Crystalfontz America, Incorporated

INTELLIGENT I^{2}C
LCD MODULE SPECIFICATIONS
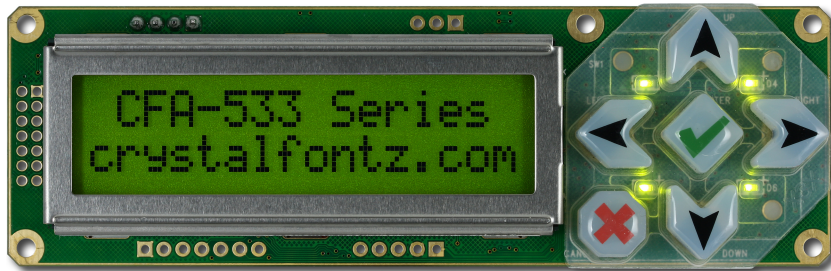
Crystalfontz Model Number	**CFA533-YYH-KC**
Hardware Version	**Revision 1.0, March 2010**
Firmware Version	**Revision c1.0, April 2010**
Data Sheet Version	**Revision 2.0, March 2011**
Product Pages	http://www.crystalfontz.com/product/CFA533-YYH-KC.html

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 2

REVISION HISTORY

HARDWARE	
2010/06/06	Current hardware version: **v1.0** (revision number has not changed) Changes: We are transitioning to an improved keypad from "10.5" millimeters to "12.00" millimeters high. See PCN 10282.

2010/03/18	Hardware version: v1.0
	Changes since hardware version (v0.1):
	As the inventory of hardware version v0.1 depletes, we are phasing in the improved v1.0.
	The v1.0 should be a drop-in replacement for any v0.1 application.
	1. Added a JPUSBSENSE jumper to help user configuration. Does not apply to I2C. See Jumper Locations and Functions (Pg. 13).
	2. Changed JPFG to RFG. RFG connects 1MΩ between logic ground and frame ground. See Jumper Locations and Functions (Pg. 13).
	3. PCB layout changes:
	    - Added more breaks to the frameground (FG) trace that surrounds the board so there is not a loop.
	    - Added masking on FG trace where connectors may be hand soldered to ease assembly.
	    - Moved vias from under the zebra near the ends of the bezel to assure no shorts between the vias and the bezel.
	    - Made mounting hole annular ring (pads) larger to avoid component damage by tools during assembly.
	    - Resistors are loaded wherever needed.
	4. Module weight increased from "40" grams to "41" grams.

2007/12/31	Hardware version: v0.1 New module.

FIRMWARE	
2010/05/15	Current firmware version: **c1.0** Changes since last version (i0.2): Made firmware version match hardware version.
2009/05/15	Firmware version: i0.2 New firmware.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 3

DATA SHEET

2011/03/07	Current Data Sheet version: **v2.0**
	Changes since Data Sheet version (v1.0):
	● Wherever listed, changed part numbers for I^{2}C variants of CFA533 modules to end in "-KC" instead of "-KI".
	  - Part number "CFA533-TMI-KI" changed to "CFA533-TMI-KC".
	  - Part number "CFA533-YYH-KI" changed to "CFA533-YYH-KC".
	● Wherever listed, deleted dash ("-") from module part numbers to match how they now appear on our website without the dash ("-").
	● Wherever listed or shown in illustration, changed keypad from "10.5" millimeters to "12.00" millimeters high. This transition started 2010/06/06. See PCN 10282.
	● In Features (Pg. 8), added
	  - "RoHS compliant".
	  - "Factories are ISO certified".
	  - Materials are in compliance with EU Directive "REACH'.
	  - Note on CE approved modules.
	● In Physical Characteristics (Pg. 10),
	  - Changed module nominal depth specification to include keypad "12.00" millimeters high (formerly 10.5 millimeters high).
	  - Added a maximum depth specification of "25.90".
	  - Module weight increased from "40" grams to "41" grams.
	● Updated information in Jumper Locations and Functions (Pg. 13) due to hardware revision v1.0. Changed "JPFG" to "RFG". RFG connects 1MΩ between logic ground and frame ground.
	● Added OPTICAL CHARACTERISTICS (Pg. 18) section, with definition of viewing angles.
	● Added Humidity Range specification in Temperature and Humidity Ratings (Pg. 19).
	● In CONNECTION INFORMATION (Pg. 21).
	  - Based on hardware revision v1.0, revised illustrations on all power and control connections.
	  - Because in hardware version 1.0 resistors are loaded by default, all references to "resistors not loaded" were removed.
	  - Added hyperlinks to recommended Crystalfontz cables.
	● In command 4 (0x04): Store Current State as Boot State (Pg. 38), added Errata Note. Please read boxed note.
	● In command 8 (0x08): Set LCD Contents, Line 2 (Pg. 41), corrected screen display text from
	`data[] = TOP line's display content (must supply 16 bytes)`
	 to
	`data[] = BOTTOM line's display content (must supply 16 bytes)`
	● In command 11 (0x0B): Set LCD Cursor Position (Pg. 42) , corrected from
	`data[0] = column (0-19 valid)`
	`data[1] = row (0-3 valid)`
	 to
	`data[0] = column (0-15 valid)`
	`data[1] = row (0-1 valid)`
	● In command 12 (0x0C): Set LCD Cursor Style (Pg. 42), corrected from "`3 = blinking block plus underscore`" to "`3 = blinking underscore`". This behavior is not the same as the CFA633 series which is "`3 = blinking block plus underscore`".
	● In command 22 (0x16): Send Command Directly to the LCD Controller (Pg. 46), corrected "*The Neotec NT7070B controller on the CFA533 is **S6A0073** compatible.*" to "*The controller on the CFA533 is a Neotec NT7070B (**HD44780** compatible).*"

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 4

DATA SHEET (Continued)	

2011/03/07	• In command 35 (0x23): Read GPIO Pin Levels and Configuration State (Pg. 54),
	   - Replaced upper case "X" with lower case "x".
	   - Corrected "`data length`" from "`4`" to "`1`".
	 • In APPENDIX B: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC) (Pg. 66),
	   - In sample code for Algorithm 1: "C" Table Implementation (Pg. 66) and Algorithm 2: "C" Bit Shift Implementation (Pg. 67), added typedefs for "`ubyte`" and "`word`".
	   - Added sample code Algorithm 7: For PIC18F8722 or PIC18F2685 (Pg. 74) .
	 • In addition to the list above, made minor changes in text and illustrations to improve clarity wherever needed.

2009/07/10	Data Sheet version: 1.0 New Data Sheet.

The Fine Print

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 5

CONTENTS

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 6

CONTENTS, CONTINUED

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 7

LIST OF FIGURES

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 8

MAIN FEATURES

COMPARISON TO CFA633

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

FEATURES

❑ 16 characters x 2 lines LCD with keypad and high-level interface. Will fit nicely in a 1U rack mount case (35 mm overall height).

❑ Only a single supply is needed. Wide power supply voltage range (V_{DD} = +3.3v to +5.0v) is perfect for embedded systems.

❑ Backlight and contrast are fully voltage compensated over the power supply range. No adjustments to the contrast setting or backlight brightness are needed.

❑ The CFA533 series is mechanically similar to the the CFA633 series. The CFA533 series command set is compatible with the CFA633 series. The CFA533 can be used as a "drop-in" replacement for most CFA633 series applications that do not need fan capabilities.

❑ Industry standard Philips® I^{2}C bus compatible interface.

● Only two pins (SDA and SCL) required to interface to I^{2}C bus.

● Standard data rate of 100/400 kbps, also supports 50 kbps.

● 7-bit addressing mode.

❑ Edge-lit yellow-green LED backlight with positive yellow-green STN LCD. Displays dark characters on yellow background). Integrated yellow LED backlit 6-button translucent silicon keypad.

❑ Positive mode display is sunlight readable and also readable in dark areas.

❑ Fully decoded keypad: any key combination is valid and unique.

❑ Robust packet-based communications protocol with 16-bit CRC.

❑ Nonvolatile memory capability (EEPROM):

● Customize the "power-on" display settings.

● 16-byte "scratch" register for storing data such as IP address, netmask, system serial number . . .

❑ Optional capabilities. Crystalfontz can make these modifications for you. A semi-custom part number and minimum order may apply.

● ATX power supply control functionality allows the buttons on the CFA533-YYH-KC to replace the Power and Reset switches on your system, simplifying front panel design. The ATX functionality can also implement a hardware watchdog that can reset host system on host software failure.

● Temperature monitoring: up to 32 channels at up to 0.5 degrees Celsius with absolute accuracy (using optional connector and Crystalfontz WR-DOW-Y17 cable with Dallas 1-Wire sensor).

● "Live Display" shows up to four temperature readings without host intervention, allowing temperatures to be shown immediately at boot, even before the host operating system is loaded.

● I^{2}C 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 are ISO certified.

❑ Product materials are in compliance with the regulations related to the EU Directive 2006/121/EC for Registration, Evaluation, Authorization and Restriction of Chemicals (REACH).

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 9

❑ If you need a CE approved module, please see our XES635 USB series (http://www.crystalfontz.com/product/ XES635BK-YYE-KU.html, http://www.crystalfontz.com/product/XES635BK-TMF-KU.html, or http://www.crystal-fontz.com/product/XES635BK-TFE-KU.html. The CFA533-YYH-KC does not have CE certification because it is not an end product. The module requires power and communications from another system in order to operate.

MODULE CLASSIFICATION INFORMATION

$$\underline{CFA} \quad \underline{533} \quad - \quad \underline{Y} \quad \underline{Y} \quad \underline{H} \quad - \quad \underline{K} \quad \underline{C}$$
❶   ❷   ❸ ❹ ❺   ❻ ❼

❶	Brand	Crystalfontz America, Inc.
❷	Model Identifier	533
❸	Backlight Type & Color	Y – LED, yellow-green
❹	Fluid Type, Image (positive or negative), & LCD Glass Color	Y – STN, positive yellow-green
❺	Polarizer Film Type, Temperature Range, & View Angle (O 'Clock)	H – Transflective, Wide Temperature[1], 6:00
❻	Special Code 1	K – Manufacturer's codes
❼	Special Code 2	C – I²C interface

[1]*Wide Temperature Range is -20°C minimum to +70°C maximum*

ORDERING INFORMATION

PART NUMBER	FLUID	LCD GLASS COLOR	IMAGE	POLARIZER FILM	BACKLIGHT COLOR/TYPE	
CFA533-YYH-KC (I²C)	STN	yellow-green	positive	transflective	LCD: yellow-green edge LEDs Keypad: yellow-green LEDs	
CFA533-TMI-KC (I²C)	STN	blue	negative	transmissive	LCD: white edge LEDs Keypad: blue LEDs	
CFA533-TMI-KL ("logic-level" RS-232)	STN	blue	negative	transmissive	LCD: white edge LEDs Keypad: blue LEDs	
CFA533-TMI-KS ("full swing" RS-232)	STN	blue	negative	transmissive	LCD: white edge LEDs Keypad: blue LEDs	
CFA533-TMI-KU (USB)	STN	blue	negative	transmissive	LCD: white edge LEDs Keypad: blue LEDs	
CFA533-YYH-KL ("logic-level" RS-232)	STN	yellow-green	positive	transflective	LCD: yellow-green edge LEDs Keypad: yellow-green LEDs	
CFA533-YYH-KS ("full swing" RS-232)	STN	yellow-green	positive	transflective	LCD: yellow-green edge LEDs Keypad: yellow-green LEDs	
CFA533-YYH-KU (USB)	STN	yellow-green	positive	transflective	LCD: yellow-green edge LEDs Keypad: yellow-green LEDs	

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 10

MECHANICAL SPECIFICATIONS

PHYSICAL CHARACTERISTICS

ITEM	SIZE
Module Overall Dimensions	
Width and Height	110.5 (W) x 35.0 (H)
Depth with Keypad, with Connectors	25.60 mm nominal 25.90 mm maximum
Viewing Area	61.0 (W) x 15.8 (H) mm
Active Area	56.2 (W) x 11.5 (H) mm
Character Size	2.95 (W) x 5.55 (H) mm
Character Pitch	3.55 (W) x 5.95 (H) mm
Pixel Size	0.55 (W) x 0.65 (H) mm
Pixel Pitch	0.60 (W) x 0.70 (H) mm
Keystroke Travel (approximate)	2.4 mm
Weight	41 grams (typical)

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

Figure 1. Module Outline Drawings (two page below))

110.50±0.50 Overall (PCB)
75.00 PCB Mounting Holes
30.50 PCB M.H.
68.70 Bezel
20.00
61.00 Viewing Area
56.20 Active Area
PWR
DOW
2.50
4.25
9.60
11.75

35.00±0.50 Overall (PCB)
30.00 PCB M.H.
26.50 Bezel
15.80 VA
11.50 AA

2.50
5.65
9.50
11.90

See Character
Detail A

25.90 Maximum
25.60 Nominal
12.00 [2]Keypad

I²C

3.55
2.95
.40
5.95
5.55
.60

See Pixel
Detail B

Character Detail A

.60
.55
.65
.70
.05
.05

Pixel Detail B

Note:
1. Tolerance is ±0.5 mm unless specified.
2. June 2010, we are transitioning from a
   keypad 10.5 mm high to a keypad 12 mm high.

Part No.(s): CFA533-TMI-KC CFA533-YYH-KC	Scale: Not to scale	Drawing Number: CFA533_master	Hardware Rev.: v1.0
	Units: Millimeters	Date: 2010/10/14	Sheet: 1 of 2

3.41

12.42

Pin Detail A

1.60
12.00

6.50 Bezel / PCB

10.70

12.00
10.50

I²C

J_PWR

I²C

28.00

8.00

See Pin
Detail A

60.08

24.92

J_DOW

J_PWR

27.50

1

I²C

7.50

J_PROG

J8

18.50

57.50

Back View

Note:
1. Tolerance is ±0.5 mm unless specified.
2. June 2010, we are transitioning from a
    keypad 10.5 mm high to a keypad 12 mm high.

Part No.(s):	Scale:	Drawing Number:	Hardware Rev.:
CFA533-TMI-KC	Not to scale	CFA533_master	v1.0
CFA533-YYH-KC	Units:	Date:	Sheet:
	Millimeters	2010/10/27	2 of 2

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 13

JUMPER LOCATIONS AND FUNCTIONS

The CFA533-YYH-KC has thirteen jumpers. Only the jumpers JP8 and RFG may be changed. The jumpers may be closed by melting a ball of solder across their gap. You may re-open these jumpers by removing the solder. Solder wick works well for removing solder.
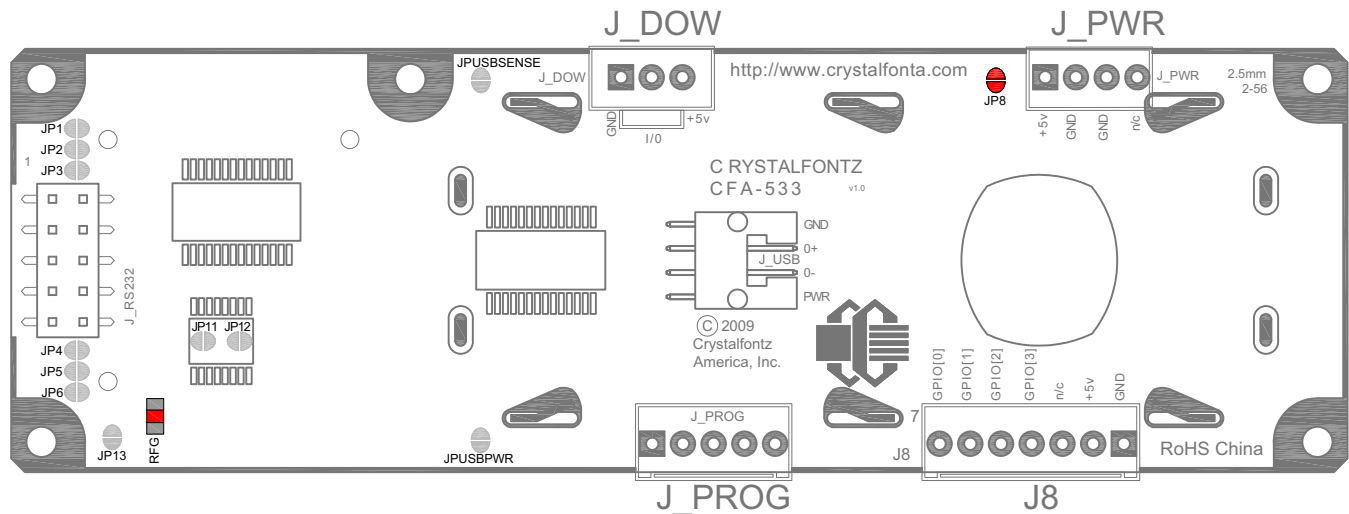
JUMPER	DEFAULT	FUNCTION
JP8	open **closed**	+5v pin from PWR connector used only for ATX SENSE +5v pin from PWR connector supplies power to module
RFG	**1MΩ**	Mounting holes and FG trace sourrunding PCB are discharged slowly to LOGIC GND.
	0Ω open	Mounting holes and FG trace are connected to LOGIC GND Mounting holes and FG trace surrounding PCB are open
Do not change JP1, JP2, JP3, JP4, JP5, JP6, JP11, JP12, JP13, JPUSBPWR, and JPUSBSENSE.		

