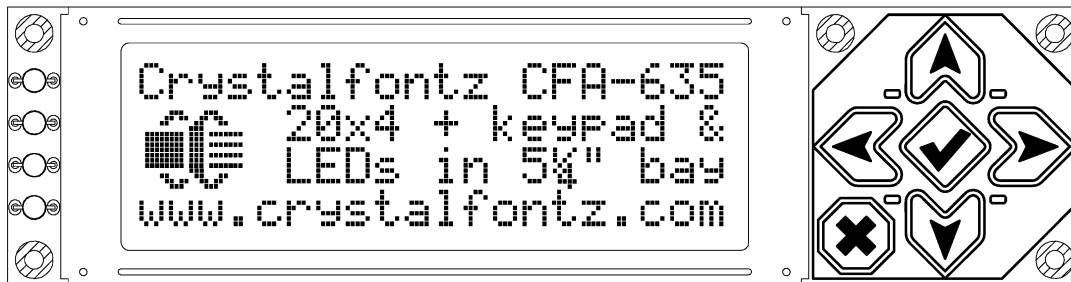


CrystalFontz America, Incorporated

LCD MODULE SPECIFICATIONS



CrystalFontz Model Number	CFA-635
Firmware Version	v1.0
Hardware Version	v1.0
Product Page	http://www.crystalfontz.com/products/635/

Customer Name	
Customer Part Number	

CrystalFontz America, Inc.

12412 East Saltese Avenue
Spokane Valley, WA 99036-0357

Phone: (509) 892-1200

Fax: (509) 892-1203

e-mail: sales@crystalfontz.com

<http://www.crystalfontz.com>



Table of Contents

■ FEATURES.....	4
■ SYSTEM BLOCK DIAGRAM	5
■ MECHANICAL CHARACTERISTICS.....	5
■ GENERAL SPECIFICATIONS	5
■ ELECTRICAL SPECIFICATIONS	6
■ RELIABILITY	6
■ HOST CONNECTIONS	6
■ GPIO/GPO CONNECTIONS.....	8
■ HOST COMMUNICATIONS	9
PACKET STRUCTURE	9
HANDSHAKING	10
REPORT CODES	10
0x80: Key Activity	11
0x81: Fan Speed Report (reserved, additional hardware required).....	11
0x82: Temperature Sensor Report (reserved, additional hardware required)	11
COMMAND CODES	11
0: Ping Command.....	11
1: Get Hardware & Firmware Version.....	12
2: Write User Flash Area	12
3: Read User Flash Area	12
4: Store Current State As Boot State.....	12
5: Reboot CFA-635, Reset Host, or Power Off Host.....	13
6: Clear LCD Screen	14
7: (deprecated, see command 31).....	14
8: (deprecated, see command 31).....	14
9: Set LCD Special Character Data.....	14
10: Read 8 Bytes of LCD Memory	15
11: Set LCD Cursor Position	15
12: Set LCD Cursor Style	16
13: Set LCD Contrast	16
14: Set LCD & Keypad Backlight.....	17
15: (deprecated)	17
16: Set Up Fan Reporting (reserved, additional hardware required)	17
17: Set Fan Power (reserved, additional hardware required)	17
18: Read DOW Device Information (reserved, additional hardware required).....	17
19: Set Up Temperature Reporting (reserved, additional hardware required).....	17
20: Arbitrary DOW Transaction (reserved, additional hardware required)	17
21: (deprecated)	17
22: Send command directly to the LCD controller	17
23: Configure Key Reporting	17
24: Read Keypad, Polled Mode	18
25: Set Fan Power Fail-Safe (reserved, additional hardware required).....	19
26: Set Fan Tachometer Glitch Delay (reserved, additional hardware required)	19
27: Query Fan Power & Fail-Safe Mask (reserved, additional hardware required).....	19



28: Set ATX Power Switch Functionality (reserved, additional hardware required).....	19
29: Enable/Disable and Reset the Watchdog (reserved, additional hardware required)	19
30: Read Reporting & Status	19
31: Send Data to LCD	19
32: reserved (CFA-631 Key Legends).....	20
33: Set Baud Rate	20
34: Set or Set and Configure GPIO pin	20
35: Read GPIO pin levels and configuration state.....	22
■ CHARACTER GENERATOR ROM (CGROM).....	24
■ MODULE OUTLINE DRAWING	25
■ KEYPAD OUTLINE DRAWING.....	26
■ JUMPER REFERENCE	27

