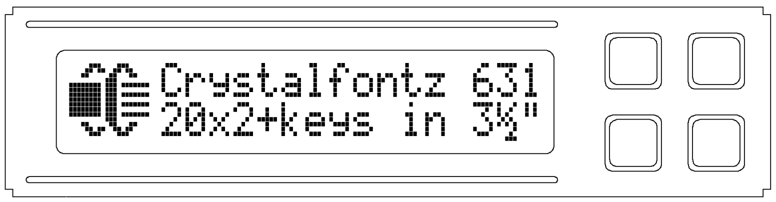
Crystalfontz America, Incorporated

USB LCD MODULE SPECIFICATIONS



Crystalfontz Model Number	**CFA-631**
Firmware Version	**V1.0**
Product Page	**http://www.crystalfontz.com/products/631**

Customer Name	
Customer Part Number	

Crystalfontz America, Inc.
15611 East Washington Road
Valleyford, WA 99036-9747

Phone: (509) 291-3514
Fax: (509) 291-3345

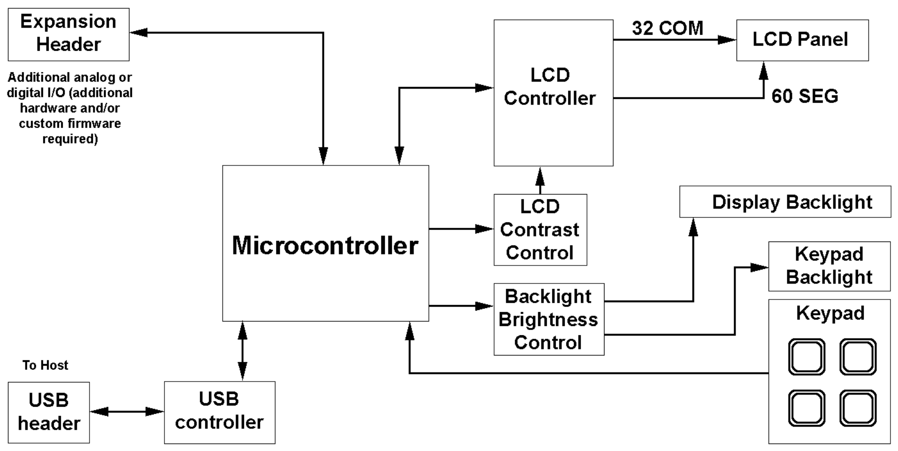
e-mail: sales@crystalfontz.com
http://www.crystalfontz.com

Table of Contents

■ FEATURES

- Compact size: “3.5½ inch floppy drive” form factor
- USB interface (115200 baud throughput)
- LED backlit negative mode STN 20x2 LCD
- Integrated LED backlit 4-button translucent silicon keypad
- “key legends” allows assignment of keys to be shown easily on the LCD
- 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 re-programmable microcontroller (factory operation)
- Non-volatile memory capability (“EEPROM”):
  Customize the "power-on" display settings
  16-byte "scratch" register for storing IP, netmask, system serial number . . .
- Expandable firmware and configurable hardware that can be customized to add specific features for your system needs:
  Hardware or software watchdog can provide system reset
  Other analog or digital I/O
  Provide "dongle" functionality for software copy protection
  Autonomous hardware monitoring
  Fan control and monitoring, temperature monitoring
  (Tooling fee may apply for custom firmware and hardware)

■ SYSTEM BLOCK DIAGRAM



■ MECHANICAL CHARACTERISTICS

Item	Description	Unit
Overall Size	101.6(W) x 25.4(H)	mm
Viewing Area	66.0(W) x 13.8(H)	mm
Active Area	63.55(W) x 10.35(H)	mm
Character Size	3.13(W) x 5.15(H)	mm
Dot Size	0.48(W) x 0.60(H)	mm
Dot Pitch	0.53(W) x 0.65(H)	mm
Depth:		
Without Keypad or Overlay	90.0	mm
Without Keypad, with Overlay	90.6	mm
With Keypad	93.1	mm
Keystroke Travel (approximate)	2	mm