Figure 2. Jumper Locations and Functions

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

LED dice dimension:
1.6mm(L)x0.8mm(W)x0.8mm(H)

Figure 3. Keypad Detail Drawing
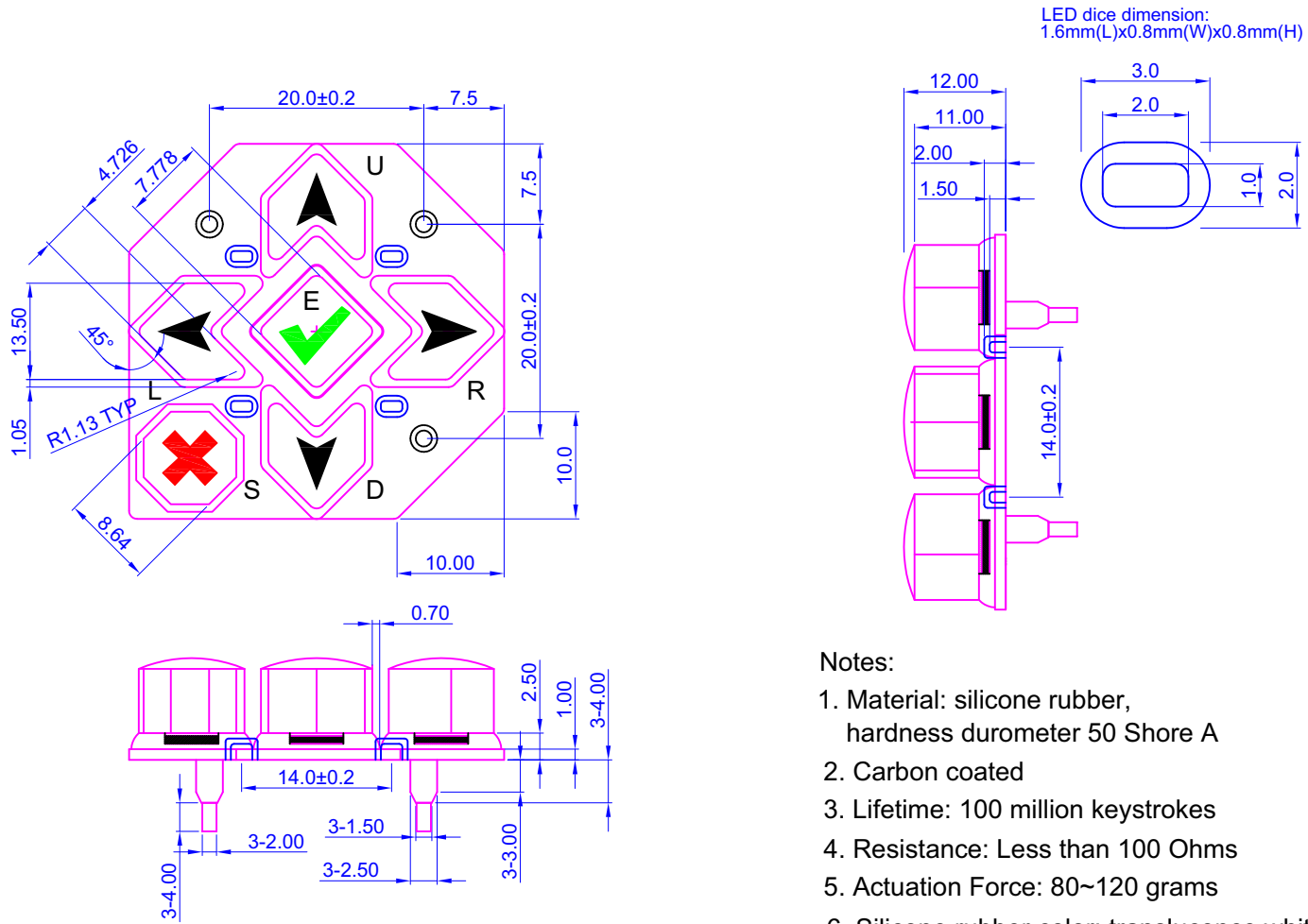
Notes:

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: translucence white

7. All corners have a fillet radius of 0.75 mm

copyright © 2010 by **Crystalfontz America, Inc.** www.crystalfontz.com/products/	Part No.(s): CFA533 Keypad Detail	Scale: Not to scale	Drawing Number: Keypad_master	Hardware Rev.: v1.1
		Units: Millimeters	Date: 2010/10/14	Sheet: 1 of 1

Figure 4. Panel Mounting Application Cutout Drawing for Optional Bracket

Cutout Detail

75.00   15.00   15.50

30.00   27.01   1.50   2.90   69.21

D   D   D   D

5-Ø2.16 PTH
5-Ø3.66 Pad

See Keypad Cutout Detail

Keypad Cutout Detail

Detail A
Detail B
Detail C

1.89   15.00
33.41   24.01   21.77   12.01
15.00
1.39
12.01
21.77
24.01
33.41
D   D   D   D

Detail A

5.29   90°   45°   4.58   3.04   R1.13 TYP

Detail B

90°   6.28 TYP   R1.13 TYP

Detail C

135°   2.96 TYP

Typical mounting hardware at locations "D" (5 places):
- PEM FH-256-8
- Bivar Inc. 9913-5 mm Spacer
- 2-56 "Small Profile" Hex Nut
- Use appropriate screen printed overlay to cover display bezel and mounting hardware, and to protect LCD from scratching. Sample fabrication drawings are available on request.

Part No.(s):	CFA533 Panel Mounting Application Cutout		
Scale: Not to scale		*Drawing Number:* CFA533_master	*Hardware Rev.:* v1.0
Units: Millimeters		*Date:* 2010/04/05	*Sheet:* 1 of 1

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 16

OPTICAL CHARACTERISTICS

Viewing Direction	6 o'clock

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

This module has a 6:00 o'clock viewing angle. A 6:00 o'clock viewing angle is a bottom viewing angle like what you would see when you look at a cell phone or calculator. A 12:00 o'clock viewing angle is a top viewing angle like what you would see when you look at the gauges in a golf cart or airplane.

Eyes look down

Eyes look up

6:00 O'Clock
Bottom Viewing Angle

12:00 O'Clock
Top Viewing Angle
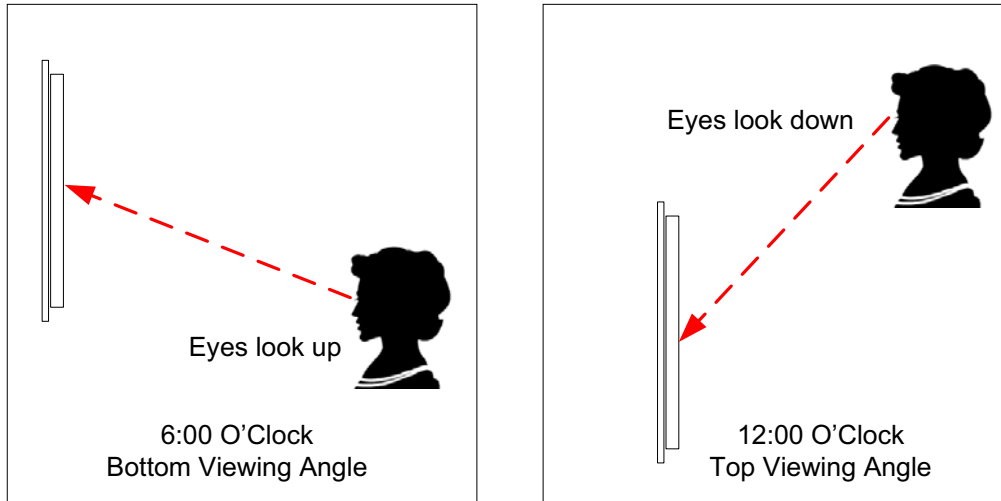
Figure 5. Definition of 6:00 O'clock and 12:00 O'clock Viewing Angles

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 17

ELECTRICAL SPECIFICATIONS

SYSTEM BLOCK DIAGRAM
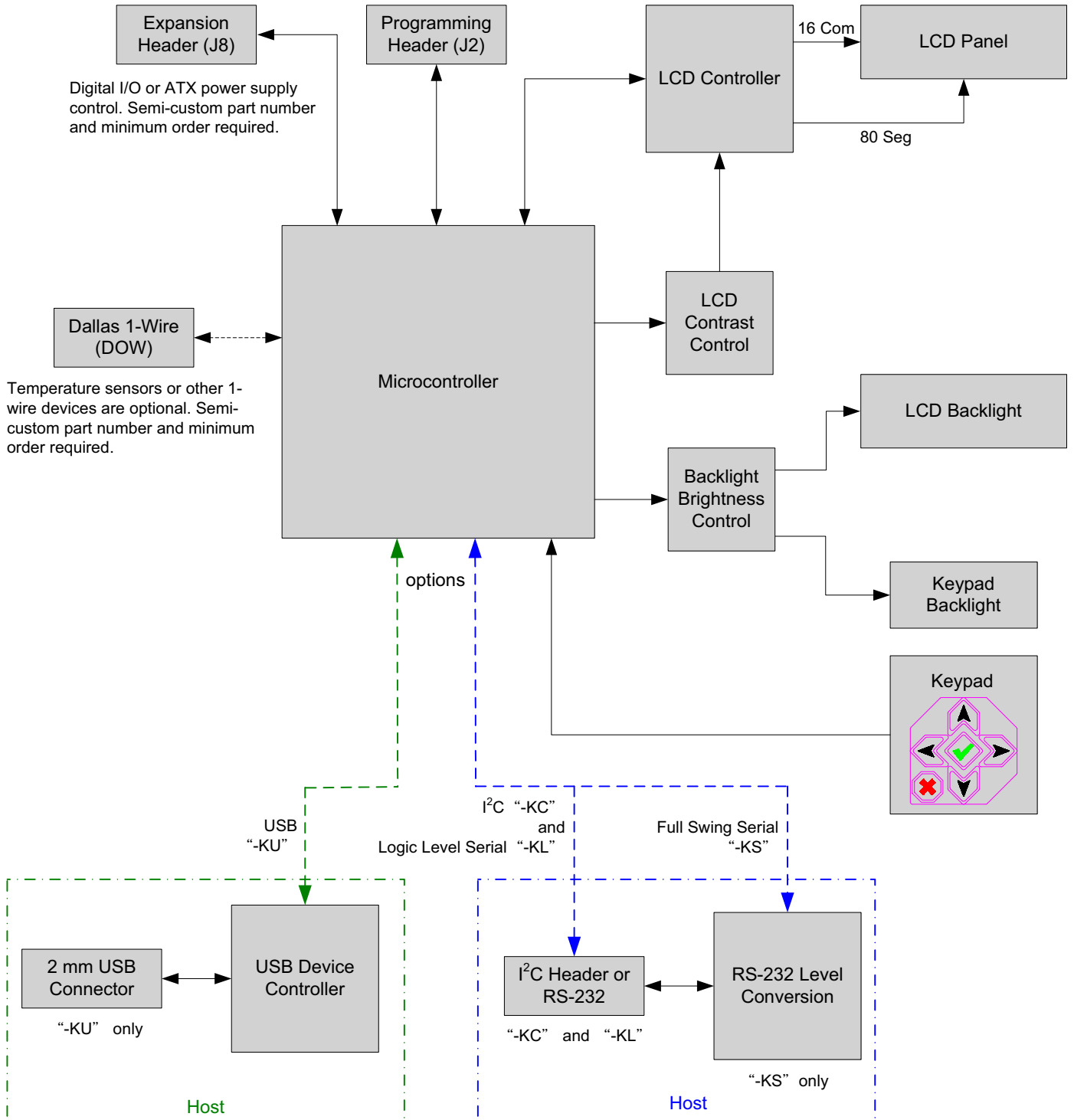
Figure 6. System Block Diagram

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 18

SUPPLY VOLTAGES AND CURRENT

SUPPLY VOLTAGE	MINIMUM	MAXIMUM
Power Supply voltage (V_{DD})	+3.3v	+5.5v
Pull-in voltage		+3.2v
Drop-out voltage		3+.0v

ITEMS ENABLED	TYPICAL CURRENT CONSUMPTION
+5v for logic (LCD + microcontroller)	< 20mA
+5v for logic (LCD + microcontroller) + YYH backlight	< 120mA

GPIO CURRENT LIMITS	SPECIFICATION
Sink	25 mA
Source	10 mA

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 19

TEMPERATURE AND HUMIDITY RATINGS

ABSOLUTE MAXIMUM RATINGS	SYMBOL	MINIMUM	MAXIMUM
Operating Temperature	T_{OP}	0°C	+50°C
Storage Temperature	T_{ST}	-10°C	+60°C
Humidity Range (noncondensing)	RH	10%	90%

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 20

CFA 533 supply current vs supply voltage (typical)
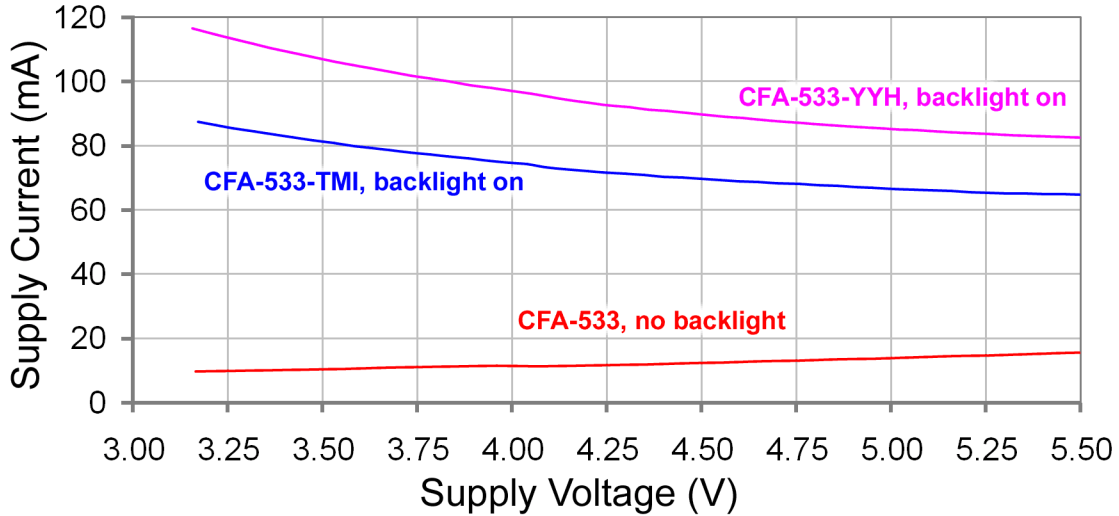
CFA 533-YYH supply current vs backlight setting (typical)
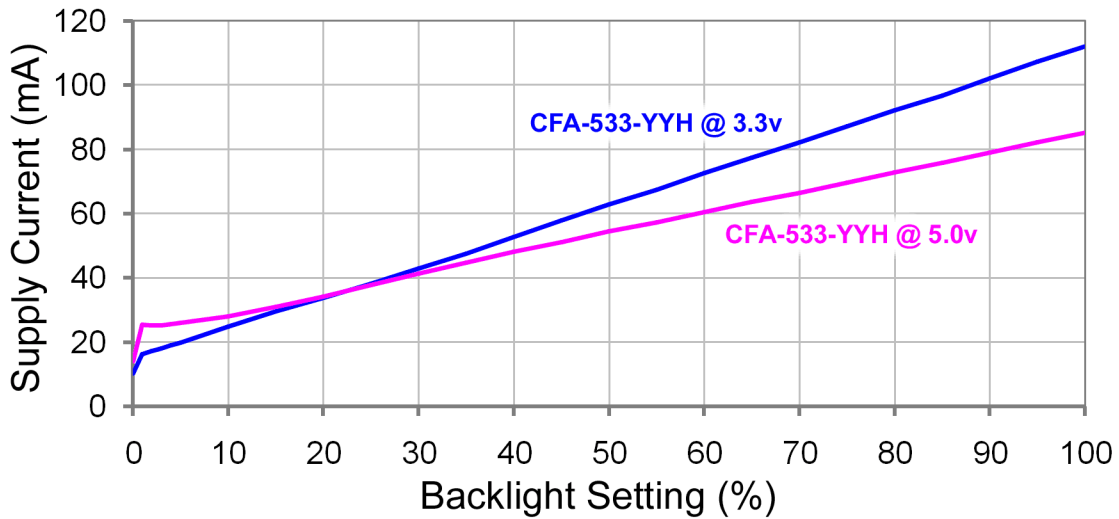
Figure 7. CFA533-YYH-KC Current Usage

ESD (ELECTRO-STATIC DISCHARGE) SPECIFICATIONS

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 21

CONNECTION INFORMATION

OVERVIEW OF CONNECTION INFORMATION

This section describes your choices of methods to connect power and host power sense to the LCD module. The section also describes connections for optional accessories.