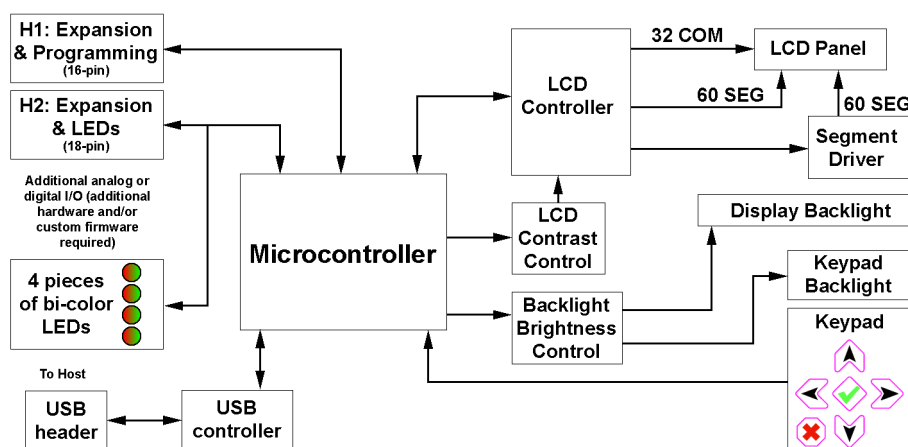


■ FEATURES

- LED backlit STN 20x4 LCD
- Compact size: fits in a 1U rack mount case (37mm overall height)
- USB interface for both power and data (115200 baud equivalent throughput)
- LED backlit 6-button translucent silicon keypad with screened legend
- Two color options:
 - Yellow-green backlit positive mode LCD, yellow/green keypad backlight ("YYE")
 - White backlit negative mode LCD, blue keypad backlight ("TMF")
- Four bi-color (red + green) LEDs. The LEDs' brightness can set by the host software, which allows mixing the LEDs to produce other colors (yellow and orange)
- LCD characters are contiguous in both X and Y directions to allow the host software to display "gapless" bar graphs in horizontal or vertical directions
- Fully decoded keypad: any key combination is valid and unique
- Robust packet-based communications protocol with 16-bit CRC
- Built-in factory re-programmable microcontroller
- Non-volatile memory capability ("EEPROM"):
 - Customize the "power-on" display and LED settings
 - 16-byte "scratch" register for storing IP, netmask, system serial number . . .
- 5.25" half-height drive-bay mounting bracket available (optional)



■ SYSTEM BLOCK DIAGRAM



■ MECHANICAL CHARACTERISTICS

Item	Dimensions	Unit
PCB Outline Size	142.0(W) x 37.0(H)	mm
Viewing Area	82.95(W) x 27.5(H)	mm
Active Area	77.95(W) x 22.35(H)	mm
Character Size	3.85(W) x 5.55(H)	mm
Dot Size	0.60(W) x 0.65(H)	mm
Dot Pitch	0.65(W) x 0.70(H)	mm
Thickness:		
Without Keypad or Connectors	10.6	mm
With Keypad, without Connectors	14.4	mm
Without Keypad, with Connectors	15.5	mm
With Keypad, with Connectors	19.3	mm
Keystroke Travel (approximate)	2.4	mm

■ GENERAL SPECIFICATIONS

Operating Temperature: 0°C minimum, 50°C maximum

Storage Temperature: -10°C minimum, 60°C maximum

LCD Glass Type: STN, Yellow/Green Positive Mode or Blue Negative Mode

Viewing Direction: 12 O'clock

Polarizer Type: Transflective

Driving Method: 1/32 Duty, 1/6.7 Bias

Backlight: LED, Yellow/Green (568nm nominal) or White

Backlight PWM frequency: 320Hz nominal

Weight: 65 grams typical



■ ELECTRICAL SPECIFICATIONS

Required voltage supplies:

V_{USB}: 4.75v minimum, 5.0v nominal, 5.25v maximum
(normally supplied through USB connection)

Typical current consumption for CFA-635-YYE (yellow backlight):

Items Enabled			Current Consumption	
Logic	LCD backlight	All LEDs (4 red + 4 green)	V _{USB} =4.75v	V _{USB} =5.25v
X	-	-	35mA	42mA
X	X	-	108mA	153mA
X	-	X	147mA	175mA
X	X	X	218mA	282mA

Typical current consumption for CFA-635-TMF (white/blue backlight):

Items Enabled			Current Consumption	
Logic	LCD backlight	All LEDs (4 red + 4 green)	V _{USB} =4.75v	V _{USB} =5.25v
X	-	-	35mA	42mA
X	X	-	129mA	161mA
X	-	X	147mA	175mA
X	X	X	239mA	290mA

The CFA-635 circuitry is industry standard CMOS logic, which is susceptible to ESD damage. Please exercise industry standard static precautions as you would for any other bare PCB such as expansion cards or motherboards.

■ RELIABILITY

LCD portion (less the keypad & backlight):

50,000 to 100,000 hours.

Keypad:

1,000,000 keystrokes.

Yellow/Green and Red LED backlights:

50,000 to 100,000 hours.

White / Blue LED backlights:

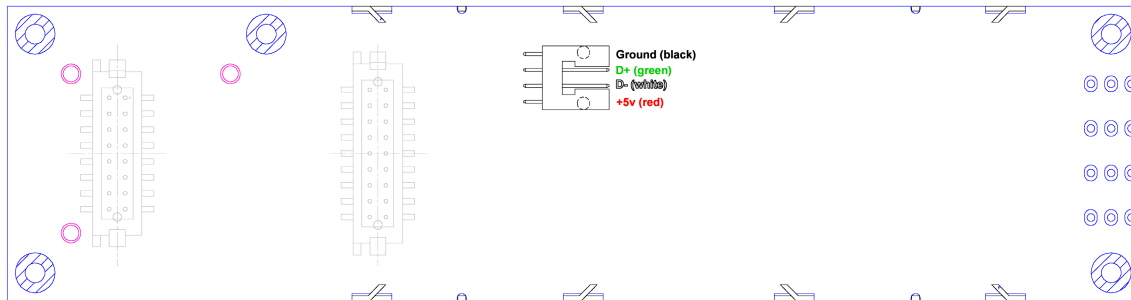
10,000 hours to 70% of original brightness

■ HOST CONNECTIONS

The CFA-635 is a USB peripheral, requiring only one connection to the host for both data communications and power supply.