■ GENERAL SPECIFICATIONS

Operating Temperature: 0°C minimum, 50°C maximum
Storage Temperature: -10°C minimum, 60°C maximum
LCD Glass Type: STN negative blue mode
Viewing Direction: 12 O'clock
Polarizer Type: Transmissive
Driving Method: 1/32 Duty, 1/.67 Bias
Backlight: LED, Red (-RMF) or White (-TMF)
Weight: 80 grams typical
Lifetime:
  LCD and logic: 50,000 to 100,000 hours typical
  Red LED backlight: 50,000 to 100,000 hours typical
  White LED backlight: 10,000 hours to 70% brightness

■ ELECTRICAL SPECIFICATIONS

Required voltage:
+5v (supplied by USB): 4.5v minimum, 5.0v nominal, 5.5v maximum

Current consumption:
+5v (logic + USB controller): 30mA typical
+5v (white backlight "-TMC"): 60mA typical / 90mA total including logic
+5v (red backlight "-RMC"): 150mA typical / 180mA total including logic

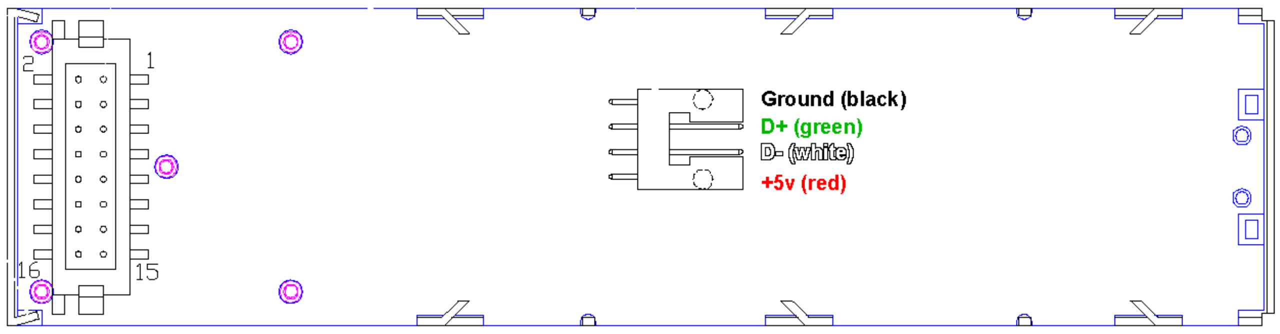
ESD (Electro-Static Discharge) Specifications:

D+ and D- pins of USB connector only:
Electrostatic Discharge Voltage (I < 1uA): +/- 2000V

The remainder of the 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.

■ USB CONNECTIONS

The USB connector on the 631 mates with the Crystalfontz WRUSBY03 and WRUSBY11 cables. The connector is 2mm pitch, polarized and latching.



If you would like to make your own cable, the connector on the 631 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)

■ HOST COMMUNICATIONS

The CFA-631 communicates with its host using the USB interface. The easiest and most common way for the host software to access the USB is through the Crystalfontz supplied virtual COM port (VCP) drivers. The VCP drivers can be downloaded from the Crystalfontz web site. Using these drivers makes it appear to the host software as if there is an additional serial port (the VCP) on the host system when the CFA-631 is connected. This VCP should be opened at 115200 baud, 8 data bits, no parity, 1 stop bit.

All communication between the CFA-631 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-631 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 sync 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-631
             01 = normal response from CFA-631 to host
             10 = normal report from CFA-631 to host (not in
                  direct response to a command from the host)
             11 = error response from CFA-631 to host (a packet
                  with valid structure but illegal content
                  was received by the CFA-631)
```

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

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

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

Crystalfontz will supply a demonstration and test program, 631_WinTest, along with its C source code. Included in the 631_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.

Please follow the algorithm in the sample code closely in order to realize the benefits of using the packet communications.

"HANDSHAKING"

The packet nature of the CFA-631 makes it unnecessary to implement traditional hardware or software handshaking.

The host should wait for a corresponding acknowledge packet from the 631 before sending the next command packet. The CFA-631 will respond to all packets within 250mS, so the host software should stop waiting and retry the packet if the CFA-631 fails to respond within 250mS. The host software should report an error if a packet is not acknowledged after several retries. This situation would indicate a hardware problem—a disconnected cable for instance.

Since the CFA-631 can be configured to send several types of report packets, along with regular acknowledge packets, the host should be able to buffer several incoming packets and must guarantee that it can process and remove packets from its input buffer faster than the packets can arrive given the 115200 equivalent baud rate of the VCP and the reporting configuration of the CFA-631. 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-631 can be configured to report the following items. These reports will be sent automatically by the CFA-631 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-631 will send a Key Activity report packet to the host. Key events may be individually enabled or disabled by "23: Configure Key Reporting", below.