The host power supply can power the CFA533-YYH-KC in one of two ways:
1. *Standard:* This is the basic method to supply power to the module ("non-ATX").
2. *ATX:* This method supplies power to the module and has power on, power off, and reset functionality.

For your convenience, here are links to the connection descriptions:

In the sections listed above, we describe which jumpers, if any, must be opened or closed for the different connection methods. A helpful reference is Jumper Locations and Functions (Pg. 13). The table lists the open/close defaults for all jumpers.

STANDARD (+5V) POWER SUPPLY CONNECTION

For a standard power connection from your host to the CFA533-YYH-KC, choose one of the three methods described below.
1. Standard (+5v) Connection through J_PWR Connector
2. Standard (+5v) Connection through J8 Connector
3. Standard (+5v) Connection through J_RS232 Connector

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 22

1. Standard (+5v) Connection through J_PWR Connector
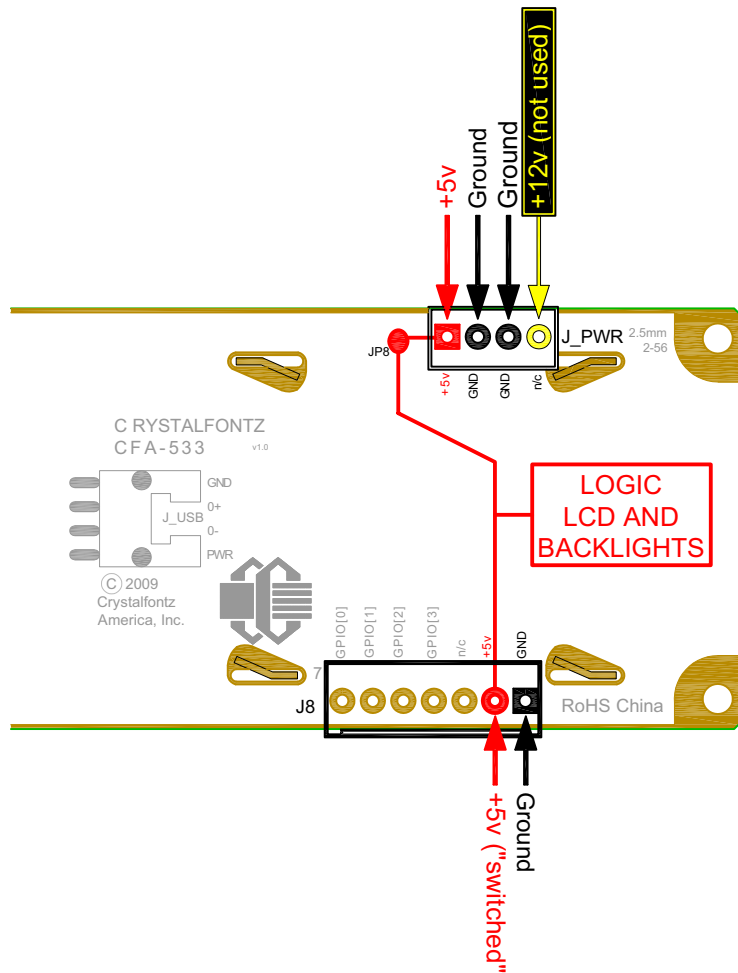
Figure 8. Standard (+5v) Connection through J_PWR

1. JP8 is closed by default. Leave JP8 closed so that the J8 connector is electrically connected to J_PWR.
2. You will need to supply +5v to pin 1 and Ground to pin 2 or 3 on the J_PWR connector. Use the Crystalfontz cable WR-PWR-Y12 (or equivalent) to connect from the host's 4-pin power supply cable connector to the CFA533-YYH-KC's J_PWR connector.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 23

2. Standard (+5v) Connection through J8 Connector

Figure 9. Standard (+5v) Connection through J8

Please see the previous illustration (Standard (+5v) Connection through J_PWR (Pg. 22)) for this connection method.

1. Open JP8 when powering from the J8 connector. (JP8 is closed by default. JP8 should remain closed only when powering from J_PWR.)
2. You will need to supply +5v to pin 2 and Ground to pin 1 on the J8 connector.

Modification by Crystalfontz

Crystalfontz can configure the LCD modules so they will be ready to use in your application without modification. We can open JP8 and load a connector at J8. For information, please contact technical support (+1-888-206-9720 or email techinfo@crystalfontz.com). We will provide you with a semi-custom part number and pricing. A minimum order quantity may apply.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 24

3. Standard (+5v) Connection through J_RS232 Connector
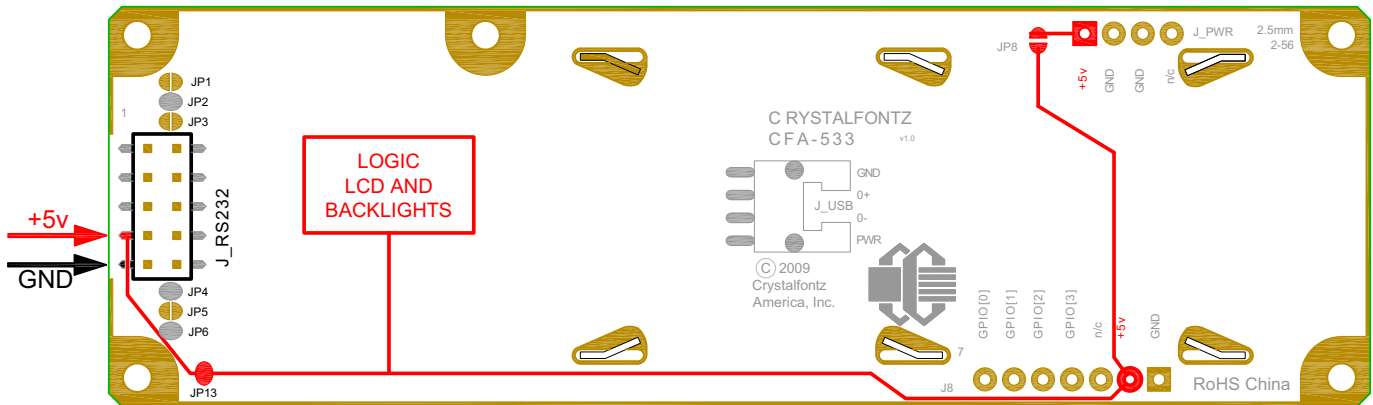
Figure 10. CFA533-YYH-KC Standard (+5v) Connection through J_RS232 Connector

The +5v power can be supplied through connector J_RS232, allowing a single cable to contain both power and data connections.

1. Close JP13. (JP13 is open by default.)
2. Open JP8.
3. If the "Default RS-232 Pin Assignments" (see are selected, the five connections needed to operate the LCD module are all on a single column of pins on J_RS232. You can connect a single 0.1-inch spacing 5-conductor cable to connect between the CFA533-YYH-KC and your host.

Modification by Crystalfontz

Crystalfontz can configure the LCD modules so they will be ready to use in your application without modification. We will close JP13 and open JP8. For information, please contact technical support (+1-888-206-9720 or email techinfo@crystalfontz.com). We will provide you with a semi-custom part number and pricing. A minimum order quantity may apply.

ATX POWER SUPPLY AND CONTROL CONNECTION FOR HOST POWER SENSE

ATX Power Supply Connection

The CFA533-YYH-KC has the ability to control power on/off and reset functions of an ATX power supply.

> **NOTE**
>
> The GPIO pins used for ATX control must not be configured as user GPIO. If ATX Host Power Sense to LCD module is being used, do not reconfigure the GPIO pins.

For this functionality, the CFA533-YYH-KC is powered from the host's V_{SB} signal (V_{SB} is the standby power which is always-on +5v ATX power supply output).

ATX Control Connections for Host Power Sense through J_PWR or J8 Connector

For ATX control, choose one of these two connection methods described below.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 25

1. ATX Host Power Sense through +5v on J_PWR Connector
2. ATX Host Power Sense through GPIO[1] on J8 Connector

1. ATX Host Power Sense through +5v on J_PWR Connector
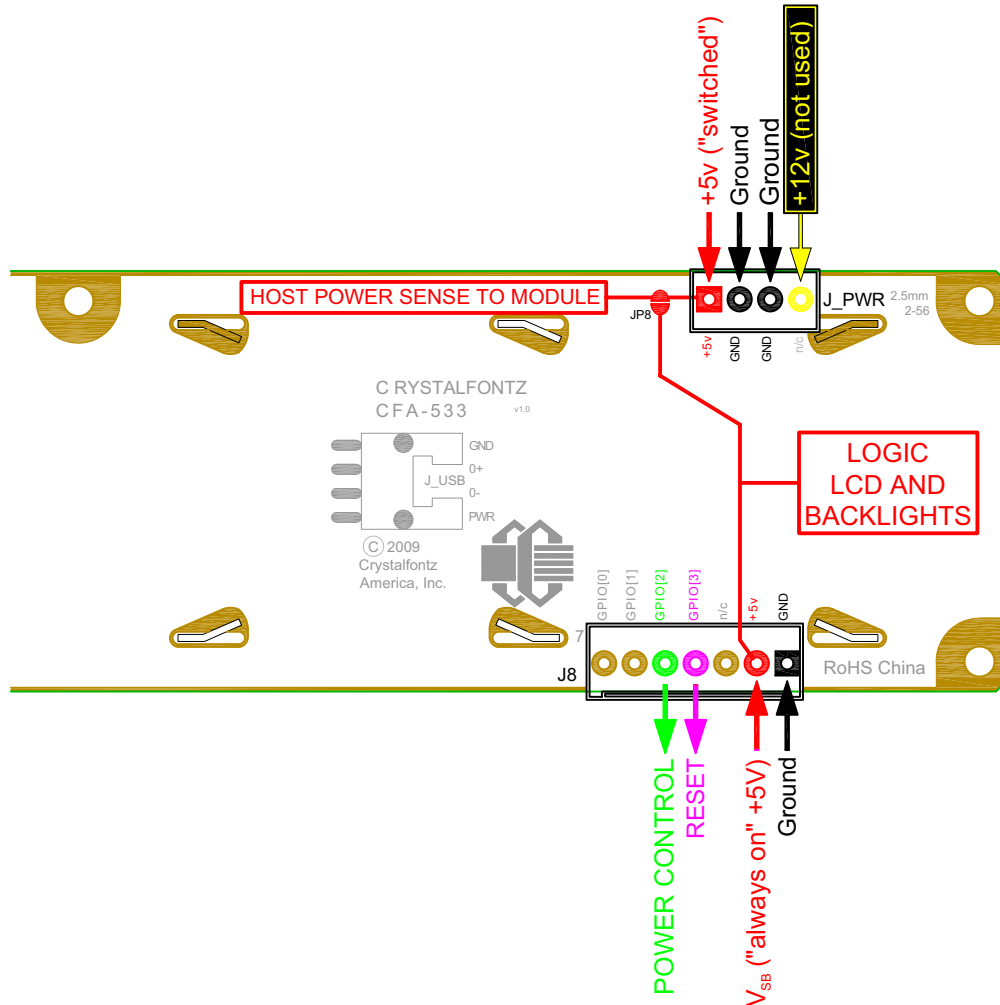
Figure 11. ATX Host Power Sense through +5v on J_PWR Connector

By default, the pin labeled +5v on the CFA533-YYH-KC's J_PWR connector is electrically connected to the +5v pin on the J8 connector through the normally closed JP8. If you want to use the CFA533-YYH-KC to do ATX power supply control, open jumper JP8. This will disconnect the +5v pin of the J_PWR connector from the +5v of the J8 connector. The +5v pin of the J_PWR connector will then function as the "Host Power Sense". The +5v pin of the J8 connector will function as V_{SB} power to the LCD module.

The motherboard's power switch input is connected to Pin 5 of the CFA533-YYH-KC's connector J8 (labeled as GPIO[2]). This pin functions as POWER CONTROL. The POWER CONTROL pin is configured as a high-impedance input until the LCD module wants to turn the host on or off, then it will change momentarily to low impedance output, driving either low or high depending on the setting of POWER_INVERT. (See command 28 (0x1C): Set ATX Switch Functionality (Pg. 49).)

The motherboard's reset switch input is connected to Pin 4 of the CFA533-YYH-KC connector J8 (labeled as GPIO[3]). This pin functions as RESET. The RESET pin is configured as a high-impedance input until the LCD module wants to reset the host. Then it will change momentarily to low impedance output, driving either low or high.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 26

The optional Crystalfontz WR-PWR-Y14 cable simplifies ATX power supply control connections. JP8 from connector J_PWR is closed by default. When using this cable, open jumper JP8 in order to ensure correct operation.
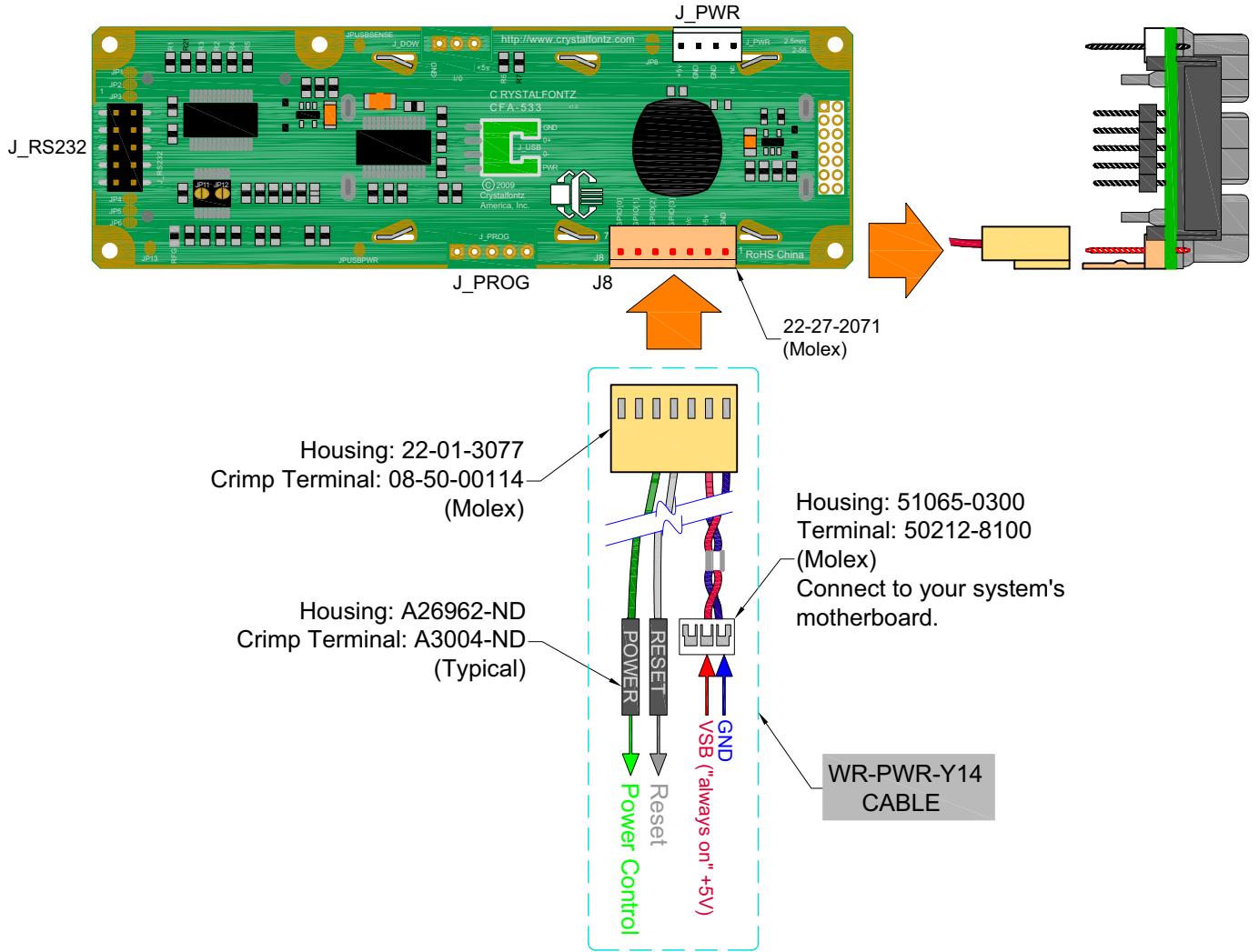
Figure 12. ATX Power Supply and Control Using Crystalfontz WR-PWR-Y14 Cable

