



Crystalfontz America, Incorporated

INTELLIGENT USB LCD MODULE SPECIFICATIONS



Crystalfontz Model Number	CFA635-TMF-KU (Sold as part of a kit.)
Hardware Version	Revision v1.1, July 2008
Firmware Version	Revision v1.6, May 2010
Data Sheet Version	Revision 2.1, September 2010
Product Pages	www.crystalfontz.com/product/CFA635TMFKU.html

Crystalfontz America, Incorporated

12412 East Saltese Avenue
Spokane Valley, WA 99216-0357

Phone: 888-206-9720

Fax: 509-892-1203

Email: techinfo@crystalfontz.com

URL: www.crystalfontz.com



REVISION HISTORY

HARDWARE CFA635 LCD MODULE (CFA635-TMF-KU identical to serial version CFA635-TMF-KL)	
2010/06/01	Current Hardware Revision: v1.1 <ul style="list-style-type: none">● Effective June 2010, we are transitioning to an improved keypad from 10.5 mm to 12.00 mm high. See PCN 10282.● Factories have ISO certification.
2009/06/01	Hardware Revision: v1.1 Improved bezel. Changed from nickel plated bezel to stainless steel bezel.
2008/07/01	New Hardware Revision: v1.1 Improved backlight. Display is lit more evenly, contrast can be adjusted to higher, and lifetime is longer.
2005/08/01	Start Public Version Tracking. Hardware version: v1.0

FIRMWARE	
2010/05/24	Current Firmware Revision: v1.6 <ul style="list-style-type: none">● Command 1 (0x01): Get Hardware & Firmware Version (Pg. 32) returns: "CFA635:h1.1,v1.6".● Improved reset function to make module less sensitive to supply variations.● Fixed range checking for command 0 (0x00): Ping Command (Pg. 32).● Fixed parameter checking for command 20 (0x14): Arbitrary DOW Transaction (SCAB required) (Pg. 41).
2005/07/01	Start Public Version Tracking. Firmware Revision: v1.4 Command 1: Get Hardware & Firmware Version returns: "CFA635:h1.0,v1.4".



DATA SHEET	
2010/09/09	<p>Current Data Sheet Revision: v2.1: Changes since last revision (v2.0):</p> <ul style="list-style-type: none">● Wherever listed or shown, changed keypad dimensions from “10.5 mm” height to “12 mm” height and module overall height nominal from “19.3 mm” to “20.80 mm”. We started this gradual transition in June 2010.● In Figure 5.ATX Power Supply and Control Connections Using Crystallfontz Optional SCAB (Pg. 23), corrected illustration so Y14 cable red and blue twisted wires are aligned correctly to the connector. Ground (blue) is on the far right. Cable connector order has not changed.● In Typical Current Consumption (Pg. 19), made slight changes to specifications based on improved backlight.● Wherever needed, minor modifications in text and illustrations to improve clarity. Includes how Hardware revisions are listed above.
2010/05/24	<p>Data Sheet Revision: v2.0 Changes since last revision (v1.4c):</p> <ul style="list-style-type: none">● Added boxed note “Fine Print” at the bottom of this Revision History. Please read these disclaimers.● Removed PDF sticky note titled “Changes” from this Revision History. The note read “2008-10-07 Corrected specification of GPIO pull-up/pull-down mode resistance values from “approximately 5Ω” to “approximately 5kΩ”. See page 39.” Note is no longer needed because document has been revised.● Wherever listed, deleted dash (“-”) from module part numbers to match how they now appear on our website without the dash (“-”).● Wherever listed, added dash (“-”) to cable part numbers to match how they now appear on our website.● Wherever needed, added information about which optional Crystallfontz cables to use.● Expanded ORDERING INFORMATION (Pg. 10) to include overview of kits and accessories.● Added new section ATX Power Supply Power and Control Connections (Pg. 22).● In command 33 (0x21): Set Baud Rate (Pg. 64), corrected from “data_length = 1” to “data_length = 0”.● In command 34 (0x22): Set or Set and Configure GPIO Pin (Pg. 48), corrected specification of GPIO pull-up/pull-down mode resistance values from “approximately 5Ω” to “approximately 5kΩ”.● In command 35 (0x23): Read GPIO Pin Levels and Configuration State (Pg. 50),<ul style="list-style-type: none">- Corrected from “data_length: 4” to “data_length: 1”.- Index for “data[0]: index of GPIO to query” previously listed only 1-4. Added 5-12.- Corrected from “5-255: reserved” to “13-255: reserved”.- Improved LED descriptions for H1 pins. Pins did not change.- Clarified parameter usage in GPIOs versus GPOs.



DATA SHEET (Continued)	
2010/05/24	<ul style="list-style-type: none">● Please read expanded section CARE AND HANDLING PRECAUTIONS (Pg. 55).● In APPENDIX C: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC) (Pg. 62).<ul style="list-style-type: none">- Added section with hypertext links to our free downloadable code.- Added section Algorithm 2B: "C" Improved Bit Shift Implementation (Pg. 65). This is a simplified algorithm that implements the CRC.- In sample code for Algorithm 1: "C" Table Implementation (Pg. 62) and Algorithm 2: "C" Bit Shift Implementation (Pg. 63), added typedefs for "ubyte" and "word".- In Algorithm 6: "Perl" Table Implementation (Pg. 70), corrected code from "my \$packet = \$type . \$length . \$data ;" to "my \$packet = chr(hex \$type) .chr(hex \$length) .chr(hex \$data);".● Reorganized content, improved drawings, and modified text to improve readability.
2007/05/01	<p>Data Sheet Revision: v1.4c Changes since last revision (v1.4b):</p> <ul style="list-style-type: none">● In MAIN FEATURES, added "RoHS Compliant".● Added section on Module Classification Information.● In Ordering Information section, added a description of other variants available.● In Physical Characteristics section, weight (typical) was changed from 60 grams to 66 grams.● Added GPIO input high and input low voltage specifications to DC Characteristics section.● Reorganized content, improved drawings, and edited to improve readability.
2005/12/20	<p>Data Sheet Revision: v1.4b Changes since last revision (v1.4a):</p> <ul style="list-style-type: none">● Corrected "Character Size" and added "Character Pitch" dimension.● Corrected specification for supply voltage maximum.● Corrected "data [1]" for command 11: Set LCD Cursor Position.● 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).● Formatting, content organization, and minor rewording to improve readability.● For added convenience, a separate data sheet is available for each CFA635 module variant.
2005/10/20	<p>Data Sheet Revision: v1.4a: Changes since last revision (v1.4):</p> <ul style="list-style-type: none">● Corrected list of GPIO connections in Figure 4.● Corrected "type" for command 33: Set Baud Rate.● Corrected length returned by reply for command 35: Read GPIO Pin Levels and Configuration State.● Minor formatting and text changes to improve readability.



DATA SHEET (Continued)	
2005/08/01	<p>Start Public Version Tracking. Data Sheet Revision: v1.4: Changes since last revision (v1.0):</p> <ul style="list-style-type: none">● Added Revision History (this page).● Added Electrical Specifications.● 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 reply for command 30: Read Reporting & Status.

The Fine Print

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

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

The information in this publication is deemed accurate but is not guaranteed.

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

Copyright © 2010 by Crystalfontz America, Inc., 12412 East Saltese Avenue, Spokane Valley, WA 99216-0357 U.S.A.



CONTENTS

MAIN FEATURES	9
ORDERING INFORMATION	10
Module Variants Choices	10
Module Classification Information	10
Kits With Accessories	11
MECHANICAL SPECIFICATIONS	12
Physical Characteristics	12
Module Outline Drawing	13
Jumper Locations and Functions	15
Keypad Detail Drawing	16
ELECTRICAL SPECIFICATIONS	17
System Block Diagram	17
Viewing Direction	18
Driving Method	18
Absolute Maximum Ratings	18
DC Characteristics	19
Typical Current Consumption	19
ESD (Electro-Static Discharge) Specifications	20
Additional Criteria	20
CONNECTION INFORMATION	21
ATX Power Supply Power and Control Connections	21
GPIO/GPO Connections	25
GPIO Connections - H1 Pin Assignments	25
GPO Connections - H2 Pin Assignments	26
HOST COMMUNICATIONS	28
Packet Structure	28
About Handshaking	29
Report Codes	30
0x80: Key Activity	30
0x81: Fan Speed Report (SCAB required)	30
0x82: Temperature Sensor Report (SCAB required)	31
Command Codes	32
0 (0x00): Ping Command	32
1 (0x01): Get Hardware & Firmware Version	32
2 (0x02): Write User Flash Area	32
3 (0x03): Read User Flash Area	33
4 (0x04): Store Current State As Boot State	33
5 (0x05): Reboot CFA635, Reset Host, or Power Off Host	34
6 (0x06): Clear LCD Screen	35
9 (0x09): Set LCD Special Character Data	35
10 (0x0A): Read 8 Bytes of LCD Memory	36
11 (0x0B): Set LCD Cursor Position	36
12 (0x0C): Set LCD Cursor Style	36
13 (0x0D): Set LCD Contrast	37



CONTENTS, CONTINUED

14 (0x0E): Set LCD & Keypad Backlight	37
16 (0x10): Set Up Fan Reporting (SCAB required)	37
17 (0x11): Set Fan Power (SCAB required)	38
18 (0x12): Read DOW Device Information (SCAB required)	38
19 (0x13): Set Up Temperature Reporting (SCAB required)	39
20 (0x14): Arbitrary DOW Transaction (SCAB required)	41
22 (0x16): Send Command Directly to the LCD Controller	41
23 (0x17): Configure Key Reporting	42
24 (0x18): Read Keypad, Polled Mode	42
25 (0x19): Set Fan Power Fail-Safe (SCAB required)	43
26 (0x1A): Set Fan Tachometer Glitch Filter (SCAB required)	43
27 (0x1B): Query Fan Power & Fail-Safe Mask (SCAB required)	44
28 (0x1C): Set ATX Power Switch Functionality	45
29 (0x1D): Enable/Disable and Reset the Watchdog	46
30 (0x1E): Read Reporting & Status	47
31 (0x1F): Send Data to LCD	48
33 (0x21): Set Baud Rate	48
34 (0x22): Set or Set and Configure GPIO Pin	48
35 (0x23): Read GPIO Pin Levels and Configuration State	50
CHARACTER GENERATOR ROM (CGROM)	52
MODULE RELIABILITY AND LONGEVITY	53
Module Reliability	53
Module Longevity (EOL / Replacement Policy)	53
CARE AND HANDLING PRECAUTIONS	54
APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER (SCAB REQUIRED)	56
APPENDIX B: CONNECTING A DS1963S SHA IBUTTON (SCAB REQUIRED)	58
APPENDIX C: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC)	61
Sample Code	61
Algorithms to Calculate the CRC	61
Algorithm 1: "C" Table Implementation	61
Algorithm 2: "C" Bit Shift Implementation	62
Algorithm 2B: "C" Improved Bit Shift Implementation	64
Algorithm 3: "PIC Assembly" Bit Shift Implementation	64
Algorithm 4: "Visual Basic" Table Implementation	66
Algorithm 5: "Java" Table Implementation	67
Algorithm 6: "Perl" Table Implementation	69
APPENDIX D: QUALITY ASSURANCE STANDARDS	71



LIST OF FIGURES

Figure 1. Module Outline Drawing (2 pages below) -----	13
Figure 2. Jumper Locations and Functions -----	15
Figure 3. Keypad Detail Drawing -----	16
Figure 4. System Block Diagram -----	17
Figure 5. ATX Power Supply and Control Connections Using CrystalFontz Optional SCAB -----	23
Figure 6. ATX Power Supply and Control Connections Using CrystalFontz WR-PWR-Y25 Cable- -----	24
Figure 7. Location of USB Host Connection -----	25
Figure 8. Five GPIO Connections on Module's Connector H1 -----	26
Figure 9. Eight GPO Connections on Module's Connector H2 -----	27
Figure 10. Module Connected to a SCAB using CrystalFontz WR-EXT-Y19 Cable -----	28
Figure 11. Character Generator ROM (CGROM) -----	52
Appendix A Figure 1. CFA635 Test Circuit Schematic -----	56
Appendix B Figure 1. Connect CFA635 to Maxim/Dallas DS19632 SHA iButton using DS9094 iButton Clip	58



MAIN FEATURES

- ☐ Large easy-to-read 20 characters x 4 lines LCD has a large display area in a compact size. Fits in a 1U rack mount case (37 mm overall height).
- ☐ USB interface (115200 baud equivalent throughput).
- ☐ White edge LED backlit with STN blue negative mode LCD (displays light characters on blue background).
- ☐ Integrated blue LED backlit 6-button translucent silicone keypad.
- ☐ Adjustable backlight contrast.
- ☐ Four bicolor (red + green) LED indicators. The LEDs' brightness can be set by the host software, which allows smoothly mixing the LEDs to produce other colors (for example, yellow and orange).
- ☐ ATX power supply control functionality allows the keypad buttons on the CFA635-TMF-KU to replace the power and reset switches on your system, simplifying front panel design. The optional 16-pin [WR-PWR-Y25](#) ATX power cable may be used for direct connection to the host's PC power supply (written as CFA635+[WR-PWR-Y25](#) cable in this Data Sheet). Or use the CFA635-TMF-KU with our optional [SCAB](#) and [WR-PWR-Y14](#) cable (written as "CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable" in this Data Sheet).
- ☐ LCD characters are contiguous in both X and Y directions to allow the host software to display "gapless" bar graphs in horizontal or vertical directions.
- ☐ Unique "Scrolling Marquee" feature continuously scrolls a message across the display without host intervention.
- ☐ Fully decoded keypad: any key combination is valid and unique.
- ☐ Robust packet-based communications protocol with 16-bit CRC.
- ☐ Built-in microcontroller.
- ☐ Nonvolatile memory capability (EEPROM):
 - Customize the "power-on" display settings.
 - 16-byte "scratch" register for storing IP address, netmask, system serial number . . .
- ☐ Firmware support for the optional System Cooling Accessory Board ([SCAB](#)). The combination of the CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable allows:
 - Four functional fan connectors with RPM monitoring and variable PWM fan power control. Commonly available PC cooling fans may be used. Fail-safe fan power settings allows host to safely control four fans based on temperature. See [25 \(0x19\): Set Fan Power Fail-Safe \(SCAB required\) \(Pg. 43\)](#).
 - Using optional CrystalFontz [WR-DOW-Y17](#) cable with Dallas 1-Wire sensor, you can monitor temperatures up to 32 channels at up to 0.5°C absolute accuracy.
 - Hardware watchdog can reset host on host software failure.
 - RS-232 to Dallas Semiconductor 1-Wire bridge functionality allows control of other 1-Wire compatible devices (ADC, voltage monitoring, current monitoring, RTC, GPIO, counters, identification/encryption). Additional hardware required.
- ☐ RoHS compliant.
- ☐ Factories have ISO certification.



