

# LCD160CRv1.0 Ref. Manual

|                         |
|-------------------------|
| <b>REVISION HISTORY</b> |
|-------------------------|

| NUMBER | DATE        | DESCRIPTION | NAME |
|--------|-------------|-------------|------|
|        | 2017-Jan-20 |             | tsp  |

# Contents

|           |                                 |           |
|-----------|---------------------------------|-----------|
| <b>1</b>  | <b>Ref. Manual</b>              | <b>1</b>  |
| <b>2</b>  | <b>Physical Properties</b>      | <b>2</b>  |
| 2.1       | Key Features . . . . .          | 2         |
| 2.2       | Operation Modes . . . . .       | 2         |
| 2.3       | I2C Status . . . . .            | 2         |
| <b>3</b>  | <b>Commands</b>                 | <b>3</b>  |
| 3.1       | Basic commands . . . . .        | 3         |
| 3.2       | Setup commands . . . . .        | 4         |
| 3.3       | Advanced commands . . . . .     | 4         |
| 3.4       | Obsolete commands . . . . .     | 6         |
| <b>4</b>  | <b>Polypoints and Polyline</b>  | <b>7</b>  |
| <b>5</b>  | <b>Linegraph and Dotgraph</b>   | <b>7</b>  |
| <b>6</b>  | <b>Colors</b>                   | <b>7</b>  |
| <b>7</b>  | <b>Fonts</b>                    | <b>7</b>  |
| <b>8</b>  | <b>Windows</b>                  | <b>8</b>  |
| <b>9</b>  | <b>Sprites</b>                  | <b>9</b>  |
| 9.1       | Bitmaps . . . . .               | 9         |
| 9.2       | Sprite Format . . . . .         | 9         |
| 9.3       | JPEG . . . . .                  | 9         |
| <b>10</b> | <b>Touch Panel</b>              | <b>10</b> |
| <b>11</b> | <b>ANSI Mode</b>                | <b>11</b> |
| <b>12</b> | <b>NVRAM Parameters</b>         | <b>11</b> |
| <b>13</b> | <b>Firmware Upgrade</b>         | <b>12</b> |
| <b>14</b> | <b>Technical Specifications</b> | <b>13</b> |
| 14.1      | Power Supply . . . . .          | 13        |
| 14.2      | Ports . . . . .                 | 13        |
| 14.3      | Connectors . . . . .            | 14        |
| 14.4      | Communication . . . . .         | 14        |
| <b>15</b> | <b>Drawings</b>                 | <b>15</b> |

---

# 1 Ref. Manual

|                |                     |
|----------------|---------------------|
| <b>Product</b> | <b>LCD160CRv1.0</b> |
| Rev            | 6.9                 |
| Date           | 2017-Dec-20         |
| Author         | tsp                 |

## 2 Physical Properties

LCD160CR is a powerful color LCD with integrated touch controller. It provides all the necessary hard- and software allow both low- and high-level operation of the display and the touch screen.

### 2.1 Key Features

- stand alone module with integrated power control
- self configured operation, no need to initialize hardware components
- factory calibrated touch screen with proper coordinate translation
- multiple communication channels
- adequate ANSI terminal emulation for REPL output
- integrated character fonts
- one character set with UTF-8 capabilities
- compact and simple character based commands
- high-speed host to screen region transfer

### 2.2 Operation Modes

The LCD160CR accepts commands and data from all available interfaces. All communication channels share the same input and output queues. It is up to the user to avoid collisions between concurrent data streams.

The raw SPI mode does not use an input queue. If raw SPI mode is activated, it will be active until a reset or power cycle occurs. Activating raw SPI mode will prepare a window in the frame buffer. All raw SPI data will be copied to this window. If an overflow occurs, the data will wrap to the head of the window.

Sending commands and data via regular communication channels will not terminate raw SPI mode. However, it is highly recommended to restart raw SPI mode whenever data was entered via a regular communication channel <sup>1</sup>.

It is mandatory to enter raw SPI mode only if the LCD160CR input queue is empty. Also any kind of auto-scrolling has to be terminated prior to entering raw SPI mode.