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 27

NOTE

If the Crystalfontz WR-PWR-Y14 cable is ordered at the same time as the LCD module, Crystalfontz will install the WR-PWR-Y14 connector, open jumper JP8 and send the following software configuration commands (unless we are otherwise instructed).

```
command = 28  // Set ATX Switch Functionality
length = 3
data[0] = 241 // Enable:
              //    KEYPAD_POWER_OFF
              //    KEYPAD_POWER_ON
              //    KEYPAD_RESET
              //    LCD_OFF_IF_HOST_IS_OFF
              //    AUTO_POLARITY
data[1] = 16  // One half second power pulses
data[2] = 0   // sense ATX host power state on P0.7 (J_PWR, +5v)
command = 4   // Store current state as boot state
length = 0
```

2. ATX Host Power Sense through GPIO[1] on J8 Connector
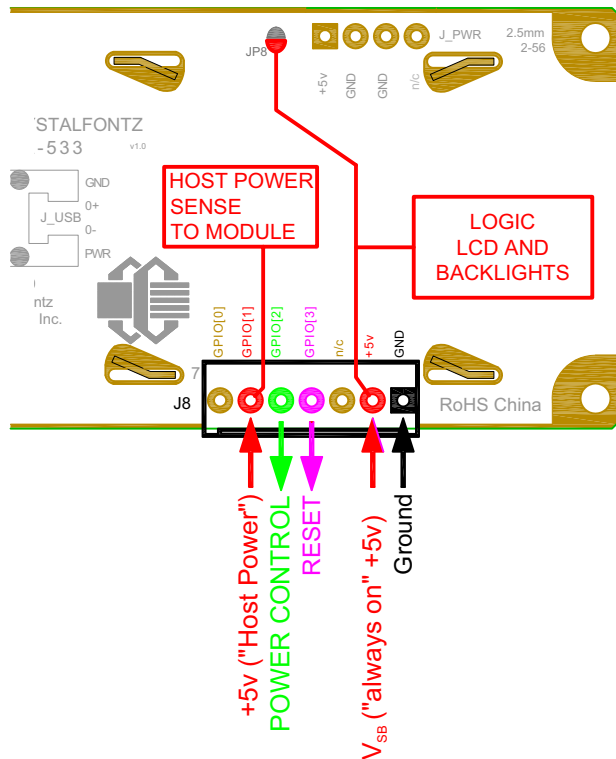
Figure 13. ATX Host Power Sense through GPIO[1] on J8 Connector

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 28

The CFA533-YYH-KC can be configured to sense host power through GPIO[1] on connector J8. In addition to +5$_{VSB}$, Ground, Power Control (GPIO[2]), and Reset Control (GPIO[3]) connections, you will need to supply connection to the host's +5v power to GPIO[1]. JP8 is closed by default. To properly function, JP8 must be opened.

The POWER-ON SENSE can be provided through Pin 6 of J8 (GPIO[1]). This option is only provided to allow backwards compatibility for legacy CFA633 applications. R21 is loaded with a 5kΩ 0805 SMT resistor for this functionality. See location of R21 illustration in Jumper Locations and Functions (Pg. 13).

Here is an excerpt from command 28 (0x1C): Set ATX Switch Functionality (Pg. 49) :

NOTE ON COMMAND 28: SET ATX SWITCH FUNCTIONALITY

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

These settings are factory default but may be changed by the user. Please see command 34 (0x22): Set/Configure GPIO (Pg. 53). 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.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 29

ATX Keypad Control

Once configured by the host software (see command 28 (0x1C): Set ATX Switch Functionality (Pg. 49)), the following functions may be individually enabled:

- **System power on.** If POWER-ON SENSE is low (0th), pressing the green check key (Enter key) for 0.25 seconds will turn the unit on by driving POWER CONTROL line for the pulse width set by command 28 (0x1C): Set ATX Switch Functionality (Pg. 49) (1.0 seconds default).

- **System hard power off.** If POWER-ON SENSE is high (+5v power, V$_{DD}$) pressing the red X key (Cancel key) for 4 seconds will turn the system off by driving the POWER CONTROL line. The line will be driven for a minimum of the pulse width set by command 28 (0x1C): Set ATX Switch Functionality (Pg. 49) (1.0 seconds default). If the user continues to press the key, the CFA533-YYH-KC will continue to drive the line for up to an additional 5 seconds.

- **System hard reset.** If POWER-ON SENSE is high (+5v power, V$_{DD}$) pressing the green check key (Enter key) for 4 seconds will reset the system by driving the RESET line for 1 second. The CFA533-YYH-KC will reboot itself immediately after resetting the host.

Since the computer and LCD module must look off if the computer's power is off, the CFA533-YYH-KC can be configured to monitor the POWER-ON SENSE line and blank its display any time the POWER-ON SENSE line is low.

DATA COMMUNICATIONS: I²C CONNECTIONS

Connect SDA (Serial DAta) and SCL (Serial CLock) to your I^{2}C host as shown in the figure below. The I^{2}C bus requires external pull-up resistors. The pull-up resistors (RP) are determined by the supply voltage, clock speed, and bus capacitance. The minimum sink current for any device (master or slave) should be no less than 3 mA at VOL$_{MAX}$ = 0.4v for the output stage. This limits the minimum pull-up resistor value for a 5-volt system to about 1.5kΩ. The maximum value for RP is dependent on the bus capacitance and the clock speed. For a 5-volt system with a bus capacitance of 150 pF, the pull-up resistors should be no larger than 6kΩ. For more information see the _UM10204 I^{2}C-bus specification and user manual_ on the NXP Semiconductors website.
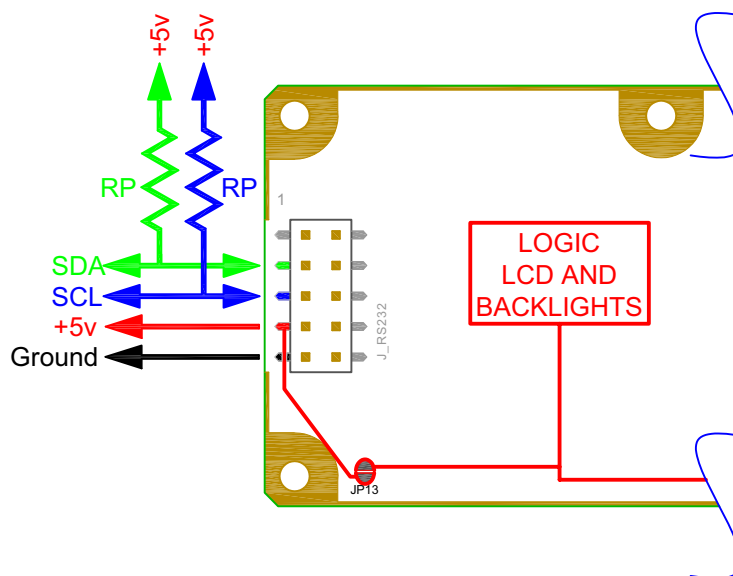
Figure 14. I^{2}C Connections

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 30

DATA COMMUNICATIONS: DETAILS FOR RS232 CONNECTIONS

JP2, JP4, and JP6 are closed at the factory, selecting the "Default RS-232 Pin Assignments" (see Figure 15. on Pg. 31). This connection allows a low-cost ribbon cable (Crystalfontz WR-232-Y08) to connect the CFA533-YYH-KC to a host's DB-9 COM port.

If you are connecting the CFA533-YYH-KC to a host system (such as a PC motherboard) that has a similar 10-pin 0.1-inch connector, rather than a standard RS-232 DB-9 connector common to rear panels, the pinouts may need to be changed from "Default" to "Alternate".

For an RS-232 connection, you can modify the LCD modules or have Crystalfontz modify them for you.

Choose one of two methods described below to make the connection.

Method 1: Use Crystalfontz WR-232-Y22 cable
The easiest method is to use a Crystalfontz WR-232-Y22 cable. Connect the single end of the WR-232-Y22 cable to the CFA533-YYH-KC. On the double end of the WR-232-Y22 cable, one connector will work for host connections that use "default" numbering; the other end will work for host connections that use the "alternate" numbering.

Method 2: Use Straight-through 10-pin to 10-pin ribbon cable
Use a straight-through 10-pin to 10-pin ribbon cable (for example, CW Industries' C3AAG-1018G-ND cable available from Digi-Kith pin order of your motherboard's header will determine if the CFA533-YYH-KC's pin assignments need to be "Default" or "Alternate". Open or close jumpers JP1-JP6 as necessary to set the CFA533-YYH-KC to "Default" or "Alternate" that matches your motherboard.

Modification by Crystalfontz

We can configure the LCD module for you by opening JP8, loading the mating connector for the Crystalfontz WR-PWR-Y14 cable in J8, and configuring the ATX firmware functions. The modules will be ready to use in your application without modification by you. For information, please contact technical support (+1-888-206-9720, techinfo@crystalfontz.com). We will provide you with a semi-custom part number and pricing. A minimum order quantity may apply.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 31

Please note that the CFA533-YYH-KC can be powered through this header. Please refer to 3. Standard (+5v) Connection through J_RS232 Connector (Pg. 24).

Default RS-232 Pin Assignment

LCD Tx/Host Rx
LCD Rx/Host Tx

JP1: open   Ground
JP2: **closed**
JP3: open
JP4: **closed**
JP:5 open
JP:6 **closed**

Alternate RS-232 Pin Assignment

LCD Tx/Host Rx
LCD Tx/Host Rx
Ground

JP1: closed
JP2: **open**
JP3: closed
JP4: **open**
JP:5 closed
JP:6 **open**
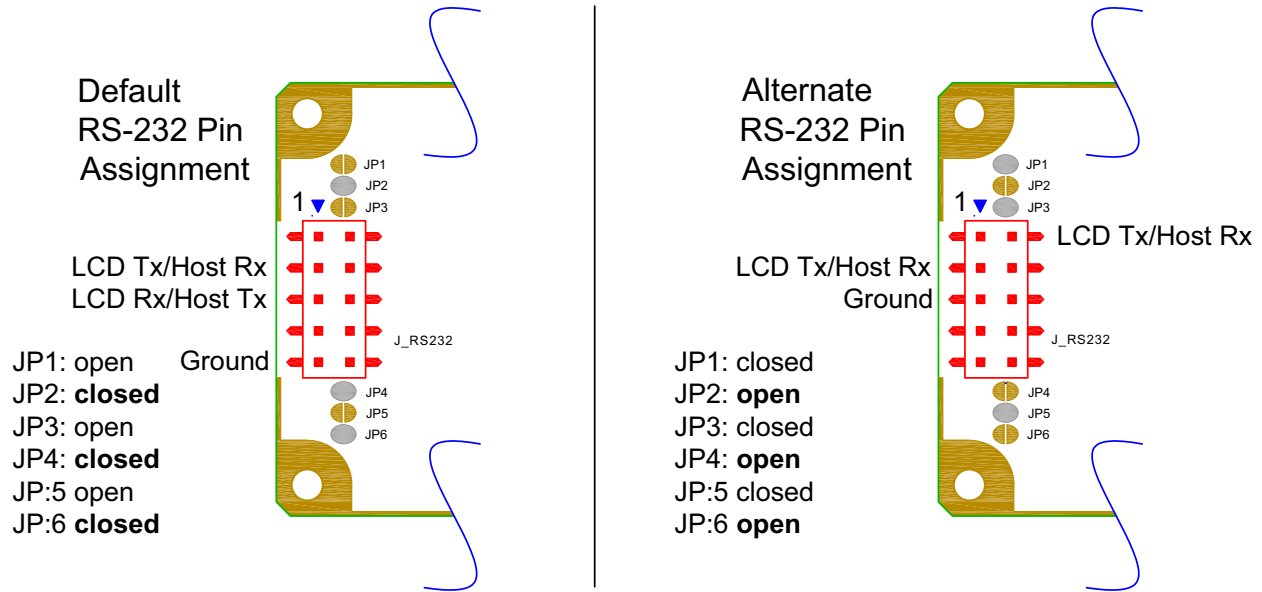
Figure 15. J_RS232 Default and Alternate Pin Assignments

GPIO CONNECTIONS

The CFA533-YYH-KC has five General-Purpose Input/Output (GPIO) pins. The GPIO are port pins from the CFA533-YYH-KC's microcontroller brought out to connectors. As an output, a GPIO can be used to turn on an LED, or perhaps drive a relay. As an input, a GPIO can be used to read a switch or a button. Most of the GPIOs have a default function that allows the LCD module to perform some special purpose activity with the pin.

```
GPIO[0] = J8, Pin 7
GPIO[1] = J8, Pin 6 (may be used as ATX Host Power Sense, has R21 in series)
GPIO[2] = J8, Pin 5 (default is ATX Host Power Control)
GPIO[3] = J8, Pin 4 (default is ATX Host Reset Control)
GPIO[4] = J_DOW, Pin 2 (default is DOW I/O--may have 1 KΩ hardware pull-up: R7)
```

GPIO[0], GPIO[2] and GPIO[3] are connected directly from the microcontroller port pin to the connector pin.

GPIO[1] has a series 5kΩ resistor in R21.

GPIO[4] is also used as the DOW I/O pin. Since the DOW requires a pull-up on the I/O pin, a 1kΩ resistor in R7 is loaded to pull GPIO[4] to V_{DD} (+5v power).

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 32

Please refer to commands 34 (0x22): Set/Configure GPIO (Pg. 53) and 35 (0x23): Read GPIO Pin Levels and Configuration State (Pg. 54) for additional details concerning the GPIO operation.
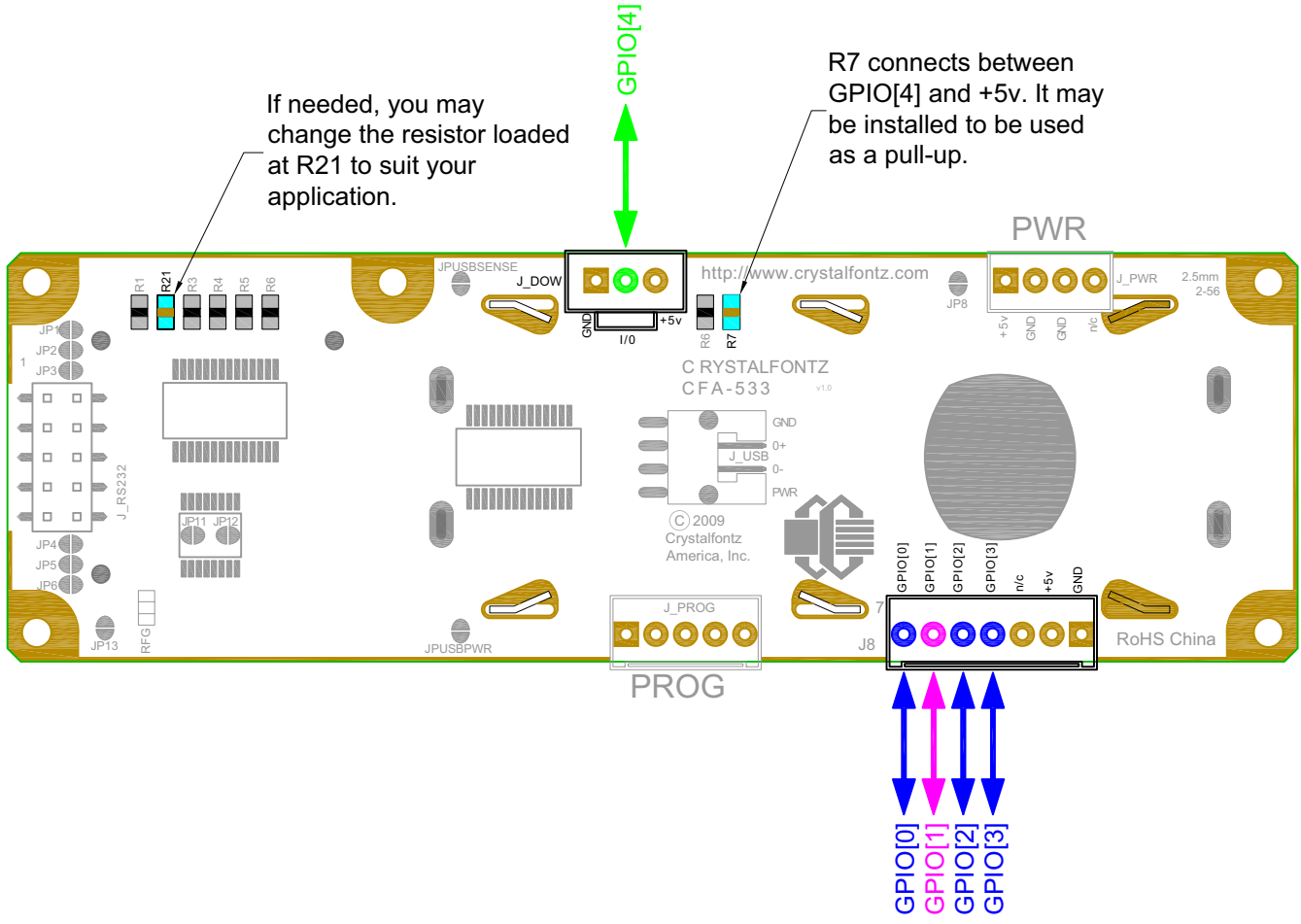
Figure 16. Location of GPIO Connections, Resistors, and J_DOW

DALLAS SEMICONDUCTOR 1-WIRE DEVICE CONNECTIONS FOR OPTIONAL ACCESSORIES

Temperature Sensors

The CFA533-YYH-KC supports Dallas Semiconductor 1-Wire (DOW) temperature sensors. See location of J_DOW in Figure 16. on Pg. 32. By default, J_DOW is not loaded with a connector. If you request it, Crystalfontz can load a Molex 70543-0002. For information, please contact technical support (+1-888-206-9720 or email techinfo@crystalfontz.com). We will provide you with a semi-custom part number and pricing. A minimum order quantity may apply.

