.0	0	
7	Bubbles between polarizer film and glass		*Defect Size (mm)*	*Acceptable Qty*	Minor
			≤0.20	Ignore	
			0.20 to 0.40	3	
			0.40 to 0.60	2	
			≥0.60	0	

<table>
<tr><th>#</th><th>DEFECT TYPE</th><th>ACCEPTANCE STANDARDS CRITERIA (Continued)</th><th>MAJOR / MINOR</th></tr>
<tr><td>8</td><td>Display pattern defect</td><td> 

<table>
<tr><th>Dot Size (mm)</th><th>Acceptable Qty</th></tr>
<tr><td>((A+B)/2)≤0.2</td><td rowspan="4">≤3 total defects
≤2 pinholes per digit</td></tr>
<tr><td>C>0</td></tr>
<tr><td>((D+E)/2)≤0.25</td></tr>
<tr><td>((F+G)/2)≤0.25</td></tr>
</table></td><td>Minor</td></tr>
<tr><td>9</td><td>Backlight defects</td><td>1. Light fails or flickers.*
2. Color and luminance do not correspond to specifications.*
3. Exceeds standards for display’s blemishes or foreign matter (see test 5, Pg. 80), and dark lines or scratches (see test 6, Pg. 80).
*Minor if display functions correctly. Major if the display fails.</td><td>Minor</td></tr>
<tr><td>10</td><td>COB defects</td><td>1. Pinholes >0.2 mm.
2. Seal surface has pinholes through to the IC.
3. More than 3 locations of sealant beyond 2 mm of the sealed areas.</td><td>Minor</td></tr>
<tr><td>11</td><td>PCB defects</td><td>1. Oxidation or contamination on connectors.*
2. Wrong parts, missing parts, or parts not in specification.*
3. Jumpers set incorrectly.
4. Solder (if any) on bezel, LED pad, zebra pad, or screw hole pad is not smooth.
*Minor if display functions correctly. Major if the display fails.</td><td>Minor</td></tr>
<tr><td>12</td><td>Soldering defects</td><td>1. Unmelted solder paste.
2. Cold solder joints, missing solder connections, or oxidation.*
3. Solder bridges causing short circuits.*
4. Solder balls.
*Minor if display functions correctly. Major if the display fails.</td><td>Minor</td></tr>
</table>