# 1 ezLCD-001

## 1.1 Overview

### Congratulation with your ezLCD-001.

The ezLCD-001 is an all-in-one advanced color TFT LCD panel which includes:

- 240x160 pixels 512 colors 2.7" TFT LCD (Sony ACX705AKM)
- LCD controller (Epson SED1375)
- Embedded processor (Atmel ATmega128L)
- Power supply, which generates all the voltages needed by the logic and the display itself
- Interface drivers and other circuitry.

The ezLCD-001 communicates with outside world through many implemented interfaces:

- RS232
- USB
- I2C
- SPI
- 8 bit parallel (Centronix printer protocol)

*Figure 1. ezLCD-001 Top*

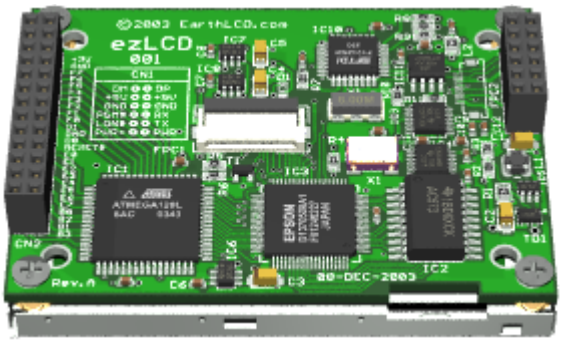



*Figure 2. ezLCD-001 Bottom*

The ezLCD-001 is driven by a set of commands, which can be fed through any of the implemented interfaces. The device may be used as an "intelligent" display or as a stand alone device as well. There is plenty of flash memory left in ATmega128 to incorporate additional graphic instructions, or to customize the software for particular tasks. Possible applications include automotive, avionics, nautical, industrial control, hobby, etc.

## 1.2 Operation

The ezLCD-001 is driven by a set of 8 bit commands, which can be received by any of the implemented interfaces.

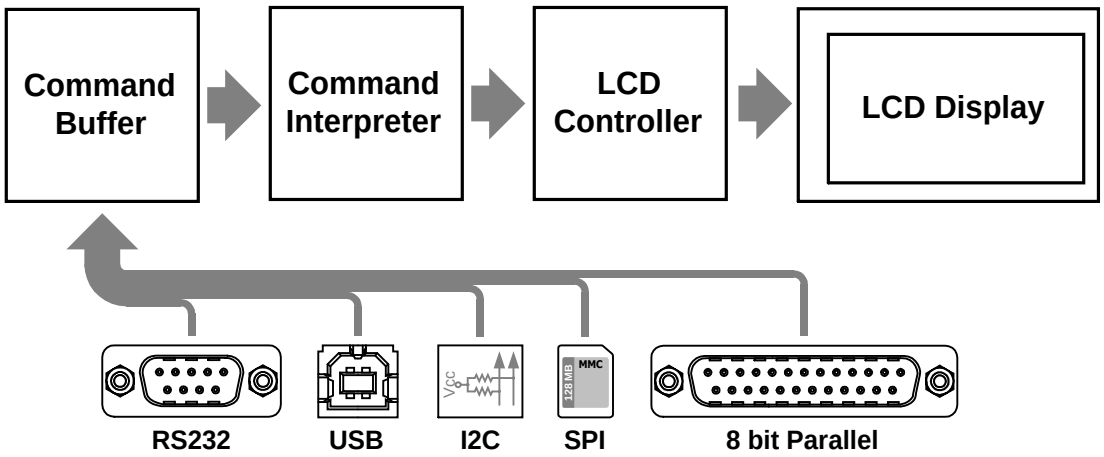



*Figure 3. ezLCD-001 Data flow Diagram*

Each of the implemented interfaces uses the same set of the ezLCD Commands.
Upon arrival, the ezLCD Commands are stored into 1024 byte long **Command Buffer** as shown on the *Figure 3*.
All interfaces use the same Command Buffer. The **Command Interpreter** (*Figure 3*.), picks up byte-by-byte the commands stored in the Command Buffer and drives the **LCD Controller** with the corresponding set of signals and instructions. The commands are processed on a First-In, First-Out principle.
Such data flow architecture makes possible implementation of some advanced graphic commands, like CIRCLE_R, LINE_TO_XY, PUT_BITMAP, etc.

**Example:**

**The following commands will draw a green circle with the radius of 60 pixels and the center positioned at the column 120 and row 80.**

**Pseudo-Code (ANSI C format):**

```
SetColor(GREEN);   /* Set the drawing color to green */
SetXY(120, 80);    /* Set the position to x = 120, y = 80 */
CircleR(60);       /* Draw the circle with the radius of 60 pixels */
```

**Data sent to the ezLCD** (Columns: Value and Format)**:**

| Mnemonic | Value | Format | Comment |
|---|---|---|---|
| SET_COLOR | 24 | hex | Set the drawing color to: |
| green | 00111000 | bin | green |
| SET_XY | 25 | hex | Set the drawing position to: |
| 120 | 120 | dec | x (column) = 120 |
| 80 | 80 | dec | y (row) = 80 |
| CIRCLE_R | 29 | hex | Draw the circle with the radius of: |
| 60 | 60 | dec | 60 pixels |

## 1.3 Hardware & Interfaces

### 1.3.1 Block Diagram

The ezLCD-001 Hardware Block Diagram is shown on the *Figure 4.* below.

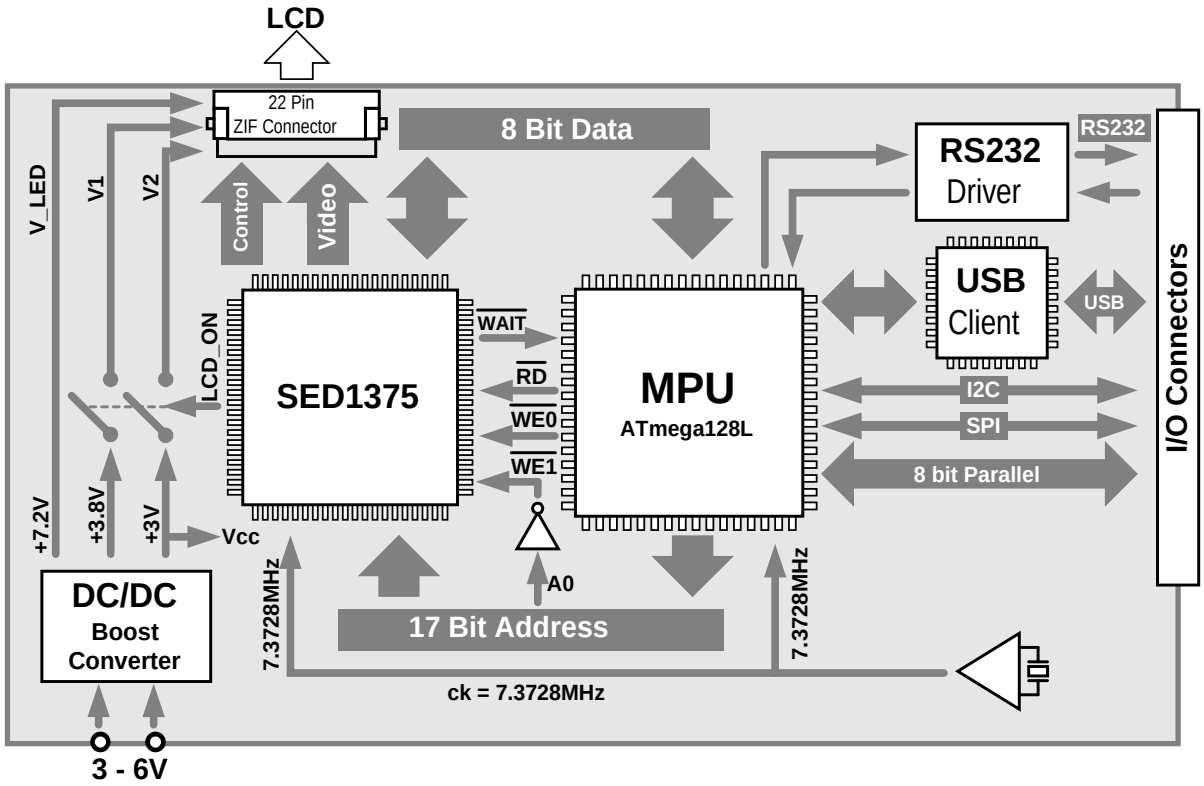



*Figure 4. ezLCD-001 Block Diagram*

The ezLCD-001 receives the commands through any of the available interfaces (RS232, USB, I2C, SPI and Parallel).
The MPU (ATmega128L) processes the received data and writes the resulting pixels into the Video RAM of the SED1375 LCD controller.
The SED1375 generates the "Digital CRT" video signals, using the data stored in the Video RAM.

### 1.3.2 Power Distribution

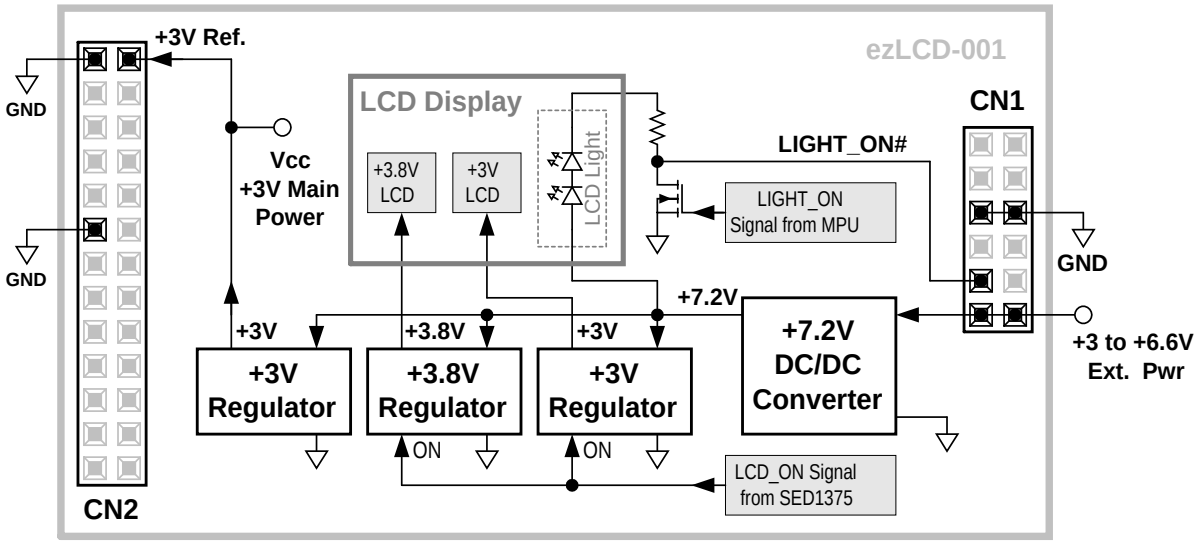



*Figure 5. ezLCD-001 Power Supply and Distribution*

The ezLCD-001 Power Supply System generates the following voltages:

- +3V Main Power Vcc (MPU, SED1375 and Interfaces)
- +3.8V LCD (LCD Screen $V_1$)
- +3V LCD (LCD Screen $V_2$)
- +7.2V (Voltage Regulators and LCD Light, $V_{LED}$)

## Operation

**External Power**

The ezLCD-001 is powered by the External Voltage of 3V to 6.6V DC. The External Voltage is first converted to the regulated +7.2V by the high efficiency (97%) DC/DC Converter. The 7.2V is than used by other regulators to generate all required voltages.

**LCD Display**

The LCD Display requires 3 different voltages: 3.8V ($V_1$), 3V ($V_2$) and 7.2V ($V_{LED}$). $V_1$ and $V_2$ are used by LCD screen and logic. They can be turned ON or OFF by the SED1735. $V_{LED}$ powers the LCD Light.

**LCD Light**

The LCD Light is powered by 7.2V ($V_{LED}$) generated by the DC/DC Converter. The LCD Light can be turned on or off by the LIGHT_ON signal from MPU ( ezLCD commands: LIGHT_ON and LIGHT_OFF ). Additionally, the light can be turned on by jumping the signal LIGHT_ON# to the GND on the CN1 connector. Light On condition has the priority over Light Off. For example, once LIGHT_ON# is jumpered to the GND, the light cannot be extinguished be sending LIGHT_OFF command to the ezLCD-001. The following table shows the LCD Light logic.

| LIGHT_ON | LIGHT_ON# | LCD Light |
|---|---|---|
| OFF | Open | Off |
| OFF | GND | On |
| ON | Open | On |
| ON | GND | On |

**Vcc +3V Main Power**
This voltage powers MPU, SED1375, interfaces and other circuits on the ezLCD-001 board. Vcc is outputted on the connector CN2, where it is called +3V Ref.

**NOTE:** The +3V Ref is an I/O reference voltage.
It may be used as a pull-up source (I2C etc.).
It SHOULD NOT be used as a power source.

### 1.3.3 RS-232

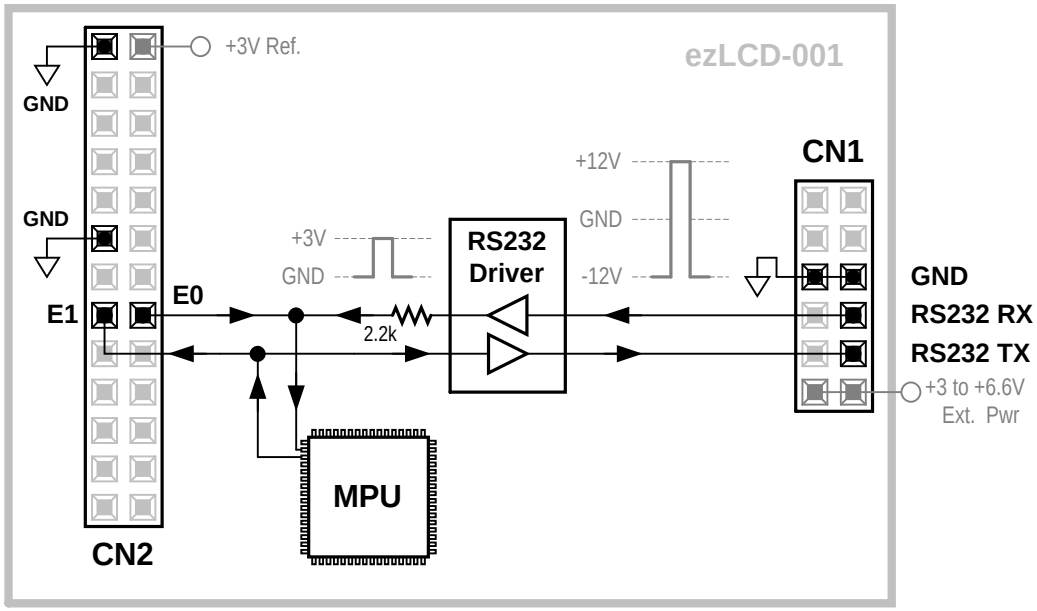



*Figure 6. ezLCD-001 RS232 Interface*

## Default Communication Parameters

**Baudrate:** ..... 115200 bps
**No of Stop Bits:** .... 1
**Parity:** ..... Off
**Handshake:** ..... None

## Operation

**RS232:**
The ezLCD-001 uses 3 wires for a non-handshake RS232 communication:
- RS232 RX (ezLCD receive)
- RS232 TX (ezLCD transmit)
- GND (common ground)

The voltage levels and limits are as per RS232 standard.
The MPU handles the asynchronous communication protocol. The RS232 Driver converts voltage levels from MPU 0V(Lo) and 3V(Hi) to RS232 -12V(Lo) and +12V(Hi).