Any combination of up to 32 DS1822 Econo 1-Wire Digital Thermometer (2°C absolute accuracy) or DS18B20 High Precision 1-Wire Digital Thermometer (0.5°C absolute accuracy) temperature sensors or other DOW compatible devices are directly supported.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 33

The Crystalfontz WR-DOW-Y17 has a DS18B20 attached to a "daisy chain" cable. If a WR-DOW-Y17 is ordered at the same time as a CFA533-YYH-KC, Crystalfontz can load the WR-DOW-Y17's mating connector into the CFA533-YYH-KC's DOW position. For reference, the mating connector for the WR-DOW-Y17 is Molex 70543-0002 available from Digi-Key or other parts suppliers.

The temperature sensor can be configured to be automatically read and displayed on the CFA533-YYH-KC's LCD in °C or °F (see command 21 (0x15): Set Up Live Temperature Display (Pg. 46)). Independently, any temperature sensor can be configured to report to the host (see 19 (0x13): (reserved) (Pg. 45)). The sensors configured to be reported are updated once each second.

Other 1-Wire Devices

Other Dallas Semiconductor 1-Wire devices may be connected to the 1-Wire bus, with the CFA533-YYH-KC acting as a bridge between RS-232 and the 1-Wire bus (see command 21 (0x15): Set Up Live Temperature Display (Pg. 46)). The total number of 1-Wire devices supported is 32, including directly supported temperature sensors and any other user-provided 1-Wire devices. (See CFA533-YYH-KC's DOW connection location in Figure 16. on Pg. 32.) The LCD module 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 module.

The CFA533-YYH-KC has a 1kΩ hardware pull-up on the DOW connector's I/O line.

Connect the 1-Wire sensors as detailed in the sensor's data sheet.

HOST COMMUNICATIONS

> **NOTE**
>
> Because there is no difference in communications and commands for I^{2}C variants (part numbers ending in "-KC"), serial variants (part numbers ending in "-KL" or "-KS") and USB variants (part numbers ending in "-KU") of the CFA533, the Host Communications section of this Data Sheet uses the shorter term "CFA533" instead of CFA533-YYH-KC".

I^{2}C BUFFERS

The I^{2}C specification allows the I^{2}C master (host) to run at clock speeds from 100 kHz down to DC.

Reading and writing data within the I^{2}C slave (CFA533) is accomplished using buffers. The foreground process in the I^{2}C slave (CFA533) can then operate on data deposited in its write buffer by the I^{2}C master (host) and prepare responses to be read by the I^{2}C master (host) from its read buffer.

During Read: If the I^{2}C master (host) attempts to read more data than is contained in a buffer, the last byte will be retransmitted until the I^{2}C master (host) stops reading. The I^{2}C protocol does not define a method for the I^{2}C slave to stop a master from reading.

During Write: When an I^{2}C master (host) writes one or more data bytes to the I^{2}C slave (CFA533), upon receiving the last byte for which storage is available, the I^{2}C slave (CFA533) generates a NAK (Negative AcKnowledgement). If the I^{2}C master (host) continues to write data, the slave (CFA533) will continue to NAK it. Once the first NAK is generated (data is stored in the last available location), further data is not stored.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 34

I^{2}C ADDRESSING

I^{2}C addresses are contained in the upper 7-bits of the first byte of a read or write transaction. This byte is used by the I^{2}C master to address the slave (CFA533). Valid selections are from 0-127 decimal. The lsb (least significant bit) of the byte contains the R/W bit. If this bit is *0*, the address will be *written* to it; if the lsb is a *1* then the addressed slave will have data *read* from it.

The default address of the CFA533-YYH-KC is 42_{10} (84_{10} writes, 85_{10} reads), in hexademicals 0x2A (0x54 writes, 0x55 reads).

Display Address: To display the I^{2}C address of the module on the LCD, hold both the up and the down arrows for 4 seconds.

Change Address: The I^{2}C address may be changed by command 33 (0x21): Set I^{2}C Address (Pg. 52). To make the change permanent, use command 4 (0x04): Store Current State as Boot State (Pg. 38).
The CFA533-YYH-KC expects to find a valid packet within the data area of I^{2}C transactions. (See Packet Structure (Pg. 35)).

I²C Start	I²C Address	I²C R/W	I²C data[0]	I²C data[1]	....	I²C Stop

I²C Payload

533 command	533 data_length	533 data[0]	533 data[1]	....	533 data[data_length-1]	533 CRC LSB	533 CRC MSB

533 Packet
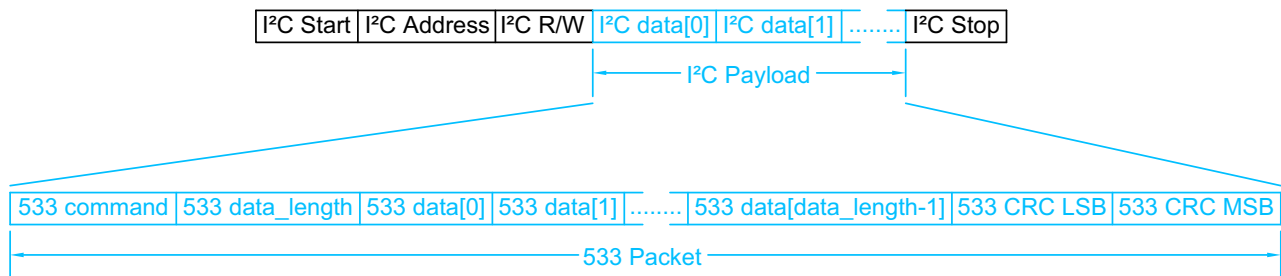
Figure 17. I^{2}C Transactions

The host must read at least as many bytes as the size of the CFA533-YYH-KC response. The host may read additional bytes. These additional bytes will contain undetermined data. The host must NAK the last byte that it reads (the NAK comes just before the STOP) so the CFA533 knows that the read is terminated. See pages 14-15 in NXP's I2C-bus specification and user manual (UM10204) (www.standardics.nxp.com/support/documents/i2c/pdf/i2c.bus.specification.pdf).

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 35

I^{2}C DEBUGGING

The Aardvark I^{2}C/SPI Host Adapter by Total Phase may be used for debugging. We have a Windows application 533_I2C_WinTest that can be used with the Aardvark to test and debug CFA533 I^{2}C modules.

Total Phase TP240141
"Aardvark I2C/SPI Host Adapter"

Total Phase TP240212
"10-Pin Split Cable"

ORANGE(SDA)
BROWN(SCL)
YELLOW(NC/+5v)
RED(GND)

LOGIC LCD AND BACKLIGHTS

Figure 18. CFA533-YYH-KC Connection to Aardvark I^{2}C/SPI Host Adapter

PACKET STRUCTURE

All communication between the CFA533 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 CFA533 and the host without the traditional problems that occur in a stream-based serial communication (such as having to send data in inefficient ASCII format, to "escape" certain "control characters", or losing sync if a character is corrupted, missing, or inserted)..

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

All packets have the following structure:

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

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

```
TTcc cccc
|||| ||||--Command, response, error or report code 0-63
||---------Type:
        00 = normal command from host to CFA533
        01 = normal response from CFA533 to host
        10 = normal report from CFA533 to host (not indirect response to a command
             from the host)
        11 = error response from CFA533 to host (a packet with valid structure but
             illegal content was received by the CFA533)
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 36

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

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

`CRC` is a standard 16-bit CRC of all the bytes in the packet except the CRC itself. The CRC is sent LSB first. At the port, the CRC immediately follows the last used element of data `[]`. See APPENDIX B: SAMPLE CODE (INCLUDES ALGORITHMS TO CALCULATE THE CRC) (Pg. 66) for several examples of how to calculate the CRC in different programming languages.

The following concept may be useful for understanding the packet structure.

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

Crystalfontz supplies a demonstration and test program 533_I2C_WinTest along with its C source code. Included in the 533_I2C_WinTest source is a CRC algorithm and an algorithm that validates packets. Please follow the validation algorithm in the sample code closely in order to realize the benefits of using the packet communications.

ABOUT HANDSHAKING

The I²C bus master (host) must initiate all transactions. When the host writes a command to the input buffer of the slave (CFA533), the CFA533 will execute the appropriate functions, then write the resulting acknowledge or response packet into the CFA533's output buffer.

The host can read the CFA533's output buffer to verify the command or to read the result of a query.

The CFA533 takes a small amount of time to execute the command before the result can be read. For many commands 5mS is plenty of time for the CFA533 to complete its processing and have the result ready for the host to read.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 37

Commands that may take longer are:
    2 (0x02): Write User Flash Area
      25mS

    4 (0x04): Store Current State as Boot State
      50mS

    5 (0x05): Reboot CFA533, Reset Host, or Power Off Host
      500mS (for parameters \008\018\099, Reboot CFA533)
      2S (for parameters \012\028\097 Reset host)
      2S (for parameters \003\011\095 Power off host)

    14 (0x0E): Set LCD & Keypad Backlight
      50mS (all change)

    20 (0x14): Arbitrary DOW Transaction
      50mS (execution time depends on the transaction)

COMMAND CODES

Below is a list of valid commands for the CFA533. 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 CFA533 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 CFA533 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[] = "CFA533:hX.X,yY.Y"

hX.X is the hardware revision, "1.0" for example
yY.Y is the firmware version, "c1.0" for example
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 38

2 (0x02): Write User Flash Area

The CFA533 reserves 16 bytes of nonvolatile memory for arbitrary use by the host. This memory can be used to store data such as a serial number, IP address, gateway address, netmask, or any other data required. All 16 bytes must be supplied.

You must wait up to 25mS after the I^{2}C write phase completes to guarantee the CFA533 will have the acknowledge or response I^{2}C packet ready to be read by the I^{2}C host.

```
type: 0x02 = 2₁₀
valid data_length is 16
data[] = 16 bytes of arbitrary user data to be stored in the CFA533'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 CFA533's non-volatile memory
```

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

> **ERRATA NOTE**
>
> At voltages lower than +5v, the operation of reading from the LCD to the microcontroller may not work correctly. If this happens, you will see corrupted characters on the LCD after using Command 4 and rebooting the module.
>
> - One workaround is to supply +5v to the module during the Command 4 operation.
> - Another workaround is to have Crystalfontz set the boot state for you at the factory, using a semi-custom part number.
>
> Please write techinfo@crystalfontz.com if you have questions regarding this errata.

The CFA533 loads its power-up configuration from nonvolatile memory when power is applied. The CFA533 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. 40).
  - command 7 (0x07): Set LCD Contents, Line 1 (Pg. 40).
  - command 8 (0x08): Set LCD Contents, Line 2 (Pg. 41).
  - command 31 (0x1F): Send Data to LCD (Pg. 52).
- Special character font definitions (command 9 (0x09): Set LCD Special Character Data (Pg. 41)).

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 39

- Cursor position (command 11 (0x0B): Set LCD Cursor Position (Pg. 42)).
- Cursor style (command 12 (0x0C): Set LCD Cursor Style (Pg. 42)).
- Contrast setting (command 13 (0x0D): Set LCD Contrast (Pg. 42)).
- LCD backlight setting (command 14 (0x0E): Set LCD & Keypad Backlight (Pg. 43)).
- Keypad backlight setting (command 14 (0x0E): Set LCD & Keypad Backlight (Pg. 43)).
- Settings of any live displays (command 21 (0x15): Set Up Live Temperature Display (Pg. 46)).
- ATX function enable and pulse length settings (command 28 (0x1C): Set ATX Switch Functionality (Pg. 49)).
- Baud rate (command 33 (0x21): Set I²C Address (Pg. 52)).
- GPIO settings (command 34 (0x22): Set/Configure GPIO (Pg. 53)).

You cannot store the temperature reporting (although the live display of temperatures can be saved). You cannot store the host watchdog.The host software should enable this item once the system is initialized and it is ready to receive the data.

You must wait up to 50mS after the I²C write phase completes to guarantee the CFA533 will have the acknowledge or response I²C packet ready to be read by the I²C host.

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

The return packet will be:

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

5 (0x05): Reboot CFA533, Reset Host, or Power Off Host

This command instructs the CFA533 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.

You must wait after the I²C write phase completes to guarantee the CFA533 will have the acknowledge or response I²C packet ready to be read by the I²C host. Execute times are up to:

    500mS (for parameters \008\018\099, Reboot CFA533)
    2S (for parameters \012\028\097 Reset host)
    2S (for parameters \003\011\095 Power off host)

> **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/Configure GPIO (Pg. 53).

Rebooting the CFA533 may be useful when testing the boot configuration. It may also be useful to re-enumerate the devices on the 1-Wire bus. To reboot the CFA533, send the following packet: .

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 40

To reset the host, assuming the host's reset line is connected to GPIO[3] as described in command 28 (0x1C): Set ATX Switch Functionality (Pg. 49), send the following packet:

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

To turn the host's power off, assuming the host's power control line is connected to GPIO[2] as described in command 28 (0x1C): Set ATX Switch Functionality (Pg. 49), send the following packet:

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

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

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

6 (0x06): Clear LCD Screen

Sets the contents of the LCD screen DDRAM to '' = 0x20 = 32$_{10}$ 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. 38).

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

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

> **NOTE**
>
> Please use this command only if you need backwards compatibility with older CFA633 units. For new applications, please use the more flexible command 31 (0x1F): Send Data to LCD (Pg. 52) which is also supported by the CFA631 and CFA635.

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

The return packet will be:

```
type: 0x40 | 0x07 = 0x47 = 71₁₀
data_length: 0
```

Set LCD Contents, Line 1 is one of the items stored by the command 4 (0x04): Store Current State as Boot State (Pg. 38).

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 41

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

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

> **NOTE**
>
> Please use this command only if you need backwards compatibility with older CFA633 units. For new applications, please use the more flexible command 31 (0x1F): Send Data to LCD (Pg. 52) which is also supported by the CFA631 and CFA635.

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

The return packet will be:

```
type: 0x40 | 0x08 = 0x48 = 72₁₀
data_length: 0
```

Set LCD Contents, Line 2 is one of the items stored by the command 4 (0x04): Store Current State as Boot State (Pg. 38).

9 (0x09): Set LCD Special Character Data

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

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

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

The return packet will be:

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

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

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

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

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

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

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

The return packet will be:

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 42

`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 CFA533'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. 42).