```
type = 0x80
data_length = 1
data[0] is the type of keyboard activity:
     KEY_UL_PRESS   13
     KEY_UR_PRESS   14
     KEY_LL_PRESS   15
     KEY_LR_PRESS   16
     KEY_UL_RELEASE 17
     KEY_UR_RELEASE 18
     KEY_LL_RELEASE 19
```

```
KEY_LR_RELEASE 20
```

0x81: Fan Speed Report (reserved, additional hardware required)
0x82: Temperature Sensor Report (reserved, additional hardware required)

COMMAND CODES

This is a list of valid commands for the CFA-631. Each command packet will be answered by either a response packet or an error packet. 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-631 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-631 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[] = "CFA631:hX.X,vY.Y"

X.X is the hardware revision, "1.0" for example
vY.Y is the firmware version, "v1.0" for example
```

2: Write User Flash Area

The CFA-631 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-631'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-631's
         non-volatile memory
```

4: Store Current State As Boot State

The CFA-631 loads its power-up configuration from non-volatile memory when power is applied. The CFA-631 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
Contrast setting
Cursor position
Cursor style
The characters shown on the LCD
The special character font definitions
The key legends

You cannot store the “reporting” configuration. The host should enable the appropriate reporting once it is ready to receive the data.

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

The return packet will be:

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

5: Reboot CFA-631

This command instructs the CFA-631 to simulate a power-on restart of itself.

Rebooting the CFA-631 may be useful when testing the boot configuration. It may also be useful to re-enumerate the devices on the 1-Wire bus (additional hardware required). To reboot the CFA-631, send the following packet:

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

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.

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

The return packet will be:

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

7: Set LCD Contents, Line 1 (CFA-633 compatibility)

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

This command is provided to allow legacy software that displays data on the CFA-633 to work unchanged on the CFA-631. For new applications, please use the more flexible **“31: Send Data to LCD”** command.

```
type = 7
valid data_length is 16
data[] = top line’s display content (must supply 16 bytes)
```

The return packet will be:

```
type = 0x40 | 7
data_length = 0
```

8: Set LCD Contents, Line 2 (CFA-633 compatibility)

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

This command is provided to allow legacy software that displays data on the CFA-633 to work unchanged on the CFA-631. For new applications, please use the more flexible **"31: Send Data to LCD"** command.

```
type = 8
valid data_length is 16
data[] = top line's display content (must supply 16 bytes)
```

The return packet will be:

```
type = 0x40 | 8
data_length = 0
```

9: Set LCD Special Character Data

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

```
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.

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

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 (\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 | 10
data_length = 8
```

`data[0-7]` 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-631's LCD screen. 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 (0-1 valid)
```

The return packet will be:

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

12: Set LCD Cursor Style

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

```
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
      4 = inverting, blinking block
```

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.

```
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.

```
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: Set Up Live Fan or Temperature Display (reserved, additional hardware required)

22: Send command directly to the LCD controller

The LCD controller on the CFA-631 is S6A0073 compatible. Generally you would not need low-level access the LCD controller, but there are some arcane functions of the S6A0073 that are not exposed by the CFA-631's command set. This command allows you to access the CFA-631's LCD controller directly. Please note that it is quite possible to corrupt the CFA-631's display using this command.

```
type = 22
data_length = 2
data[0]: location code
```

```
           0 = "Data" register
           1 = "Control" register, RE=0
           2 = "Control" register, RE=1
data[1]: data to write to the selected register
```

The return packet will be:

```
type = 0x40 | 22
```
data_length = 0

23: Configure Key Reporting

By default, the CFA-631 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_UL     0x01 //(upper-left)
#define KP_UR     0x02 //(upper-right)
#define KP_LL     0x04 //(lower-left)
#define KP_LR     0x08 //(lower-right)

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-631 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 would be set to 0 if the host is reading the keypad in polled mode.