### 2.3 I2C Status

The I2C communication port provides two addresses. The primary address is always an even value. The status address is always primary address + 1. The only valid operation for the status address is to read a single byte. This byte gives the status of the input queue.

Table 1: Status Encoding

| Value  | Description                              |
|--------|--|
| 0      | Input queue full                         |
| 1..253 | <Value> bytes free space available       |
| 254    | more than 254 bytes free space available |
| 255    | input queue empty & idle                 |

<sup>1</sup> Commands that do not alter frame buffer content may be used intermixed with raw SPI transfers.

### 3 Commands

All bytes entered through each of the communication paths (I2C, UART and cooked SPI, if supported) are treated as ASCII characters and handled in a dumb terminal style according to the font and topbot parameters. Input buffer is shared among all input channels. The host is responsible to avoid buffer overflow and protocol sequence errors in case more than one channel is used. The available space of the input buffer can be polled if bidirectional communication is possible (cooked SPI is output only). The I2C slave interface has an extra address to poll input queue status. The queue status is reported as one byte giving the available space (0..255). If more than 253 bytes available, 254 is reported. If the input queue is empty (eg. all commands processed) 255 is reported.

Commands are initiated by an STX character (0x02). If enabled at compilation time, an ENQ character (0x05) enters debug mode.

Screen coordinates are specified as uint8\_t values.

A timeout of 5 s is applied for all composite commands. After timeout the parser falls back to ASCII mode.

Notes:

- Jpeg and sprite as well as debug mode will not timeout!
- Cooked SPI mode is currently not supported.

Parameter values are of type uint8 except specified otherwise.

#### 3.1 Basic commands

Table 2: Basic Commands

| Type | Cmd  | nParam | Description                   | Notes  | Result                         |
|------|------|--------|-------------------------------|--|--------------------------------|
| 2    | 0x10 | 3      | read pixels from frame buffer | P1: <1..127> P2: <x> P3: <y>, send buffer cleared              | 2*n bytes                      |
| 2    | A    | 4      | set pixel                     | P1: <x> P2: <y> P3: <gggr rrrr> P4: <bbbb bggg>                |                                |
| 2    | E    | 0      | erase window with fill color  |  |                                |
| 2    | F    | 2      | font                          | see section 7 for font parameters                              |                                |
| 2    | K    | 2      | set pixel pen                 | P1: <x> P2: <y>  |                                |
| 2    | L    | 4      | line                          | P1..P4: <xa><ya><xe><ye>                                       |                                |
| 2    | P    | 4      | pen & fill                    | P1: <pen color:b5g6r5>, P2: <fill color:b5g6r5>                |                                |
| 2    | T    | 0      | get touch coordinates         |  | 3 bytes                        |
| 2    | X    | 2      | pos                           | P1..P2: <x><y>   |                                |
| 2    | a    | 2      | get pixel                     | P1: <x> P2: <y>  | 2 bytes <gggr rrrr><bbbb bggg> |
| 2    | c    | 4      | colors                        | P1: <foreground color:b5g6r5>, P2: <background color:b5g6r5>   |                                |
| 2    | g    | 1      | get status, P1: ['1','2',-]   | '1': get SW version, '2': get serial & I2C, -: <w><h><hf>0x4c  | 31, 20 or 4 bytes              |
| 2    | l    | ?      | linegraph                     | P1: is number of bytes to follow                               |                                |
| 2    | m    | ?      | dotgraph                      | P1: is number of bytes to follow                               |                                |
| 2    | p    | 3      | draw sprite                   | P1: <n:int16> P2: <type>, see section 9 for parameters details |                                |
| 2    | q    | ?      | polydot                       | P1: is number of (x,y) coordinates to follow                   |                                |
| 2    | Q    | 4      | fill rect                     | P1..P4: <xa><ya><w><h>   |                                |
| 2    | W    | 4      | draw rect                     | P1..P4: <xa><ya><w><h>   |                                |
| 2    | r    | 4      | rect                          | P1..P4: <xa><ya><w><h>   |                                |