ORDERING INFORMATION

MODULE VARIANTS CHOICES

PART NUMBER	FLUID	LCD GLASS COLOR	IMAGE	POLARIZER FILM	BACKLIGHTS
CFA635-TMF-KU	STN	blue	negative	transmissive	LCD: white edge LEDs Keypad: blue LEDs
<i>Additional variants available (same form factor and interface).</i>					
CFA635-YYE-KU	STN	yellow-green	positive	transflective	LCD: yellow-green edge LEDs Keypad: yellow-green LEDs
CFA635-TFE-KU	FSTN	neutral	positive	transflective	LCD: white edge LEDs Keypad: white LEDs
<i>Two different serial interfaces are also available. See www.crystalfontz.com/products/635serial/.</i>					

MODULE CLASSIFICATION INFORMATION

CFA 635 - T M F - K U
① ② ③ ④ ⑤ ⑥ ⑦

①	Brand	CrystalFontz America, Inc.
②	Model Identifier	635
③	Backlight Type & Color	T – LED, white
④	Fluid Type, Image (positive or negative), & LCD Glass Color	M – STN, negative, blue
⑤	Polarizer Film Type, Normal (NT) Temperature Range, & View Angle (O 'Clock)	F – Transmissive, NT ¹ , 12:00
⑥	Special Code 1	K – Manufacturer's code
⑦	Special Code 2	U – USB interface
¹ Normal Temperature Range is 0°C minimum to +50°C maximum.		



KITS WITH ACCESSORIES

The CFA635-TMF-KU module is ordered as part of a kit. See [CFA635-xxx-KU Kit Configurator](#).

$\frac{xx}{1} \quad \frac{635}{2} \quad \frac{xx}{3} - \frac{xxx}{4} - \frac{KUx}{5}$

①	DB – 5.25-inch half-height drive bay mounting bracket. or DS – SLED, a chassis that fits in 5.25-inch half-height drive bay. The SLED can hold a CFA635 module, a SCAB (System Cooling Accessory Board) , and a 3.5-inch hard disk drive. Hard drive is not included.
②	AK, AL, BG, or BK – Overlay for front of module (choice of aluminum, beige, and two different blacks).
③	Backlight, fluid, and LCD glass is described on previous page.
④	KU1 – CFA635+ WR-USB-Y03 USB A (male) to 2 mm connector cable for external connection to host's USB port. or KU2 – CFA635+ WR-USB-Y11 cable with 2 mm motherboard header to connect to the USB pins on your motherboard. or KU3 – CFA635+ WR-USB-Y03 USB A (male) to 2 mm connector cable connects CFA635 to host's USB port. WR-USB-Y11 cable with 2 mm motherboard header connects to the USB pins on host's motherboard. WR-PWR-Y12 "3.5-inch floppy" 4-pin hard drive-to-floppy connector and hard drive splitter power cable. SCAB (System Cooling Accessory Board) . WR-EXT-Y19 16-pin conductor cable connects the CFA635 to the SCAB . 3.5 inches. WR-EXT-Y15 16-pin conductor cable connects the CFA635 to the SCAB . 16 inches. 4 x WR-FAN-X01 3-pin fan extension cables to use with the SCAB . 4 x WR-DOW-Y17 Dallas One Wire DS18B20 temperature sensors to use with the SCAB . You can order more to connect up to 32 sensors.

We also sell this module in an external enclosure with a captive USB "A" cable connection. See [XES635BK-TMF-KU](#).



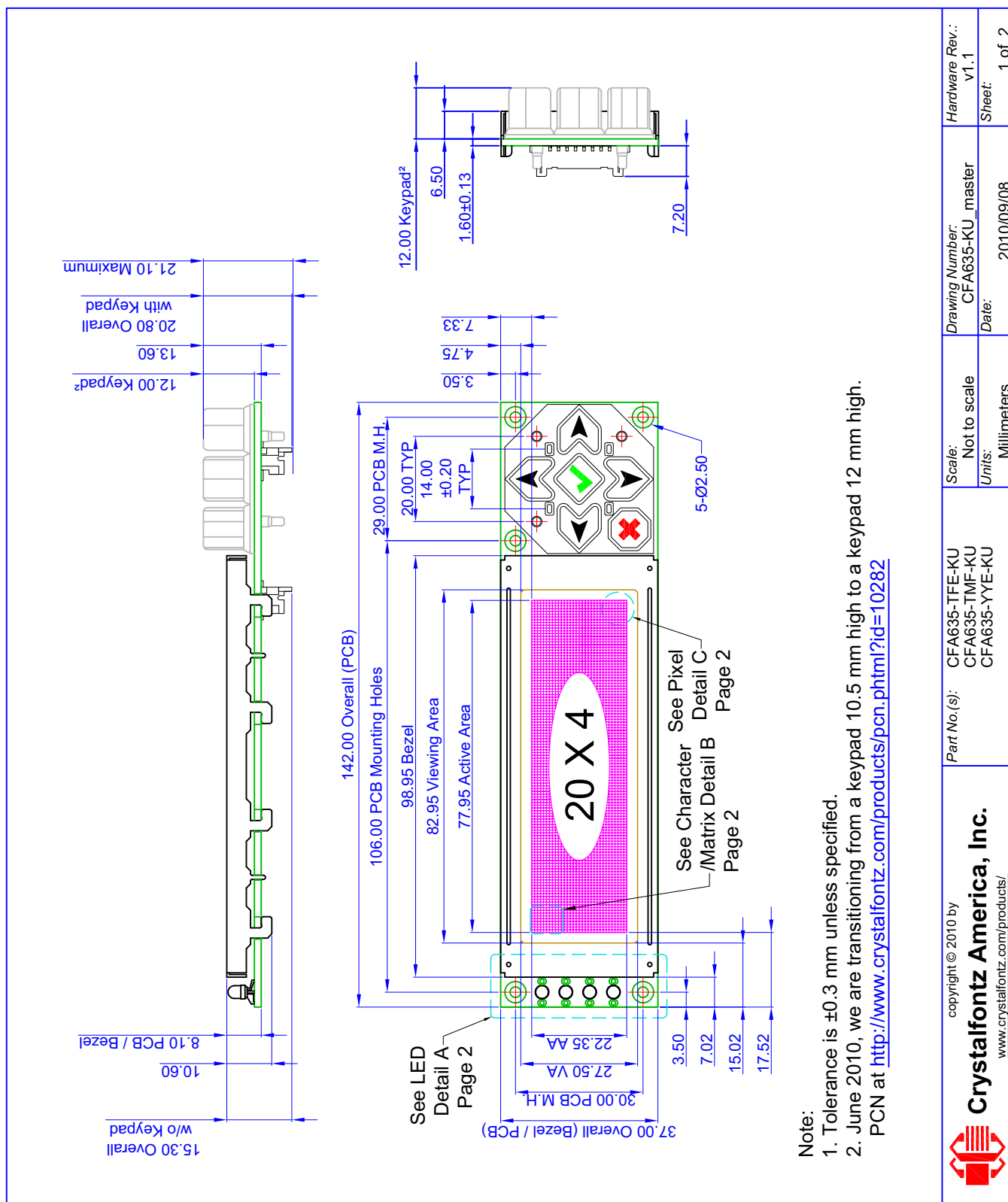
MECHANICAL SPECIFICATIONS

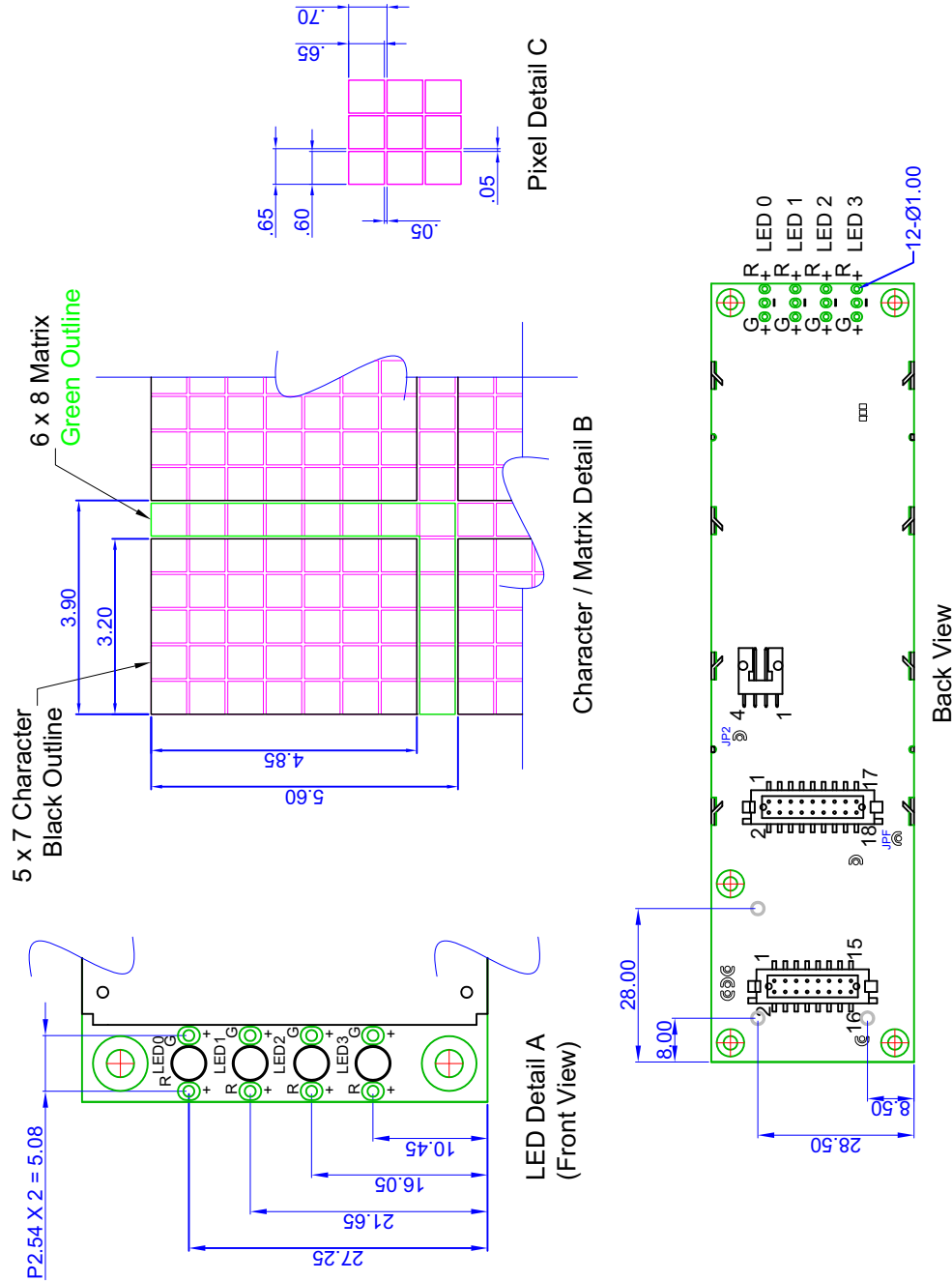
PHYSICAL CHARACTERISTICS

ITEM	SIZE
Module Width and Height	142.0 (W) x 37.0 (H) mm
Module Depth: without Keypad, with Connectors with Keypad, with Connectors	15.30 mm nominal 20.80 mm nominal 21.10 mm maximum
Viewing Area	82.95 (W) x 27.50 (H) mm
Active Area	77.95 (W) x 22.35 (H) mm
Standard Character Size	3.20 (W) x 4.85 (H) mm
Standard Character Pitch	3.90 (W) X 5.6 (H) mm
Dot Size	0.60 (W) x 0.65 (H) mm
Dot Pitch	0.65 (W) x 0.70 (H) mm
Keystroke Travel (approximate)	2.4 mm
Weight	66 grams (typical)



Figure 1. Module Outline Drawing (2 pages below)





Note:

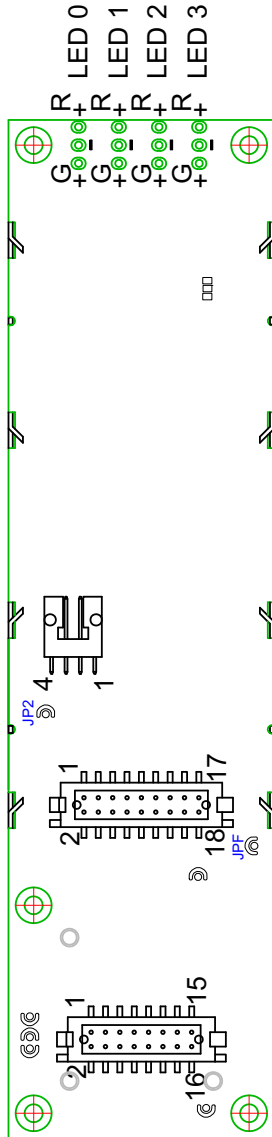
1. Tolerance is ± 0.3 mm unless specified.
2. June 2010, we are transitioning from a keypad 10.5 mm high to a keypad 12 mm high.
PCN at <http://www.crystalfontz.com/products/pcn.php?id=10282>

copyright © 2010 by



JUMPER LOCATIONS AND FUNCTIONS

This USB LCD module has eight jumpers. Only JP2 and JPF may be changed. The JP2 jumper is normally closed. To open the jumper, remove the solder. Solder wick works well for this. The JPF jumper is normally open. The jumper may be closed by melting a ball of solder across the gap.