```
type: 0x0B = 11₁₀
valid data_length is 2
data[0] = column (0-15 valid)
data[1] = row (0-1 valid)
```

The return packet will be:

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

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

12 (0x0C): Set LCD Cursor Style

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

```
type: 0x0C = 12₁₀
valid data_length is 1
data[0]: cursor style (0-3 valid)
      0 = no cursor
      1 = blinking block cursor
      2 = underscore cursor
      3 = blinking underscore (Note: This behavior is not the same as the CFA633 series
          which is: blinking block plus underscore.
```

The return packet will be:

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

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

13 (0x0D): Set LCD Contrast

This command sets the contrast or vertical viewing angle of the display. (Initiated by the host, responded to by the CFA533.)

● CFA633 Compatible
  If only one byte of data is supplied, then it is the "CFA633 Compatible" version of the command. Requires 1 byte (0-200) are valid, but only (0-50) are useful for this LCD.

```
type: 0x0D = 13₁₀
valid data_length is 1
data[0]: contrast setting (0-50 valid)
      0 = light
     16 = about right
     29 = dark
  30-50 = very dark
```

The return packet will be:

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 43

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

- CFA533 Enhanced
  If two bytes of data are supplied, then the command takes advantage of the CFA533s native enhanced contrast resolution. Requires 2 bytes.
  - The first byte data[0] is ignored, any value from 0 to 255 is accepted.
  - The second byte data[1] controls the CFA533 contrast with better resolution.

```
type: 0x0D = 13₁₀
valid data_length is 1
data[0]: required but ignored
data[1]: contrast setting (0-200 valid)
      0-99 = lighter
        100 = no correction
    101-200 = darker
```

The return packet will be:

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

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

14 (0x0E): Set LCD & Keypad Backlight

This command sets the brightness of the LCD and keypad backlights. (Initiated by the host, responded to by the CFA533.)

You must wait up to 50mS (all change) after the I^{2}C write phase completes to guarantee the CFA533 will have the acknowledge or response I^{2}C packet ready to be read by the I^{2}C host.

- If one byte is supplied, both the keypad and LCD backlights are set to that brightness (CFA633 compatible).

```
type: 0x0E = 14₁₀
valid data_length is 1
data[0]: keypad and LCD backlight power setting (0-100 valid)
      0 = off
   1-100 = variable brightness
```

The return packet will be:

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

- If two bytes are supplied, the LCD is set to the brightness of the first byte, the keypad is set to the brightness of the second byte.

```
type: 0x0E = 14₁₀
valid data_length is 2
data[0]: LCD backlight power setting (0-100 valid)
      0 = off
   1-100 = variable brightness

data[1]: keypad backlight power setting (0-100 valid)
      0 = off
   1-100 = variable brightness
```

The return packet will be:

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 44

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

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

15 (0x0F): Read Temperature

Each temperature sensor is read once every second. The most recent reading can be retrieved by the host by using this command.

```
type: 0x0F = 15₁₀
valid data_length: 1
data[0]: 0 to 31 DOW device index
```

The family code for the device at "device index" must be 0x22 (DS1822) or 0x28 (DS12B20). This can be verified with Command 18 (0x12): Read DOW Device Information (Pg. 44).

The return packet will be:

```
type: 0x40 | 0x0F = 0x4F = 79₁₀
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:

```c
void OnReceivedTempReport(COMMAND_PACKET *packet, char *output)
  {
  //First check the DOW CRC return code from the CFA533
  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);
    }
  }
```

16-17 (0x10-0x11): (reserved)

18 (0x12): Read DOW Device Information

When power is applied to the CFA533, 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.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 45

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/Configure GPIO (Pg. 53).

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 CFA533 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
```

19 (0x13): (reserved)

20 (0x14): Arbitrary DOW Transaction

The CFA533 can function as an I^{2}C to Dallas 1-Wire bridge. The CFA533 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 CFA533.

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 C: CONNECTING A DS2450 1-WIRE QUAD A/D CONVERTER (Pg. 78) for an example of using this command.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 46

You must wait up to 50mS after the I^{2}C write phase completes to guarantee the CFA533 will have the acknowledge or response I^{2}C packet ready to be read by the I^{2}C host.

```
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): Set Up Live Temperature Display

You can configure the CFA533 to automatically update a portion of the LCD with a live temperature reading. Once the display is configured using this command, the CFA533 will continue to display the live reading on the LCD without host intervention. The Set Up Live Temperature Display is one of the items stored by command 4 (0x04): Store Current State as Boot State (Pg. 38), so you can configure the CFA533 to immediately display system temperatures as soon as power is applied.

The live display is based on a concept of display slots. There are 4 slots, and each of the 4 slots may be enabled or disabled independently.

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

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 47

```
type: 0x15 = 21₁₀
valid data_length is 7 or 2 (for turning a slot off)
data[0]: display slot (0-7)
data[1]: type of item to display in this slot
        0 = nothing (data_length then must be 2)
        1 = (invalid)
        2 = temperature (data_length then must be 7)
data[2]: index of the sensor to display in this slot:
        0-31 are valid for temperatures (and the temperature device must be attached)
data[3]: number of digits
        for a temperature: 3 digits (-XX or XXX)
        for a temperature: 5 digits (-XX.X or XXX.X)
data[4]: display column
        0-13 valid for a 3-digit temperature
        0-11 valid for a 5-digit temperature
data[5]: display row (0-1 valid)
data[6]: temperature units(0 = deg C, 1 = deg F)
```

If a 1-Wire CRC error is detected, the temperature will be displayed as "ERR" or "ERROR".

The return packet will be:

```
type: 0x40 | 15 = 0x55 = 85₁₀
data_length: 0
```

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

The controller on the CFA533 is a Neotec NT7070B (HD44780 compatible). Generally you won't need low-level access to the LCD controller but some arcane functions of the NT7070B are not exposed by the CFA533's command set. This command allows you to access the CFA533's LCD controller directly. Note: It is possible to corrupt the CFA533 display using this command.

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

The return packet will be:

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

23 (0x17): (reserved)

24 (0x18): Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the CFA533 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.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 48

```
#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
```

25-27 (0x19-0x1B): (reserved)

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 49

28 (0x1C): Set ATX Switch Functionality

The combination of the CFA533 with the Crystalfontz WR-PWR-Y14 cable can be used to replace the function of the power and reset switches in a standard ATX-compatible system. The ATX Power Switch Functionality is one of the items stored by the command 4 (0x04): Store Current State as Boot State (Pg. 38).

NOTE ON COMMAND 28: SET ATX SWITCH FUNCTIONALITY

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

These settings are factory default but may be changed by the user. Please see command 34 (0x22): Set/Configure GPIO (Pg. 53). 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 CFA533 are normally high-impedance. Electrically, they appear to be disconnected or floating. When the CFA533 asserts the RESET or POWER CONTROL lines, they are momentarily driven high or low (as determined by the AUTO_POLARITY, RESET_INVERT or POWER_INVERT bits, detailed below). To end the power or reset pulse, the CFA533 changes the lines back to high-impedance.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 50

FOUR FUNCTIONS MAY BE ENABLED BY COMMAND 28

Function 1: KEYPAD_RESET

If POWER-ON SENSE (GPIO[1]) is high, holding the green check key for 4 seconds will pulse RESET (GPIO[3]) pin for 1 second. During the 1-second pulse, the CFA533 will show "RESET", and then the CFA533 will reset itself, showing its boot state as if it had just powered on. Once the pulse has finished, the CFA533 will not respond to any commands until after it has reset the host and itself.

Function 2: KEYPAD_POWER_ON

If POWER-ON SENSE (GPIO[1]) is low, pressing the green check key for 0.25 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified by in data[1] or the default of 1 second. During this time the CFA533 will show "POWER ON", then the CFA533 will reset itself.

Function 3: KEYPAD_POWER_OFF

If POWER-ON SENSE (GPIO[1]) is high, holding the red X key for 4 seconds will pulse POWER CONTROL (GPIO[2]) for the duration specified by in data[1] or the default of 1 second. If the user continues to hold the power key down, then the CFA533 will continue to drive the line for a maximum of 5 additional seconds. During this time the CFA533 will show "POWER OFF".

Function 4: LCD_OFF_IF_HOST_IS_OFF

If LCD_OFF_IF_HOST_IS_OFF is set, the CFA533 will blank its screen and turn off its backlight to simulate its power being off any time POWER-ON SENSE is low.

NOTE

By default there is an internal POWER-ON-SENSE connected to the +5v pin of J_PWR, selected by setting data[2] to 1. Alternatively, GPIO[1] may be configured to act as POWER-ON-SENSE through R21 of 5k, and specifying data[2] as 0. The CFA533 will still be active (since it is powered by V_{SB}), monitoring the keypad for a power-on keystroke. Once POWER-ON SENSE goes high, the CFA533 will reboot as if power had just been applied to it.

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

type: 0x1C = 28₁₀
data_length: 1, 2 or 3
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
data[2]: (optional) atx_sense_on_floppy
        0: sense ATX host state on P2.1 (J8, pin 6 / GPIO [1] -- R21 must be loaded)
        1: sense ATX host state on P0.7 (JPWR,+5v -- recommended configuration))
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 51

The return packet will be:

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

29 (0x1D): Enable/Feed Host Watchdog Reset

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 CFA533. If the system monitor program fails to feed the CFA533's watchdog timer, the CFA533 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 Switch Functionality (Pg. 49) or command 34 (0x22): Set/Configure GPIO (Pg. 53).

```
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 feed
the watchdog and 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 CFA533 will reset the
host (see command 28 for details). Since the watchdog is off by default when the CFA533
powers up, the CFA533 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/ATX/Watchdog (debug)

This command can be used to verify the current items configured to report to the host, as well as some other miscellaneous status information. Please note that the information returned by the CFA533 is not identical to the information returned by other models in this family (CFA631, CFA633, CFA635).

```
type: 30
data_length: 0
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 52

The return packet will be:

```
type: 0x1E = 30₁₀
data_length: 15
data[0] = 0 (reserved)
data[1] = temperatures 1-8 reporting status (reserved)
data[2] = temperatures 9-15 reporting status (reserved)
data[3] = temperatures 16-23 reporting status (reserved)
data[4] = temperatures 24-32 reporting status (reserved)
data[5] = key presses (reserved)
data[6] = key releases (reserved)
data[7] = ATX Power Switch Functionality (as set by command 28)
data[8] = current watchdog counter (as set by command 29)
data[9] = User Contrast Adjust (as set by command 13, data[1])
data[10] = Key backlight setting (as set by command 14, data[1])
data[11] = atx_sense_on_floppy (as set by command 28)
data[12] = 0 (reserved)
data[13] = CFA633-style contrast setting (as set by command 13, data[0])
data[14] = LCD backlight setting (as set by command 14, data[0])
```

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 18
data[0]: col = x = 0 to 15
data[1]: row = y = 0 to 1
data[2-21]: text to place on the LCD, variable from 1 to 16 characters
```

The return packet will be:

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

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

32 (0x20): (Reserved for CFA631 Key Legends)

33 (0x21): Set I^{2}C Address

This command sets the I^{2}C address. This number is shifted left by one and combined with the R/$\overline{W}$ bit to make the actual I^{2}C address byte. For example, if 10 is set as the address, the I^{2}C address byte for *writes* is *20* and for *reads* is *21*.

The default address of the CFA533-YYH-KC is 42₁₀ (84₁₀ writes, 85₁₀ reads), in hexademicals 0x2A (0x54 writes, 0x55 reads).

Debugging Tip: To display the I^{2}C address of the module on the LCD, hold both the up and the down arrows for 4 seconds.

```
type: 0x21 = 33₁₀
data_length: 1
data[0]: 0 to 127
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 53

The return packet will be:

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

34 (0x22): Set/Configure GPIO

The CFA533 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 CFA533 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. (PWM Pulse Width Modulation is a way to simulate intermediate levels by switching a level between full on and full off. PWM is typically used to control the brightness of LED backlights, relying on the natural averaging done by the human eye.)

In output mode using the PWM (and a suitable supporting circuit), 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 CFA533 continuously polls the GPIOs as inputs at 32 Hz. The present level can be queried by the host software at a lower rate. The CFA533 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 CFA533 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.

GPIO[1] may be connected to the host's power in order to sense the host's power on/off state. There is resistor for R21 in series with GPIO[1] to limit the possibility of latchup. To use GPIO[1] as a general-purpose input/output, you may need to change R21 with a resistor suitable for your application. It is loaded with a 5kΩ resistor that is suitable for most applications.

> ### NOTE ON SETTING AND CONFIGURING GPIO PINS
> The GPIO pins may also be used for ATX control through header J8 and temperature sensing through the CFA533'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 GPIO configuration is one of the items stored by the command 4 (0x04): Store Current State as Boot State (Pg. 38).

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 54

```
type: 0x22 = 34₁₀
```
type: 0x22 = 34_{10}
```
data_length:
 2 bytes to change value only
 3 bytes to change value and configure function and drive mode

data[0]: index of GPIO to modify
        0 = GPIO[0] = J8, Pin 7
        1 = GPIO[1] = J8, Pin 6 (may be ATX Host Power Sense, as configured by
                        command 28, data[2])
        2 = GPIO[2] = J8, Pin 5 (default is ATX Host Power Control)
        3 = GPIO[3] = J8, Pin 4 (default is ATX Host Reset Control)
        4 = GPIO[4] = J9, Pin 2 (default is DOW I/O--has 1kΩ hardware pull-up)
    5-255 = 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
  101-255 = invalid

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

The return packet will be:

```
type: 0x40 │ 0x22 = 0x62 = 98₁₀
data_length: 0
```
type: 0x40 │ 0x22 = 0x62 = 98_{10}

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

Please see command 34 (0x22): Set/Configure GPIO (Pg. 53) for details on the GPIO architecture.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 55

```
type: 0x23 = 35₁₀
data_length: 1

data[0]: index of GPIO to query
        0 = GPIO[0] = J8, Pin 7
        1 = GPIO[1] = J8, Pin 6 (may be ATX Host Power Sense, as configured
                                  by command 28, data[2])
        2 = GPIO[2] = J8, Pin 5 (default is ATX Host Power Control)
        3 = GPIO[3] = J8, Pin 4 (default is ATX Host Reset Control)
        4 = GPIO[4] = J9, Pin 2 (default is DOW I/O--may have a 1kΩ hardware pull-up)
5-255 = reserved

 Please note: Future versions of this command on future hardware models may accept
 additional values for data[0],which would return the status of future additional GPIO
 pins.
returns:
  data[0]: index of GPIO read
  data[1]: Pin state & changes since last poll
    ---- -RFS
    │││││ ││││-- S = state at the last reading
    │││││ ││││--- F = at least one falling edge has
    │││││ ││││         been detected since the last poll
    │││││ ││││---- R = at least one rising edge has
    │││││ ││││         been detected since the last poll
    │││││ │----- reserved

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

    │││││ │----- F = Function
                ========================================================
                0: Port unused for GPIO. It will take on the default
                   function such as ATX, DOW or unused. The user is
                   responsible for setting the drive to the correct
                   value in order for the default function to work
                   correctly.
                1: Port used for GPIO under user control. The user is
                   responsible for setting the drive to the correct
                   value in order for the desired GPIO mode to work
                   correctly.
    │││││----- reserved, will return 0
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 56

CHARACTER GENERATOR ROM (CGROM)

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

Figure 19. Character Generator ROM (CGROM)

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 57

MODULE RELIABILITY AND LONGEVITY

MODULE RELIABILITY

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

MODULE LONGEVITY (EOL / REPLACEMENT POLICY)

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

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

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

Although the replacement module is still within the stated Data Sheet specifications and tolerances of the discontinued module, changes may require modification to your circuit and/or firmware. Possible changes include:
- *Backlight LEDs.* Brightness may be affected (perhaps the new LEDs have better efficiency) or the current they draw may change (new LEDs may have a different VF).
- *Controller.* A new controller may require minor changes in your code.
- *Component tolerances.* Module components have manufacturing tolerances. In extreme cases, the tolerance stack can change the visual or operating characteristics.

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 58

CARE AND HANDLING PRECAUTIONS

For optimum operation of the CFA533-YYH-KC and to prolong its life, please follow the precautions described below.

ELECTROSTATIC DISCHARGE (ESD)

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

CLEANING

- The polarizer (laminated to the glass) is soft plastic. The soft plastic is easily scratched or damaged. Be very careful when you clean the polarizer.
- Do not clean the polarizer with liquids. Do not wipe the polarizer with any type of cloth or swab (for example, Q-tips).
- Use the removable protective film to remove smudges (for example, fingerprints) and any foreign matter. If you no longer have the protective film, use standard transparent office tape (for example, Scotch® brand "Crystal Clear Tape"). If the polarizer is dusty, you may carefully blow it off with clean, dry, oil-free compressed air.
- CFA533-YYH-KC *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.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 59

OPERATION

- Your circuit should be designed to protect the CFA533-YYH-KC from ESD and power supply transients.
- Observe the operating temperature limitations: a minimum of 0°C to a maximum of 50°C with minimal fluctuation. Operation outside of these limits may shorten life and/or harm display.
    - At lower temperatures of this range, response time is delayed.
    - At higher temperatures of this range, display becomes dark. (You may need to adjust the contrast.)
- Operate away from dust, moisture, and direct sunlight.