In order to keep the CFA-635 as thin as possible, the CFA-635 uses a very low profile 2mm latching polarized connector for USB connection. Crystalfontz offers two cables that will make the connection between the CFA-635 and the host. The [WRUSBY03](#) has the mating 2mm connector on one end, and a standard “USB A” on the other end. The [WRUSBY11](#) has the mating 2mm connector on one end, and standard single pin connectors on the opposite end. These single pin connectors are suitable to plug directly onto the USB headers typically found on motherboards.



If you would like to make your own cable, the connector on the CFA-635 is:

- FCI/Berg 95000-004: SMT 2mm connector, 4-position, polarized

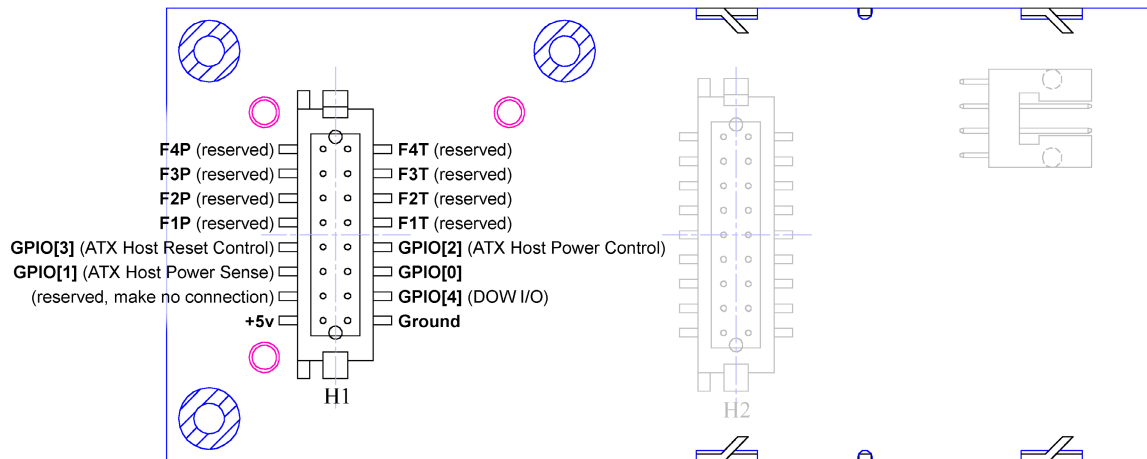
The mating housing and terminals for the cable are.

- FCI/Berg 90312-004: Housing, 2mm connector, 4-position, polarized
- FCI/Berg 77138-001: Terminal (4 pieces required)



■ GPIO/GPO CONNECTIONS

The CFA-635 has 5 GPIO available on header “H1”:



Please see the commands [“34: Set or Set and Configure GPIO pin”](#), and [“35: Read GPIO pin levels and configuration state”](#), below for details on how to control the GPIO.

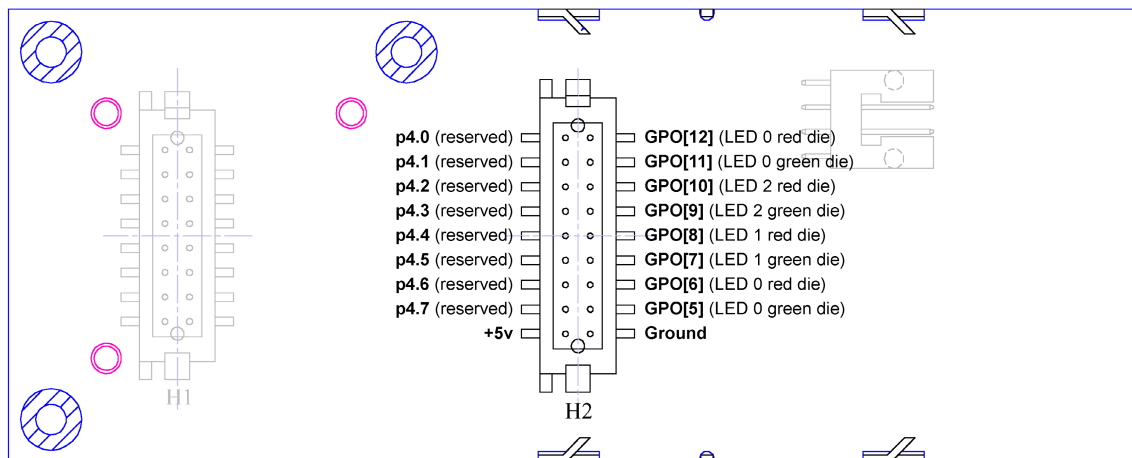
The following parts may be used to make a mating cable for H1:

16-position housing: Hirose DF11-16DS-2C / Digi-key H2025-ND

Terminal (tape & reel): Hirose DF11-2428SCF / Digi-Key H1504TR-ND

Terminal (loose): Hirose DF11-2428SC / Digi-Key H1504-ND

The CFA-635 has 8 GPO available on header “H2”. By factory default, these GPO drive the front panel LEDs. , by removing the LEDs these GPO could be used for other purposes:



Please see the commands [“34: Set or Set and Configure GPIO pin”](#), and [“35: Read GPIO pin levels and configuration state”](#), below for details on how to control the GPO.