Table 2: (continued)

| Type | Cmd | nParam | Description              | Notes   | Result                |
|------|-----|--------|--------------------------|---|-----------------------|
| 2    | v   | 4      | viewport                 | P1..P4:<br><left><top><right><bottom>   | set clipping viewport |
| 2    | x   | ?      | polyline                 | P1: is number of (x,y) coordinates to follow                                  |                       |
| ?    | -   | -      | print as ASCII character | handle \r, \n according to font, ANSI ESC support to allow REPL compatibility |                       |

### 3.2 Setup commands

Table 3: Setup commands

| Type | Cmd  | nParam | Description   | Notes   | Result |
|------|------|--------|---|---|--------|
| 2    | 0x14 | 1      | set topbot  | 0,2 portrait, 1,3 landscape   |        |
| 2    | 0x16 | 1      | set brightness  | 0x00..0xff  |        |
| 2    | 0x0e | 4      | set I2C address   | 0x49 0x32 0x43 (addr*2) & 0xfc  |        |
| 2    | 0x18 | 1      | UART baud rate P1 (0:2400, 1:4800, 2:9600, 3:19200, 4:38400, 5:57600, 6:115200, 7:230400, 8:460800) | save to NVRAM, effective after reset  |        |
| 2    | 0x19 | 1      | startup P1=bbbb bDSL b: brightness, D: use default brightness L: show logo S: show SN               |   |        |
| 2    | 0x1a | 0      | trigger touch calibration   | after successful calibration, touch parameters are stored in NVRAM  |        |
| 2    | f    | 1      | save to flash   | P1 n: saves NVRAM to flash, N: factory reset  |        |
| 2    | z    | 2      | calibrate touch panel   | P1: . . . . .TNC (T set touch mode (P2), N save to NVRAM, C calibrate) P2: IFff ffff (I pull down host interrupt, F output every <f> frame) |        |

### 3.3 Advanced commands

Table 4: Advanced commands

| Type | Cmd  | nParam | Description            | Notes                                 | Result   |
|------|------|--------|------------------------|---------------------------------------|----------|
| 2    | 0x11 | 1      | set window scroll text |                                       |          |
| 2    | 0x12 | 0      | enter SPI raw mode     | transfer raw SPI data to frame buffer |          |
| 2    | 0x15 | 1      | set window scroll mode |                                       |          |
| 2    | 0x17 | 0      | watchdog trigger       |                                       |          |
| 2    | 0x3f | 0      | get status             |                                       | 20 bytes |

Table 4: (continued)

| Type | Cmd | nParam | Description            | Notes   | Result     |
|------|-----|--------|------------------------|---|------------|
| 2    | u   | 4      | window parameter value | P1: <window:int8>, P2: <index:int8>, P3:<value:int16>   |            |
| 2    | U   | 17     | window parameters      |   |            |
| 2    | Y   | 4      | sysres                 | P1 0xdeadbeef:int32   |            |
| 2    | j   | 1      | draw jpeg              | P1: <i:int16>, number of bytes to follow, see section 9.2   |            |
| 2    | y   | 4      | window                 | define window for ERASE and POLY??? commands, P1..P4:<br><xul:int8><yul:int8><br><ww:int8><wh:int8> |            |
| 5    | -   | -      | enter debug mode       | needs compile time option   | DEBUG only |



### 3.4 Obsolete commands

These commands will be removed in future versions.

Table 5: Obsolete commands

| Type | Cmd  | nParam | Description  | Notes                            | Result             |
|------|------|--------|--|----------------------------------|--------------------|
| 2    | 0x08 | 1      | set character rotation matrix (0..3<br>0,90,180,270) |                                  |                    |
| 2    | 0x0f | 1      | read energy counter (always 0)                       |                                  | 6 bytes            |
| 2    | 0x1c | 0      | continue SPI raw mode                                |                                  |                    |
| 2    | @    | 1      | set config   | P1 . . . . . F F: backlight fade |                    |
| 2    | B    | 10     | bulk output to frame buffer                          | report CRC, match and input CRC  | 12 bytes           |
| 2    | C    | 10     | doCRC  |                                  | 4 bytes            |
| 2    | D    | 6      | dump   |                                  | DEBUG only         |
| 2    | G    | 0      | run buffer code                                      |                                  | 1 byte             |
| 2    | I    | 4      | init   |                                  |                    |
| 2    | O    | 0      | set SWD on   |                                  | DEBUG only         |
| 2    | R    | 4      | calculate CRC from frame buffer                      |                                  | 4 bytes            |
| 2    | S    | 8      | set32  |                                  | 1 bytes DEBUG only |
| 2    | V    | 0      | turn backlight on                                    |                                  |                    |
| 2    | b    | 10     | bulk output to buffer                                | report CRC, match and input CRC  |                    |
| 2    | o    | 0      | set SWD off  |                                  | DEBUG only         |
| 2    | s    | 6      | set16  |                                  | 1 bytes DEBUG only |
| 2    | v    | 0      | turn backlight off                                   |                                  |                    |