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

type = 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 Filter (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 = 13
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 (as set by command 23)
data[ 6] = key releases (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)
```

31: Send Data to LCD

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

```
type = 31
data_length = 3 to 22
data[0]: col = x = 0 to 19
data[1]: row = y = 0 to 1
data[2-21]: text to place on the LCD, variable from 1 to 20 characters
```

The return packet will be:

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

32: Key Legends

The CFA-631 offers firmware support for “soft keys”. There are 8 pre-defined icons that correspond to common key functions:

```
#define KEY_LEGEND_BLANK    0 ■
#define KEY_LEGEND_CANCEL   1 ✖
#define KEY_LEGEND_CHECK    2 ✓
#define KEY_LEGEND_UP       3 ▲
#define KEY_LEGEND_DOWN     4 ▼
#define KEY_LEGEND_RIGHT    5 ►
#define KEY_LEGEND_LEFT     6 ◄
#define KEY_LEGEND_PLUS     7 +
#define KEY_LEGEND_MINUS    8 −
#define KEY_LEGEND_NONE     9 //no key or symbol
```

The host simply enables key legends—specifying the icon to display corresponding to each key—and then the CFA-631 firmware handles drawing the legends. Each soft-key legend will “invert” when the corresponding hard key is pressed, providing instant user feedback that the key has been actuated.

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

The key legends will act as a second “layer” of the display over the 6 rightmost characters. Text written to the key legends area will be overwritten instantly by the key legends.