STORAGE AND RECYCLING

- Store in an ESD-approved container away from dust, moisture, and direct sunlight.
- Observe the storage temperature limitations: -10°C minimum, 60°C maximum with minimal fluctuation. Rapid temperature changes can cause moisture to form, resulting in permanent damage.
- Do not allow weight to be placed on the CFA533-YYH-KCs while they are in storage.
- Please recycle your outdated Crystalfontz modules at an approved facility.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 60

APPENDIX A: QUALITY ASSURANCE STANDARDS

INSPECTION CONDITIONS

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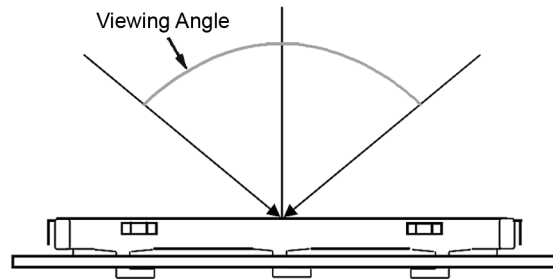
COLOR DEFINITIONS

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

DEFINITION OF ACTIVE AREA AND VIEWABLE AREA

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 61
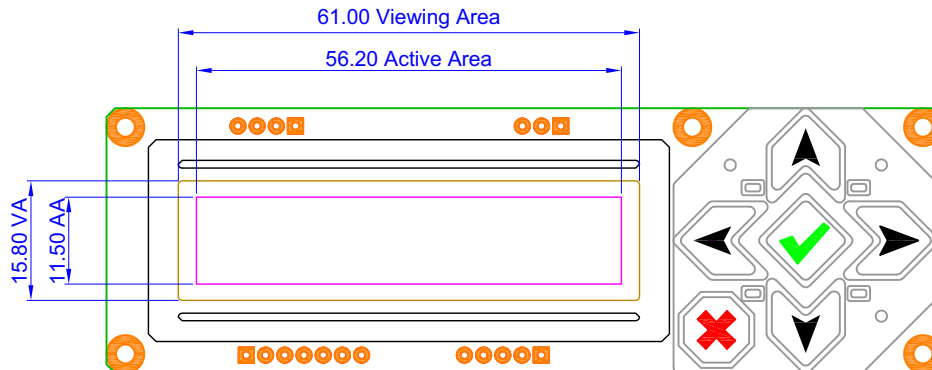
ACCEPTANCE SAMPLING
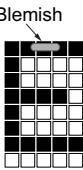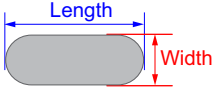
DEFECT TYPE	AQL*
Major	≤0.65%
Minor	≤1.00%
Acceptable Quality Level: maximum allowable error rate or variation from standard.	

DEFECTS CLASSIFICATION
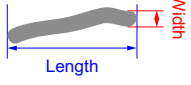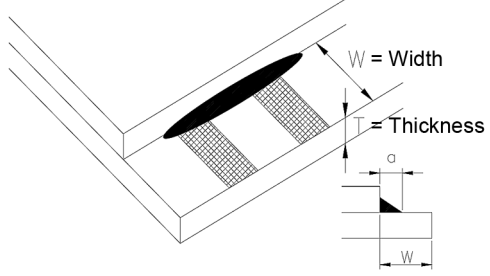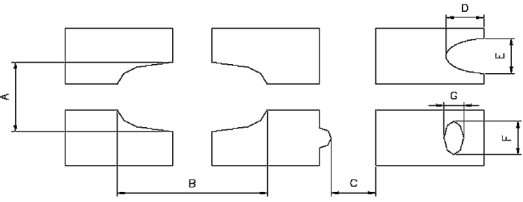
Defects are defined as:

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

ACCEPTANCE STANDARDS

#	DEFECT TYPE	CRITERIA			MAJOR / MINOR
1	Electrical defects	1. No display, display malfunctions, or shorted segments. 2. Current consumption exceeds specifications.			Major
2	Viewing area defect	Viewing area does not meet specifications. (See Inspection Conditions (Pg. 60).			Major
3	Contrast adjustment defect	Contrast adjustment fails or malfunctions.			Major
4	Blemishes or foreign matter on display segments	Blemish	*Defect Size (mm)*	*Acceptable Qty*	Minor
			≤0.3	3	
			≤2 defects within 10 mm of each other		
5	Other blemishes or foreign matter outside of display segments	Defect size = (A + B)/2	*Defect Size (mm)*	*Acceptable Qty*	Minor
			≤0.15	Ignore	
			0.15 to 0.20	3	
			0.20 to 0.25	2	
			0.25 to 0.30	1	

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 62

#	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	W = Width T = Thickness a≤1/4W			Minor
9	Display pattern defect				Minor
		Dot Size (mm)	*Acceptable Qty*		
		((A+B)/2)≤0.2			
		C>0	≤3 total defects ≤2 pinholes per digit		
		((D+E)/2)≤0.25			
		((F+G)/2)≤0.25			

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 63

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 64

#	DEFECT TYPE	CRITERIA				MAJOR / MINOR
13	Chip on "contact" edge of LCD, on the inactive side					Minor
		a	*b*	*c*	*Acceptable Qty*	
		≤3 mm	≤1 mm	≤T	Ignore	
		≤4 mm	≤1.5 mm	≤T	3	
14	Chip in seal area					
		a	*b*	*c*	*Acceptable Qty*	
		<3 mm	≤1.5 mm	≤1/2 T	3	Minor
		Unacceptable if c>50% of glass thickness or if the seal area is damaged.				Major
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. 61), and dark lines or scratches (see test 6, Pg. 62). *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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 65

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 66

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

SAMPLE CODE

Free downloadable code is on our website http://www.crystalfontz.com/products/document/2097/CFA-533-I2C-AVR.zip (demonstration code for Atmel AVR ATMega 2561) and http://www.crystalfontz.com/product/533_I2C_WinTest (Windows compatible test/demonstration program and source). Please leave the original copyrights in the code.

ALGORITHMS TO CALCULATE THE CRC

Below are eight sample algorithms that will calculate the CRC of a CFA533 packet. Some of the algorithms were contributed by forum members and originally written for the CFA631 or CFA635. The CRC used in the CFA533 is the same one that is used in IrDA, which came from PPP, which seems to be related to a CCITT (ref: Network Working Group Request for Comments: 1171) standard. At that point, the trail was getting a bit cold and diverged into several referenced articles and papers, dating back to 1983.

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

Algorithm 1: "C" Table Implementation

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

```
//This code is from the IRDA LAP documentation, which appears to
//have been copied from PPP:
//
// http://irda.affiniscape.com/associations/2494/files/Specifications/
IrLAP11_Plus_Errata.zip
//
//I doubt that there are any worries about the legality of this code,
//searching for the first line of the table below, it appears that
//the code is already included in the linux 2.6 kernel "Driver for
//ST5481 USB ISDN modem". This is an "industry standard" algorithm
//and I do not think there are ANY issues with it at all.
typedef unsigned char ubyte;
typedef unsigned short word;
word get_crc(ubyte *bufptr,word len)
  {
  //CRC lookup table to avoid bit-shifting loops.
  static const word crcLookupTable[256] =
    {0x00000,0x01189,0x02312,0x0329B,0x04624,0x057AD,0x06536,0x074BF,
     0x08C48,0x09DC1,0x0AF5A,0x0BED3,0x0CA6C,0x0DBE5,0x0E97E,0x0F8F7,
     0x01081,0x00108,0x03393,0x0221A,0x056A5,0x0472C,0x075B7,0x0643E,
     0x09CC9,0x08D40,0x0BFDB,0x0AE52,0x0DAED,0x0CB64,0x0F9FF,0x0E876,
     0x02102,0x0308B,0x00210,0x01399,0x06726,0x076AF,0x04434,0x055BD,
     0x0AD4A,0x0BCC3,0x08E58,0x09FD1,0x0EB6E,0x0FAE7,0x0C87C,0x0D9F5,
     0x03183,0x0200A,0x01291,0x00318,0x077A7,0x0662E,0x054B5,0x0453C,
     0x0BDCB,0x0AC42,0x09ED9,0x08F50,0x0FBEF,0x0EA66,0x0D8FD,0x0C974,
     0x04204,0x0538D,0x06116,0x0709F,0x00420,0x015A9,0x02732,0x036BB,
     0x0CE4C,0x0DFC5,0x0ED5E,0x0FCD7,0x08868,0x099E1,0x0AB7A,0x0BAF3,
     0x05285,0x0430C,0x07197,0x0601E,0x014A1,0x00528,0x037B3,0x0263A,
     0x0DECD,0x0CF44,0x0FDDF,0x0EC56,0x098E9,0x08960,0x0BBFB,0x0AA72,
     0x06306,0x0728F,0x04014,0x0519D,0x02522,0x034AB,0x00630,0x017B9,
     0x0EF4E,0x0FEC7,0x0CC5C,0x0DDD5,0x0A96A,0x0B8E3,0x08A78,0x09BF1,
     0x07387,0x0620E,0x05095,0x0411C,0x035A3,0x0242A,0x016B1,0x00738,
     0x0FFCF,0x0EE46,0x0DCDD,0x0CD54,0x0B9EB,0x0A862,0x09AF9,0x08B70,
     0x08408,0x09581,0x0A71A,0x0B693,0x0C22C,0x0D3A5,0x0E13E,0x0F0B7,
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 67

```
      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;
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 68

```
      //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);
}
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 69

Algorithm 2B: "C" Improved Bit Shift Implementation

This is 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 CFA533 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 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
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 70

```
savchr      equ     46h             ; BYTE - temp storage for CRC routine
;
seedlo      equ     021h            ; initial seed for CRC reg lo byte
seedhi      equ     0F3h            ; initial seed for CRC reg hi byte
;
polyL       equ     008h            ; polynomial low byte
polyH       equ     084h            ; polynomial high byte
;=================================================================
;  CRC Test Program
;-----------------------------------------------------------------
        org     0               ; reset vector = 0000H
;
        clrf    PCLATH          ; ensure upper bits of PC are cleared
        clrf    STATUS          ; ensure page bits are cleared
        goto    main            ; jump to start of program
;
; ISR Vector
;
        org     4               ; start of ISR
        goto    $               ; jump to ISR when coded
;
        org     20              ; start of main program
main
        movlw   seedhi          ; setup intial CRC seed value.
        movwf   accumh          ; This must be done prior to
        movlw   seedlo          ; sending string to CRC routine.
        movwf   accuml          ;
        clrf    index           ; clear string read variables
;
main1
        movlw   HIGH InputStr ; point to LCD test string
        movwf   PCLATH          ; latch into PCL,
        movfw   index           ; get index
        call    InputStr        ; get character
        movwf   Zero            ; setup for terminator test
        movf    Zero,f          ; see if terminator
        btfsc   STATUS,Z        ; skip if not terminator
        goto    main2           ; else terminator reached, jump out of loop
        call    CRC16           ; calculate new crc
        call    SENDUART        ; send data to LCD
        incf    index,f         ; bump index
        goto    main1           ; loop
;
main2
        movlw   00h             ; shift accumulator 16 more bits.
        call    CRC16           ; This must be done after sending
        movlw   00h             ; string to CRC routine.
        call    CRC16           ;
;
        comf    accumh,f        ; invert result
        comf    accuml,f        ;
;
        movfw   accuml          ; get CRC low byte
        call    SENDUART        ; send to LCD
        movfw   accumh          ; get CRC hi byte
        call    SENDUART        ; send to LCD
;
stop    goto    stop                ; word result of 0x93FA is in accumh/accuml
;=================================================================
; calculate CRC of input byte
;-----------------------------------------------------------------
CRC16
        movwf   savchr          ; save the input character
        movwf   datareg         ; load data register
        movlw   .8              ; setup number of bits to test
        movwf   j               ; save to incrementor
_loop
        clrc                    ; clear carry for CRC register shift
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 71

```
        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 CFA533 packet) data. This routine was adapted from the C table implementation. The complete project can be found in our forums.

```
'This program is brutally blunt. Just like VB. No apologies.
'Written by Crystalfontz America, Inc. 2004 http://www.crystalfontz.com
'Free code, not copyright copyleft or anything else.
'Some visual basic concepts taken from:
'http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=21434&lngWId=1
'most of the algorithm is from functions in 633_WinTest:
'https://www.crystalfontz.com/product/633WinTest#docs
'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
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 72

```
'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];
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 73

```java
  // 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,
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 74

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

Algorithm 6: "Perl" Table Implementation

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

```perl
#!/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
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 75

```perl
my $type: '80';
my $length = '01';
my $data = '05';
my $packet = chr(hex $type) . chr(hex $length) . chr(hex $data) ;


my $valid_crc = '5584' ;

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

my $crc = 0xFFFF ;

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

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

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

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

Algorithm 7: For PIC18F8722 or PIC18F2685

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

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 76

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 77

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
**Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0**
Page 78

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

This appendix describes a simple test circuit that demonstrates how to connect a Dallas Semiconductor DS2450 4-channel ADC to the CFA533'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 CFA533. In this example the DS2450 appears at device index 0. Your software should query the connected devices using command 18 (0x12): Read DOW Device Information (Pg. 44) to verify the locations and types of DOW devices connected in your application.

Please refer to the DS2450 Data Sheet and the description for command 20 (0x14): Arbitrary DOW Transaction (Pg. 45) more information.
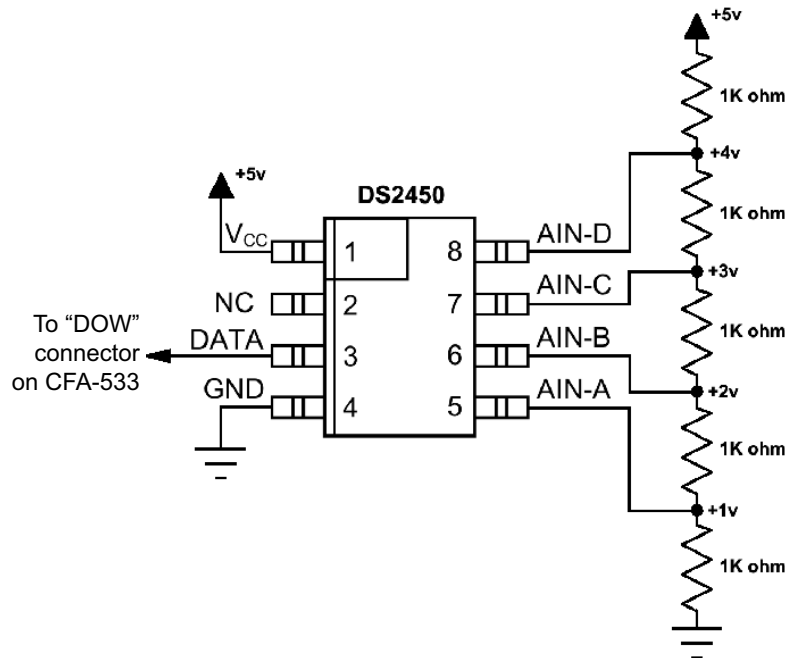
Appendix C Figure 1. Test Circuit Schematic

Start 533_I2C_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.

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I2C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 79

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 80

APPENDIX D: EXAMPLE OF I²C SOURCE CODE FOR THE ATMEL AVR ATMEGA2561

Sample code for I²C communications between the CFA533-YYH-KC and an Atmel AVR ATMega2561 is below. Files may be downloaded at http://www.crystalfontz.com/products/document/2097/CFA-533-I2C-AVR.zip.