**Asynchronous Serial:**
The ezLCD-001 uses 3 wires for a non-handshake
Asynchronous Serial (RS232-TTL) communication:
- E0 (ezLCD receive)
- E1 (ezLCD transmit)
- GND (common ground)

The voltage levels are:
- 0V to +1V = Lo (logical "0")
- +2V to +3V = Hi (logical "1")
- Absolute minimum: -0.2V
- Absolute maximum: +3.2V

The MPU handles the asynchronous communication protocol. The Asynchronous Serial Interface uses the same MPU lines as the RS232 does. The 2.2k resistor is used to separate the receive signals from both interfaces. The Asynchronous Serial receive has the priority over the RS232 receive

### 1.3.4 USB

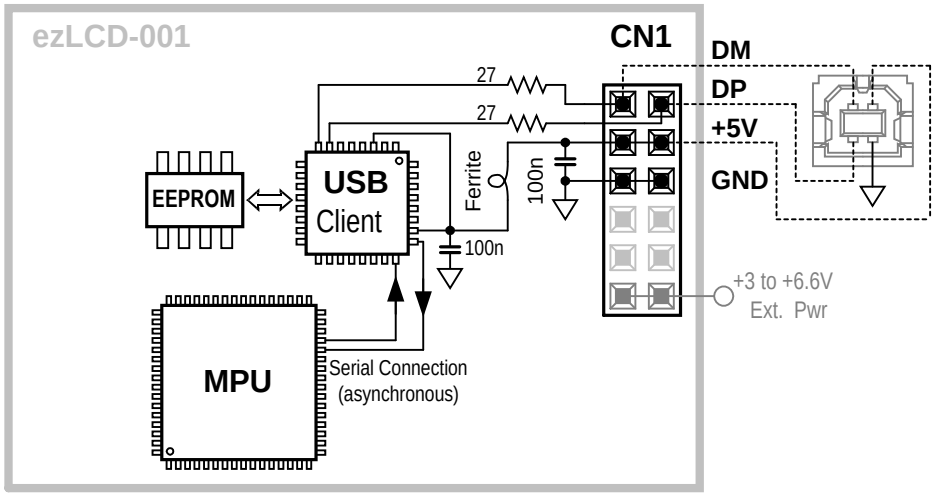



*Figure 7. ezLCD-001 USB Interface*

## Operation

**Connector CN1**

The ezLCD-001 USB Interface uses 4 lines:

- DM (USB Data Minus)
- DP (USB Data Plus)
- +5V
- GND

The DM and DP lines are connected, through 27 Ohm resistors, to the USB Client IC.
The +5V line first goes through EMI filter and then is used to supply power to the USB Client IC and the EEPROM IC.

**USB Client IC**

The USB Client IC (FT232BM by FTDI Chip), handles all protocol and physical layer aspects of the USB communication.
MPU communicates with the USB Client through standard asynchronous serial connection using the following communication parameters:

- Baudrate: ..... 115200 bps
- No of Stop Bits: .... 1
- Parity: ..... Off

**EEPROM IC**

The EEPROM IC (93C46 type) is used to store the USB configuration data like:

- USB Vendor ID and Product ID
- USB Version (1.0, 1.1 or 2.0)
- Product and Manufacturer Description Strings
- USB Serial Number
- Etc.

The USB Client IC retrieves all the above data from the EEPROM IC and uses it in the USB communication.
The data stored in the EEPROM IC may be modified by using the MProg utility, which is available for download on the FTDI Chip site: http://www.ftdichip.com

**Host Configuration**

FTDI Chip provides ready-to-go royalty free USB drivers, which can configure the operating system of the Host Computer (Windows, Linux, OSX, etc) to "see" the ecLCD-001 as an additional RS232 port or as a custom USB device.

When ezLCD-001 USB is configured as a RS232 port, the following communication parameters should be used:

**Baudrate:** ..... 115200 bps
**No of Stop Bits:** .... 1
**Parity:** ..... Off
**Handshake:** ..... None

**Drivers , Software and Documentation**

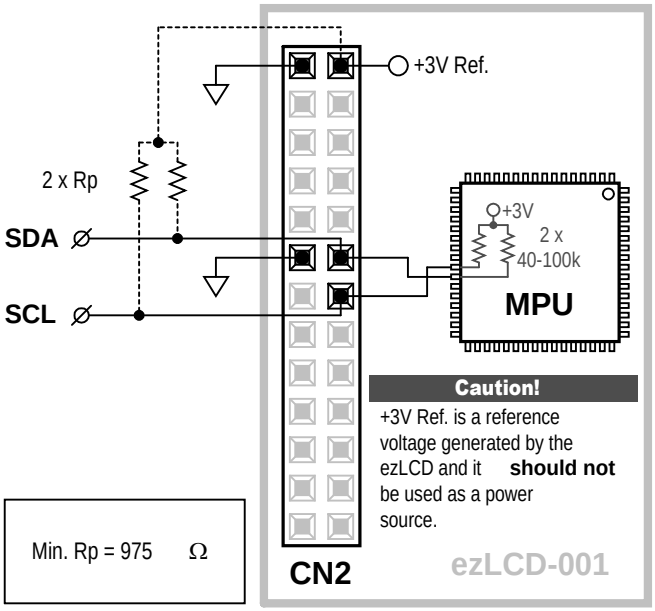
The latest documentation, software and drivers are available for download on the FTDI Chip site: http://www.ftdichip.com.

The following links were last checked on the August, 1, 2004 and may not be valid anymore:

- Drivers: http://www.ftdichip.com/FTDriver.htm
- Utilities: http://www.ftdichip.com/FTUtilities.htm
- Documentation:
  - Application Notes: http://www.ftdichip.com/FTApp.htm
  - Datasheets: http://www.ftdichip.com/FTProduct.htm
  - MProg Manual: http://www.ftdichip.com/Documents/MProg.pdf
  - Other Resources: http://www.ftdichip.com/FTResource.htm

Please, note that the chip used by ezLCD-001 is: **FT232BM**

### 1.3.5 I2C




| When $f_{SCL} \leq 100kHz$ | When $f_{SCL} > 100kHz$ |
|---|---|
| Max. Rp = $\frac{1000}{C_b+10}$ kΩ | Max. Rp = $\frac{300}{C_b+10}$ kΩ |

$C_b$ [pF] = capacitance of one bus line

Max $C_b$ = 400pF (10 feet, or 3 meters)

*Figure 8. ezLCD-001 I2C Interface*

## Operation

**Connector CN2**

The ezLCD-001 I2C Interface uses 3 wires:

- SCL (Clock)
- SDA (Data)
- GND

**Pull-Up Resistors**

The pull-up resistors (Rp) should be connected to +3V.
The ezLCD-001 outputs +3V reference voltage, which may be used as a pull-up source, as it is shown on the *Figure 8.* above.

**Protocol**

- Configuration:

The ezLCD-001 is configured as an I2C Slave.

- Address:

The default I2C address of the ezLCD-001 is 111 dec (6F hex).

- Handshake:

The ezLCD-001 responds with NACK (non-acknowledge) if it's 1024 byte command circular buffer runs out of space.

Reminder:
I2C address byte consists of the 7 address bits and the R/W bit.
This means that the address byte should be 222 dec (DE hex).

### 1.3.6 Board Layout

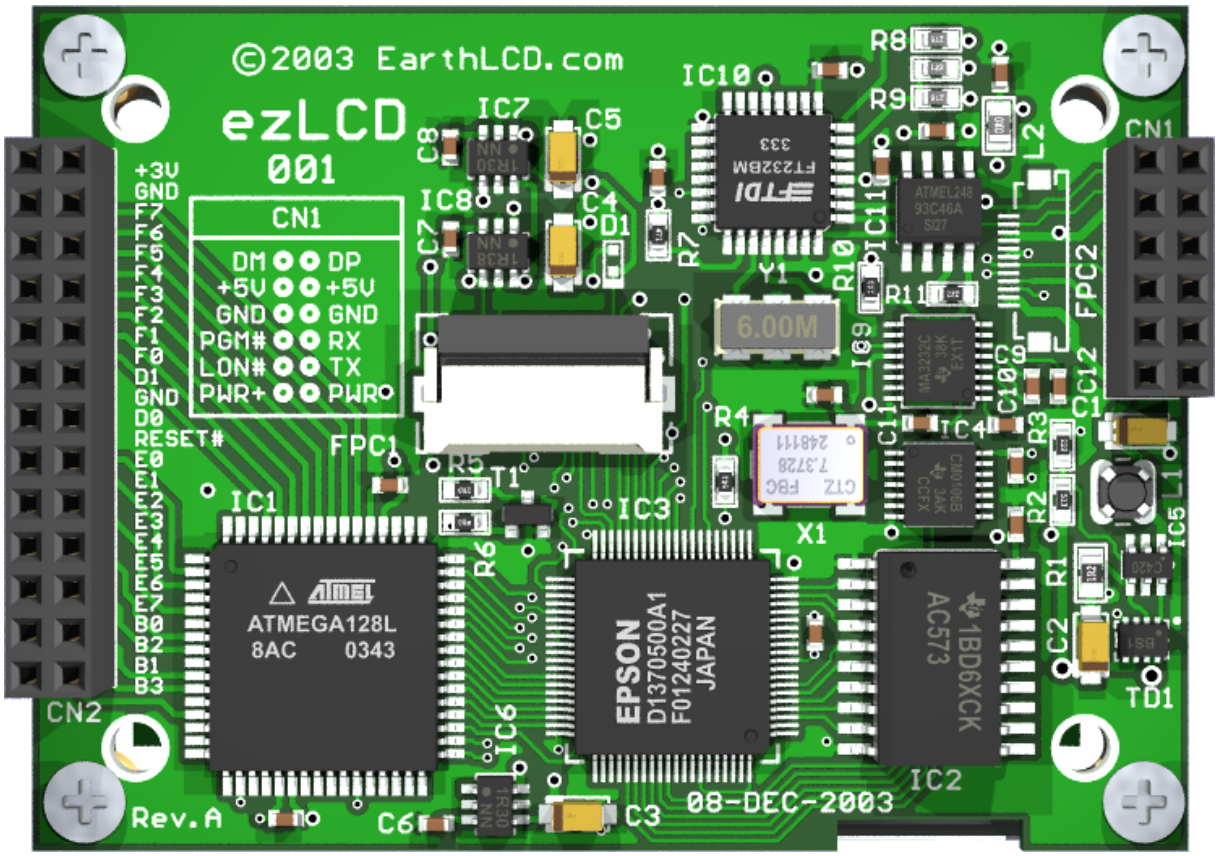



*Figure 10. ezLCD-001 Board Layout*

### 1.3.7 Board Dimensions

**NOTE:** All dimensions are in Inches

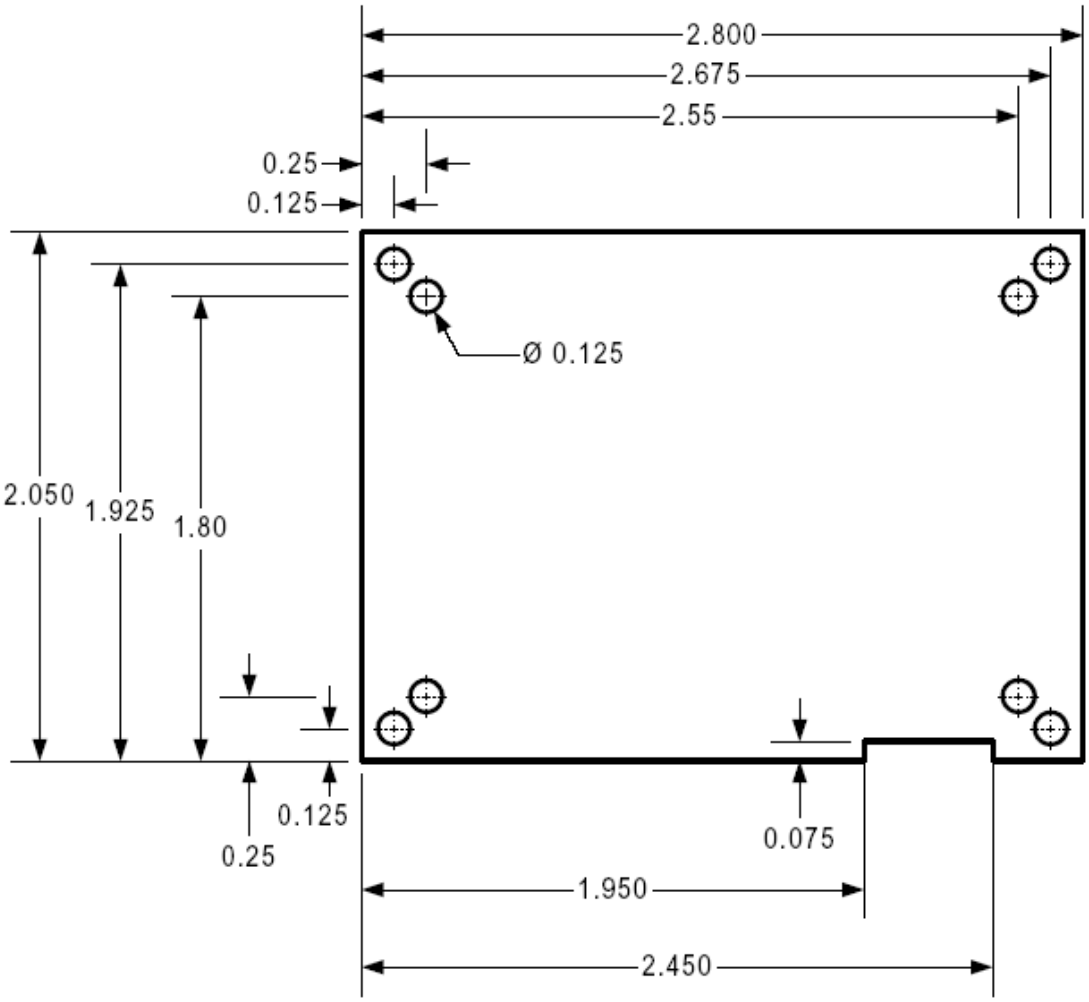



*Figure 11. ezLCD-001 Board Dimensions*

## 1.4 ezLCD Commands

The instructions may be fed to ezLCD through the Evaluation Board's RS232 and USB connectors.
The USB-Client port is based on the FTDI **FT232BM** chip, which is USB-RS232 bridge.
Upon installation of the driver, an Operating System of a Personal Computer treats USB port of ezLCD as an additional COM port.