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

18- position housing: Hirose DF11-18DS-2C / Digi-key H2026-ND
Terminal (tape & reel): Hirose DF11-2428SCF / Digi-Key H1504TR-ND
Terminal (loose): Hirose DF11-2428SC / Digi-Key H1504-ND

■ HOST COMMUNICATIONS

The CFA-635 communicates with its host using a USB interface. In most cases the host will have “virtual com port” drivers loaded. These drivers make the CFA-635 look like an additional serial port on the host system. The virtual serial port settings are 115,200 baud, 8 data bits, no parity, 1 stop bit by factory default. The speed can be set to 19,200 baud under software control (see command [33: Set Baud Rate](#)).

All communication between the CFA-635 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 CFA-635 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, having to “escape” certain “control characters”, or losing synchronization if a character is corrupted, missing or inserted).

PACKET STRUCTURE

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 CFA-635
      01 = normal response from CFA-635 to host
      10 = normal report from CFA-635 to host (not in
           direct response to a command from the host)
      11 = error response from CFA-635 to host (a packet
           with valid structure but illegal content
           was received by the CFA-635)

```

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

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

`CRC` is a standard 16-bit CRC of all the bytes in the packet except the CRC itself.



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

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

CrystalFontz supplies a demonstration and test program, [635 WinTest](#), along with its C source code. Included in the 635_WinTest source is the CRC algorithm, and an algorithm that will detect packets. The algorithm will automatically re-synchronize to the next valid packet in the event of any communications errors.

HANDSHAKING

The packet nature of the CFA-635 communications interface makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for an acknowledge packet from the CFA-635 before sending the next command packet. The CFA-635 will respond to all packets within 250mS, so the host software should stop waiting and retry the packet if the CFA-635 fails to respond within 250mS. The host software should report an error if a packet is not acknowledged after 1 or 2 retries. This situation would indicate a hardware problem.

Since the CFA-635 can be configured to send several types of report packets, along with regular acknowledge packets, the host should be able to buffer several incoming packets and must guarantee that it can process and remove packets from its input buffer faster than the packets can arrive given the baud rate and the reporting configuration of the CFA-635. For any modern PC using reasonably efficient software, this requirement will not pose a challenge.

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

REPORT CODES

The CFA-635 can be configured to report the following items. These reports will be sent automatically by the CFA-635 when the data becomes available. They are not sent in response to a particular packet received from the host.



0x80: Key Activity

If a key is pressed or released, the CFA-635 will send a Key Activity report packet to the host. Key events may be individually enabled or disabled by command “[23: Configure Key Reporting](#)”, below.

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

These codes are identical to the codes returned by the [CFA-633](#). Please note that the [CFA-631](#) will return codes 13 through 20. See the [CFA-631 data sheet](#) for more details.

0x81: Fan Speed Report (reserved, additional hardware required)

0x82: Temperature Sensor Report (reserved, additional hardware required)

COMMAND CODES

This is a list of valid commands for the CFA-635. Each valid command packet will be answered by either a response packet or an error packet. The data_length must be less than or equal to 18 in order for a packet to be valid. The low 6 bits of the type field of the response or error packet will be the same as the low 6 bits of the type field of the command packet being acknowledged.

0: Ping Command

The CFA-635 will return the Ping Command to the host.

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

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

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



1: Get Hardware & Firmware Version

The CFA-635 will return the Hardware and Firmware version information to the host.

```
type = 1
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 1
data_length = 16
data[] = "CFA635:hX.X,fY.Y"
```

hX.X is the hardware revision, "h1.0" for example
sY.Y is the firmware version, "v1.0" for example

2: Write User Flash Area

The CFA-635 reserves 16 bytes of non-volatile memory for arbitrary use by the host. This memory could be used to store a serial number, IP address, gateway address, netmask, or any other data required. All 16 bytes must be supplied.

```
type = 2
valid data_length is 16
data[] = 16 bytes of arbitrary user data to be stored in
         the CFA-635's non-volatile memory
```

The return packet will be:

```
type = 0x40 | 2
data_length = 0
```

3: Read User Flash Area

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

```
type = 3
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 3
data_length = 16
data[] = 16 bytes user data recalled from the CFA-635's
         non-volatile memory
```

4: Store Current State As Boot State

The CFA-635 loads its power-up configuration from non-volatile memory when power is applied. The CFA-635 is configured at the factory to display a "welcome" screen when



power is applied. This command can be used to customize that welcome screen, as well as many other settings.

The following items are stored by this command:

- Backlight setting (command 14)
- Contrast setting (command 13)
- Cursor position (command 11)
- Cursor style (command 12)
- The characters shown on the LCD (commands 6, 7, 8 & 31)
- The special character font definitions (command 9)
- The fan power settings (command 17) (additional hardware required)
- The fan glitch delay settings (command 26) (additional hardware required)
- The key press and release masks (command 23)
- The ATX function enable and pulse length settings (command 28) (additional hardware required)
- The baud rate (command 33)
- The GPIO settings (command 34)
- The front panel LED/GPO settings (command 34)

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