Jumper Locations and Functions

JP2	open	Power is not supplied through USB connector. Typically JP2 is open for ATX functionality.
	closed	Power is supplied through USB connector.
JPF	open	Frame ground is isolated from logic/USB ground.
	closed	Frame ground is connected to Logic/USB ground.
All other jumpers (JP3, JPA1, JPA2, JPB1, JPB2, JPC) are factory build options. Do not change.		

Note:
1. Tolerance is ± 0.3 mm unless specified.



copyright © 2010 by
Crystalfontz America, Inc.
www.crystalfontz.com/products/

Part No (s):
CFA635-TFE-KU
CFA635-TMF-KU
CFA635-YYE-KU

Scale:
Not to scale
Units:
Millimeters

Drawing Number:
CFA635-KU_master
Date:
2010/09/08

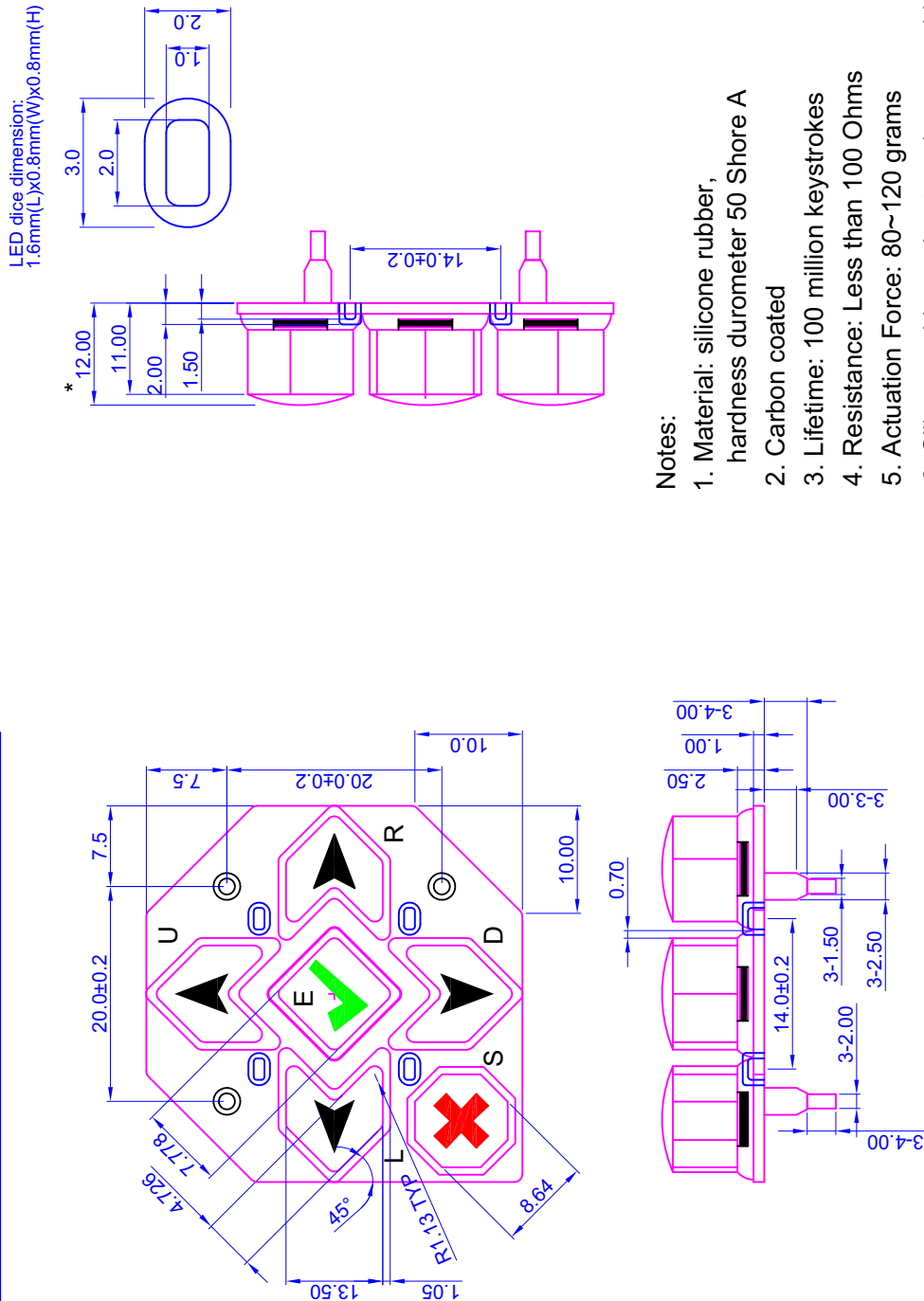
Hardware Rev.:
v1.1
Sheet:
1 of 1

Figure 2. Jumper Locations and Functions



KEYPAD DETAIL DRAWING

June 2010, we are transitioning from a keypad 10.5 mm high to a keypad 12 mm high. PCN at <http://www.crystalfontz.com/products/pcn.phtml?id=10282>



Notes:

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

 copyright © 2009 by CrystalFontz America, Inc. www.crystalfontz.com/products/	Part No.(s): CFA635 Keypad Detail	Scale: Not to scale Units: Millimeters	Drawing Number: Keypad_Detail_master Date: 2010/09/09	Hardware Rev.: V1.1 Sheet: 1 of 1

Figure 3. Keypad Detail Drawing



ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM

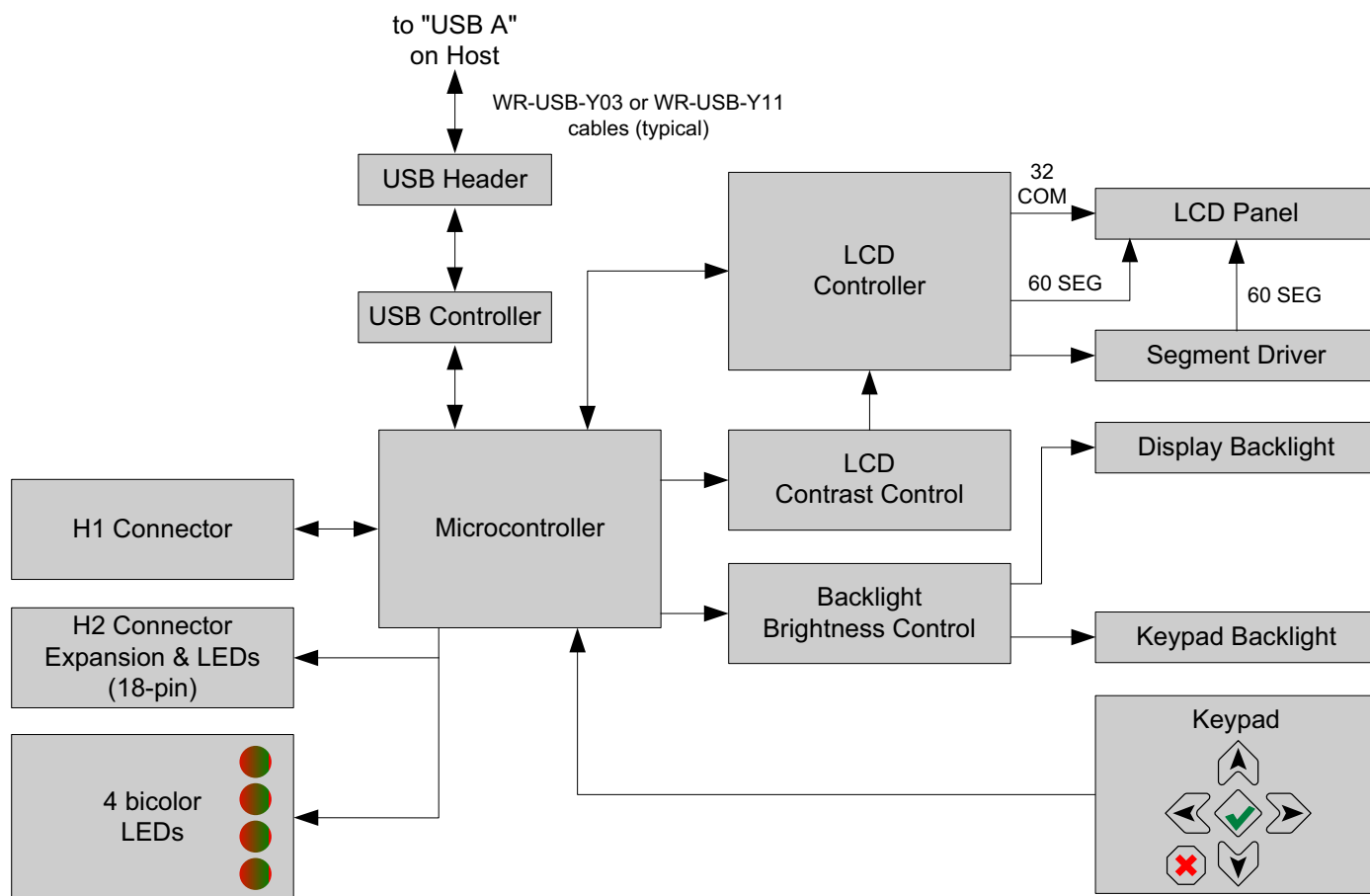


Figure 4. System Block Diagram



VIEWING DIRECTION

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

DRIVING METHOD

DRIVING METHOD	SPECIFICATION
Duty	1/32
Bias	6.7

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
Supply Voltage for Logic	V _{DD}	0	5.25v
<i>*Note: Prolonged exposure at temperatures outside of this range may cause permanent damage to the module.</i>			



DC CHARACTERISTICS

DC CHARACTERISTICS	SYMBOL	MINIMUM	TYPICAL	MAXIMUM
Supply Voltage	$V_{DD} - V_O$	+4.75v	+5.0v	+5.25v
GPIO Input High Voltage	V_{IH}	+2.1v		V_{DD}
GPIO Input Low Voltage	V_{IL}	V_{SS}		+0.08v

TYPICAL CURRENT CONSUMPTION

ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION	
Logic	LCD and Keypad Backlights	All Indicator LEDs (4 Red + 4 Green)	$V_{DD}=4.75V$	$V_{DD}=5.25V$
X	-	-	25 mA	26 mA
X	X	-	124 mA	148 mA
X	-	X	142 mA	163 mA
X	X	X	240 mA	285 mA

ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION	
Logic	LCD and Keypad Backlights	All Indicator LEDs (4 Red + 4 Green)	$V_{DD}=4.75V$	$V_{DD}=5.25V$
X	-	-	27 mA	29 mA
X	X	-	150 mA	180 mA
X	-	X	143 mA	166 mA
X	X	X	260 mA	330 mA



ITEMS ENABLED			TYPICAL CURRENT CONSUMPTION	
Logic	LCD and Keypad Backlights	All Indicator LEDs (4 Red + 4 Green)	V _{DD} =4.75V	V _{DD} =5.25V
X	-	-	24 mA	26 mA
X	X	-	109 mA	131 mA
X	-	X	140 mA	162 mA
X	X	X	225 mA	267 mA

GPIO CURRENT LIMITS	SPECIFICATION
Sink	25 mA
Source	10 mA

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS

D+ and D- pins of USB connector only: Electrostatic Discharge Voltage ($I < 1 \mu A$): +/- 2000 v.

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.

ADDITIONAL CRITERIA

ADDITIONAL CRITERIA	SPECIFICATION
Backlight PWM ¹ Frequency	300 Hz nominal
Fan Tachometer Speed Range (assuming two PPR ²)	600 RPM to 3,000,000 RPM
Fan Power Control PWM ¹ Frequency (When used with optional SCAB .)	18 Hz nominal
<u>Definitions</u> ¹ PWM is <i>Pulse Width Modulation</i> . PWM is a way to simulate intermediate levels by switching a level between full on and full off. PWM can be used to control the brightness of LED backlights, relying on the natural averaging done by the human eye, as well as for controlling fan power. ² PPR is <i>Pulses Per Revolution</i> , also written as p/r.	



CONNECTION INFORMATION

ATX POWER SUPPLY POWER AND CONTROL CONNECTIONS

ATX power supply control functionality allows the buttons on the CFA635-TMF-KU to replace the power and reset button on your system, simplifying front panel design. This ATX power supply control functionality can be accomplished with or without our optional [SCAB](#) (System Cooling Accessory Board). The [SCAB](#) provides fan monitoring and control as well as DOW temperature probe monitoring.

NOTE

The GPIO pins used for ATX control must not be configured as user GPIO. The GPIO pins must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are factory default but may be changed by the user. Please see command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 48\)](#).

When configuring the CFA635-TMF-KU for ATX functionality, open jumper JP2 in order to ensure correct operation. See [Jumper Locations and Functions \(Pg. 15\)](#) for jumper positions and locations. This is required whether the optional [SCAB](#) is or is not used.

For this functionality, this configuration for the CFA635-TMF-KU is powered from the PC's V_{SB} signal, the “standby” 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 connector pins labeled GND and +5v.

GPIO[1] ATX Host Power Sense

Since the CFA635-TMF-KU must act differently depending on whether the host's power supply is on or off, you must also connect the host's “switched +5v” to GPIO[1]. This GPIO line functions as POWER SENSE. The POWER SENSE pin is configured as an input with a pull-down, 5k Ω nominal.

GPIO[2] ATX Host Power Control

The motherboard's power switch input is connected to GPIO[2]. This GPIO line functions as POWER CONTROL. The POWER CONTROL pin is configured as a high impedance input until the LCD module instructs the host to turn on or off. Then it will change momentarily to low impedance output, driving either low or high depending on the setting of POWER INVERT. See command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 45\)](#).

GPIO[3] ATX Host Reset Control

The motherboard's reset switch input is connected to GPIO[3]. This GPIO line functions as RESET. The RESET pin is configured as a high-impedance input until the LCD module wants to reset the host. Then it will change momentarily to low impedance output, driving either low or high depending on the setting of RESET_INVERT. See command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 45\)](#). This connection is also used for the hardware watchdog.