### The default parameters of the RS232 and USB are:

**Baudrate:** ..... 115200 bps
**No of Stop Bits:** .... 1
**Parity:** ..... Off
**Handshake:** ..... None

The new USB Drivers and software may be downloaded from USB Drivers & Software

**Note:** This chapter describes only a few graphic instructions. Additional instructions will be added with each firmware upgrade.

**General**
CLS
LIGHT_ON
LIGHT_OFF
SET_COLOR
SET_XY

**Points**
PLOT
PLOT_XY

**Lines**
H_LINE
V_LINE
LINE_TO_XY

**Figures**
ARC
CIRCLE_R
CIRCLE_R_FILL
BOX
BOX_FILL

**Bitmaps**
PUT_BITMAP
PUT_ICON
PICTURE

**Text and Fonts**
SELECT_FONT
SET_BG_COLOR
TEXT_NORTH
TEXT_EAST
TEXT_SOUTH
TEXT_WEST
PRINT_CHAR
PRINT_CHAR_BG
PRINT_STRING
PRINT_STRING_BG

### 1.4.1 ARC

**Description:** Draws an Arc in Current Color, with the center at Current Position, starting on Begin Angle and ending on the End Angle.
**Class:** Multi Byte Command
**Code:** **2F**hex, **47**dec, **/** ASCII

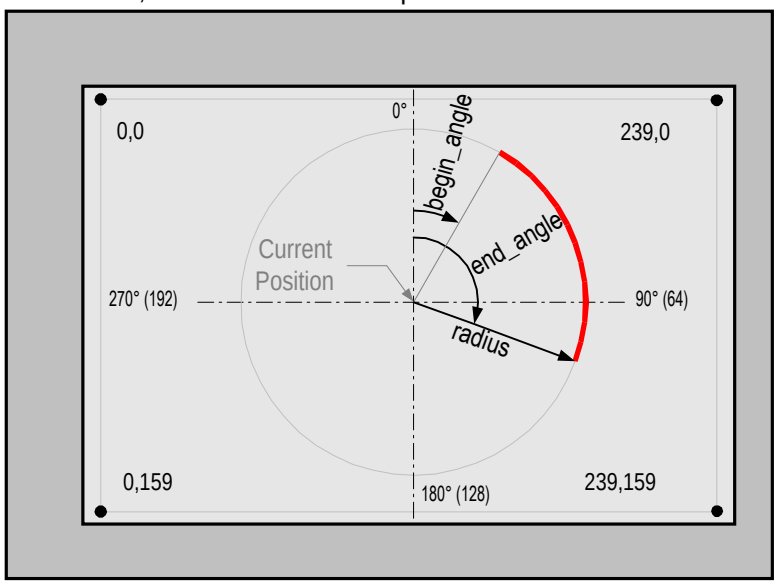
| 7 6 5 4 3 2 1 0 | |
|---|---|
| **ARC** | **Byte 0 (Command)** |
| **r a d i u s** | **Byte 1 (Radius)** |
| **begin_angle** | **Byte 2 (Arc Begin Angle)** |
| **end_angle** | **Byte 3 (Arc End Angle)** |

**See Also:** SET_XY, SET_COLOR, CIRCLE_R

**Angle Coding:** The angle range is from 0 to 255.
To transform degrees to ARC angle units:
**`Angle_lcd = Angle_deg x 32 / 45`**
For example:

```
 32 =  45°
 64 =  90°
128 = 180°
192 = 270°
  0 = 0° = 360°
```

The angle is drawn clockwise with the zero positioned at the top of a screen, as it is shown on the picture below




## Example:

**The following sequence will draw a green arc from 45 to 225 degrees with the center positioned in the middle of a screen.**

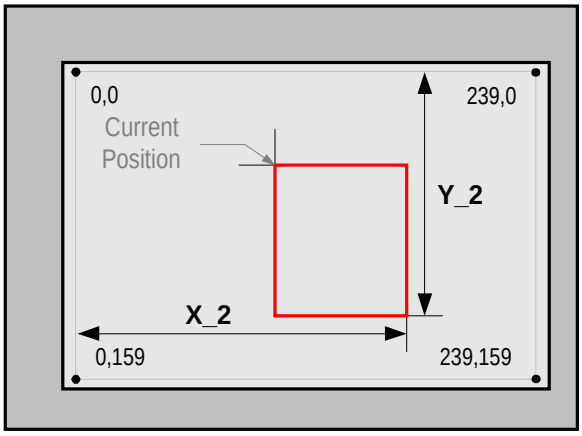
```
SET_COLOR        24 hex
```

```
GREEN          00111000 bin
SET_XY         25 hex
120           120 dec
 80            80 dec
ARC            2F hex
 60            60 dec (radius)
 32            32 dec (begin_angle = 45 degrees)
160           160 dec (end_angle = 225 degrees)
```

## 1.4.2 BOX

**Description:** Draws a rectangle.
**Class:** Multi Byte Command
**Code:** **42**hex, **66**dec, **B** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| BOX | | | | | | | | **Byte 0 (Command)** |
| X_2 | | | | | | | | **Byte 1 (Corner Column)** |
| Y_2 | | | | | | | | **Byte 2 (Corner Row)** |




**See Also:** SET_XY, BOX_FILL

**Example:**

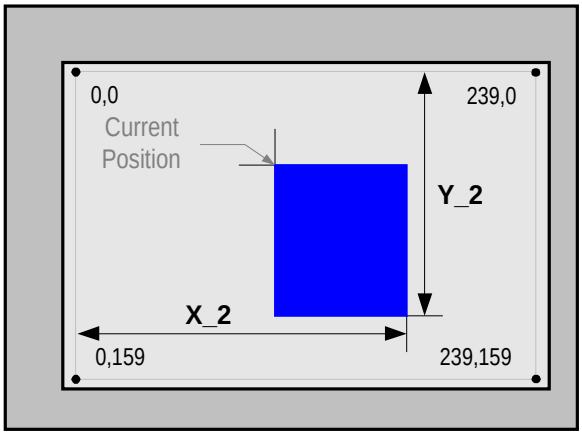
**The following sequence will draw the red rectangle**

```
SET_COLOR    24 hex
RED        00000111 bin
SET_XY       25 hex
 95          95 dec
 40          10 dec
BOX          42 hex
180         180 dec (X_2)
120         120 dec (Y_2)
```

### 1.4.3 BOX_FILL

**Description:** Draws a rectangle filled with Current Color
**Class:** Multi Byte Command
**Code:** **43**hex, **67**dec, **C** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| BOX_FILL | | | | | | | | **Byte 0 (Command)** |
| X_2 | | | | | | | | **Byte 1 (Corner Column)** |
| Y_2 | | | | | | | | **Byte 2 (Corner Row)** |




**See Also:** SET_XY, BOX

## Example:

**The following sequence will draw the rectangle filled with blue color**

```
SET_COLOR    24 hex
RED        11000000 bin
SET_XY       25 hex
 95          95 dec
 40          10 dec
BOX_FILL     43 hex
180         180 dec (X_2)
120         120 dec (Y_2)
```

### 1.4.4 CIRCLE_R

**Description:** Draws a circle in Current Color at Current Position
**Class:** Double Byte Command
**Code:** **29**hex, **41**dec, **)** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| CIRCLE_R | | | | | | | | **Byte 0 (Command)** |
| r a d i u s | | | | | | | | **Byte 1 (radius)** |

**See Also:** SET_XY, SET_COLOR

**Example:**

**The following sequence will draw a green circle in the middle of the screen.**

```
SET_COLOR   24 hex
GREEN       00111000 bin
SET_XY      25 hex
120        120 dec
80          80 dec
CIRCLE_R    29 hex
60          60 dec
```

### 1.4.5 CIRCLE_R_FILL

**Description:** Draws a circle in Current Color at Current Position, filled with Current Color
**Class:** Double Byte Command
**Code:** **39**hex, **57**dec, **9** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| CIRCLE_R_FILL | | | | | | | | **Byte 0 (Command)** |
| r a d i u s | | | | | | | | **Byte 1 (Radius)** |

**See Also:** SET_XY, SET_COLOR

**Example:**

**The following sequence will draw a red filled circle in the middle of the screen.**

```
\SET_COLOR      24 hex
RED            00000111 bin
SET_XY          25 hex
120            120 dec
80              80 dec
CIRCLE_R_FILL   39 hex
60              60 dec
```

### 1.4.6 CLS

**Description:** Clears screen by filling it with the Current Color
**Class:** Single Byte Command
**Code:** **21**hex, **33**dec, **!** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| | | | C | L | S | | | **Byte 0 (Command)** |

**See Also:** SET_COLOR

**Example:**

**The following sequence will clear the screen**

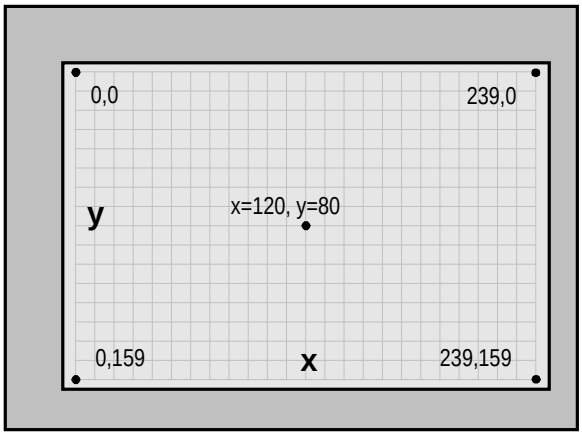
```
SET_COLOR    24 hex
WHITE        11111111 bin
CLS          21 hex
```

## 1.4.7 H_LINE

**Description:** Fast draws a horizontal line from Current Position, to the column specified by the parameter.
**Class:** Double Byte Command
**Code:** **40**hex, **64**dec, @ ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| H_LINE | | | | | | | | **Byte 0 (Command)** |
| x7 | x6 | x5 | x4 | x3 | x2 | x1 | x0 | **Byte 1 ( X )** |

**Note:** The screen size is 240x160. However, the valid X range is 0 - 255




**See Also:** V_LINE, SET_XY

## Example:

**The following sequence will draw the horizontal green line from (20, 60) to (170, 60)**

```
SET_COLOR    24 hex
GREEN      00111000 bin
SET_XY       25 hex
 20          20 dec
 60          60 dec
H_LINE       40 hex
170         170 dec
```

### 1.4.8 LIGHT_OFF

**Description:** Turns off the screen light
**Class:** Single Byte Command
**Code:** **23**hex, **35**dec, # ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| LIGHT_OFF | | | | | | | | **Byte 0 (Command)** |

**See Also:** LIGHT_ON

**Example:**

**The following sequence will turn off the screen light**

```
LIGHT_OFF   23 hex
```

### 1.4.9 LIGHT_ON

**Description:** Turns on the screen light
**Class:** Single Byte Command
**Code:** **22**hex, **34**dec, **"** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| LIGHT_ON | | | | | | | | **Byte 0 (Command)** |

**See Also:** LIGHT_OFF

**Example:**

**The following sequence will turn on the screen light**

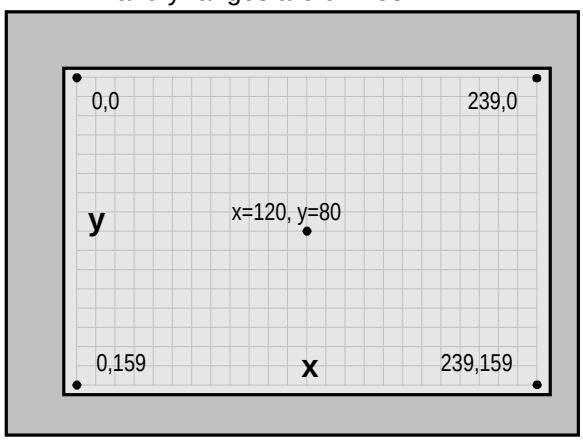
```
LIGHT_ON     22 hex
```

## 1.4.10 LINE_TO_XY

**Description:** Draws a line in Current Color, from the Current Position to the to specified position
**Class:** Multi Byte Command
**Code:** **28**hex, **40**dec, **(** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| LINE_TO_XY | | | | | | | | **Byte 0 (Command)** |
| x7 | x6 | x5 | x4 | x3 | x2 | x1 | x0 | **Byte 1 (x)** |
| y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0 | **Byte 2 (y)** |

**Note:** The screen size is 240x160. However, the valid x and y ranges are 0 - 255




**See Also:** SET_XY, SET_COLOR, PLOT

**Example:**

**The following sequence will draw a red line across the screen.**

```
SET_COLOR     24 hex
RED           00000111 bin
SET_XY        25 hex
0              0 dec
0              0 dec
LINE_TO_XY    28 hex
239          239 dec
159          159 dec
```

## 1.4.11 PICTURE

**Description:** Puts a bitmap picture over the entire screen
**Class:** Multi Byte Command
**Code:** **2A**hex, **42**dec, ***** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| PICTURE | | | | | | | | **Byte 0 (Command)** |
| b1 | b0 | g2 | g1 | g0 | r2 | r1 | r0 | **Byte 1 (x=0, y=159)** |
| b1 | b0 | g2 | g1 | g0 | r2 | r1 | r0 | **Byte 2 (x=1, y=159)** |
| b1 | b0 | g2 | g1 | g0 | r2 | r1 | r0 | **Byte 3 (x=2, y=159)** |
| ⋮ | | | | | | | | |
| b1 | b0 | g2 | g1 | g0 | r2 | r1 | r0 | **Byte 240 (x=239, y=159)** |
| b1 | b0 | g2 | g1 | g0 | r2 | r1 | r0 | **Byte 241 (x=0, y=158)** |
| ⋮ | | | | | | | | |
| b1 | b0 | g2 | g1 | g0 | r2 | r1 | r0 | **Byte 38399 (x=238, y=0)** |
| b1 | b0 | g2 | g1 | g0 | r2 | r1 | r0 | **Byte 38400 (x=239, y=0)** |