## 4 Polypoints and Polylines

Byte sequences are handled as y-values where x is incremented by one. Either a dot or a line is drawn at or to the new position. All coordinates are offset by <xul, yul> (see window command)

## 5 Linegraph and Dotgraph

Byte sequences are handled as (x,y)-coordinates. Either a dot or a line is drawn at or to the new position. All coordinates are offset by <xul, yul> (see window command)

## 6 Colors

The frame buffer handles 16-bit colors (bbbbbggg ggrrrrr).

## 7 Fonts

LCD160CRv1.0 has three fonts. Font 0 & 3 contain 95 glyphs.

Font 1 & 2 are identical and provide more than 1500 UTF8 glyphs.

Table 6: Available Fonts

| Font | Size | Glyphs       | Description |
|------|------|--------------|-------------|
| 0    | 4x5  | 0x20..0x7e   | tiny font   |
| 1    | 6x8  | 0x20..0xffff | UTF-8       |
| 2    | 6x8  | 0x20..0xffff | UTF-8       |
| 3    | 9x14 | 0x20..0x7e   | big font    |

Font attributes include size, boldness and transparency.

fontparameter: 0b..ss ssss STff hhww

Table 7: Font Parameter Fields

| Field | Size | Description             |
|-------|------|-------------------------|
| s     | 6    | pixel replication (s+1) |
| S     | 1    | soft scroll flag        |
| T     | 1    | transparency flag       |
| f     | 2    | font number             |
| h     | 2    | horizontal bold offst   |
| v     | 2    | vertical bold offst     |

## 8 Windows

The device supports up to 8+2 regions on screen that allow either horizontal or vertical scrolling in both directions. Window 8 is special, it handles a rolling message of up to 32 character. Window 9 is reserved for future use.

LCD160CRv1.0 handles scrolling without host intervention. Scrolling might be turned on or off globally or by modifying individual windows. A shift value of zero disables scrolling of an individual window.

A window description consists of 17 bytes.

<window\_no:8> <xstart:16> <ystart:16> <width:16> <height:16> <vector:16> <pattern:16> <fill:16> <color:16>

Table 8: Window Parameters

| Parameter | Values                | Description  | Notes  |
|-----------|-----------------------|--|--|
| window_no | 0..7, 8 and 9         | 8 is hor. text scroll ( <i>ticker</i> ), 9 is reserved   | other window numbers will modify raw SPI target window |
| xstart    | 0..W-1                |  | 16-bit value   |
| ystart    | 0..H-1                |  | 16-bit value   |
| width     | 2..W                  |  | 16-bit value   |
| height    | 2..H                  |  | 16-bit value   |
| vector    | 0bF.dd SSSS SSSS SSSS | dd: 0=+x, 1=+y, 2=-x, 3=-y F: frame mode, 1=scroll every <S> frame(s)<br>0=shift <S>/%256 pixels |  |
| pattern   | 0..65535              | 16-bit pattern mask  | alternates fill and color if fill != color             |
| fill      | 0..65535              | fill color   |  |
| color     | 0..65535              | extra color  |  |

For window 8 the pattern parameter selects the font for the ticker message. The maximum scale is limited to four. A zero value selects the default ticker font (font 3, transparent).

Only limited parameter checking takes place. It is the users obligation to provide correct and meaningful parameters.