```
type = 4
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 4
data_length = 0
```

5: Reboot CFA-635, Reset Host, or Power Off Host

This command instructs the CFA-635 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.

Additional hardware is required to connect the GPIO pins to the 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 Set or Set and Configure GPIO pin (additional hardware required).

Rebooting the CFA-635 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 CFA-635, send the following packet:



```
type = 5
valid data_length is 3
data[0] = 8
data[1] = 18
data[2] = 99
```

To reset the host (assuming the host's reset line is connected to GPIO[3] as described in "[ATX POWER SUPPLY CONTROL CONNECTIONS](#)" above), send the following packet (additional hardware required):

```
type = 5
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 "[ATX POWER SUPPLY CONTROL CONNECTIONS](#)" above), send the following packet (additional hardware required):

```
type = 5
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 | 5
data_length = 0
```

6: Clear LCD Screen

Sets the contents of the LCD screen DDRAM to ' ' = 0x20 = 32 and moves the cursor to the left-most column of the top line. The LCD contents are one of the items stored by the "[4: Store Current State As Boot State](#)" command.

```
type = 6
valid data_length is 0
```

The return packet will be:

```
type = 0x40 | 6
data_length = 0
```

7: (deprecated, see command 31)

8: (deprecated, see command 31)

9: Set LCD Special Character Data



Sets the font definition for one of the special characters (CGRAM). The LCD CGRAM contents are one of the items stored by the "[4: Store Current State As Boot State](#)" command.

```
type = 9
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 | 9
data_length = 0
```

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

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

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

The return packet will be:

```
type = 0x40 | 10
data_length = 9
```

data[0] of the return packet will be the address code

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

11: Set LCD Cursor Position

This command allows the cursor to be placed at the desired location on the CFA-635's LCD screen. The LCD cursor position is one of the items stored by the "[4: Store Current State As](#)



[Boot State](#)” command. If you want the cursor to be visible, you may also need to send a [“Set LCD Cursor Style”](#) command.

```
type = 11
valid data_length is 2
data[0] = column (0-19 valid)
data[1] = row (0-3 valid)
```

The return packet will be:

```
type = 0x40 | 11
data_length = 0
```

12: Set LCD Cursor Style

This command allows you to determine the style of the hardware-generated cursor shown on the LCD. The LCD cursor style is one of the items stored by the [“4: Store Current State As Boot State”](#) command.

```
type = 12
valid data_length is 1
data[0] = cursor style (0-3 valid)
    0 = no cursor
    1 = blinking block cursor
    2 = underscore cursor
    3 = blinking block plus underscore
```

The return packet will be:

```
type = 0x40 | 12
data_length = 0
```

13: Set LCD Contrast

This command sets the contrast or vertical viewing angle of the display. The LCD contrast is one of the items stored by the [“4: Store Current State As Boot State”](#) command.

```
type = 13
valid data_length is 1
data[0] = contrast setting (0-255 valid)
    0-65 = very light
    66 = light
    95 = about right
    125 = dark
    126-255 = very dark
```

The return packet will be:

```
type = 0x40 | 13
data_length = 0
```




14: Set LCD & Keypad Backlight

This command sets the brightness of the LCD and keypad backlights. The backlight brightness is one of the items stored by the [“4: Store Current State As Boot State”](#) command.

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

The return packet will be:

```
type = 0x40 | 14
data_length = 0
```

15: (deprecated)

16: Set Up Fan Reporting (reserved, additional hardware required)

17: Set Fan Power (reserved, additional hardware required)

18: Read DOW Device Information (reserved, additional hardware required)

19: Set Up Temperature Reporting (reserved, additional hardware required)

20: Arbitrary DOW Transaction (reserved, additional hardware required)

21: (deprecated)

22: Send command directly to the LCD controller

The LCD controller on the CFA-635 is HD44780 compatible. Generally you would not need low-level access the LCD controller, but there are some arcane functions of the HD44780 that are not exposed by the CFA-635's command set. This command allows you to access the CFA-635's LCD controller directly. Please note that it is quite possible to corrupt the CFA-635's display using this command.

```
type = 22
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 | 22
data_length = 0
```

23: Configure Key Reporting



By default, the CFA-635 reports any key event to the host. This command allows the key events to be enabled or disabled on an individual basis. The key events set to report are one of the items stored by the [“4: Store Current State As Boot State”](#) command.

```
#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 = 23
data_length = 2
data[0]: press mask
data[1]: release mask
```

The return packet will be:

```
type = 0x40 | 23
data_length = 0
```

24: Read Keypad, Polled Mode

In some situations, it may be convenient for the host to poll the CFA-635 for key activity. This command allows the host to detect which keys are currently pressed, which keys have been pressed since the last poll, and which keys have been released since the last poll.

This command is independent of the key reporting masks set by command 23--all keys will always be visible to this command. Typically both masks of command [23: Configure Key Reporting](#) would be set to 0 if the host is reading the keypad in polled mode.