**See Also:** SET_XY, SET_COLOR, PUT_BITMAP

### 1.4.12 PLOT

**Description:** Plots a point at Current Position in Current Color
**Class:** Single Byte Command
**Code:** **26**hex, **38**dec, **&** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| | | | | **PLOT** | | | | **Byte 0 (Command)** |

**See Also:** SET_XY, SET_COLOR

**Example:**

**The following sequence will put the blue point in the middle of the screen.**

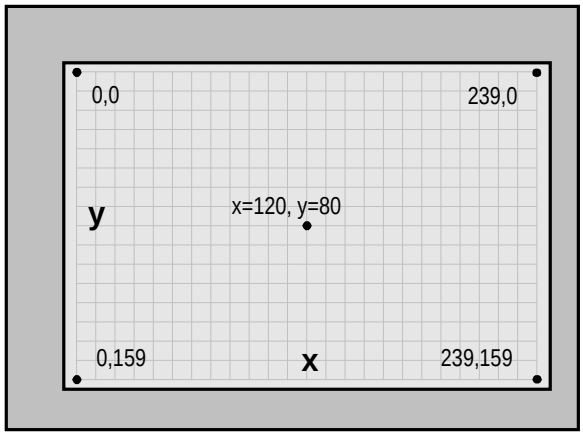
```
SET_COLOR    24 hex
BLUE         11000000 bin
SET_XY       25 hex
120         120 dec
80           80 dec
PLOT         26 hex
```

## 1.4.13 PLOT_XY

**Description:** Plots a point in Current Color, at specified position.
**Class:** Multi Byte Command
**Code:** **27**hex, **39**dec, **'** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| PLOT_XY | | | | | | | | **Byte 0 (Command)** |
| x7 | x6 | x5 | x4 | x3 | x2 | x1 | x0 | **Byte 1 (x)** |
| y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0 | **Byte 2 (y)** |

**Note:** The screen size is 240x160. However, the valid x and y ranges are 0 - 255




**See Also:** SET_XY, SET_COLOR, PLOT

**Example:**

**The following sequence will put the red point in the middle of the screen.**

```
SET_COLOR    24 hex
RED          00000111 bin
PLOT_XY      27 hex
120          120 dec
80           80 dec
```

### 1.4.14 PRINT_CHAR

**Description:** Prints a character at Current Position
**Class:** Double Byte Command
**Code:** **2C**hex, **44**dec, **,** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| PRINT_CHAR | | | | | | | | **Byte 0 (Command)** |
| A S C I I | | | | | | | | **Byte 1 (ASCII Character)** |

**See Also:** SELECT_FONT, PRINT_STRING

**Example:**

**The following sequence will print black character 'M' in the middle of the screen, using font number 2**

```
SELECT_FONT    2B hex
2               2 dec
SET_COLOR      24 hex
BLACK    00000000 bin
SET_XY         25 hex
120           120 dec
80             80 dec
PRINT_CHAR     2C hex
'M'            4D hex
```

## 1.4.15 PRINT_CHAR_BG

**Description:** Prints a character at Current Position on the background specified by SET_BG_COLOR command
**Class:** Double Byte Command
**Code:** **3C**hex, **60**dec, **<** ASCII

| 7 6 5 4 3 2 1 0 | |
|---|---|
| **PRINT_CHAR_BG** | **Byte 0 (Command)** |
| **A S C I I** | **Byte 1 (ASCII Character)** |

**See Also:** SELECT_FONT, SET_BG_COLOR, PRINT_STRING_BG

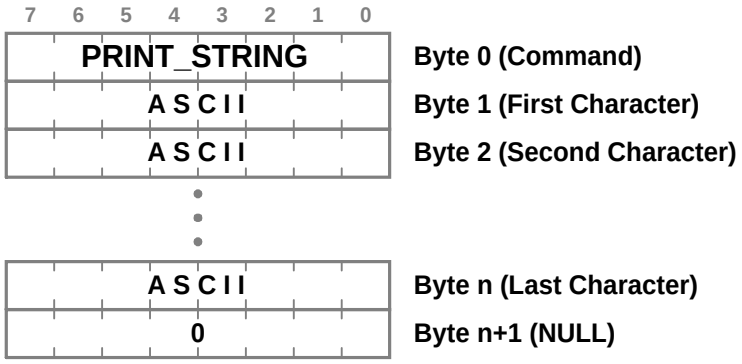
### Example:

**The following sequence will print white character 'M', on a black background in the middle of the screen, using font number 2**

```
SELECT_FONT   2B hex
2              2 dec
SET_BG_COLOR  34 hex
BLACK         00000000 bin
SET_COLOR     24 hex
WHITE         11111111 bin
SET_XY        25 hex
120          120 dec
80            80 dec
PRINT_CHAR_BG 3C hex
'M'           4D hex
```

## 1.4.16 PRINT_STRING

**Description:** Prints null-terminated String starting at Current Position
**Class:** Multi Byte Command
**Code:** **2D**hex, **45**dec, **-** ASCII




**See Also:** SELECT_FONT, PRINT_CHAR

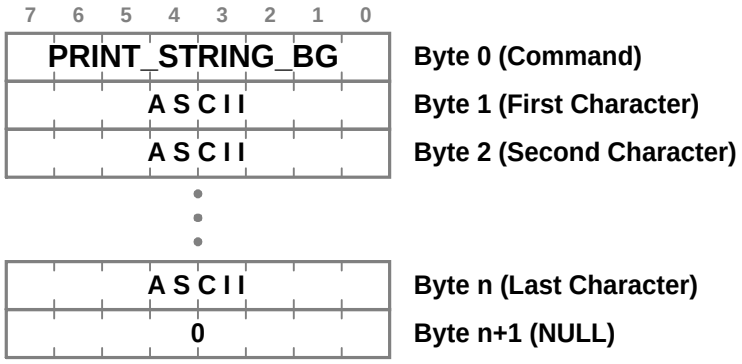
**Example:**

**The following sequence will print violet sign "LCD" in the middle of the screen, using font number 1**

```
SELECT_FONT   2B hex
1              1 dec
SET_COLOR     24 hex
VIOLET        11000100 bin
SET_XY        25 hex
120          120 dec
80            80 dec
PRINT_STRING  2D hex
'L'           4C hex
'C'           43 hex
'D'           44 hex
NULL           0 hex
```

## 1.4.17 PRINT_STRING_BG

**Description:** Prints null-terminated String starting at Current Position on the background specified by SET_BG_COLOR command
**Class:** Multi Byte Command
**Code:** **3D**hex, **61**dec, **=** ASCII




**See Also:** SELECT_FONT, SET_BG_COLOR, PRINT_CHAR_BG

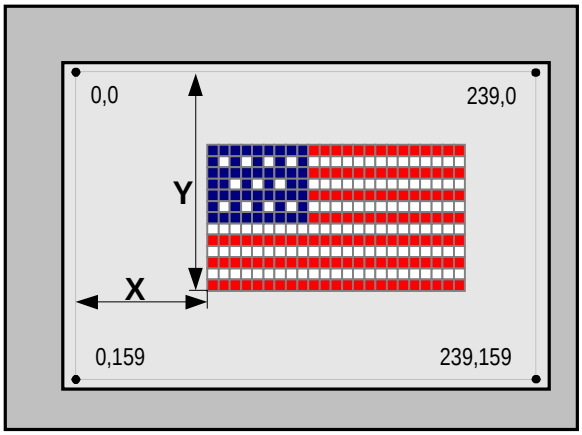
**Example:**

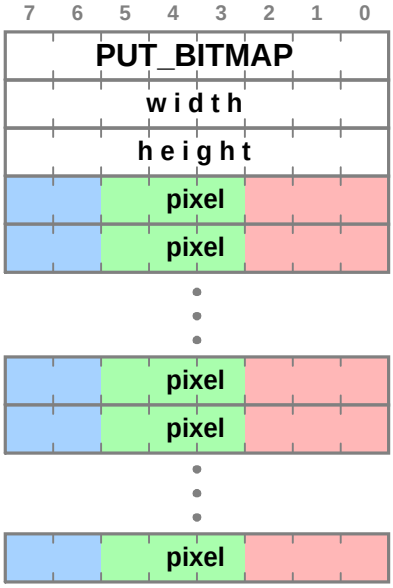
**The following sequence print Yellow "LCD" on the Navy background, in the middle of a screen, using font no 0.**

```
SET_BG_COLOR     34 hex
NAVY             10000000 bin
SET_COLOR        24 hex
YELLOW           00111111 bin
SET_XY           25 hex
120             120 dec
80               80 dec
SELECT_FONT      2B hex
0                 0 dec
PRINT_STRING_BG  3D hex
'L'              4C hex
'C'              43 hex
'D'              44 hex
NULL              0 hex
```

## 1.4.18 PUT_BITMAP

**Description:** Puts Bitmap on the screen starting at Current Position, then UP and RIGHT
**Class:** Multi Byte Command
**Code:** **2E**hex, **46**dec, **.** ASCII




**Byte 0 (Command)**
**Byte 1 (Bitmap Width)**
**Byte 2 (Bitmap Height)**
**Byte 3 (pixel at: X, Y)**
**Byte 4 (pixel at: X+1, Y)**

**Byte width+2 (pixel at: X+width-1, Y)**
**Byte width+3 (pixel at: X, Y-1)**

**Byte height x width + 2 (pixel at: X+width-1, Y-height+1)**

**Note:** The total number of bytes is: width x height + 3

**See Also:** SET_XY, SET_COLOR, PICTURE

**Example:**

**The following sequence will put 4x3 bitmap at x = 60, y = 80**

```
SET_XY        25 hex
x             60 dec
y             80 dec
PUT_BITMAP    2E hex     ----------+--
width          4 dec               |
height         3 dec               |
```

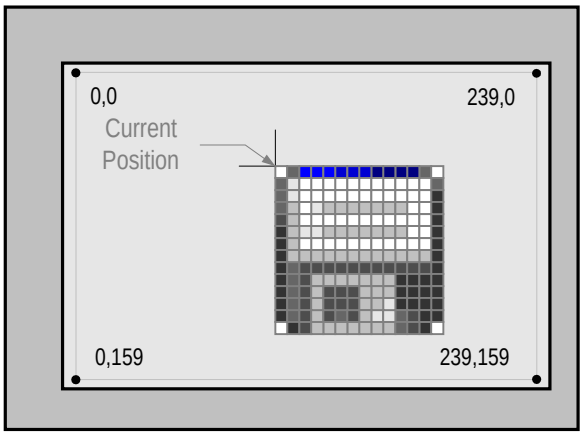
```
pixel (x = 60, y = 80)            |
pixel (x = 61, y = 80)            |
pixel (x = 62, y = 80)            |
pixel (x = 63, y = 80)          TOTAL:
pixel (x = 60, y = 79)     4 x 3 + 3 = 15 bytes
pixel (x = 61, y = 79)            |
pixel (x = 62, y = 79)            |
pixel (x = 63, y = 79)            |
pixel (x = 60, y = 78)            |
pixel (x = 61, y = 78)            |
pixel (x = 62, y = 78)            |
pixel (x = 63, y = 78)  ----------+
```

| | | | |
|---|---|---|---|
| 11 | 12 | 13 | 14 |
| 7 | 8 | 9 | 10 |
| 3 | 4 | 5 | 6 |

## 1.4.19 PUT_ICON

**Description:** Displays the icon with it's upper-left corner positioned at the Current Position.
The icon is read from the ezLCD ROM.
Use ezLCDrom.exe utility to store icons in the ezLCD ROM

**Class:** Double Byte Command
**Code:** **57**hex, **87**dec, **W** ASCII




| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| PUT_ICON | | | | | | | | **Byte 0 (Command)** |
| Icon ID | | | | | | | | **Byte 1 (Icon ID)** |

**See Also:** SET_XY

**Example:**

**The following sequence will display an icon no 3 with it's upper-left corner positioned at X = 60, Y = 43**

```
SET_XY      25 hex
60          60 dec
43          43 dec
PUT_ICON    57 hex
3            3 dec
```

## 1.4.20 SELECT_FONT

**Description:** Sets the Current Font
**Class:** Double Byte Command
**Code:** **2B**hex, **43**dec, **+** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| SELECT_FONT | | | | | | | | **Byte 0 (Command)** |
| font number | | | | | | | | **Byte 1 (font number)** |

**Note:** The following fonts are implemented

Font 0: ezLCD-001
Font 1: ezLCD-001
Font 2: ezLCD-001
Font 3: ezLCD-001
Font 4: ezLCD-001
Font 5: ezLCD-001

**See Also:** PRINT_STRING, PRINT_CHAR

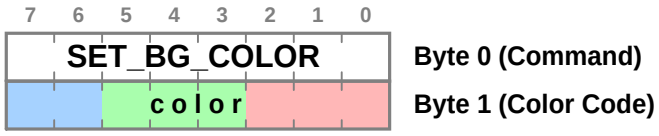
**Example:**

**The following sequence will print black character 'M' in the middle of the screen, using font number 2**

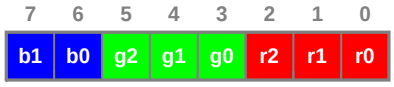
```
SELECT_FONT    2B hex
2               2 dec
SET_COLOR      24 hex
BLACK    00000000 bin
SET_XY         25 hex
120           120 dec
80             80 dec
PRINT_CHAR     2C hex
'M'            4D hex
```

### 1.4.21 SET_BG_COLOR

**Description:** Sets the Background Color for the following instructions:
PRINT_CHAR_BG
PRINT_STRING_BG
**Class:** Double Byte Command
**Code:** **34**hex, **52**dec, **4** ASCII




**Note:** The default NATURAL palette has the following color coding:




**See Also:** PRINT_CHAR_BG, PRINT_STRING_BG, PALETTE

## Example:

**The following sequence print Yellow "LCD" on the Navy background, in the middle of a screen, using font no 0.**

```
SET_BG_COLOR     34 hex
NAVY             10000000 bin
SET_COLOR        24 hex
YELLOW           00111111 bin
SET_XY           25 hex
120             120 dec
80               80 dec
SELECT_FONT      2B hex
0                 0 dec
PRINT_STRING_BG  3D hex
'L'              4C hex
'C'              43 hex
'D'              44 hex
NULL              0 hex
```

## 1.4.22 SET_COLOR

**Description:** Sets the Current Color
**Class:** Double Byte Command
**Code:** **24**hex, **36**dec, **$** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| SET_COLOR | | | | | | | | **Byte 0 (Command)** |
| | | c | o | l | o | r | | **Byte 1 (Color Code)** |