**Raw SPI Target Window** The width and height parameters specify the coordinates of the last pixel inside the window.

## 9 Sprites

### 9.1 Bitmaps

The LCD supports drawing of raster pattern in various formats. The SPRITE command takes up to 65535 bytes (specified in the 16-bit parameter) and a format byte. The parameters are:

Table 9: Bitmap Parameters

| Parameter | Values       | Description      | Notes           |
|-----------|--------------|------------------|-----------------|
| P1        | 1..65535     | size of data + 2 |                 |
| P2        | ZZTb<br>bbbb | sprite format    |                 |
| P3        | 1..W         | width            |                 |
| P4        | 1..H         | height           |                 |
| <data>    |              |                  | P2-2 data bytes |

### 9.2 Sprite Format

Table 10: Sprite Format Fields

| Parameter | Values | Description    | Notes  |
|-----------|--------|----------------|--|
| Z         | 0..3   | zoom-1         | replicate each pixel   |
| T         | 0..1   | transparent    | pixel value 0 is transparent value   |
| b         | 1..16  | bits per pixel | legal values 1, 2, 4, 8, 16, for bpp values below 8 a lookup table has to be specified in the datastream |

### 9.3 JPEG

LCD160CR accepts JPEG data streams up to 65535 bytes.

Format

Table 11: JPEG Attributes

| Parameter            | Values                       |
|----------------------|------------------------------|
| JFIF Version         | 1.01                         |
| Resolution Unit      | None                         |
| X Resolution         | 1                            |
| Y Resolution         | 1                            |
| Encoding Process     | Baseline DCT, Huffman coding |
| Bits Per Sample      | 8                            |
| Color Components     | 3                            |
| Y Cb Cr Sub Sampling | YCbCr4:2:2 (2 1)             |

## 10 Touch Panel

The touch panel is factory calibrated. A built-in calibration procedure allows re-calibration anytime. Calibration parameters are stored in NVRAM. While touched, the interrupt line is driven low.

Touch coordinates can be retrieved via status query. Valid touch coordinates are mapped to screen coordinates, upper left is (0,0). Global screen orientation applies also to touch coordinates.

## 11 ANSI Mode

So far the following sequences are decoded.

Table 12: Supported ANSI Sequences

| Code | Parameter | Values               | Description              | Notes   |
|------|-----------|----------------------|--------------------------|---|
| D    | 1         | 1                    | move cursor backward     | default parameter 1                           |
| K    | 1         | 0, 1, 2              | clear to end of line     | only default value 0 is supported             |
| H    | 1..2      | goto screen position | upper left corner is 1,1 | default (1,1)                                 |
| m    | 1         | 30..39, 40..49       | set colors               | default is foreground green, background black |

Note: Other sequences are not decoded. Usually they are silently ignored. Using unsupported ANSI ESC sequences may result in undefined behavior. It is up to the user to provide reasonable parameters, like coordinates within screen etc.

## 12 NVRAM Parameters

A number of parameters are saved in non-volatile RAM.

Table 13: NVRAM Parameters

| Parameter          | Format | Values  | Default | Description   | Notes                                  |
|--------------------|--------|---|---------|---|--|
| backlight fade     | uint2  | 0..3  | 0       | enable backlight fading   | only valid for OLED displays           |
| boot decoration    | uint2  | 0..3  | 3       | 0bIL, I:show information, L:show logo                           |  |
| screen orientation | uint2  | 0..3  | 0       | initial screen orientation                                      |  |
| touch mode         | uint8  | 0..255  | 0x8000  | 0bIFff ffff, I: activate touch interrupt, F: frame/delta mode   | frame/delta mode currently ignored     |
| winscroll          | uint1  | 0..1  | 0       | enable or disable window scroll mode on reset                   | use carefully                          |
| brightness         | uint5  | 0..31   | 31      | initial display brightness                                      |  |
| baudrate           | uint32 | 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800 | 115200  | UART baudrate   |  |
| flags              | uint3  | 0..7  | 3       | 0bSUC S: initial SPI mode, U: echo on UART, C: need calibration | use initial SPI mode with special care |
| i2c address        | uint8  | 0x00, 0x04..0xfc  | 0xc4    | initial I2C address   | default addresses are 98,99            |

A firmware upgrade will reset all NVRAM parameters to their default state. This includes the touch calibration parameters. After a firmware upgrade the LCD160CR enters touch calibration mode on reset until the calibration finally succeeds.