```
type = 32
data_length = 1 (to disable) or 5 (to enable and specify)
data[0]: enable = 1, disable = 0
data[1] = code for icon to be displayed for upper-left key
data[2] = code for icon to be displayed for upper-right key
data[3] = code for icon to be displayed for lower-left key
data[4] = code for icon to be displayed for lower-right key
```

The return packet will be:

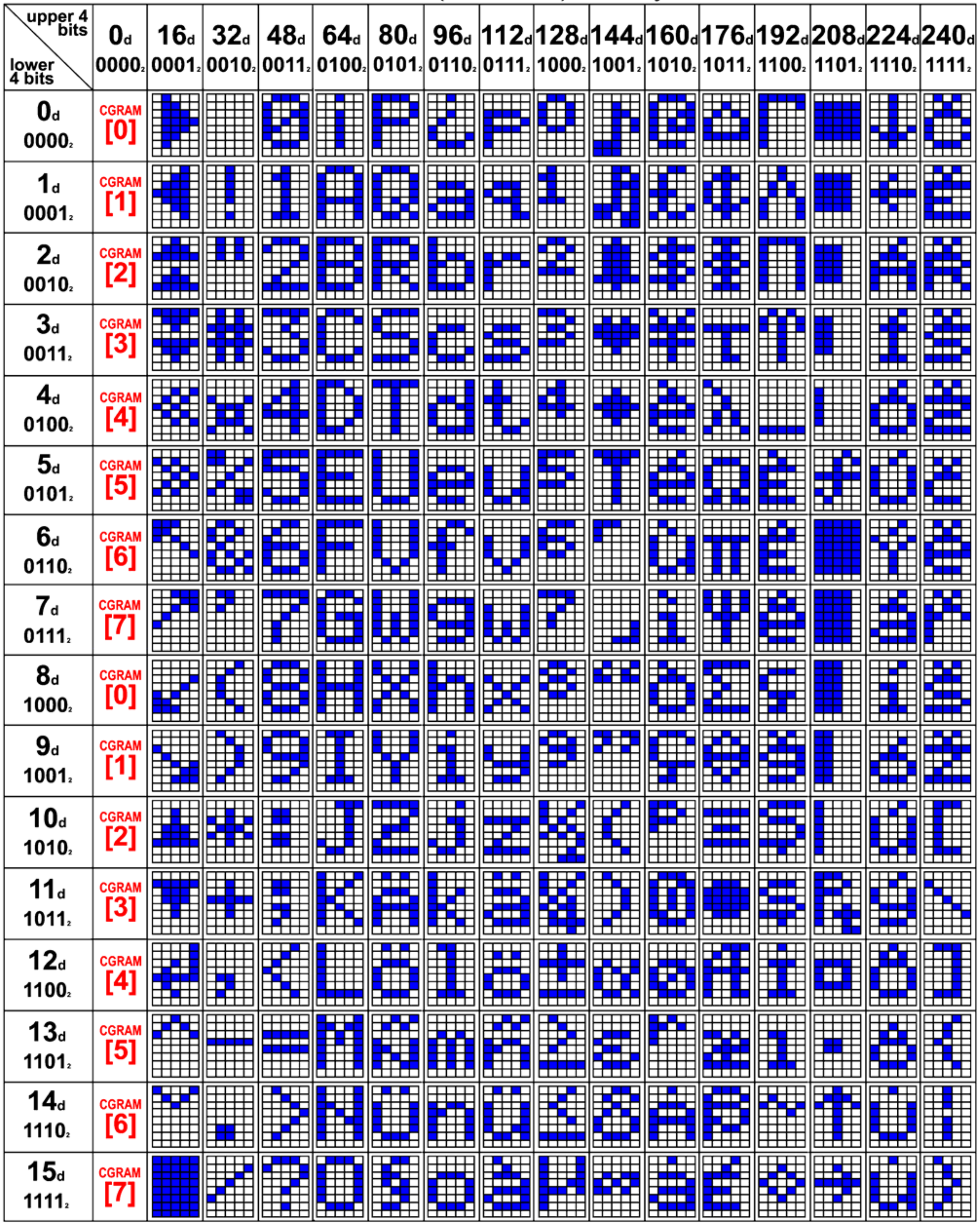
```
type = 0x40 | 32
data_length = 0
```

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

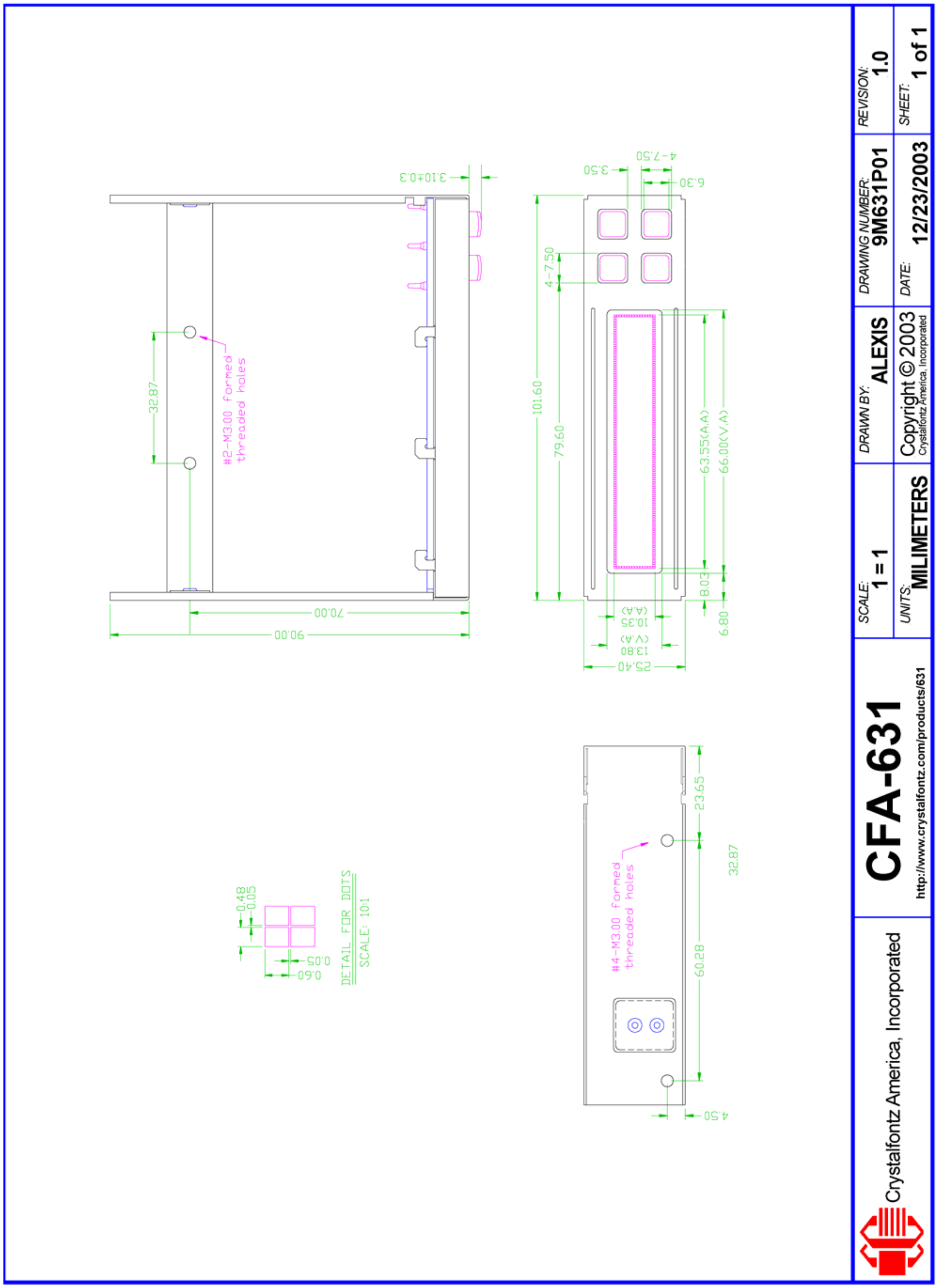
By using special character definitions and key reports, the functionality of the key legends could be emulated in host software, which would allow unlimited icon definitions.

■ CHARACTER GENERATOR ROM (CGROM)

Character Generator ROM (CGROM) for Crystalfontz CFA-631


upper 4 bits
lower 4 bits
0d 0000₂
16d 0001₂
32d 0010₂
48d 0011₂
64d 0100₂
80d 0101₂
96d 0110₂
112d 0111₂