**Note:** The default NATURAL palette has the following color coding:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| b1 | b0 | g2 | g1 | g0 | r2 | r1 | r0 |

**See Also:** CLS, PLOT, PALETTE

**Example:**

**The following sequence will fill the whole display with green**

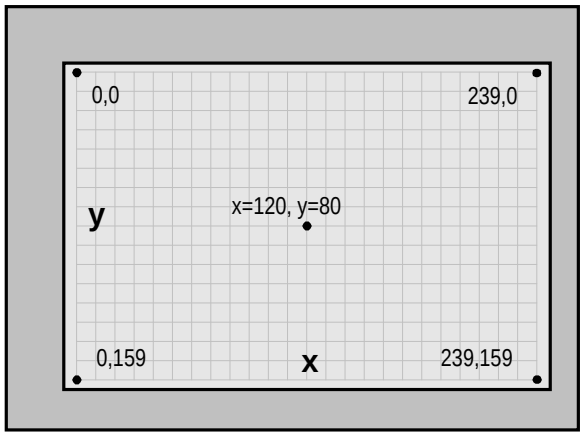
```
SET_COLOR    24 hex
GREEN        00111000 bin
CLS          21 hex
```

## 1.4.23 SET_XY

**Description:** Sets the Current Position
**Class:** Multi Byte Command
**Code:** **25**hex, **37**dec, **%** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| SET_XY | | | | | | | | **Byte 0 (Command)** |
| x7 | x6 | x5 | x4 | x3 | x2 | x1 | x0 | **Byte 1 (x)** |
| y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0 | **Byte 2 (y)** |

**Note:** The screen size is 240x160. However, the valid x and y ranges are 0 - 255




**See Also:** PLOT, LINE_TO_XY, CIRCLE_R

**Example:**

**The following sequence will put the blue point in the middle of the screen.**

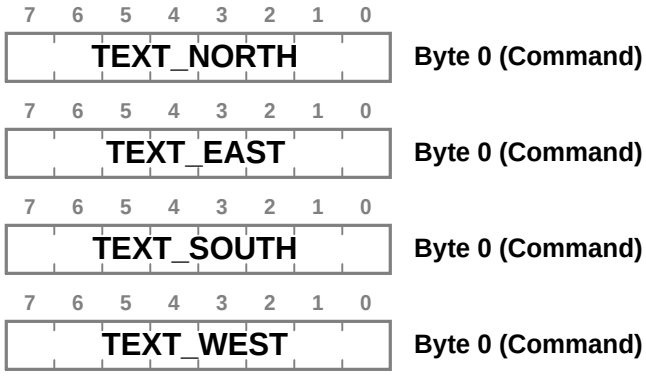
```
SET_COLOR    24 hex
BLUE         11000000 bin
SET_XY       25 hex
120         120 dec
80           80 dec
PLOT         26 hex
```

## 1.4.24 TEXT_EAST

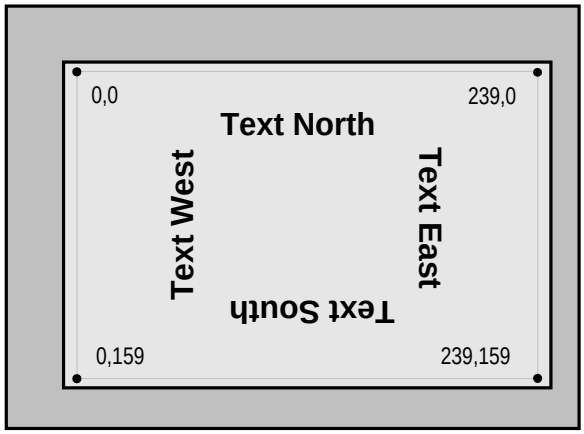
**Description:** Set the orientation of the text, as shown on the picture below

**Class:** Single Byte Commands

**Code:**
**TEXT_NORTH** : **60**hex, **96**dec, **'** ASCII
**TEXT_EAST** : **61**hex, **97**dec, **a** ASCII
**TEXT_SOUTH** : **62**hex, **98**dec, **b** ASCII
**TEXT_WEST** : **2F**hex, **99**dec, **c** ASCII




**Note:** TEXT_NORTH is the default text orientation




**See Also:** PRINT_CHAR, PRINT_STRING, SELECT_FONT

## Example:

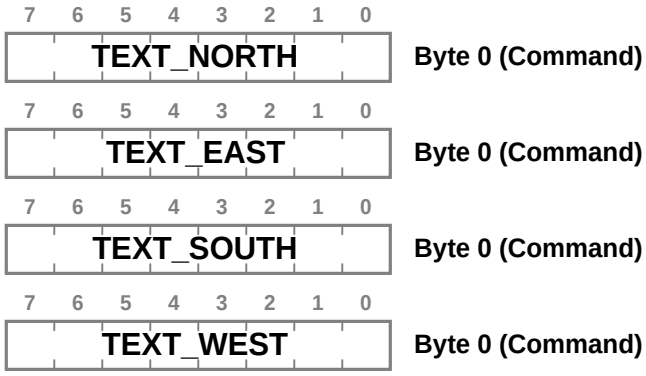
**The following sequence will print the text pattern similar to the one on the picture above.**

```
SET_XY          25 hex
60              60 dec
10              10 dec
SELECT_FONT     2B hex
0                0 dec
TEXT_NORTH      60 hex
PRINT_STRING    2D hex
"Text North "
```

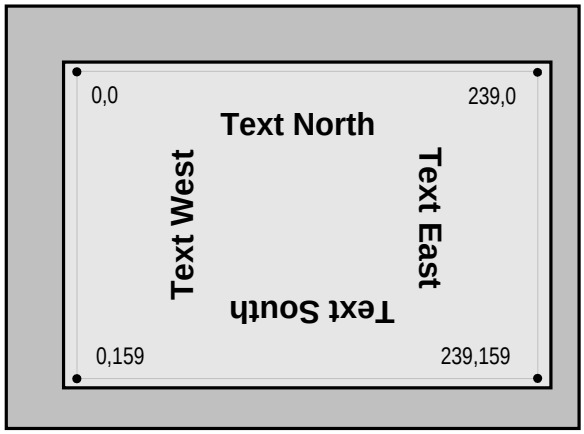
```
NULL           0 hex
TEXT_EAST     61 hex
PRINT_STRING  2D hex
"  Text East  "
NULL           0 hex
TEXT_SOUTH    62 hex
PRINT_STRING  2D hex
" Text South "
NULL           0 hex
TEXT_WEST     63 hex
PRINT_STRING  2D hex
"  Text West "
NULL           0 hex
```

## 1.4.25 TEXT_NORTH

**Description:** Set the orientation of the text, as shown on the picture below
**Class:** Single Byte Commands
**Code:**
`TEXT_NORTH` : **60**hex, **96**dec, **'** ASCII
`TEXT_EAST` : **61**hex, **97**dec, **a** ASCII
`TEXT_SOUTH` : **62**hex, **98**dec, **b** ASCII
`TEXT_WEST` : **2F**hex, **99**dec, **c** ASCII




**Note:** `TEXT_NORTH` is the default text orientation




**See Also:** PRINT_CHAR, PRINT_STRING, SELECT_FONT

**Example:**

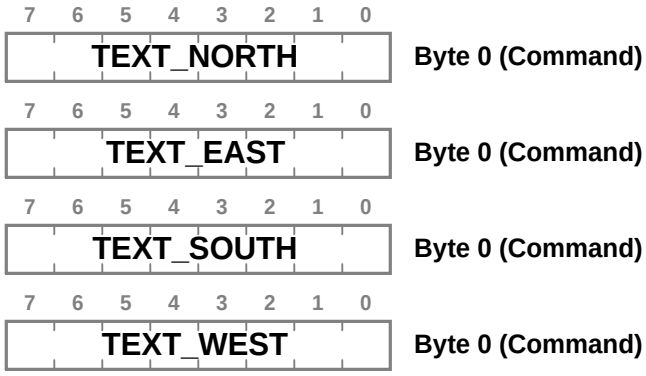
**The following sequence will print the text pattern similar to the one on the picture above.**

```
SET_XY        25 hex
60            60 dec
10            10 dec
SELECT_FONT   2B hex
0              0 dec
TEXT_NORTH    60 hex
PRINT_STRING  2D hex
"Text North "
```

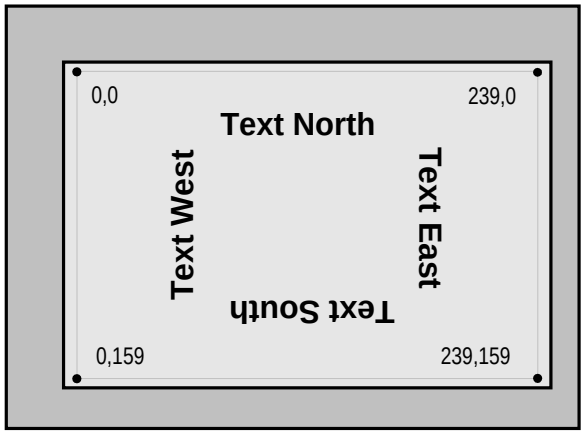
```
NULL           0 hex
TEXT_EAST     61 hex
PRINT_STRING  2D hex
"  Text East  "
NULL           0 hex
TEXT_SOUTH    62 hex
PRINT_STRING  2D hex
" Text South "
NULL           0 hex
TEXT_WEST     63 hex
PRINT_STRING  2D hex
"  Text West "
NULL           0 hex
```

## 1.4.26 TEXT_SOUTH

**Description:** Set the orientation of the text, as shown on the picture below
**Class:** Single Byte Commands
**Code:**
**TEXT_NORTH** : **60**hex, **96**dec, **'** ASCII
**TEXT_EAST** : **61**hex, **97**dec, **a** ASCII
**TEXT_SOUTH** : **62**hex, **98**dec, **b** ASCII
**TEXT_WEST** : **2F**hex, **99**dec, **c** ASCII




**Note:** TEXT_NORTH is the default text orientation




**See Also:** PRINT_CHAR, PRINT_STRING, SELECT_FONT

## Example:

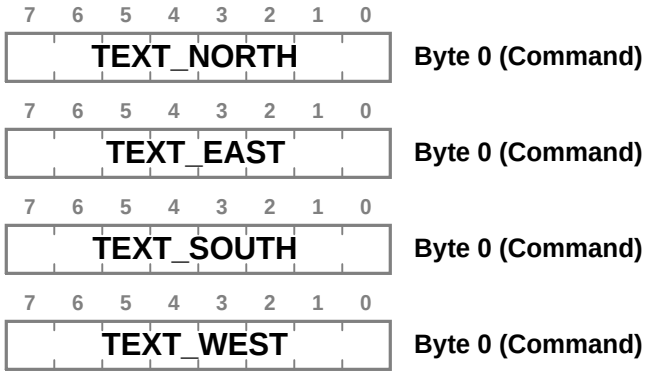
**The following sequence will print the text pattern similar to the one on the picture above.**

```
SET_XY          25 hex
60              60 dec
10              10 dec
SELECT_FONT     2B hex
0                0 dec
TEXT_NORTH      60 hex
PRINT_STRING    2D hex
"Text North "
```

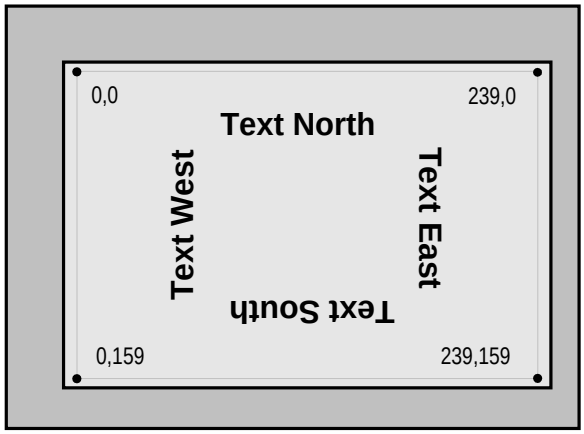
```
NULL           0 hex
TEXT_EAST     61 hex
PRINT_STRING  2D hex
"  Text East  "
NULL           0 hex
TEXT_SOUTH    62 hex
PRINT_STRING  2D hex
" Text South "
NULL           0 hex
TEXT_WEST     63 hex
PRINT_STRING  2D hex
"  Text West "
NULL           0 hex
```

## 1.4.27 TEXT_WEST

**Description:** Set the orientation of the text, as shown on the picture below
**Class:** Single Byte Commands
**Code:**
**TEXT_NORTH** : **60**hex, **96**dec, **'** ASCII
**TEXT_EAST** : **61**hex, **97**dec, **a** ASCII
**TEXT_SOUTH** : **62**hex, **98**dec, **b** ASCII
**TEXT_WEST** : **2F**hex, **99**dec, **c** ASCII




**Note:** TEXT_NORTH is the default text orientation




**See Also:** PRINT_CHAR, PRINT_STRING, SELECT_FONT

## Example:

**The following sequence will print the text pattern similar to the one on the picture above.**

```
SET_XY        25 hex
60            60 dec
10            10 dec
SELECT_FONT   2B hex
0              0 dec
TEXT_NORTH    60 hex
PRINT_STRING  2D hex
"Text North "
```

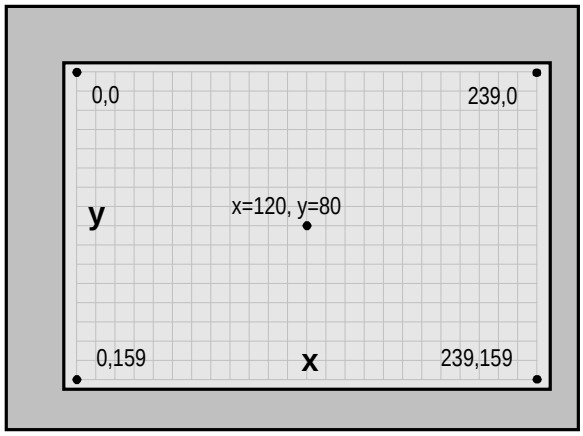
```
NULL           0 hex
TEXT_EAST     61 hex
PRINT_STRING  2D hex
"  Text East  "
NULL           0 hex
TEXT_SOUTH    62 hex
PRINT_STRING  2D hex
" Text South "
NULL           0 hex
TEXT_WEST     63 hex
PRINT_STRING  2D hex
"  Text West "
NULL           0 hex
```

## 1.4.28 V_LINE

**Description:** Fast draws a vertical line from Current Position, to the row specified by the parameter.
**Class:** Double Byte Command
**Code:** **41**hex, **65**dec, **A** ASCII

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|
| V_LINE | | | | | | | | **Byte 0 (Command)** |
| y7 | y6 | y5 | y4 | y3 | y2 | y1 | y0 | **Byte 1 ( Y )** |

**Note:** The screen size is 240x160. However, the valid Y range is 0 - 255




**See Also:** H_LINE, SET_XY

## Example:

**The following sequence will draw the vertical blue line from (95, 10) to (95, 110)**

```
SET_COLOR    24 hex
BLUE       11000000 bin
SET_XY       25 hex
 95          95 dec
 10          10 dec
V_LINE       41 hex
110         110 dec
```

## 1.5 Evaluation Board

**Note:**
This document is only a short preliminary version of the *ezLCD-001 Evaluation Board Manual* and should be treated only as a "Quick Start" reference. The full documentation will follow soon.