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

```
type = 24
data_length = 0
```

The return packet will be:

```
type = 0x40 | 24
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: Set Fan Power Fail-Safe** (reserved, additional hardware required)
- 26: Set Fan Tachometer Glitch Delay** (reserved, additional hardware required)
- 27: Query Fan Power & Fail-Safe Mask** (reserved, additional hardware required)
- 28: Set ATX Power Switch Functionality** (reserved, additional hardware required)
- 29: Enable/Disable and Reset the Watchdog** (reserved, additional hardware required)

30: Read Reporting & Status

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

```
type = 30
data_length = 0
```

The return packet will be:

```
type = 0x40 | 30
data_length = 15
data[0] = (reserved, additional hardware required)
data[1] = (reserved, additional hardware required)
data[2] = (reserved, additional hardware required)
data[3] = (reserved, additional hardware required)
data[4] = (reserved, additional hardware required)
data[5] = key presses enabled (as set by command 23)
data[6] = key releases enabled (as set by command 23)
data[7] = (reserved, additional hardware required)
data[8] = (reserved, additional hardware required)
data[9] = (reserved, additional hardware required)
data[10] = (reserved, additional hardware required)
data[11] = (reserved, additional hardware required)
data[12] = (reserved, additional hardware required)
data[13] = contrast setting (as set by command 13)
data[14] = backlight setting (as set by command 14)
```

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

31: Send Data to LCD

This command allows any length of data to be placed at any position on the LCD. The LCD contents are one of the items stored by the [“4: Store Current State As Boot State”](#) command.

```
type = 31
data_length = 3 to 22
data[0]: col = x = 0 to 19
data[1]: row = y = 0 to 3
data[2-21]: text to place on the LCD, variable from 1 to 16 characters
```

The return packet will be:



```
type = 0x40 | 31  
data_length = 0
```

32: reserved (CFA-631 Key Legends)

33: Set Baud Rate

This command will change the CFA-635's baud rate. The CFA-635 will send the acknowledge packet for this command and then change its baud rate to the new value. The host should send the baud rate command, wait for a positive acknowledge and then switch to the new baud rate itself. The baud rate must be saved by the "[4: Store Current State As Boot State](#)" command if you want the CFA-635 to power up at the new baud rate.

The factory default baud rate is 115,200.

```
type = 27  
data_length = 1  
data[1]: 0 = 19,200 baud  
         1 = 115,200 baud
```

The return packet will be:

```
type = 0x40 | 27  
data_length = 0
```

34: Set or Set and Configure GPIO pin

The CFA-635 has thirteen pins that can be used for user-definable general-purpose input / output (GPIO) or general-purpose output (GPO). The first five of these pins (GPIO0 – GPIO4) require additional hardware for full capability and should be considered reserved.

The remaining eight pins (GPO5 – GPO12) control the bi-color LEDs that are installed on the CFA-635 at the left side of the display.

The architecture of the CFA-635 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 100Hz PWM signal.

In output mode using the PWM (and a suitable current limiting resistor—which is standard on the CFA-635 for GPO5 – GPO12), LEDs may be turned on or off and even dimmed under host software control. With suitable external circuitry, the GPIOs or GPOs could also be used to drive external logic or power transistors.

The CFA-635 continuously polls the GPIOs as inputs at 32Hz. The present level can be queried by the host software at a lower rate. The CFA-635 also keeps track of whether there were rising or falling edges since the last host query (subject to the resolution of the 32Hz sampling), so the host is not forced to poll quickly in order to detect short events. The algorithm used by the CFA-635 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 20mA per GPIO.

The default drive mode for GPO5 – GPO12 are correct for driving the factory-installed bi-color LEDs. We recommend that you use the “change value only” form of the command, since the LEDs may not function correctly if the GPO’s drive mode is changed.

The GPIO/GPO configuration is one of the items stored by the [“4: Store Current State As Boot State”](#) command.

```
type: 34
data_length:
    2 bytes to change value only
    3 bytes to change value and configure function and drive mode
```

```
data[0]: index of GPIO/GPO to modify
0 = GPIO[0] = (reserved, additional hardware required)
1 = GPIO[1] = (reserved, additional hardware required)
2 = GPIO[2] = (reserved, additional hardware required)
3 = GPIO[3] = (reserved, additional hardware required)
4 = GPIO[4] = (reserved, additional hardware required)
5 = GPO[5] = LED 3 (bottom) green die
6 = GPO[6] = LED 3 (bottom) red die
7 = GPO[7] = LED 2 green die
8 = GPO[8] = LED 2 red die
9 = GPO[9] = LED 1 green die
10 = GPO[10] = LED 1 red die
11 = GPO[11] = LED 0 (top) green die
12 = GPO[12] = LED 0 (top) red die
```

13-255: reserved

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

```
data[1] = Pin output state (actual behavior depends on drive mode):
0 = Output set to low
1-99: Output duty cycle percentage (100Hz 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 | 34
data_length = 0

```

35: Read GPIO pin levels and configuration state

Please see "[34: Set or Set and Configure GPIO pin](#)", above for details on the GPIO architecture.