Connection Overview with and without Optional SCAB

ATX Power Supply & Control Connections	With Optional SCAB+ WR-PWR-Y14 cable, Pins on SCAB's 7-pin and 4-pin Connectors*	Without Optional SCAB, Pins on CFA635-TMF-KU's Connector H1
V _{SB} (+5v)	7-pin connector, +5v pin	Pin 16
Ground	7-pin connector, GND pin	Pin 15
GPIO[1] ATX Host POWER SENSE	4-pin connector, +5v pin	Pin 12
GPIO[2] ATX Host POWER CONTROL	7-pin connector, GPIO[2] pin	Pin 9
GPIO[3] ATX Host RESET CONTROL	7-pin connector, GPIO[3] pin	Pin 10
*SCAB's JP8 must be open and JP9 must be closed. For quick reference of location, see Figure 5. on Pg. 23 . For details, see the SCAB Data Sheet on www.crystalfontz.com/product/SCAB.html#docs .		

Details of Connection with Optional SCAB and WR-PWR-Y14 Cable

The CrystalFontz [WR-PWR-Y14](#) cable allows ATX power control connections through the optional [SCAB](#). This allows additional flexibility in cabling and overall functionality of the CFA635-TMF-KU in system control and monitoring. You will need to order either the [WR-EXT-Y15](#) or [WR-EXT-Y19](#) to connect the [SCAB](#) to the CFA635-TMF-KU's connector H1.

NOTE

If the CrystalFontz [WR-PWR-Y14](#) cable and [SCAB](#) is ordered at the same time as the module, CrystalFontz will open JP2 on the CFA635-TMF-KU, open JP8 and close JP9 on the [SCAB](#), and send the following software configuration commands unless we are otherwise instructed. Please note that once these changes are made, for the module to power up, power must be applied to the 7-pin connector on the [SCAB](#) as well as the 4-pin power connector on module.

```
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 [SCAB](#) and [WR-PWR-Y14](#) cable connects to the CFA635-TMF-KU's connector H1, and your system motherboard/ATX power supply:

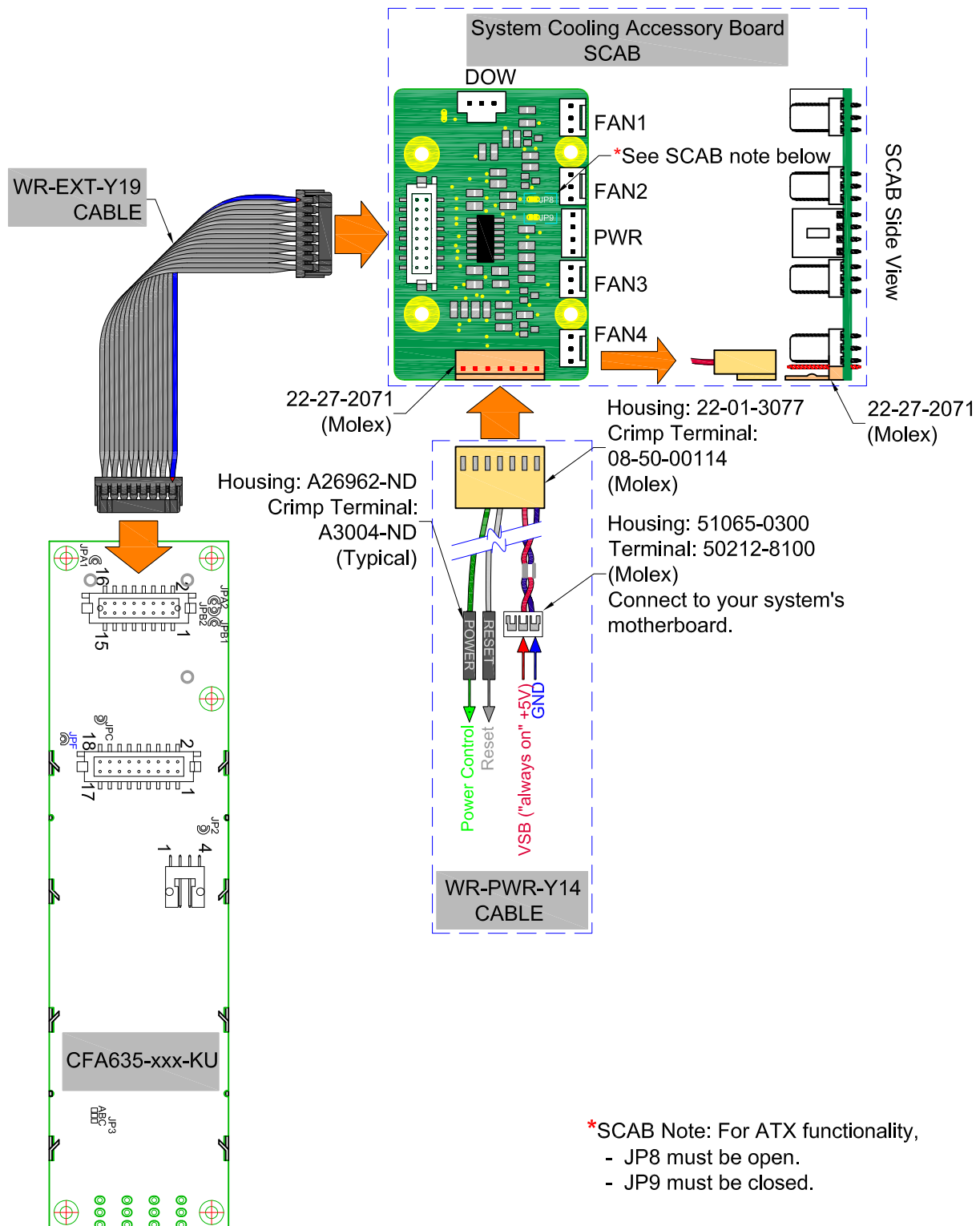


Figure 5. ATX Power Supply and Control Connections Using Crystalfontz Optional SCAB



Details of Connection without Optional SCAB using WR-PWR-Y25 Cable

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

NOTE

If the Crystalfontz [WR-PWR-Y25](#) cable is ordered at the same time as the module, we will open jumper JP2 and send the following software configuration commands unless we are otherwise instructed. Please note that once these changes are made, for the module to power up, power must be applied to connector H1 with +5v applied to pin 12 and ground to pin 16. (See figure below.)

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

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

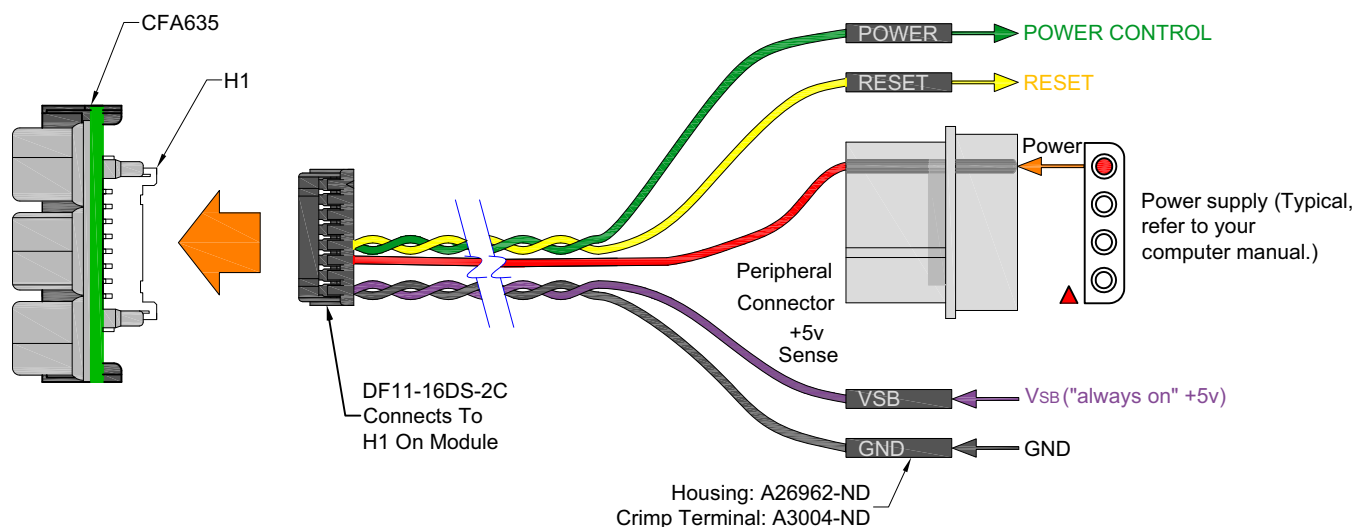


Figure 6. ATX Power Supply and Control Connections Using Crystalfontz WR-PWR-Y25 Cable

USB Connection to Host

The CFA635-TMF-KU is a USB peripheral, requiring only one connection to the host for both data communications and power supply. To keep the CFA635-TMF-KU as thin as possible, the CFA635-TMF-KU uses a very low profile 2 mm latching polarized connector for USB connection.

Crystalfontz offers two cables to connect between the CFA635-TMF-KU and the host:

1. The [WR-USB-Y03](#) (75-inch) has a mating 2 mm connector on one end and a standard "USB A" on the other end.



- The [WR-USB-Y11](#) (27-inch) has a mating 2 mm connector on one end and standard single pin connectors on the opposite end. These single pin connectors are suitable to plug directly onto the USB headers typically found on motherboards.

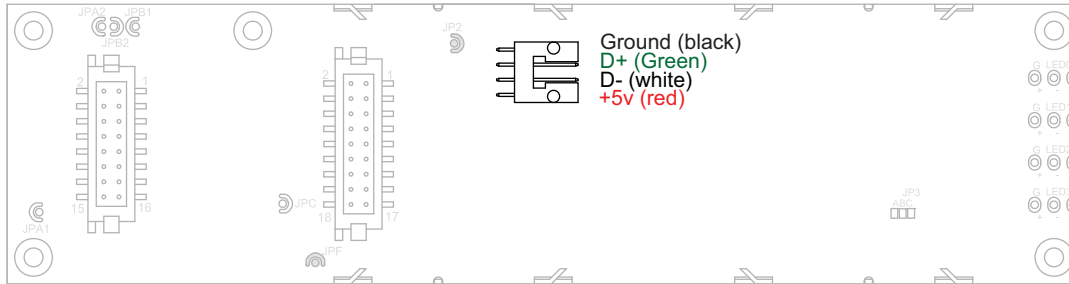


Figure 7. Location of USB Host Connection

If you would like to make your own cable, the connector on the CFA635-TMF-KU is:
FCI/Berg 95000-004: SMT 2mm connector, 4-position, polarized

The mating housing and terminals for the cable are:
FCI/Berg 90312-004: Housing, 2 mm connector, 4-position, polarized
FCI/Berg 77138-001: Terminal (4 pieces required)

GPIO/GPO CONNECTIONS

GPIO connections are on the H1 connector. GPO connections are on the H2 connector. These connections are described below.

GPIO Connections - H1 Pin Assignments

CFA635-TMF-KU has five GPIOs available on connector H1. These GPIOs can be accessed directly through H1 or through the CrystalFontz optional [SCAB \(System Cooling Accessory Board\)](#) when it is connected to H1. The [SCAB](#) can be easily connected to the CFA635-TMF-KU by using either the CrystalFontz [WR-EXT-Y15](#) or [WR-EXT-Y19](#) cables. For



more information on the [SCAB](#), see [How to Connect the Optional Scab \(Pg. 27\)](#) below and the [SCAB Data Sheet](#) on our website.

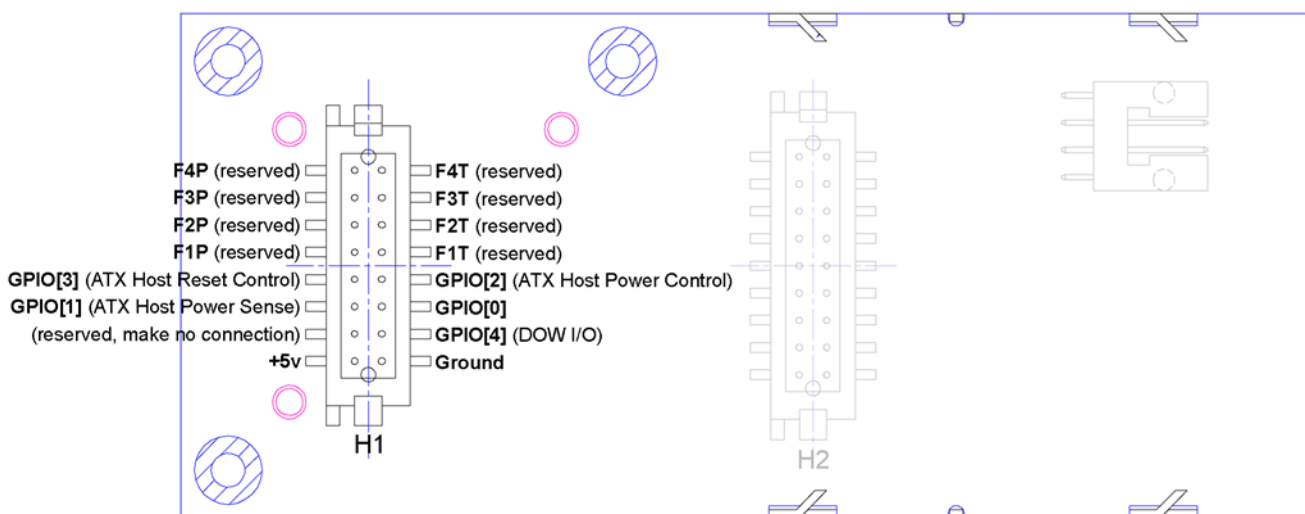


Figure 8. Five GPIO Connections on Module's Connector H1

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

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

GPO Connections - H2 Pin Assignments

The CFA635-TMF-KU has eight GPOs available on connector H2. By factory default, these GPOs drive the front panel LEDs. If you remove the LEDs, you can use the GPOs for other purposes.

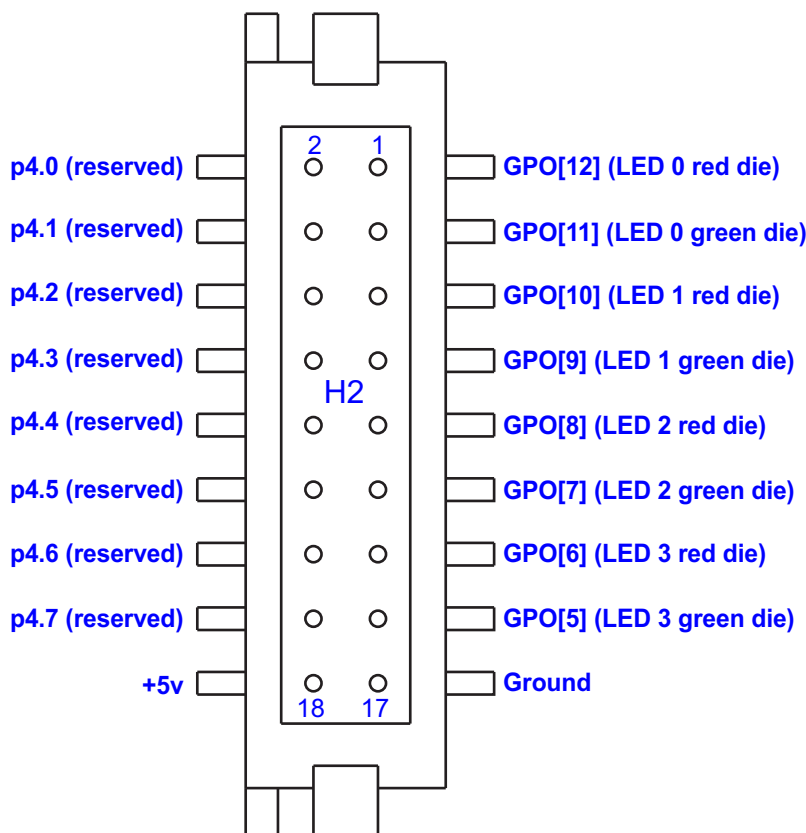


Figure 9. Eight GPO Connections on Module's Connector H2

Please see the commands [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 48\)](#), and [35 \(0x23\): Read GPIO Pin Levels and Configuration State \(Pg. 50\)](#) below for details on how to control the GPIOs.

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

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

HOW TO CONNECT THE OPTIONAL SCAB

The CrystalFontz [SCAB](#) (System Cooling Accessory Board) is designed to connect to a CFA635-TMF-KU's H1 connector. The [SCAB](#) will receive the correct signals to operate from the CFA635-TMF-KU.

Purchase one of the following two CrystalFontz cables to make the connection between the [SCAB](#) and the CFA635-TMF-KU:

1. Use the [WR-EXT-Y15](#) 18-inch cable if the [SCAB](#) is mounted some distance away from the CFA635-TMF-KU. For instance, the [SCAB](#) could be mounted in a central location within the PC's case. The WR-EXT-Y15 would connect from this central location to the LCD module that is mounted in a drive bay. Then the connections to the fans and temperature sensors would only need to be run to the [SCAB](#), not all the way to the front panel where the LCD module is mounted.
2. Use the [WR-EXT-Y19](#) 3.5-inch cable when the [SCAB](#) is mounted close to the CFA635-TMF-KU, as is the case when the [SCAB](#) is fastened directly to the LCD module's mounting bracket.



Here is a photo of a CFA635 USB LCD module connected to a [SCAB](#) using the [WR-EXT-Y19](#) cable:

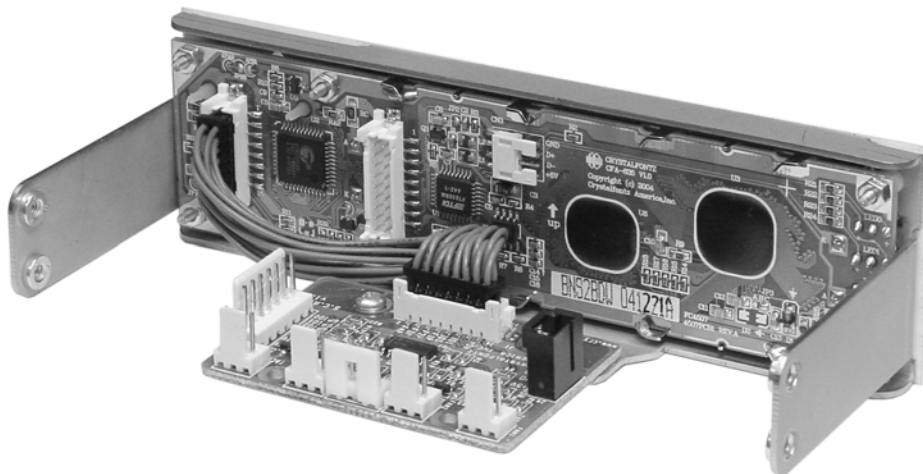


Figure 10. Module Connected to a SCAB using CrystalFontz WR-EXT-Y19 Cable

For more information on the [SCAB](#), see the SCAB Data Sheet (www.crystalfontz.com/product/SCAB#docs).

HOST COMMUNICATIONS

The CFA635-TMF-KU 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. A link to VCP drivers download and installation instructions can be found on the CrystalFontz website at [USB LCD Drivers](#). 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 CFA635-TMF-KU is connected. This VCP should be opened at 115200 baud, 8 data bits, no parity, 1 stop bit.

NOTE

Because there is no difference in communications and commands for *serial* and *USB* variants of the CFA635, the remaining sections in this Data Sheet use the shorter term “CFA635” instead of “CFA635-TMF-KU”.

PACKET STRUCTURE

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

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 CFA635
    01 = normal response from CFA635 to host
    10 = normal report from CFA635 to host (not in
        direct response to a command from the host)
    11 = error response from CFA635 to host (a packet
        with valid structure but illegal content
        was received by the CFA635)
```