### 1.5.1 Introduction

#### Congratulation with your ezLCD-001 Evaluation Board.

The ezLCD-001 Evaluation Board is a starter kit and development system for the ezLCD-001 from EarthLCD. It's purpose is to give the designers a quick start to write and check ezLCD graphic commands.

### 1.5.2 Quick Start

The ezLCD-001 may be checked by invoking the Self Test, or running the Av232 Utility. Additionally the Av232 Utility may be used to send bitmap files to ezLCD-001 or to make various drawings on the screen of the ezLCD-001.

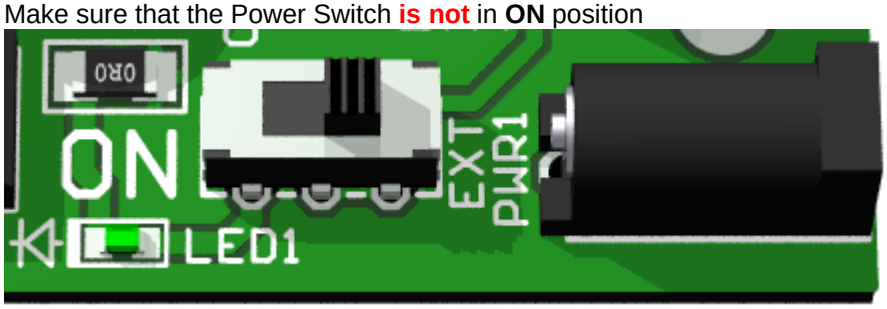
Self Test
Av232 Utility

### 1.5.2.1 Self Test

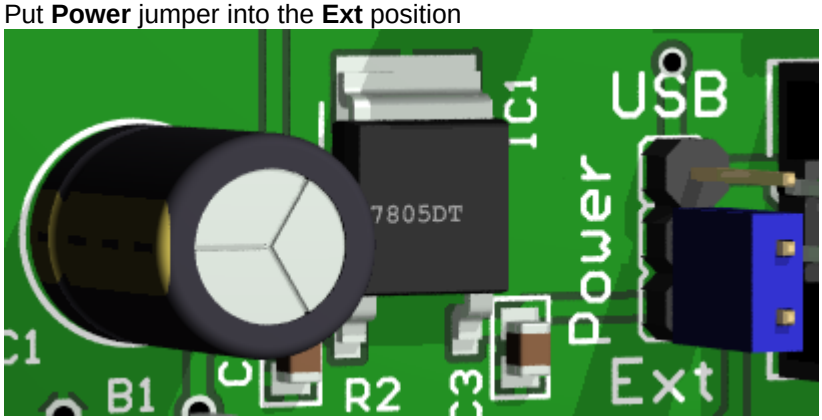
Self Test can be executed by powering the EzLCD while **TEST** jumper is closed.
During the Self Test RS232 cable should be disconnected from the Evaluation Board.

## In order to invoke the Self Test:

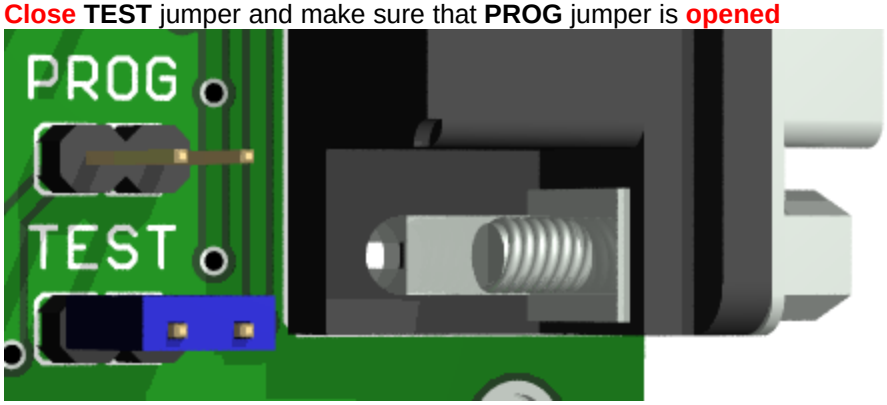
1. Plug ezLCD-001 into the Evaluation Board, making sure that the connectors are not misaligned
2. Make sure that the Power Switch **is not** in **ON** position




3. Put **Power** jumper into the **Ext** position

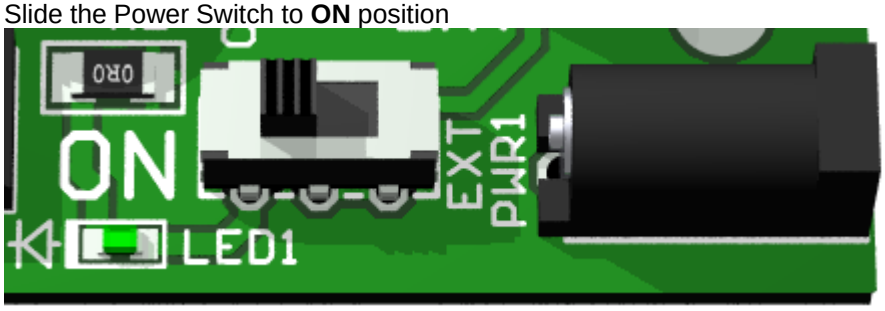



4. **Close TEST** jumper and make sure that **PROG** jumper is **opened**




5. **Make sure** that **no RS232 cable** is connected to the Evaluation Board
6. Connect External Power (6.5 to 12V DC or AC)

**7.** Slide the Power Switch to **ON** position




## EzLCD-001 should now display the test pattern

**NOTE:** The **TEST** jumper connects ezLCD RS232 Tx to Rx in order to execute a wrap-around test.
The **TEST** jumper should be opened for normal operation.

### 1.5.2.2 Av232 Utility

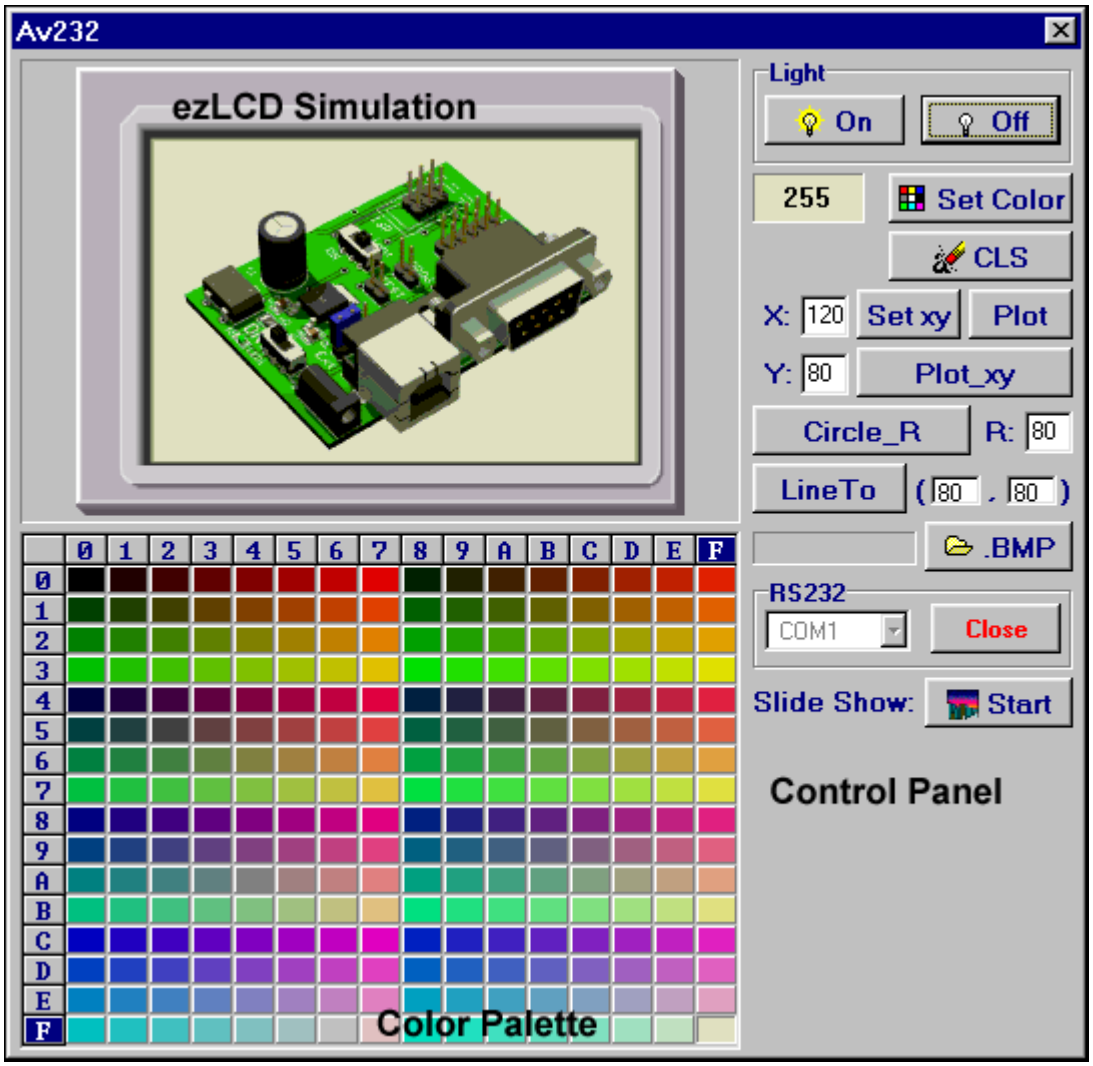
The Av232 Utility may be used to send bitmap files to ezLCD-001 or to make various drawings on the screen of the ezLCD-001.

**NOTE:** The Av232 Utility is now under development. The version described in this chapter is a preliminary one at best.

The Av232 utility is in the Av232 directory on the ezLCD-001 CD.

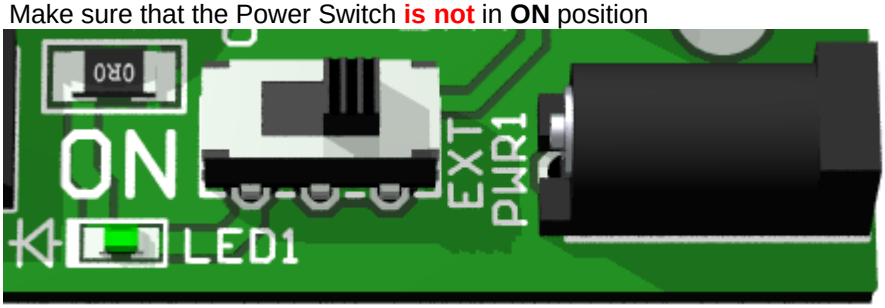
## To drive ezLCD-001 by Av232 Utility:

1. Connect ezLCD
2. Run Av232 Utility
3. Open PC COM Port
4, Send Commands



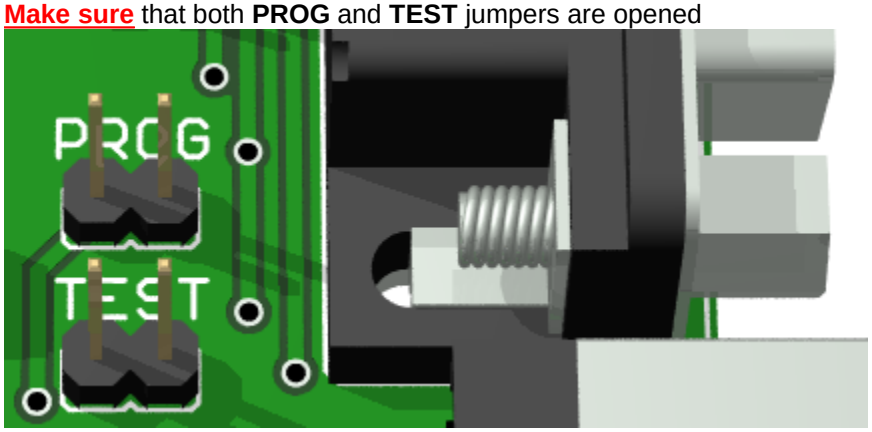
1.5.2.2.1 Connect ezLCD

1. Make sure that the Power Switch **is not** in **ON** position




2. Plug ezLCD-001 into the Evaluation Board, making sure that the connectors are not misaligned

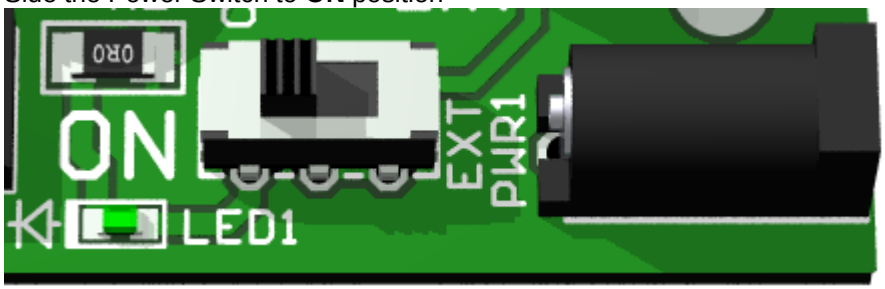
3. **Make sure** that both **PROG** and **TEST** jumpers are opened