```

type: 35
data_length: 1

```

```

data[0]: index of GPIO to query
0 = GPIO[0] = (reserved, additional hardware required)
1 = GPIO[1] = (reserved, additional hardware required)
2 = GPIO[2] = (reserved, additional hardware required)
3 = GPIO[3] = (reserved, additional hardware required)
4 = GPIO[4] = (reserved, additional hardware required)
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 to read

data[1] = Pin state & changes since last poll

```
---- -RFS Enable Reporting of this Fan's Tach Input
|||| | | | | -- S = state at the last reading
|||| | | | | --- F = at least one falling edge has
|||| | | | |         been detected since the last poll
|||| | | | | ---- R = at least one rising edge has
|||| | | | |         been detected since the last poll
|||| | | | | ----- reserved
```

(This reading is the actual pin state, which may or may not agree with the pin setting, depending on drive mode and the load presented by external circuitry. The pins are polled at approximately 32Hz asynchronously with respect to this command. Transients that happen between polls will not be detected.)

data[2] = Requested Pin level/PWM level

0-100: Output duty cycle percentage

(This value is the requested PWM duty cycle. The actual pin may or may not be toggling in agreement with this value, depending on the drive mode and the load presented by external circuitry)

data[3] = Pin function select and drive mode

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



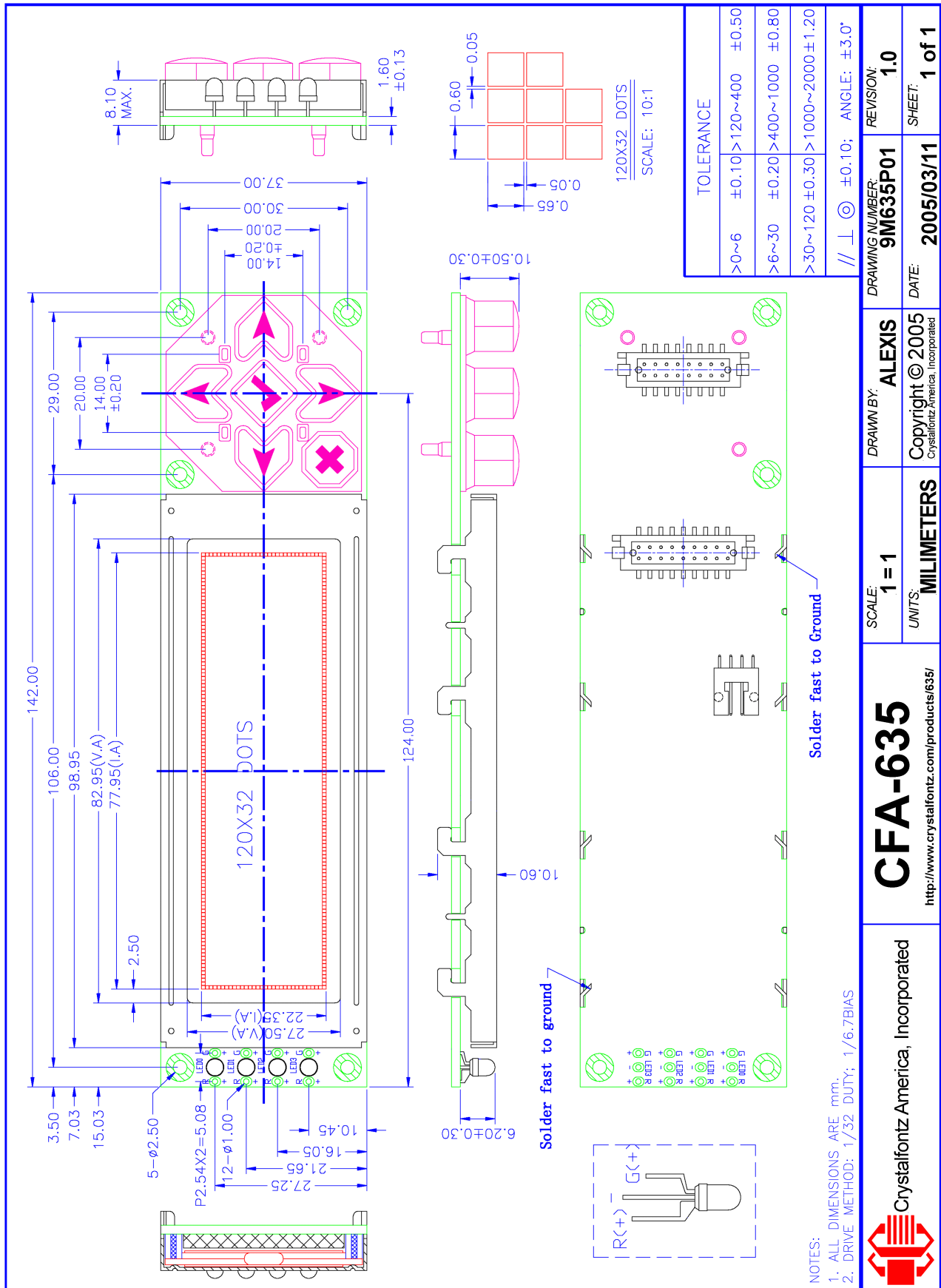
■ CHARACTER GENERATOR ROM (CGROM)