`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 [APPENDIX C: SAMPLE CODE \(INCLUDES ALGORITHMS TO CALCULATE THE CRC\) \(Pg. 61\)](#) 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, [635_WinTest](#) along with its C source code. Included in the `635_WinTest` source is a CRC algorithm and an algorithm that detects packets. The algorithm will automatically re-synchronize to the next valid packet in the event of any communications errors. Please follow the algorithm in the sample code closely in order to realize the benefits of using the packet communications.

ABOUT HANDSHAKING

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

The host should wait for a corresponding acknowledge packet from the CFA635 before sending the next command packet. The CFA635 will respond to all packets within 250 mS. The host software should stop waiting and retry the packet if the CFA635 fails to respond within 250 mS. The host software should report an error if a packet is not acknowledged after several retries. This situation indicates a hardware problem — for example, a disconnected cable. Please note that some operating systems may introduce delays between when the data arrives at the physical port from the CFA635 until it is available to the user program. In this case, the host program may have to increase its timeout window to account for the additional overhead of the operating system.

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



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

REPORT CODES

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

0x80: Key Activity

If a key is pressed or released, the CFA635 sends a Key Activity report packet to the host. Key event reporting may be individually enabled or disabled by command [23 \(0x17\): Configure Key Reporting \(Pg. 42\)](#).

```
type = 0x80
data_length = 1
data[0] is the type of keyboard activity:
    KEY_UP_PRESS           1
    KEY_DOWN_PRESS        2
    KEY_LEFT_PRESS        3
    KEY_RIGHT_PRESS       4
    KEY_ENTER_PRESS       5
    KEY_EXIT_PRESS        6
    KEY_UP_RELEASE        7
    KEY_DOWN_RELEASE      8
    KEY_LEFT_RELEASE      9
    KEY_RIGHT_RELEASE     10
    KEY_ENTER_RELEASE     11
    KEY_EXIT_RELEASE      12
```

These codes are identical to the codes returned by the [CFA633](#). Please note that the CFA631 will return codes 13 through 20. (See the [CFA631](#) Data Sheet on our website for more details.)

0x81: Fan Speed Report (SCAB required)

If any of up to four fans connected to CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable is configured to report its speed information to the host, the CFA635 will send Fan Speed Reports for each selected fan every 1/2 second. See command [16 \(0x10\): Set Up Fan Reporting \(SCAB required\) \(Pg. 37\)](#) below.

```
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 CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable will send Temperature Sensor Reports for each selected sensor every second. See the command [19 \(0x13\): Set Up Temperature Reporting \(SCAB required\) \(Pg. 39\)](#) below.

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



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

```
void OnReceivedTempReport(COMMAND_PACKET *packet, char *output)
{
    //First check the DOW CRC return code from the CFA635
    if(packet->data[3]==0)
        strcpy(output, "BAD CRC");
    else
    {
        double
            degc;
        degc = (*(short *) &(packet->data[1])) / 16.0;

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

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

COMMAND CODES

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

0 (0x00): Ping Command

The CFA635 will return the Ping Command to the host.

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

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

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

1 (0x01): Get Hardware & Firmware Version

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

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

The return packet will be:

```
type = 0x40 | 0x01 = 0x41 = 6510
data_length = 16
data[] = "CFA635:hX.X,yY.Y"
```

X.X is the hardware revision, "1.1" for example
 yY.Y is the firmware version, "v1.6" for example

2 (0x02): Write User Flash Area

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



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

The return packet will be:

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

3 (0x03): Read User Flash Area

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

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

The return packet will be:

```
type = 0x40 | 0x03 = 0x43 = 6710
data_length = 16
data[] = 16 bytes user data recalled from the CFA635's
        non-volatile memory
```

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

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

- Characters shown on LCD, which are affected by:
 - Command [6 \(0x06\): Clear LCD Screen \(Pg. 35\)](#).
 - Command [31 \(0x1F\): Send Data to LCD \(Pg. 48\)](#).
- Special character font definitions (command [9 \(0x09\): Set LCD Special Character Data \(Pg. 35\)](#)).
- Cursor position (command [11 \(0x0B\): Set LCD Cursor Position \(Pg. 36\)](#)).
- Cursor style (command [12 \(0x0C\): Set LCD Cursor Style \(Pg. 36\)](#)).
- Contrast setting (command [13 \(0x0D\): Set LCD Contrast \(Pg. 37\)](#)).
- Backlight setting (command [14 \(0x0E\): Set LCD & Keypad Backlight \(Pg. 37\)](#)).
- Fan power settings (command [17 \(0x11\): Set Fan Power \(SCAB required\) \(Pg. 38\)](#)).
- Key press and release masks (command [23 \(0x17\): Configure Key Reporting \(Pg. 42\)](#)).
- Fan glitch delay settings (command [26 \(0x1A\): Set Fan Tachometer Glitch Filter \(SCAB required\) \(Pg. 43\)](#)).
- ATX function enable and pulse length settings (command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 45\)](#)).
- Baud rate (command [33 \(0x21\): Set Baud Rate \(Pg. 48\)](#)).
- GPIO settings (command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 48\)](#)).
- The front panel LED/GPO settings ([34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 48\)](#)).

You cannot store the fan or temperature reporting, or the fan fail-safe or host watchdog. The host software should enable these items once the system is initialized and it is ready to receive the data.

```
type = 0x04 = 410
valid data_length is 0
```

The return packet will be:

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



5 (0x05): Reboot CFA635, Reset Host, or Power Off Host

This command instructs the CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable to simulate a power-on restart of itself, reset the host, or turn the host's power off. The ability to reset the host may be useful to allow certain host operating system configuration changes to complete. The ability to turn the host's power off under software control may be useful in systems that do not have ACPI compatible BIOS.

NOTE

The GPIO pins used for ATX control must not be configured as user GPIO, and must be configured to their default drive mode in order for the ATX functions to work correctly. These settings are factory default, but may be changed by the user. Please see command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 48\)](#).

Rebooting the CFA635 may be useful when testing the boot configuration. It may also be useful to re-enumerate the devices on the 1-Wire bus (CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable). To reboot the CFA635, send the following packet:

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

NOTE

The CFA635 will return the acknowledge packet immediately, then reboot itself. The module will not respond to new command packets for up to 3 seconds. 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](#)).

To reset the host (CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable), assuming the host's reset line is connected to GPIO[3] as described in command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 45\)](#), send the following packet:

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

NOTE

The CFA635 will return the acknowledge packet immediately, then reset the host. After resetting the host (~1.5 seconds), the module will reboot itself. The module will not respond to new command packets for up to 3 seconds (~4.5 seconds overall) after its reboot. Part of this delay is the intentional staggered sequencing of turning on power to the fans. If you are not using fans, you can speed the boot process by setting the fan power to 0 (command [17 \(0x11\): Set Fan Power \(SCAB required\)](#)) and saving this as the default boot state (command [4 \(0x04\): Store Current State As Boot State](#)). Normally, the host will be recovering from its own reset, so the boot delay of the module will not be of consequence.



To turn the host's power off (CFA635+[SCAB+WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable), assuming the host's power control line is connected to GPIO[2] as described in command [28 \(0x1C\): Set ATX Power Switch Functionality \(Pg. 45\)](#), send the following packet:

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

NOTE

The CFA635 will return the acknowledge packet immediately, then power cycle the host. The power cycle length is dependent on the length of the power pulse (command [28 \(0x1C\): Set ATX Power Switch Functionality](#)). After power cycling the host, the module will reboot itself. The module will not respond to new command packets for up to 3 seconds after its reboot. Part of this delay is the intentional staggered sequencing of turning on power to the fans. If you are not using fans, you can speed the boot process by setting the fan power to 0 (command [17 \(0x11\): Set Fan Power \(SCAB required\)](#)) and saving this as the default boot state (command [4 \(0x04\): Store Current State As Boot State](#)). Normally the host will be off or recovering from its own power cycle, so the boot delay of the module will not be of consequence.

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

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

6 (0x06): Clear LCD Screen

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

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

The return packet will be:

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

Clear LCD Screen is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

7 (0x07): Deprecated (See command [31 \(0x1F\): Send Data to LCD \(Pg. 48\)](#))

8 (0x08): Deprecated (See command [31 \(0x1F\): Send Data to LCD \(Pg. 48\)](#))

9 (0x09): Set LCD Special Character Data

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

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

data[1-8] are the bitmap information for this character. Any value is valid between 0 and 63, the msb is at the



left of the character cell of the row, and the lsb is at the right of the character cell.

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

Additionally, if you set bit 7 of any of the data bytes, the entire line will blink.

The return packet will be:

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

Set LCD Special Character Data is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

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

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

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

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

```
0x40 ( 64) to 0x7F (127) for CGRAM
0x80 (128) to 0x93 (147) for DDRAM, line 0
0xA0 (160) to 0xB3 (179) for DDRAM, line 1
0xC0 (192) to 0xD3 (211) for DDRAM, line 2
0xE0 (224) to 0xF3 (243) for DDRAM, line 3
```

The return packet will be:

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

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

11 (0x0B): Set LCD Cursor Position

This command allows the cursor to be placed at the desired location on the CFA635's LCD screen. If you want the cursor to be visible, you may also need to send a command [12 \(0x0C\): Set LCD Cursor Style \(Pg. 36\)](#).