4. Pug one end of the RS232 cable into any COM port of your Personal Computer and the other into ezLCD Evaluation Board

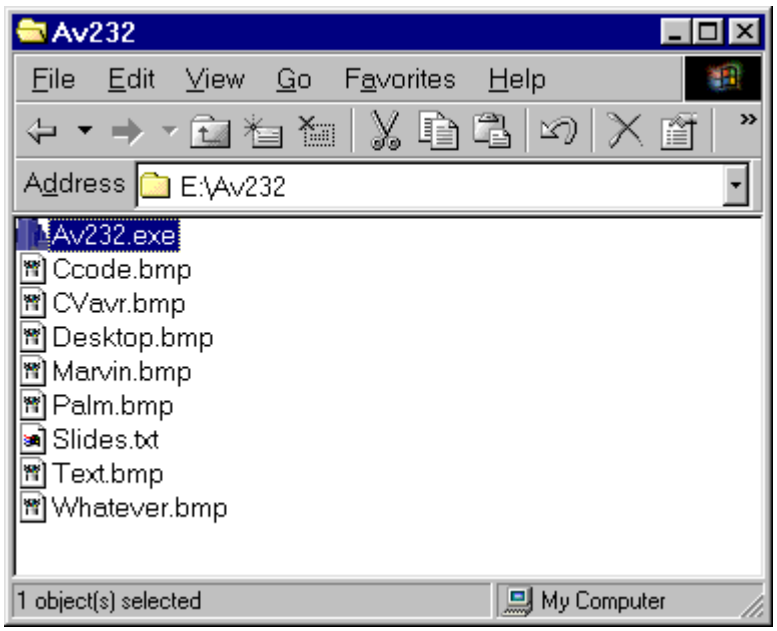
5. Cnnect External Power (6.5 to 12V DC or AC)

6. Side the Power Switch to **ON** position

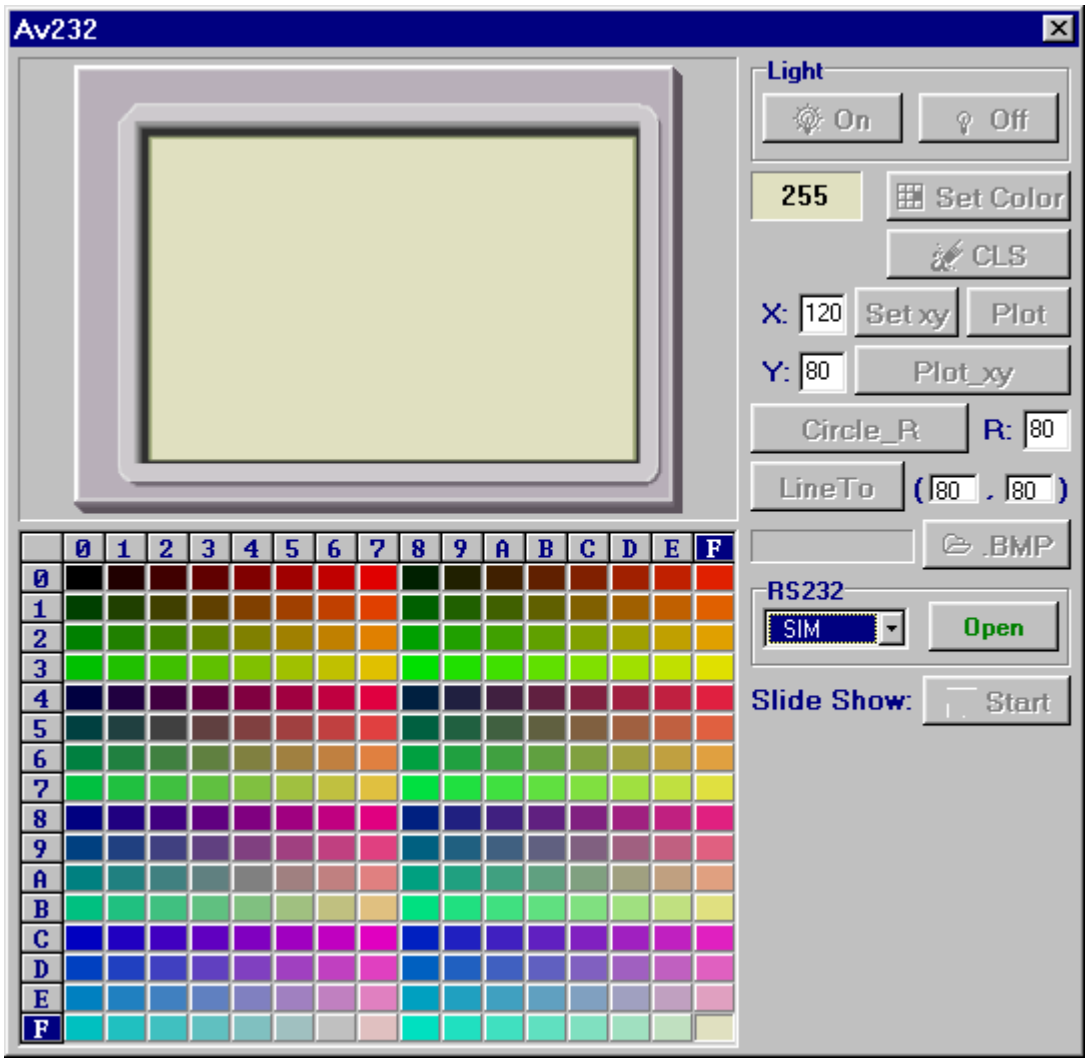


1.5.2.2.2 Run Av232 Utility

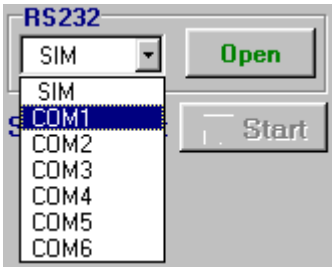
Start the Av232 Utility (file: Av232.exe).
The Av232 utility is in the Av232 directory on the ezLCD-001 CD.
Av232.exe may be started directly from CD, or from the directory on your hard drive (make sure that all the files are copied)




When Av232 starts, the following screen is displayed:

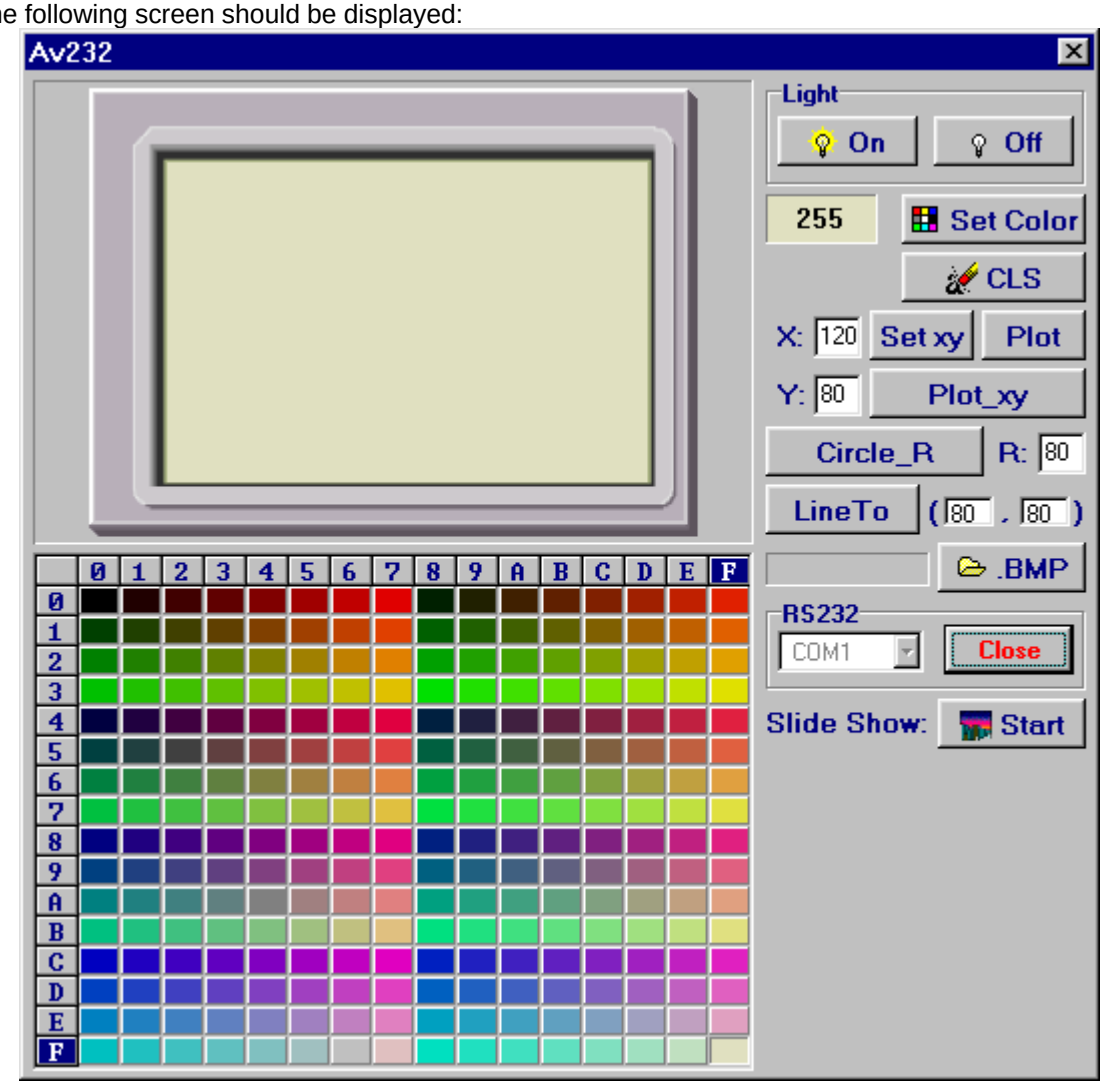

Av232
Light
On
Off
255
Set Color
CLS
X: 120
Set xy
Plot
Y: 80
Plot_xy
Circle_R
R: 80
LineTo
( 80 , 80 )
.BMP
RS232
SIM
Open
Slide Show:
Start
0 1 2 3 4 5 6 7 8 9 A B C D E F

1.5.2.2.3 Open PC COM Port




Select the COM port, which is connected to ezLCD and press the Open button.

The following screen should be displayed:



1.5.2.2.4 Send Commands

Before sending any commands to ezLCD make sure that you have:

- connected ezLCD
- run Av232 Utility
- opened PC COM Port

Now, using Av232 Control Panel you can draw various graphic on the ezLCD screen

**Example:** **SET_COLOR** and **CLS**

- Select color from the palette, for example: 


  Control Panel will display 15 the selected color

- Press Set Color

```
The following data will be sent to ezLCD:
24 hex      ( SET_COLOR command )
15 dec      ( color code )
```

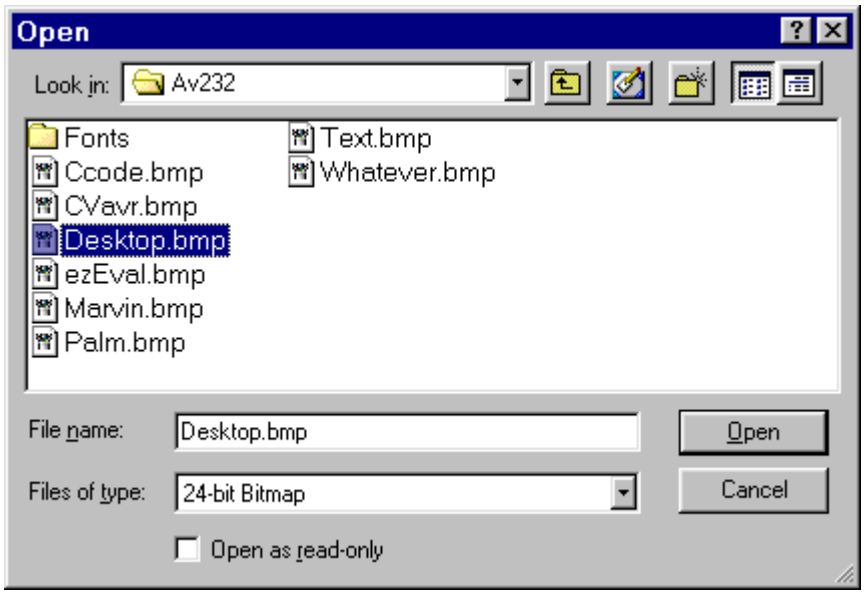
- Press CLS

```
The following data will be sent to ezLCD:
21 hex      ( CLS command )
```

The ezLCD screen should now be filled with the selected color

**Example:** **PICTURE**

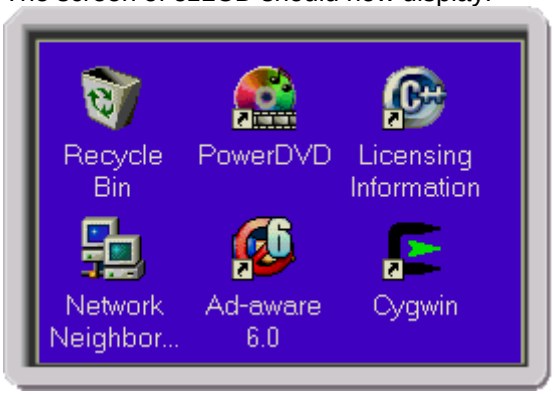
- Press .BMP
- Select file Desktop.bmp




and press Open

```
The following will be send to the ezLCD:
2A hex     ( PICTURE command )
xx hex     ( pixel x=0, y=159)
xx hex     ( pixel x=1, y=159)
           .
           .
xx hex     ( pixel x=239, y=159)
xx hex     ( pixel x=0, y=158)
           .
           .
xx hex     ( pixel x=238, y=0)
xx hex     ( pixel x=239, y=0)
The total number of 38401 (240 x 160 + 1) bytes
(including command) will be sent to the ezLCD.
```

The screen of ezLCD should now display:




**NOTE:** In order to be correctly processed by Av232, the picture has to be **24-bit .bmp** file with exact size of **240x160 pixels**.

### 1.5.3 How To

Upgrade Firmware

### 1.5.3.1 Upgrade Firmware

Firmware upgrade may be done through ezLCD-001's embedded RS232 port.