## 13 Firmware Upgrade

The firmware of the LCD160CRv1.0 can be upgraded in the field. A new firmware is encoded in a byte stream to be sent via I2C or UART interface. It is mandatory to maintain proper power while performing the upgrade.

After a successful upgrade the LCD160CRv1.0 is reset to factory state. This implies the need to calibrate the touch panel.

Calibration mode is launched on every power cycle until a successful calibration procedure has been applied. The LCD supports a minimal set of ANSI escape codes. So far the following sequences are decoded.

### Notes

- If programming the new firmware fails during flash erasure and final flash write, the firmware can only be restored with a hardware programmer. Contact manufacturer for more information.
- Errors during data transfer will terminate the upgrade process without any flash modifications.

## 14 Technical Specifications

### 14.1 Power Supply

- LCD160CR can be safely operated from 3.3 V to 12 V
- Typical power consumption is from 10mA to 300mA (peak)
- Average power consumption is from 10mA to 60mA, depending on backlight brightness
- Shut down power consumption < 1 $\mu$ A

### 14.2 Ports

All I/O ports operate at 3.3 V.

Table 14: Ports

| Port | Name    | 5V | Description                            |
|------|---------|----|--|
| X1   | UART RX | x  | serial input, default baudrate 115200  |
| Y1   | UART RX | x  | serial input, default baudrate 115200  |
| X2   | UART TX | x  | serial output, default baudrate 115200 |
| Y2   | UART TX | x  | serial output, default baudrate 115200 |
| X3   | n.c.    |    |  |
| Y3   | n.c.    |    |  |
| X4   | POWER   | x  | High level turns on LCD160CR           |
| Y4   | POWER   | x  | High level turns on LCD160CR           |
| X5   | CTS     | x  | not used                               |
| Y5   | CTS     | x  | not used                               |
| X6   | SCLK    | x  | SPI clock                              |
| Y6   | SCLK    | x  | SPI clock                              |
| X7   | MISO    | x  | SPI output (not used)                  |
| Y7   | MISO    | x  | SPI output (not used)                  |
| X8   | MOSI    | x  | SPI input                              |
| Y8   | MOSI    | x  | SPI input                              |
| X9   | I2CCLK  | x  | I2C clock (no pullup)                  |
| Y9   | I2CCLK  | x  | I2C clock (no pullup)                  |
| X10  | I2CDAT  | x  | I2C clock (no pullup)                  |
| Y10  | I2CDAT  | x  | I2C clock (no pullup)                  |
| X11  | INT     | x  | host interrupt                         |
| Y11  | INT     | x  | host interrupt                         |
| X12  | n.c.    |    |  |
| Y12  | n.c.    |    |  |
| GND  | -       |    |  |
| V+   | +       |    | power input 3.3V..12V                  |
| 3V3  | 3.3V    |    | auxiliary power supply, do not use     |
| SWC  |         | -  | res.                                   |
| SWD  |         | -  | res.                                   |



### 14.3 Connectors

Power supply as well as interface signals are routed to a set of seven connectors. Most of the signals are routed to multiple connectors to allow easy attachment to a pyboard in various combinations.

Table 15: Connectors

| Conn | Description                                  |
|------|--|
| P1   | Y1..Y8, X9..X12, GND, RST, n.c, V+           |
| P2   | X1..X8, Y9..Y12, GND, RST, n.c, V+           |
| E1   | X1..X8                                       |
| E2   | X9..X12, GND, RST, n.c, V+                   |
| E3   | X1..X8                                       |
| E4   | X9..X12, GND, RST, n.c, V+                   |
| P4   | 3V3, SWC, SWD, X1, X2, X10, X9, X11, GND, V+ |

### 14.4 Communication

- UART TTL level UART, default baudrate 115200 8 bits/char no parity
- I2C fast I2C interface (400 kHz), no pullup resistors installed
- SPI serial synchronous interface (MOSI only, up to 13.5 MHz)