Character Generator ROM (CGROM) for CrystalFontz CFA-635

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

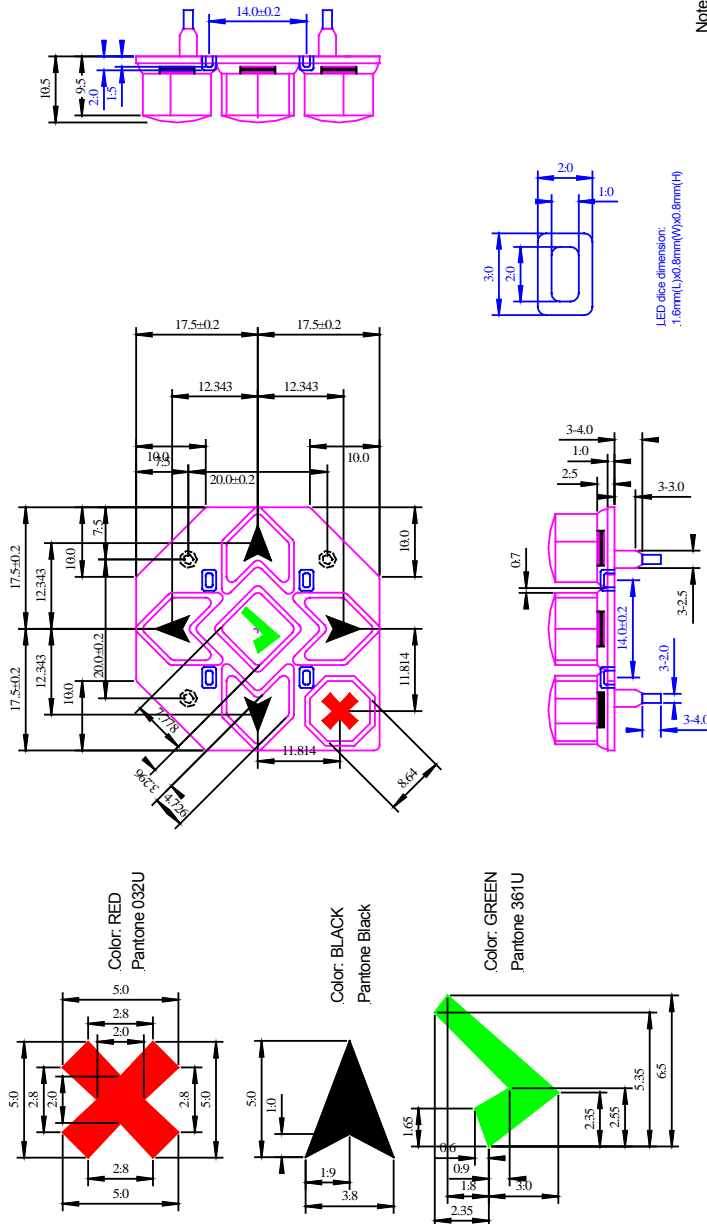


MODULE OUTLINE DRAWING






KEYPAD OUTLINE DRAWING

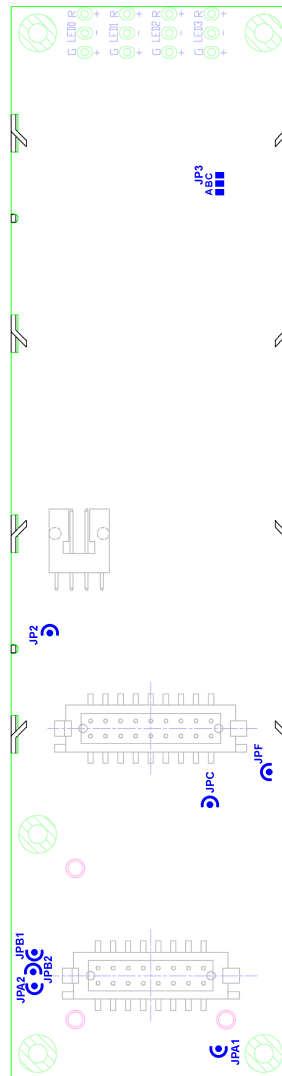


- Note:
1. Material: silicone rubber 50#
 2. Carbon coated
 3. Life time: 100 million times
 4. Resistance: Less than 100 OHMs
 5. Receive weightiness: 80~120gm
 6. Fireproofing standard: Shining 50 second
 7. Silicone rubber color: translucent white
 8. All corners are fillet 0.75mm radius


	Crystalfontz America, Incorporated http://www.crystalfontz.com/products/633	SCALE: 1 = 1 UNITS: MILLIMETERS	DRAWN BY: ALEXIS Copyright © 2002 Crystalfontz America, Incorporated	DRAWING NUMBER: 9M633P01	REVISION: 1.0
				DATE: 01/07/2001	SHEET: 2 of 2

■ JUMPER REFERENCE

CFA-635 Hardware v1.0 Jumper Locations and Functions



JP2	open	Power is not supplied through USB connector
	closed	Power is supplied through USB connector
JP6	open	Frame ground is isolated from logic/USB ground
	closed	Frame ground is connected to logic/USB ground
JP3	IDA2	IDA1 IDB2 IDC: Factory build options do not change

 CrystalFontz America, Incorporated	<h1 style="text-align: center;">CFA-635</h1> <p style="text-align: center;">http://www.crystalfontz.com/products/635/</p>	SCALE: 1 = 1	DRAWN BY: ALEXIS	DRAWING NUMBER: 9M635P02	REVISION: 1.0
		UNITS: MILIMETERS	Copyright © 2005 CrystalFontz America, Incorporated	DATE: 2005/03/11	SHEET: 1 of 1