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

The return packet will be:

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

Set LCD Cursor Position is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

12 (0x0C): Set LCD Cursor Style

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



```
type = 0x0C = 1210
valid data_length is 1
data[0] = cursor style (0-4 valid)
    0 = no cursor
    1 = blinking block cursor
    2 = underscore cursor
    3 = blinking block plus underscore
    4 = inverting, blinking block
```

The return packet will be:

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

Set LCD Cursor Style is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

13 (0x0D): Set LCD Contrast

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

```
type = 0x0D = 1310
valid data_length is 1
data[0] = contrast setting (0-254 valid)
    0-65 = very light
    66 = light
    95 = about right
    125 = dark
    126-254 = very dark
```

The return packet will be:

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

Set LCD Contrast is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

14 (0x0E): Set LCD & Keypad Backlight

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

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

The return packet will be:

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

Set LCD & Keypad Backlight is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

15 (0x0F): (Deprecated)

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

This command will configure the CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable to report the fan speed information to the host every 500 mS.



The return packet will be:

If data[0] is not 0, then the CFA635+[SCAB+WR-PWR-Y14](#) cable will start sending 0x81: Fan Speed Report packets for each enabled fan every 500 mS. (See [0x81: Fan Speed Report \(SCAB required\) \(Pg. 30\)](#).) 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 [SCAB Data Sheet \(http://www.crystallfontz.com/product/SCAB#docs\)](http://www.crystallfontz.com/product/SCAB#docs) for a detailed description.

This command will configure the power for the fan connectors. The fan power setting is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

The return packet will be:

Set Fan Power is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

When power is applied to the CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable, it detects any devices connected to the Dallas Semiconductor 1-Wire (DOW) bus and stores the device's information. This command will allow the host to read the device's information.



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

NOTE ON COMMAND 18: READ DOW DEVICE INFORMATION

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

These settings are factory default but may be changed by the user. Please see command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 48\)](#).

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

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

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

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

This setting must be saved as the boot state, so when the CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable reboots it will detect the DOW devices.

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

The return packet will be:

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

If data[1] is 0x22 ([DS1822](#) Econo 1-Wire Digital Thermometer temperature sensor) or 0x28 ([DS18B20](#) High Precision 1-Wire Digital Thermometer 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 Temperature Reporting \(SCAB required\) \(Pg. 39\)](#).

19 (0x13): Set Up Temperature Reporting (SCAB required)

This command will configure the CFA635 to report the temperature information to the host every second.



```
type = 0x13 = 1910
valid data_length is 4
data[0-3] = 32-bit bitmask indicating which temperature
            sensors fans are enabled to report (0-255
            valid in each location)
```

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

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

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

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

Any sensor enabled must have been detected as a 0x22 (DS1822 temperature sensor) or 0x28 (DS18B20 temperature sensor) during DOW enumeration. This can be verified by using the command [18 \(0x12\): Read DOW Device Information \(SCAB required\) \(Pg. 38\)](#).

The return packet will be:

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



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

The CFA635+[SCAB+WR-PWR-Y14](#) cable can function as an RS-232 to Dallas 1-Wire bridge. The CFA635+[SCAB+WR-PWR-Y14](#) cable can send up to 15 bytes and receive up to 14 bytes. This will be sufficient for many devices, but some devices require larger transactions and cannot be fully used with the CFA635+[SCAB+WR-PWR-Y14](#) cable. This command allows you to specify arbitrary transactions on the 1-Wire bus. 1-Wire commands follow this basic layout:

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

Please see [APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER \(SCAB REQUIRED\) \(Pg. 56\)](#) for an example of using this command.

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

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

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

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

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

The return packet will be:

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

21 (0x15): Deprecated

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

The LCD controller on the CFA635 is [S6A0073](#) compatible. Generally you won't need low-level access to the LCD controller but some arcane functions of the [S6A0073](#) are not exposed by the CFA635's command set. This command allows you to access the CFA635's LCD controller directly. Note: It is possible to corrupt the CFA635 display using this command.

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



The return packet will be:

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

23 (0x17): Configure Key Reporting

By default, the CFA635 reports any key event to the host. This command allows the key events to be enabled or disabled on an individual basis. The key events set to report are one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

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

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

The return packet will be:

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

Configure Key Reporting is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

24 (0x18): Read Keypad, Polled Mode

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

This command is independent of the key reporting masks set by command [23 \(0x17\): Configure Key Reporting \(Pg. 42\)](#). All keys are always visible to this command. Typically both masks of command 23 would be set to "0" if the host is reading the keypad in polled mode.

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

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

The return packet will be:

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



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

The combination of the CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable can be used as part of an active cooling system. For instance, the fans in a system can be slowed down to reduce noise when a system is idle or when the ambient temperature is low, and sped up when the system is under heavy load or the ambient temperature is high.

Since there are a very large number of ways to control the speed of the fans (thresholds, thermostat, proportional, PID, multiple temperature sensors “contributing” to the speed of several fans . . .) there was no way to foresee the particular requirements of your system and include an algorithm in the CFA635’s firmware that would be an optimal fit for your application.

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

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

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

type = 0x19 = 2510
data_length = 2
data[0] = bit mask of fans set to fail-safe (0-15 valid)
data[1] = timeout value in 1/8 second ticks:
    1 = 1/8 second
    2 = 1/4 second
    255 = 31 7/8 seconds
```

The return packet will be:

```
type = 0x40 | 0x19 = 0x59 = 8910
data_length = 0
```

26 (0x1A): Set Fan Tachometer Glitch Filter (SCAB required)

The combination of the CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable controls fan speed by using PWM. Using PWM turns the power to a fan on and off quickly to change the average power delivered to the fan. The CFA635 uses approximately 18 Hz for the PWM repetition rate. The fan’s tachometer output is only valid if power is applied to the fan. Most fans produce a valid tachometer output very quickly after the fan has been turned back on but some fans take time after being turned on before their tachometer output is valid.

This command allows you to set a variable-length delay after the fan has been turned on before the CFA635 will recognize transitions on the tachometer line. The delay is specified in counts, each count being nominally 552.5 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 \(Pg. 33\)](#).

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

The return packet will be:

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

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

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

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

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

The return packet will be:

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



28 (0x1C): Set ATX Power Switch Functionality

The combination of the CFA635+[SCAB+WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable can be used to replace the function of the power and reset switches in a standard ATX-compatible system. The ATX Power Switch Functionality is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

NOTE ON COMMAND 28: SET ATX POWER SWITCH FUNCTIONALITY

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

These settings are factory default but may be changed by the user. Please see command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 48\)](#). These settings must be saved as the boot state.

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

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

This configuration can be assured by sending the following command:

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

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

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

This configuration can be assured by sending the following command:

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

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

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

This configuration can be assured by sending the following command:

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

These settings must be saved as the boot state.

The RESET (GPIO[3]) and POWER CONTROL (GPIO[2]) lines on the CFA635+[SCAB+WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable are normally high-impedance. Electrically, they appear to be disconnected or floating. When the CFA635+[SCAB+WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable asserts the RESET or POWER CONTROL lines, they are momentarily driven high or low (as determined by the RESET_INVERT and POWER_INVERT bits, detailed below). To end the power or reset pulse, the CFA635+[SCAB+WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable changes the lines back to high-impedance.



FOUR FUNCTIONS ENABLED BY COMMAND 28

Function 1: KEYPAD_RESET

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

Function 2: KEYPAD_POWER_ON

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

Function 3: KEYPAD_POWER_OFF

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

Function 4: LCD_OFF_IF_HOST_IS_OFF

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

```
#define RESET_INVERT          0x02 //Reset pin drives high instead of low
#define POWER_INVERT         0x04 //Power pin drives high instead of low
#define LCD_OFF_IF_HOST_IS_OFF 0x10
#define KEYPAD_RESET         0x20
#define KEYPAD_POWER_ON      0x40
#define KEYPAD_POWER_OFF     0x80

type = 0x1C = 2810
data_length = 1 or 2
data[0]: bit mask of enabled functions
data[1]: (optional) length of power on & off pulses in 1/32 second
        1 = 1/32 sec
        2 = 1/16 sec
        16 = 1/2 sec
        255 = 8 sec
```

The return packet will be:

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

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

Some high-availability systems use hardware watchdog timers to ensure that a software or hardware failure does not result in an extended system outage. Once the host system has booted, a system monitor program is started. The system monitor program would enable the watchdog timer on the CFA635+[SCAB](#)+[WR-PWR-Y14](#) cable or



CFA635+[WR-PWR-Y25](#) cable. If the system monitor program fails to reset the watchdog timer, the CFA635+[SCAB+WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable 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 \(Pg. 45\)](#) or command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 48\)](#).

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

If timeout is 0, the watchdog is disabled.

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

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

If the command is not re-issued within timeout seconds, then the CFA635+[SCAB+WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable will reset the host (see command 28 for details). Since the watchdog is off by default when the CFA635+[SCAB+WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable powers up, the CFA635+[SCAB+WR-PWR-Y14](#) cable or CFA635+[WR-PWR-Y25](#) cable will not issue another host reset until the host has once again enabled the watchdog.

The return packet will be:

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

30 (0x1E): Read Reporting & Status

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

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

The return packet will be:

```
type = 0x40 | 0x1E = 0x5E = 9410
data_length = 15
data[0] = fan 1-4 reporting status (as set by command 16)
data[1] = temperatures 1-8 reporting status (as set by command 19)
data[2] = temperatures 9-15 reporting status (as set by command 19)
data[3] = temperatures 16-23 reporting status (as set by command 19)
data[4] = temperatures 24-32 reporting status (as set by command 19)
data[5] = key presses (as set by command 23)
data[6] = key releases (as set by command 23)
data[7] = ATX Power Switch Functionality (as set by command 28)
data[8] = current watchdog counter (as set by command 29)
data[9] = fan RPM glitch delay[0] (as set by command 26)
data[10] = fan RPM glitch delay[1] (as set by command 26)
data[11] = fan RPM glitch delay[2] (as set by command 26)
data[12] = fan RPM glitch delay[3] (as set by command 26)
data[13] = contrast setting (as set by command 13)
data[14] = backlight setting (as set by command 14)
```

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



31 (0x1F): Send Data to LCD

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

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

The return packet will be:

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

Send Data to LCD is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

32 (0x20): Reserved for CFA631 Key Legends

33 (0x21): Set Baud Rate

This command will change the CFA635's baud rate. The CFA635 will send the acknowledge packet for this command and change its baud rate to the new value. The host should send the baud rate command, wait for a positive acknowledge from the CFA635 at the old baud rate, and then switch itself to the new baud rate. The baud rate must be saved by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#) if you want the CFA635 to power up at the new baud rate.

The factory default baud rate is 115200.

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

The return packet will be:

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

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

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

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

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

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

The GPIOs also have "pull-up" and "pull-down" modes. These modes can be useful when using the GPIO as an input connected to a switch since no external pull-up or pull-down resistor is needed. For instance, the GPIO can be set to pull



up. Then when a switch connected between the GPIO and ground is open, reading the GPIO will return a "1". When the switch is closed, the input will return a "0".

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

NOTE ON SETTING AND CONFIGURING GPIO PINS

The GPIO pins may also be used for ATX control through the [SCAB](#)'s 7-pin connector and temperature sensing through the [SCAB](#)'s DOW header. By factory default, the GPIO output setting, function, and drive mode are set correctly to enable operation of the ATX and DOW functions. **The GPIO output setting, function, and drive mode must be set to the correct values in order for the ATX and DOW functions to work. Improper use of this command can disable the ATX and DOW functions.** The [635_WinTest](#) may be used to easily check and reset the GPIO configuration to the default state so the ATX and DOW functions will work.

The GPIO configuration is one of the items stored by the command [4 \(0x04\): Store Current State As Boot State \(Pg. 33\)](#).

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

data[0]: index of GPIO/GPO to modify
  0 = GPIO[0] = H1, Pin 11
  1 = GPIO[1] = H1, Pin 12 (default is ATX Host Power Sense)
  2 = GPIO[2] = H1, Pin 9 (default is ATX Host Power Control)
  3 = GPIO[3] = H1, Pin 10 (default is ATX Host Reset Control)
  4 = GPIO[4] = H1, Pin 13 (default is DOW I/O--has 1 KΩ hardware pull-up on SCAB)
  5 = GPO[ 5] = H2, Pin 15 = LED 3 (bottom) green die
  6 = GPO[ 6] = H2, Pin 13 = LED 3 (bottom) red die
  7 = GPO[ 7] = H2, Pin 11 = LED 2 green die
  8 = GPO[ 8] = H2, Pin 9 = LED 2 red die
  9 = GPO[ 9] = H2, Pin 7 = LED 1 green die
  10 = GPO[10] = H2, Pin 5 = LED 1 red die
  11 = GPO[11] = H2, Pin 3 = LED 0 (top) green die
  12 = GPO[12] = H2, Pin 1 = LED 0 (top) red die
  13-255 = reserved

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

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
```

(Continues on the next page.)



```
data[2] = Pin function select and drive mode (optional, 0-15 valid)
---- FDDD
||| | -- DDD = Drive Mode (based on output state of 1 or 0)
=====
000: 1=Fast, Strong Drive Up, 0=Resistive Pull Down
001: 1=Fast, Strong Drive Up, 0=Fast, Strong Drive Down
010: Hi-Z, use for input
011: 1=Resistive Pull Up,      0=Fast, Strong Drive Down
100: 1=Slow, Strong Drive Up, 0=Hi-Z
101: 1=Slow, Strong Drive Up, 0=Slow, Strong Drive Down
110: reserved, do not use
111: 1=Hi-Z,                  0=Slow, Strong Drive Down

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

The return packet will be:

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

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

Please see command [34 \(0x22\): Set or Set and Configure GPIO Pin \(Pg. 48\)](#) for details on the GPIO architecture.