TWI_driver.c

```c
// This is demo code for a CFA-10006 talking to a CFA-533-*-KC:
// http://www.crystalfontz.com/product/CFA10006
// http://www.crystalfontz.com/product/CFA533-TMI-KC.html
//
// TWI hacked to top two pins of "PORTD" on the 10K6
//   SCL = PD0
//   SDA = PD1 (lift R7 to disable LED1)
// 1K pull-ups patched onto those two pins to +5v
// CFA533-TMI-KC hooked to PD0/PD1 (SCL/SDA)

// 2000 / 12 / 05: Original source is AVR155 by Asmund Saetre
// 2010 / 01 / 07: This code was modified heavily by Crystalfontz. -BAC

//////////////////////////////////////////////////////////////////////
#include <avr/io.h>
#include "TWI_driver.h"

//////////////////////////////////////////////////////////////////////
//  Function : char Init_TWI(void)
//  Setup the TWI module
//  Baudrate    : 100kHz @ 4MHz system clock
//  Own address : OWN_ADR (Defined in TWI_driver.h)
//////////////////////////////////////////////////////////////////////
char Init_TWI(void)
  {
  TWAR = OWN_ADR;                      //Set own slave address
  TWBR = 12;                           //Set cycle time to 400 KHz
                                       // 12 -> 400 KHz measured
                                       // 72 -> 100 KHz measured
  TWCR = (1<<TWEN);                    //Enable TWI-interface
  return 1;
  }
//////////////////////////////////////////////////////////////////////
//  Function : void Wait_TWI_int(void)
//  Loop until TWI interrupt flag is set
//////////////////////////////////////////////////////////////////////
void Wait_TWI_int(void)
  {
  while (!(TWCR & (1<<TWINT)));
  }
//////////////////////////////////////////////////////////////////////
//  Function :unsigned char Send_start(void)
//  Send a START condition to the bus and wait for the TWINT get set to
//  see the result. If it failed return the TWSR value, if succes return
//  SUCCESS.
//////////////////////////////////////////////////////////////////////
unsigned char   Send_start(void)
  {
  TWCR = ((1<<TWINT)+(1<<TWSTA)+(1<<TWEN));//Send START

  Wait_TWI_int();                          //Wait for TWI interrupt flag set

  if((TWSR != START)&&(TWSR != REP_START)) //If status other than START
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 81

```
      return TWSR;                           //transmitted(0x08) or Repeated
   return SUCCESS;                           //START transmitted(0x10)
                                             //-> error  and return TWSR.
                                             //If success return   SUCCESS
   }
//////////////////////////////////////////////////////////////////////////
//  Function :
//  Send a STOP condition to the bus
//////////////////////////////////////////////////////////////////////////
void Send_stop(void)
   {
   TWCR = ((1<<TWEN)+(1<<TWINT)+(1<<TWSTO)); //Send STOP condition
   }
//////////////////////////////////////////////////////////////////////////
//  Function : unsigned char Send_byte(unsigned char data)
//  Send one byte to the bus.
//////////////////////////////////////////////////////////////////////////
unsigned char Send_byte(unsigned char data)
   {
   Wait_TWI_int();                           //Wait for TWI interrupt flag set

   TWDR = data;
   TWCR = ((1<<TWINT)+(1<<TWEN));            //Clear int flag to send byte

   Wait_TWI_int();                           //Wait for TWI interrupt flag set

   if(TWSR != MTX_DATA_ACK)                  //If NACK received return TWSR
      return TWSR;
   return SUCCESS;                           //Else return SUCCESS
   }
//////////////////////////////////////////////////////////////////////////
//  Function : unsigned char Send_adr(unsigned char adr)
//  Send a SLA+W/R to the bus
//////////////////////////////////////////////////////////////////////////
unsigned char Send_adr(unsigned char adr)
   {
   Wait_TWI_int();                           //Wait for TWI interrupt flag set

   TWDR = adr;
   TWCR = ((1<<TWINT)+(1<<TWEN));            //Clear int flag to send byte

   Wait_TWI_int();                           //Wait for TWI interrupt flag set

   if((TWSR != MTX_ADR_ACK)&&(TWSR != MRX_ADR_ACK)) //If NACK received return
         return TWSR;                        //TWSR
      return SUCCESS;                        //Else return SUCCESS
   }
//////////////////////////////////////////////////////////////////////////
unsigned char Get_byte_and_ACK(void)
   {
   Wait_TWI_int();                           //Wait for TWI interrupt flag set

   //Clear int flag and enable acknowledge to receive data.
   TWCR = ((1<<TWINT)+(1<<TWEA)+(1<<TWEN));

   Wait_TWI_int();                           //Wait for TWI interrupt flag set

   return(TWDR);                             //Save received byte
   }
//////////////////////////////////////////////////////////////////////////
unsigned char Get_byte_and_NACK(void)
   {
   Wait_TWI_int();                           //Wait for TWI interrupt flag set

   //Clear int flag to and do not enable acknowledge to tell the slave
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 82

```
  //to stop transmitting
  TWCR = ((1<<TWINT)+(1<<TWEN));

  Wait_TWI_int();                          //Wait for TWI interrupt flag set

  return(TWDR);                            //Save received byte
  }
//////////////////////////////////////////////////////////////////////////
```

CFA-533-I2C-AVR.C

```
// This is demo code for a CFA-10006 talking to a CFA-533-*-KC:
// http://www.crystalfontz.com/product/CFA10006
// http://www.crystalfontz.com/product/CFA533-TMI-KC.html
//
// TWI hacked to top two pins of "PORTD" on the 10K6
//   SCL = PD0
//   SDA = PD1 (lift R7 to disable LED1)
// 1K pull-ups patched onto those two pins to +5v
// CFA533-TMI-KC hooked to PD0/PD1 (SCL/SDA)


#include <avr/io.h>
#include <util/delay.h>

#include "TWI_driver.h"

#define LED_PORT                PORTD
#define LED2                    0x02
#define CLR_LED2                LED_PORT &= ~(1 << LED2);
#define SET_LED2                LED_PORT |= (1 << LED2);

#define CFA533_ADDRESS          42

// 1/16MHz *  80000 = 5mS
#define DELAY5mS                80000L

// 1/16MHz *  2000000 = 125mS
#define DELAY125mS              2000000L

//============================================================================
#define KP_UP                   0x01
#define KP_ENTER                0x02
#define KP_CANCEL               0x04
#define KP_LEFT                 0x08
#define KP_RIGHT                0x10
#define KP_DOWN                 0x20
//============================================================================
typedef unsigned char   ubyte;
typedef signed char     sbyte;
typedef unsigned short  uword;
typedef signed short    sword;
typedef unsigned long   ulong;
typedef signed long     slong;
//This structure allows access to the individual bytes in a word.
typedef union
  {
  ubyte
    as_bytes[2];
  uword
    as_word;
  }WORD_UNION;
//============================================================================
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 83

```
#define MAX_DATA_LENGTH 18
#define MAX_COMMAND     35

typedef struct
  {
  ubyte
    command;
  ubyte
    data_length;
  ubyte
    data[MAX_DATA_LENGTH];
  WORD_UNION
    CRC;
  }CFA533_PACKET;
//==========================================================================
CFA533_PACKET
  outgoing_command;
CFA533_PACKET
  incoming_response;
////========================================================================
//CRC lookup table to avoid bit-shifting loops.
const uword 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};
uword get_crc(ubyte *bufptr,ubyte len)
  {
  register uword
    newCrc;
  newCrc=0xFFFF;
  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);
  }
//==========================================================================
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I²C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 84

```
//                              send_packet()
//
// send_packet() will send set the CRC in outgoing_response and send it to
// the host.
//------------------------------------------------------------------------
void send_packet(void)
  {
  ubyte
    i;

  Send_start();

  Send_adr((CFA533_ADDRESS<<1)|W);  //Write

  Send_byte(outgoing_command.command);
  Send_byte(outgoing_command.data_length);

  for(i=0;i<outgoing_command.data_length;i++)
    Send_byte(outgoing_command.data[i]);

  //Set the CRC
  outgoing_command.CRC.as_word=
    get_crc((ubyte *)&outgoing_command,outgoing_command.data_length+2);

  //Now send the CRC.
  Send_byte(outgoing_command.CRC.as_bytes[0]);
  Send_byte(outgoing_command.CRC.as_bytes[1]);

  Send_stop();
  }
//============================================================================
//                            check_for_packet()
//
// check_for_packet() will read whatever is available in the CFA-533's I2C
// output buffer, and determine if it is a valid packet.
//------------------------------------------------------------------------
ubyte check_for_packet(void)
  {
  ubyte
    i;

  Send_start();

  Send_adr((CFA533_ADDRESS<<1)|R);  //Read


  incoming_response.command=Get_byte_and_ACK();

  //Only commands 0 through MAX_COMMAND are valid.
  //Mask response (0x40) and report (0x80) flags
  if(MAX_COMMAND<(incoming_response.command&0x3F))
    return(0);

  incoming_response.data_length=Get_byte_and_ACK();

  //There is a valid command byte. Get the data_length. The data length
  //must be within reason.
  if(MAX_DATA_LENGTH<incoming_response.data_length)
    return(0);

  //Read the data[] array of the packet
  for(i=0;i<incoming_response.data_length;i++)
    incoming_response.data[i]=Get_byte_and_ACK();

  //Pull in the first byte of the CRC
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 85

```
  incoming_response.CRC.as_bytes[0]=Get_byte_and_ACK();
  //Pull in the last byte, and terminate the transaction.
  incoming_response.CRC.as_bytes[1]=Get_byte_and_NACK();

  //Now check the CRC.
  if(incoming_response.CRC.as_word==
    get_crc((ubyte *)&incoming_response,incoming_response.data_length+2))
    {
    //This is a good packet. I'll be horn swaggled.
    //Let our caller know that incoming_command has good stuff in it.
    return(1);
    }
  //The CRC did not match.
  return(0);
  }
//===========================================================================
int main (void)
  {
  /* Port D - Bit 7: KEY0,                 input, pull-up
   *           6: KEY1,                 input, pull-up
   *           5: KEY2,                 input, pull-up
   *           4: KEY3,                 input, pull-up
   *           3: NC,                   input, pull-up
   *           2: LED2,                 output
   *           1: TWI_SDA
   *           0: TWI_SCL
   */
  DDRD =  0b00000100;
  PORTD = 0b11111111;

  Init_TWI();

  //Show the base screen, top line
  outgoing_command.command=31;            //31 (0x1F): Send Data to LCD
  outgoing_command.data_length=12;        //Data Length
  outgoing_command.data[0]=0;             //X = left
  outgoing_command.data[1]=0;             //Y = first line

                      //0123456789012345
                      //0000000000111111
                      //CFA-533   UECLRD
  outgoing_command.data[ 2]='C';
  outgoing_command.data[ 3]='F';
  outgoing_command.data[ 4]='A';
  outgoing_command.data[ 5]='-';
  outgoing_command.data[ 6]='5';
  outgoing_command.data[ 7]='3';
  outgoing_command.data[ 8]='3';
  outgoing_command.data[ 9]=' ';
  outgoing_command.data[10]=' ';
  outgoing_command.data[11]=' ';
  send_packet();
  __builtin_avr_delay_cycles(DELAY5mS);


  //Show the base screen bottom line
  outgoing_command.command=31;            //31 (0x1F): Send Data to LCD
  outgoing_command.data_length=12;        //Data Length
  outgoing_command.data[0]=0;             //X = left
  outgoing_command.data[1]=1;             //Y = second line

                      //0123456789012345
                      //0000000000111111
                      //I2C Demo  ------
  outgoing_command.data[ 2]='I';
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 86

```
outgoing_command.data[ 3]='2';
outgoing_command.data[ 4]='C';
outgoing_command.data[ 5]=' ';
outgoing_command.data[ 6]='D';
outgoing_command.data[ 7]='e';
outgoing_command.data[ 8]='m';
outgoing_command.data[ 9]='o';
outgoing_command.data[10]=' ';
outgoing_command.data[11]=' ';
send_packet();
__builtin_avr_delay_cycles(DELAY5mS);


while(1)
  {
  SET_LED2;
  __builtin_avr_delay_cycles(DELAY125mS); // 1/8 second
  CLR_LED2;
  __builtin_avr_delay_cycles(DELAY125mS); // 1/8 second

  //Poll the keypad ~4x per second
  outgoing_command.command=24;            //24 (0x18): Read Keypad, Polled Mode
  outgoing_command.data_length=0;         //Data Length = 0
  send_packet();
  __builtin_avr_delay_cycles(DELAY5mS);

  check_for_packet();

  // See if there is a read polled keypad response,
  // if so, put the key status on the LCD
  if(incoming_response.command == 0x58)
    {
    //Show the Key's Up/Down state
    //On the top line

    outgoing_command.command=31;               //31 (0x1F): Send Data to LCD
    outgoing_command.data_length=8;            //Data Length
    outgoing_command.data[0]=10;               //X = right
    outgoing_command.data[1]=0;                //Y = first line

    if(incoming_response.data[0]&KP_UP)
      outgoing_command.data[2]='U';
    else
      outgoing_command.data[2]='u';

    if(incoming_response.data[0]&KP_ENTER)
      outgoing_command.data[3]='E';
    else
      outgoing_command.data[3]='e';

    if(incoming_response.data[0]&KP_CANCEL)
      outgoing_command.data[4]='C';
    else
      outgoing_command.data[4]='c';

    if(incoming_response.data[0]&KP_LEFT)
      outgoing_command.data[5]='L';
    else
      outgoing_command.data[5]='l';

    if(incoming_response.data[0]&KP_RIGHT)
      outgoing_command.data[6]='R';
    else
      outgoing_command.data[6]='r';
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 87

```
if(incoming_response.data[0]&KP_DOWN)
  outgoing_command.data[7]='D';
else
  outgoing_command.data[7]='d';

send_packet();
__builtin_avr_delay_cycles(DELAY5mS);

//The default CFA-533 CGRAM has:
//  up    arrow at character 2
//  dn    arrow at character 3
//  up+dn arrow at character 7
//Use these characters to show the changes
//below the keys

outgoing_command.command=31;            //31 (0x1F): Send Data to LCD
outgoing_command.data_length=8;         //Data Length
outgoing_command.data[0]=10;            //X = right
outgoing_command.data[1]=1;             //Y = first line

if((incoming_response.data[1]&KP_UP)&&  //a press and a
   (incoming_response.data[2]&KP_UP))   //release
  outgoing_command.data[2]=7;
else
  if(incoming_response.data[1]&KP_UP)   //press
    outgoing_command.data[2]=3;
  else
    if(incoming_response.data[2]&KP_UP) //release
      outgoing_command.data[2]=2;
    else
      outgoing_command.data[2]='-';

if((incoming_response.data[1]&KP_ENTER)&&  //a press and a
   (incoming_response.data[2]&KP_ENTER))    //release
  outgoing_command.data[3]=7;
else
  if(incoming_response.data[1]&KP_ENTER)    //press
    outgoing_command.data[3]=3;
  else
    if(incoming_response.data[2]&KP_ENTER) //release
      outgoing_command.data[3]=2;
    else
      outgoing_command.data[3]='-';

if((incoming_response.data[1]&KP_CANCEL)&&  //a press and a
   (incoming_response.data[2]&KP_CANCEL))    //release
  outgoing_command.data[4]=7;
else
  if(incoming_response.data[1]&KP_CANCEL)    //press
    outgoing_command.data[4]=3;
  else
    if(incoming_response.data[2]&KP_CANCEL) //release
      outgoing_command.data[4]=2;
    else
      outgoing_command.data[4]='-';

if((incoming_response.data[1]&KP_LEFT)&&  //a press and a
   (incoming_response.data[2]&KP_LEFT))    //release
  outgoing_command.data[5]=7;
else
  if(incoming_response.data[1]&KP_LEFT)    //press
    outgoing_command.data[5]=3;
  else
    if(incoming_response.data[2]&KP_LEFT) //release
      outgoing_command.data[5]=2;
```

Crystalfontz America, Inc.
www.crystalfontz.com
March 2011

CFA533-YYH-KC Intelligent I^{2}C Module Data Sheet
Hardware v1.0 / Firmware vc1.0 / Data Sheet v2.0
Page 88

```c
          else
            outgoing_command.data[5]='-';

      if((incoming_response.data[1]&KP_RIGHT)&&  //a press and a
         (incoming_response.data[2]&KP_RIGHT))   //release
        outgoing_command.data[6]=7;
      else
        if(incoming_response.data[1]&KP_RIGHT)   //press
          outgoing_command.data[6]=3;
        else
          if(incoming_response.data[2]&KP_RIGHT) //release
            outgoing_command.data[6]=2;
          else
            outgoing_command.data[6]='-';

      if((incoming_response.data[1]&KP_DOWN)&&    //a press and a
         (incoming_response.data[2]&KP_DOWN))     //release
        outgoing_command.data[7]=7;
      else
        if(incoming_response.data[1]&KP_DOWN)    //press
          outgoing_command.data[7]=3;
        else
          if(incoming_response.data[2]&KP_DOWN) //release
            outgoing_command.data[7]=2;
          else
            outgoing_command.data[7]='-';

      send_packet();
      __builtin_avr_delay_cycles(DELAY5mS);
      }
    }
  }
//=============================================================================
```