128d 1000₂
144d 1001₂
160d 1010₂
176d 1011₂
192d 1100₂
208d 1101₂
224d 1110₂
240d 1111₂
0d 0000₂
1d 0001₂
2d 0010₂
3d 0011₂
4d 0100₂
5d 0101₂
6d 0110₂
7d 0111₂
8d 1000₂
9d 1001₂
10d 1010₂
11d 1011₂
12d 1100₂
13d 1101₂
14d 1110₂
15d 1111₂
CGRAM [0]
CGRAM [1]
CGRAM [2]
CGRAM [3]
CGRAM [4]
CGRAM [5]
CGRAM [6]
CGRAM [7]
CGRAM [0]
CGRAM [1]
CGRAM [2]
CGRAM [3]
CGRAM [4]
CGRAM [5]
CGRAM [6]
CGRAM [7]

■ MODULE OUTLINE DRAWING


3.10±0.3
#2-M3.00 formed threaded holes
32.87
70.00
90.00
3.50
4-7.50
6.30
4-7.50
101.60
79.60
63.55(A.A)
66.00(V.A)
8.03
6.80
10.35 (A.A)
13.80 (V.A)
25.40
0.48
0.05
0.05
0.60
DETAIL FOR DOTS
SCALE: 10:1
#4-M3.00 formed threaded holes
23.65
32.87
60.28
4.50
REVISION: 1.0
SHEET: 1 of 1
DRAWING NUMBER: 9M631P01
DATE: 12/23/2003
DRAWN BY: ALEXIS
Copyright © 2003 Crystalfontz America, Incorporated
SCALE: 1 = 1
UNITS: MILIMETERS
CFA-631
http://www.crystalfontz.com/products/631
Crystalfontz America, Incorporated