```
type: 0x23 = 3510
data_length: 1
data[0]: index of GPIO to query
0 = GPIO[0] = H1, Pin 11
1 = GPIO[1] = H1, Pin 12 (default is ATX Host Power Sense)
2 = GPIO[2] = H1, Pin 9 (default is ATX Host Power Control)
3 = GPIO[3] = H1, Pin 10 (default is ATX Host Reset Control)
4 = GPIO[4] = H1, Pin 13 (default is DOW I/O--has 1 K $\Omega$  hardware pull-up on SCAB)
5 = GPO[ 5] = H2, Pin 15 = LED 3 (bottom) green die
6 = GPO[ 6] = H2, Pin 13 = LED 3 (bottom) red die
7 = GPO[ 7] = H2, Pin 11 = LED 2 green die
8 = GPO[ 8] = H2, Pin 9 = LED 2 red die
9 = GPO[ 9] = H2, Pin 7 = LED 1 green die
10 = GPO[10] = H2, Pin 5 = LED 1 red die
11 = GPO[11] = H2, Pin 3 = LED 0 (top) green die
12 = GPO[12] = H2, Pin 1 = LED 0 (top) red die
13-255 = reserved
```

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

The return packet will be:

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

```

data[0] = index of GPIO to read
data[1] = Pin state & changes since last poll
    Only useful for GPIOs (index 0-4). GPOs (index of 5-12) will return 0.
    ---- -RFS Enable Reporting of this Fan's Tach Input
    ||||| |-- S = state at the last reading
    ||||| |-- F = at least one falling edge has
    |||||         been detected since the last poll
    ||||| |-- R = at least one rising edge has
    |||||         been detected since the last poll
    ||||| ----- reserved

(This reading is the actual pin state, which may
or may not agree with the pin setting, depending
on drive mode and the load presented by external
circuitry. The pins are polled at approximately
32 Hz asynchronously with respect to this command.
Transients that happen between polls will not be
detected.)
data[2] = Requested Pin level/PWM level
    0-100: Output duty cycle percentage
    (This value is the requested PWM duty cycle. The
    actual pin may or may not be toggling in agreement
    with this value, depending on the drive mode and
    the load presented by external circuitry)
data[3] = Pin function select and drive mode
    ---- FDDD
    ||||| -- DDD = Drive Mode
    ||||| =====
    ||||| 000: 1=Fast, Strong Drive Up, 0=Resistive Pull Down
    ||||| 001: 1=Fast, Strong Drive Up, 0=Fast, Strong Drive Down
    ||||| 010: Hi-Z, use for input
    ||||| 011: 1=Resistive Pull Up,      0=Fast, Strong Drive Down
    ||||| 100: 1=Slow, Strong Drive Up, 0=Hi-Z
    ||||| 101: 1=Slow, Strong Drive Up, 0=Slow, Strong Drive Down
    ||||| 110: reserved
    ||||| 111: 1=Hi-Z,                  0=Slow, Strong Drive Down

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

```



CHARACTER GENERATOR ROM (CGROM)

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

Character Generator ROM (CGROM) for CrystalFontz CFA-635

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

Figure 11. Character Generator ROM (CGROM)



MODULE RELIABILITY AND LONGEVITY

MODULE RELIABILITY

ITEM	SPECIFICATION	
LCD portion (excluding Keypad, Indicator LEDs, and Backlights)	50,000 to 100,000 hours (typical)	
Keypad	1,000,000 keystrokes	
Bicolor LED Indicators	50,000 to 100,000 hours (typical)	
White LED Display and Blue LED Keypad Backlights	<i>Power-On Hours</i>	<i>% of Initial Brightness (New Module)</i>
<i>Note: We recommend that the backlight of white LED backlit modules be dimmed or turned off during periods of inactivity to conserve the white LED backlight lifetime.</i>	<10,000	>90%
	<50,000	>50%

MODULE LONGEVITY (EOL / REPLACEMENT POLICY)

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

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

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

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

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

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



CARE AND HANDLING PRECAUTIONS

For optimum operation of the CFA635-TMF-KU and to prolong its life, please follow the precautions described below.

ELECTROSTATIC DISCHARGE (ESD) SPECIFICATIONS

D+ and D- pins of USB connector only: Electrostatic Discharge Voltage ($I < 1 \text{ uA}$): +/- 2000 v.

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 CFA635-TMF-KU ships with a protective film over the polarizer. Please peel off the protective film slowly. Peeling off the protective film abruptly may generate static electricity.
- The polarizer is made out of soft plastic and is easily scratched or damaged. When handling the module, avoid touching the polarizer. Finger oils are difficult to remove.
- To protect the soft plastic polarizer from damage, place a transparent plate (for example, acrylic, polycarbonate, or glass) in front of the CFA635-TMF-KU, leaving a small gap between the plate and the display surface. We recommend GE HP-92 Lexan, which is readily available and works well.
- Do not disassemble or modify the module.
- Do not modify the eight tabs of the metal bezel or make connections to them.
- Solder only to the I/O terminals. Use care when removing solder so you do not damage the PCB.
- Do not reverse polarity to the power supply connections. Reversing polarity will immediately ruin the module.

AVOID SHOCK, IMPACT, TORQUE, OR TENSION

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

IF LCD PANEL BREAKS

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

HOW TO CLEAN

- The polarizer (laminated to the glass) is soft plastic. The soft plastic is easily scratched or damaged. Damage will be especially obvious on a “negative” module (a module that appear dark when power is “off”). Be very careful when you clean the polarizer.
- Do not clean the polarizer with liquids. Do not wipe the polarizer with any type of cloth or swab (for example, Q-tips).



- Use the removable protective film to remove smudges (for example, fingerprints) and any foreign matter. If you no longer have the protective film, use standard transparent office tape (for example, Scotch® brand “Crystal Clear Tape”). If the polarizer is dusty, you may carefully blow it off with clean, dry, oil-free compressed air.
- CFA635-TMF-KU *without Crystalfontz overlay*: The exposed surface of the LCD “glass” is actually the front polarizer laminated to the glass. The polarizer is made out of a fairly soft plastic and is easily scratched or damaged. The polarizer will eventually become hazy if you do not take great care when cleaning it. Long contact with moisture (from condensation or cleaning) may permanently spot or stain the polarizer.

OPERATION

- Your circuit should be designed to protect the module from ESD and power supply transients.
- Observe the operating temperature limitations: a minimum of 0°C to a maximum of 50°C noncondensing with minimal fluctuation. Operation outside of these limits may shorten life and/or harm display.
 - 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.
- 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.
- Observe the storage temperature limitations: a minimum of -10°C minimum to +60°C non-condensing maximum with minimal fluctuations. Rapid temperature changes can cause moisture to form, resulting in permanent damage.
- Do not allow weight to be placed on the modules while they are in storage.
- Please recycle your outdated Crystalfontz modules at an approved facility.

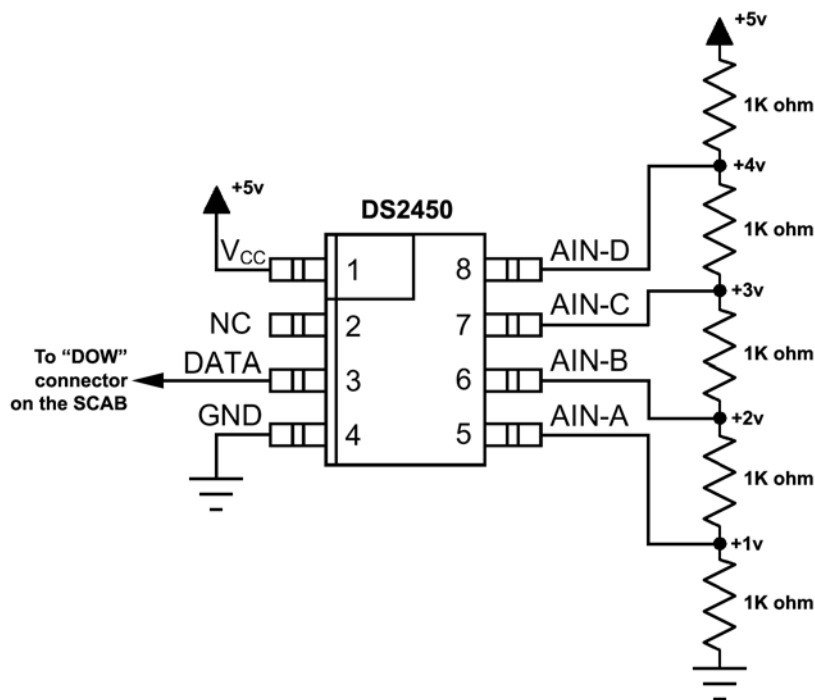


APPENDIX A: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER (SCAB REQUIRED)

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

Up to 32 DOW devices can be connected to the CFA635+[SCAB](#)+[WR-PWR-Y14](#). In this example the DS2450 appears at device index 0. Your software should query the connected devices using command [18 \(0x12\): Read DOW Device Information \(SCAB required\) \(Pg. 38\)](#) to verify the locations and types of DOW devices connected in your application.

Please refer to the [DS2450 Data Sheet](#) and the description for command [20 \(0x14\): Arbitrary DOW Transaction \(SCAB required\) \(Pg. 41\)](#) more information.



Appendix A Figure 1. CFA635 Test Circuit Schematic

Start [635 WinTest](#) and open the Packet Debugger dialog.

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

This appendix describes connecting a Dallas Semiconductor DS1963S Monetary iButton with SHA-1 Challenge Response Algorithm and 4KB of nonvolatile RAM to the CFA635+[SCAB](#)+[WR-PWR-Y14](#)'s DOW (Dallas One Wire) connector. It also gives a sample command sequence to read and write the DS1963S's scratch memory.

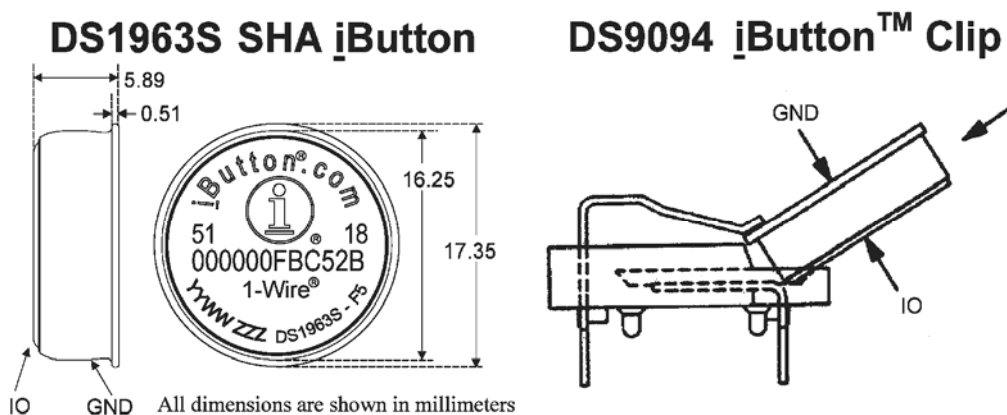
The DS1963S can be used as a secure dongle to protect your system's application software from being copied. Even if the communication channel is compromised or the host is not authentic, the SHA algorithm ensures that the data is still secure. Please see the following Maxim/Dallas white papers and application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

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

Up to 32 DOW devices can be connected to the CFA635+[SCAB](#)+[WR-PWR-Y14](#). In this example the DS1963S appears at device index 0. Your software should query the connected devices using command [18 \(0x12\): Read DOW Device Information \(SCAB required\) \(Pg. 38\)](#) to verify the locations and types of DOW devices connected in your application.

Please refer to the [DS1963S Data Sheet](#) and the description for command [20 \(0x14\): Arbitrary DOW Transaction \(SCAB required\) \(Pg. 41\)](#) for more information.

To connect the DS1963S to the CFA635+[SCAB](#)+[WR-PWR-Y14](#), simply make one connection between the DS1963S's "GND" terminal and the CFA635+[SCAB](#)+[WR-PWR-Y14](#) DOW connector's GND pin, and a second connection between the DS1963S's "IO" pin and the CFA635+[SCAB](#)+[WR-PWR-Y14](#) DOW connector's I/O pin. By using a DS9094 iButton Clip, the connection is easy.



Appendix B Figure 1. Connect CFA635 to Maxim/Dallas DS19632 SHA iButton using DS9094 iButton Clip



To demonstrate reading and writing the scratch memory on DS1963S, open the 635_WinTest Packet Debugger dialog and use it to experiment with the following commands: Erase Scratchpad, Read Scratchpad, and Write Scratchpad.

To use the full power of the DS1963S, a program based on the Dallas/Maxim application notes listed above is needed. The challenge/response sequence would be unwieldy to demonstrate using the 635_WinTest Packet Debugger dialog.

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

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

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

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

Erase Scratchpad [C3h]

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

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

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

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

Read Scratchpad Command [AAh]

HIDE = 0:

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

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

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



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

Write Scratchpad Command [0Fh]

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

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

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

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

Command:

20 = Arbitrary DOW transaction

Data sent:

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

Data received:

C=84 (d=0) : 00

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

Command:

20 = Arbitrary DOW transaction

Data sent:

\000\013\xAA

Data received:

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

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

Command:

20 = Arbitrary DOW transaction

Data sent:

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

Data received:

C=84 (d=0) : 00

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

Command:

20 = Arbitrary DOW transaction

Data sent:

\000\013\xAA

Data received:

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

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



APPENDIX C: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC)

SAMPLE CODE

Free downloadable code on our website:

- ☐ Windows compatible test/demonstration program and source.
<http://www.crystalfontz.com/product/635WinTest.html>
- ☐ Linux compatible command-line demonstration program with C source code. 8K.
http://www.crystalfontz.com/product/linux_cli_examples.html
- ☐ Supported by CrystalControl freeware.
<http://www.crystalfontz.com/product/CrystalControl2.html>
- ☐ Windows USB driver and installation instructions.
http://www.crystalfontz.com/product/USB_LCD_Driver.html

ALGORITHMS TO CALCULATE THE CRC

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

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

Algorithm 1: "C" Table Implementation

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

```
//This code is from the IRDA LAP documentation, which appears to
//have been copied from PPP:
//
// http://irda.affiniscap.com/associations/2494/files/Specifications/
//IrLAP11_Plus_Errata.zip
//
//I doubt that there are any worries about the legality of this code,
//searching for the first line of the table below, it appears that
//the code is already included in the linux 2.6 kernel "Driver for
//ST5481 USB ISDN modem". This is an "industry standard" algorithm
//and I do not think there are ANY issues with it at all.
typedef unsigned char ubyte;
typedef unsigned short word;
word get_crc(ubyte *bufptr, word len)
{
    //CRC lookup table to avoid bit-shifting loops.
    static const word crcLookupTable[256] =
    {0x00000, 0x01189, 0x02312, 0x0329B, 0x04624, 0x057AD, 0x06536, 0x074BF,
    0x08C48, 0x09DC1, 0x0AF5A, 0x0BED3, 0x0CA6C, 0x0DBE5, 0x0E97E, 0x0F8F7,
    0x01081, 0x00108, 0x03393, 0x0221A, 0x056A5, 0x0472C, 0x075B7, 0x0643E,
    0x09CC9, 0x08D40, 0x0BFDB, 0x0AE52, 0x0DAED, 0x0CB64, 0x0F9FF, 0x0E876,
    0x02102, 0x0308B, 0x00210, 0x01399, 0x06726, 0x076AF, 0x04434, 0x055BD,
    0x0AD4A, 0x0BCC3, 0x08E58, 0x09FD1, 0x0EB6E, 0x0FAE7, 0x0C87C, 0x0D9F5,
    0x03183, 0x0200A, 0x01291, 0x00318, 0x077A7, 0x0662E, 0x054B5, 0x0453C,
    0x0BDCB, 0x0AC42, 0x09ED9, 0x08F50, 0x0FBEF, 0x0EA66, 0x0D8FD, 0x0C974,
```



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

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

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

Algorithm 2: "C" Bit Shift Implementation

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

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



```
{
//Shift the CRC accumulator
newCRC>>=1;

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

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

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

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



Algorithm 2B: "C" Improved Bit Shift Implementation

This is a simplified algorithm that implements the CRC.

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

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

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

Algorithm 3: "PIC Assembly" Bit Shift Implementation

This routine was graciously donated by one of our customers.

```
;=====
; CrystalFontz CFA635 PIC CRC Calculation Example
;
; This example calculates the CRC for the hard coded example provided
; in the documentation.
;
; It uses "This is a test. " as input and calculates the proper CRC
; of 0x93FA.
;=====
#include "p16f877.inc"
;=====
; CRC16 equates and storage
;-----
accuml      equ    40h        ; BYTE - CRC result register high byte
accumh      equ    41h        ; BYTE - CRC result register high low byte
datareg     equ    42h        ; BYTE - data register for shift
j           equ    43h        ; BYTE - bit counter for CRC 16 routine
Zero        equ    44h        ; BYTE - storage for string memory read
index       equ    45h        ; BYTE - index for string memory read
savchr      equ    46h        ; BYTE - temp storage for CRC routine
```



```
;
seedlo      equ      021h      ; initial seed for CRC reg lo byte
seedhi      equ      0F3h      ; initial seed for CRC reg hi byte
;
polyL       equ      008h      ; polynomial low byte
polyH       equ      084h      ; polynomial high byte
;=====
; CRC Test Program
;-----
          org          0          ; reset vector = 0000H
;
          clrfs         PCLATH     ; ensure upper bits of PC are cleared
          clrfs         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
;
main      org          20          ; start of main program
          movlw         seedhi      ; setup initial CRC seed value.
          movwf         accumh      ; This must be done prior to
          movlw         seedlo      ; sending string to CRC routine.
          movwf         accuml      ;
          clrfs         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     clrfs         ; 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 CFA635 packet) data. This routine was adapted from the C table implementation. The complete project can be found in our forums.

```

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

```

```

Private Type WORD
    Lo As Byte
    Hi As Byte
End Type

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

Dim crcLookupTable(256) As WORD

Private Sub MSComm_OnComm()
'Leave this here
End Sub

```



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

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

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

Algorithm 5: “Java” Table Implementation

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

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



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

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

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

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

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



```

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

```

Algorithm 6: “Perl” Table Implementation

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

```

#!/usr/bin/perl

use strict;

my @CRC_LOOKUP =
(0x00000, 0x01189, 0x02312, 0x0329B, 0x04624, 0x057AD, 0x06536, 0x074BF,
0x08C48, 0x09DC1, 0x0AF5A, 0x0BED3, 0x0CA6C, 0x0DBE5, 0x0E97E, 0x0F8F7,
0x01081, 0x00108, 0x03393, 0x0221A, 0x056A5, 0x0472C, 0x075B7, 0x0643E,
0x09CC9, 0x08D40, 0x0BFDB, 0x0AE52, 0x0DAED, 0x0CB64, 0x0F9FF, 0x0E876,
0x02102, 0x0308B, 0x00210, 0x01399, 0x06726, 0x076AF, 0x04434, 0x055BD,
0x0AD4A, 0x0BCC3, 0x08E58, 0x09FD1, 0x0EB6E, 0x0FAE7, 0x0C87C, 0x0D9F5,
0x03183, 0x0200A, 0x01291, 0x00318, 0x077A7, 0x0662E, 0x054B5, 0x0453C,
0x0BDCB, 0x0AC42, 0x09ED9, 0x08F50, 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) ;

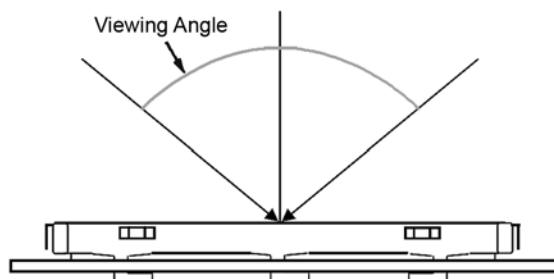
# print out the crc in hex
printf("%x\n", $crc);
```



APPENDIX D: QUALITY ASSURANCE STANDARDS

INSPECTION CONDITIONS

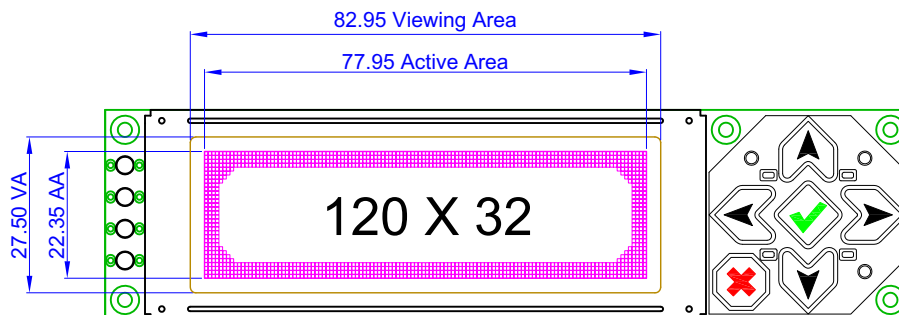
- Environment
 - Temperature: $25 \pm 5^{\circ}\text{C}$
 - Humidity: 30~85% RH
- For visual inspection of active display area
 - Source lighting: two 20 Watt or one 40 Watt fluorescent light
 - Display adjusted for best contrast
 - Viewing distance: 30 ± 5 cm (about 12 inches)
 - Viewable angle: inspect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



COLOR DEFINITIONS

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

DEFINITION OF ACTIVE AREA AND VIEWABLE AREA





ACCEPTANCE SAMPLING

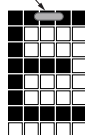
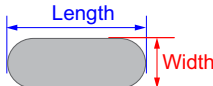
DEFECT TYPE	AQL*
Major	$\leq 0.65\%$
Minor	$\leq 1.00\%$
*Acceptable Quality Level: maximum allowable error rate or variation from standard	

DEFECTS CLASSIFICATION

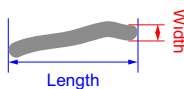
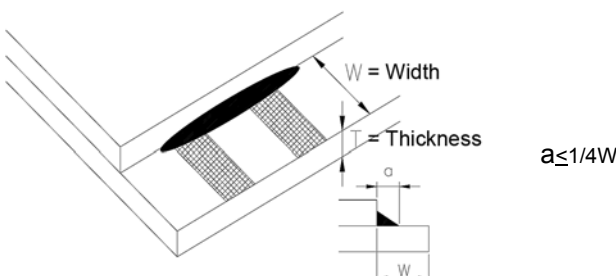
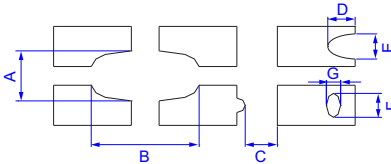
Defects are defined as:

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

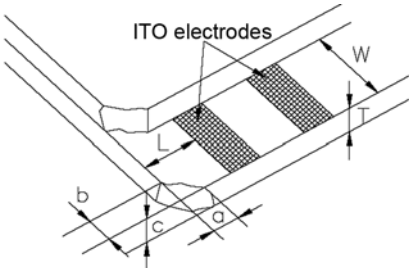
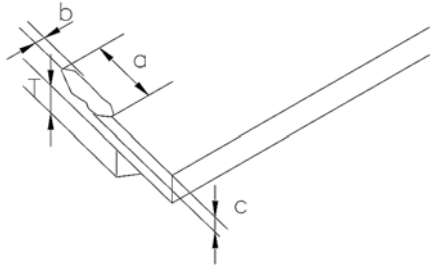
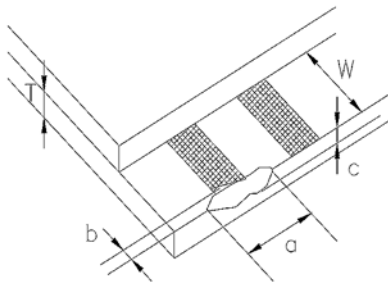
ACCEPTANCE STANDARDS

#	DEFECT TYPE	CRITERIA			MAJOR / MINOR
1	Electrical defects	1. No display, display malfunctions, or shorted segments. 2. Current consumption exceeds specifications.			Major
2	Viewing area defect	Viewing area does not meet specifications. (See Inspection Conditions (Pg. 71)).			Major
3	Contrast adjustment defect	Contrast adjustment fails or malfunctions.			Major
4	Blemishes or foreign matter on display segments	<div> <div>Blemish</div>  </div>	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	<div> <div>Defect size = (A + B)/2</div>  </div>	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	

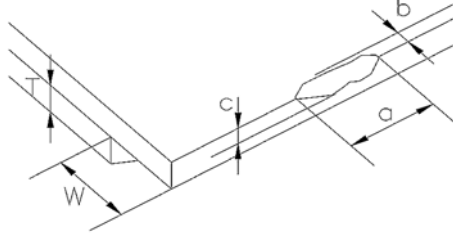
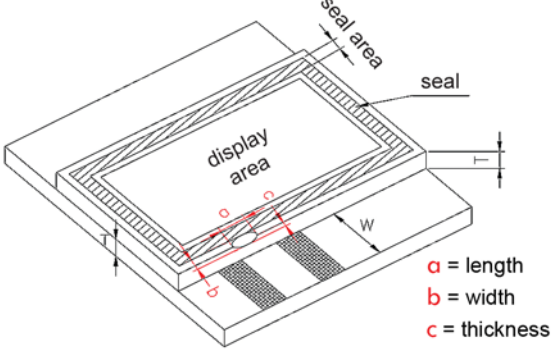


#	DEFECT TYPE	CRITERIA			MAJOR / MINOR
6	Dark lines or scratches in display area 	Defect Width (mm)	Defect Length (mm)	Acceptable Qty	Minor
		≤0.03	≤3.0	3	
		0.03 to 0.05	≤2.0	2	
		0.05 to 0.08	≤2.0	1	
		0.08 to 0.10	≤3.0	0	
		≥0.10	>3.0	0	
7	Bubbles between polarizer film and glass	Defect Size (mm)	Acceptable Qty	Minor	
		≤0.20	Ignore		
		0.20 to 0.40	3		
		0.40 to 0.60	2		
		≥0.60	0		
8	Glass rest defect				Minor
9	Display pattern defect 	Dot Size (mm)	Acceptable Qty	Minor	
		$((A+B)/2) \leq 0.2$	≤ 3 total defects ≤ 2 pinholes per digit		
		C>0			
		$((D+E)/2) \leq 0.25$			
		$((F+G)/2) \leq 0.25$			



#	DEFECT TYPE	CRITERIA				MAJOR / MINOR
10	Chip in corner					Minor
		<i>a</i>	<i>b</i>	<i>c</i>	<i>Acceptable Qty</i>	
		≤4 mm	≤W	c≤T	3	
11	Chip on “non-contact” edge of LCD					Minor
		<i>a</i>	<i>b</i>	<i>c</i>		
		≤3 mm	≤1 mm	≤T		
		≤4 mm	≤1.5 mm	≤T		
12	Chip on “contact” edge of LCD, on the active side					Minor
		<i>a</i>	<i>b</i>	<i>c</i>	<i>Acceptable Qty</i>	
		≤2 mm	≤W/4	≤T	Ignore	
		≤3 mm	≤W/4	≤T	3	



#	DEFECT TYPE	CRITERIA	MAJOR / MINOR
13	Chip on "contact" edge of LCD, on the inactive side		Minor
14	Chip in seal area	 <p> a = length b = width c = thickness </p>	Minor
15	Backlight defects	1. Light fails or flickers.* 2. Color and luminance do not correspond to specifications.* 3. Exceeds standards for display's blemishes or foreign matter (see test 5, Pg. 72), and dark lines or scratches (see test 6, Pg. 73). *Minor if display functions correctly. Major if the display fails.	Minor
16	COB defects	1. Pinholes >0.2 mm. 2. Seal surface has pinholes through to the IC. 3. More than 3 locations of sealant beyond 2 mm of the sealed areas.	Minor
17	PCB defects	1. Oxidation or contamination on connectors.* 2. Wrong parts, missing parts, or parts not in specification.* 3. Jumpers set incorrectly. 4. Solder (if any) on bezel, LED pad, zebra pad, or screw hole pad is not smooth. *Minor if display functions correctly. Major if the display fails.	Minor



#	DEFECT TYPE	CRITERIA	MAJOR / MINOR
18	Soldering defects	<ul style="list-style-type: none">1. Unmelted solder paste.2. Cold solder joints, missing solder connections, or oxidation.*3. Solder bridges causing short circuits.*4. Residue or solder balls.5. Solder flux is black or brown. <i>*Minor if display functions correctly. Major if the display fails.</i>	Minor