**Required additional equipment:**

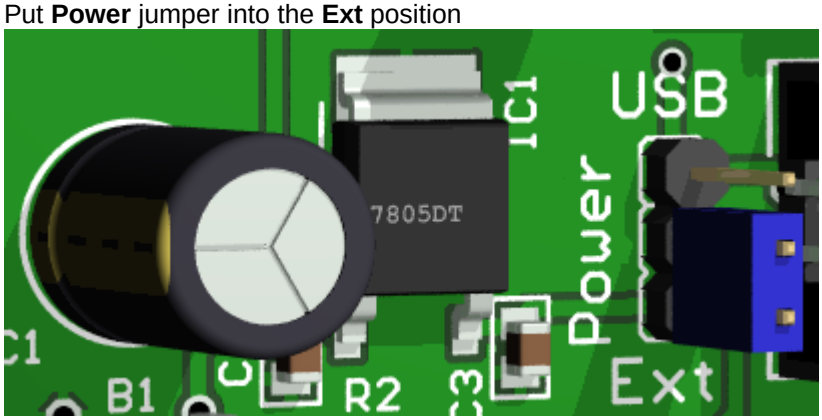
- Personal Computer running one of the following versions of Windows: 95, 98, Me, 2000 or XP
- 9 pin PC RS232 cable

## To load a new firmware into ezLCD-001:

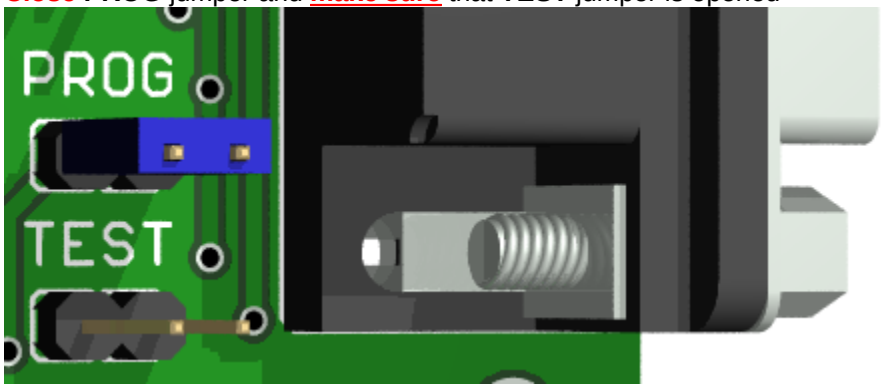
1. Make sure that the Power Switch **is not** in **ON** position


1. Plug ezLCD-001 into the Evaluation Board, making sure that the connectors are not misaligned

3. Put **Power** jumper into the **Ext** position



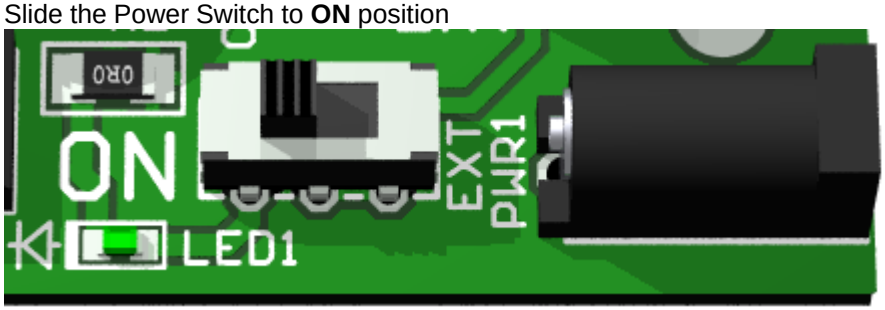

4. **Close PROG** jumper and **make sure** that **TEST** jumper is opened




5. Plug one end of the RS232 cable into any COM port of your Personal Computer and the other into ezLCD Evaluation Board

6. Connect External Power (6.5 to 12V DC or AC)

7. Slide the Power Switch to **ON** position




8. Run ezLCD_Rev_ooo.exe on your Personal Computer (ooo is the firmware revision, for example: 001).

**ezLCD_Rev_ooo.exe will:**

- extract the programming files into the temporary directory
- detect to which COM port is the ezLCD connected
- open a console window
- load a new firmware into the ezLCD

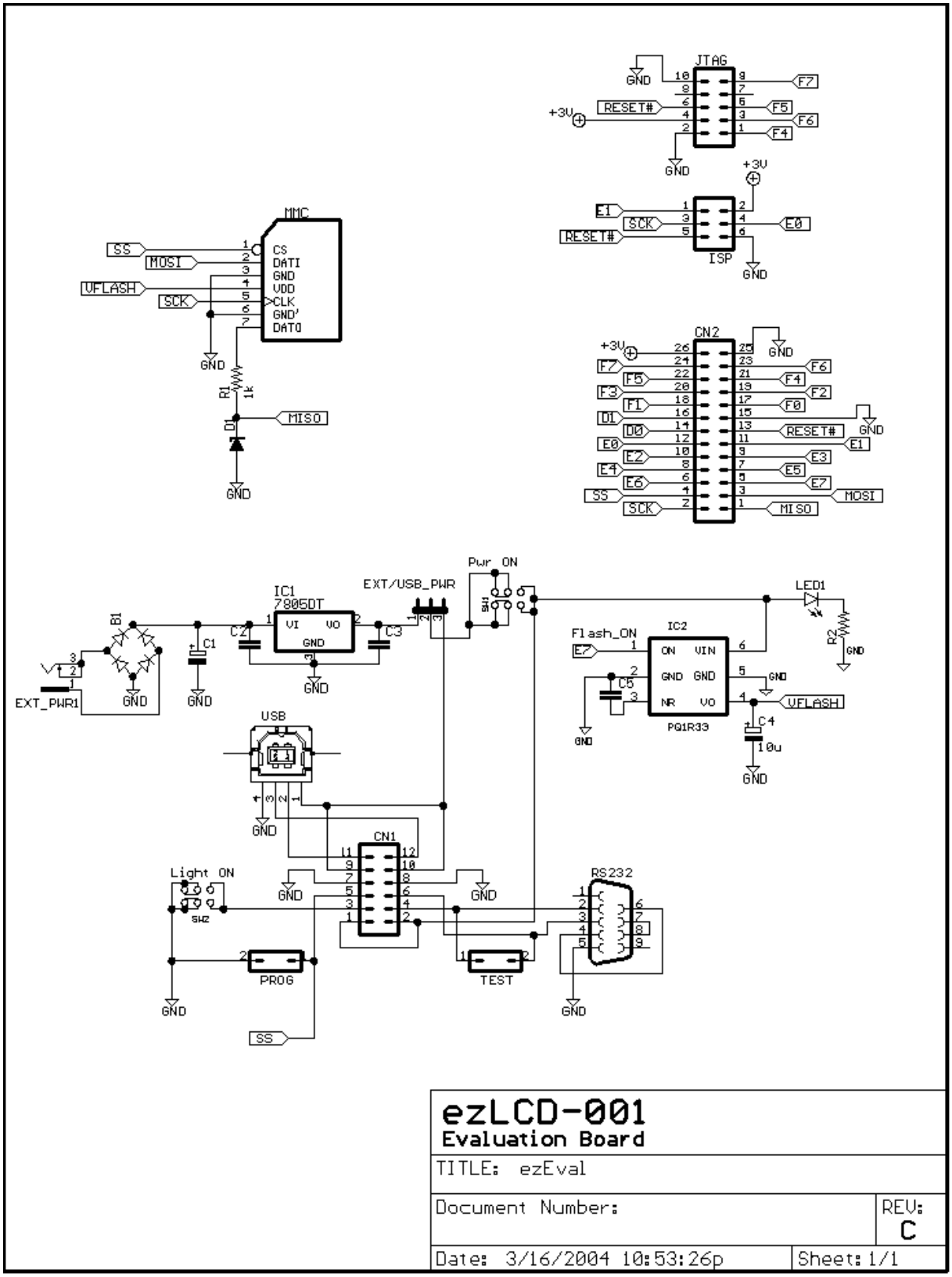
**Example of messages displayed by the console during successful firmware load:**

```
C:\tmp>stk500 -datmega128 -f0x9890

STK500 v 1.40 (C) 2000-2002 Atmel Corp.

Detecting.. AVRISP found on COM1:
Setting device parameters, serial programming mode ..OK
Entering programming mode.. OK
Programming fuses.. 0xFF, 0x9890 .. OK
Leaving programming mode.. OK

C:\tmp>stk500 -dATmega128 -ms -e -pf -ifv001.hex

STK500 v 1.40 (C) 2000-2002 Atmel Corp.

Detecting.. AVRISP found on COM1:
Reading FLASH input file.. OK
Setting device parameters, serial programming mode ..OK
Entering programming mode.. OK
Erasing device.. OK
Programming FLASH using block mode.. 100% OK
Leaving programming mode.. OK
```

### 1.5.4 Hardware Description

Schematics

### 1.5.4.1 Schematic



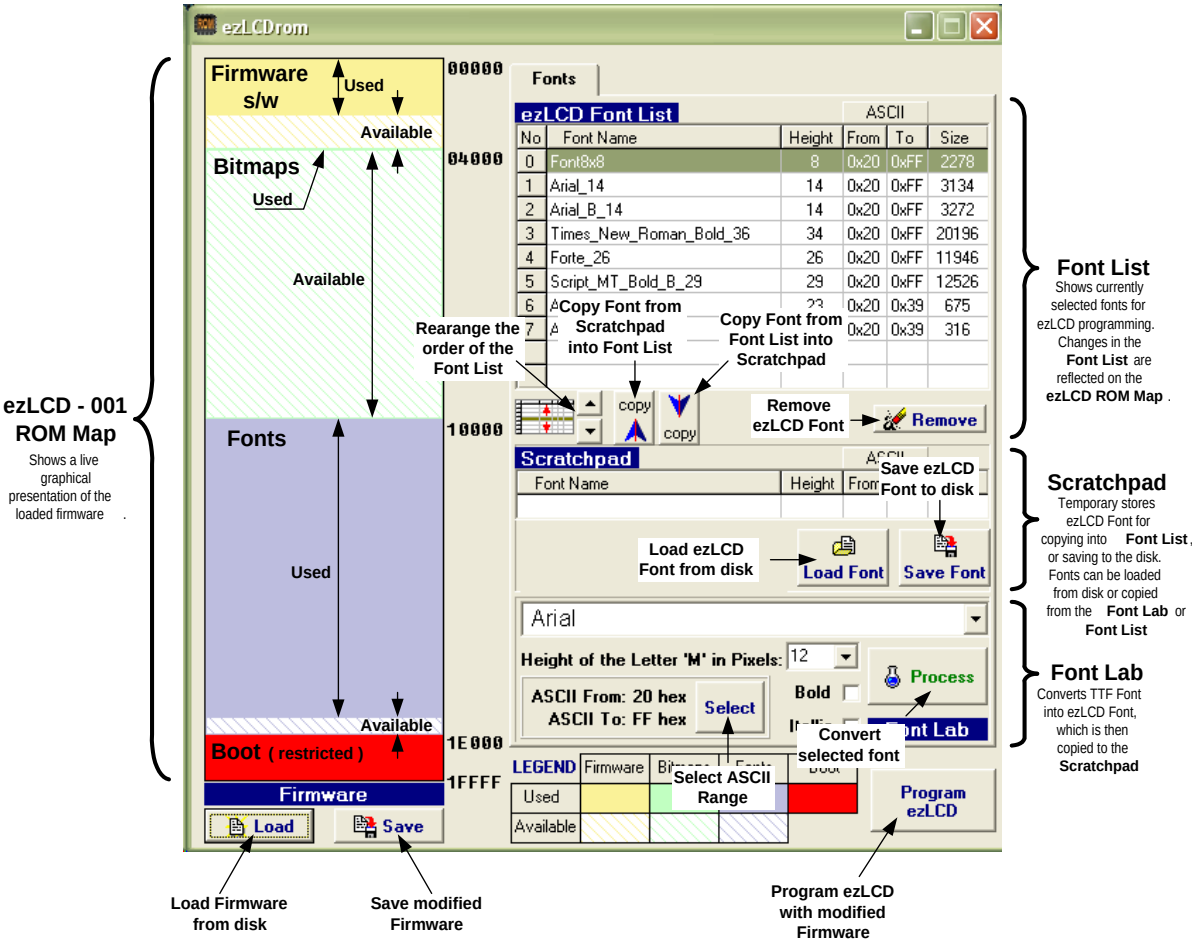
# 1.6 ezLCDrom Utility

## 1.6.1 Overview

The ezLCDrom is a utility, which allows the user to customize the Firmware of the ezLCD-001 by:

1. Adding and removing fonts
2. Adding and removing bitmaps or icons
3. Changing ezLCD settings like serial baudrate, pin assignments, etc.

**Note:** In this preliminary version only 1. is implemented

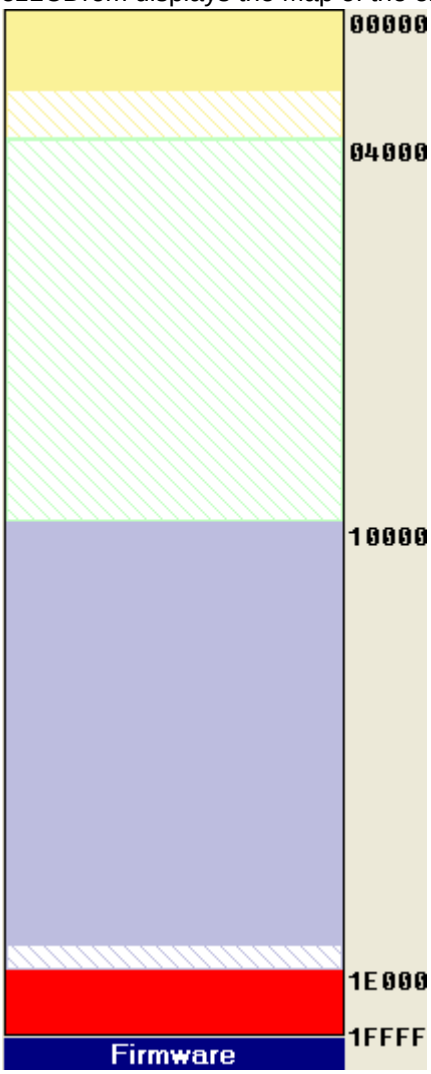


## 1.6.2 Loading Firmware file from disk

The ezLCD Firmware file is written in Intel Hex format and has an extension: `.hex`
To load the Firmware into ezLCDrom:


1. Click on Firmware **Load**
2. Select Firmware file

Upon loading the Firmware from disk,
ezLCDrom displays the Map of the ezLCD ROM:




**Where:**

**00000 - 03FFF** (16kB)

Space used by system and software.

Available space left for the system updates

**04000 - 0FFFF** (48kB)

Space used by bitmaps

Available space left for new bitmaps

**10000 - 1DFFF** (56kB)

Space used by fonts

Available space left for new fonts

**1E000 - 1FFFF** (8kB)

Boot space

### 1.6.3 Saving Firmware file

The ezLCD Firmware file will be written in Intel Hex format and should have an extension: `.hex`
To save the modified Firmware on disk:


1. Click on Firmware **Save**
2. Enter the filename and then press Save in the file save dialog

### 1.6.4 Programming ezLCD

To program the ezLCD with the modified Firmware:

Press **Program ezLCD** 


This will:

- open a console window
- load a new firmware into the ezLCD

**Example of messages displayed by the console during the successful programming:**

```
Detecting.. AVRISP found on COM1:
Reading FLASH input file.. OK
Setting device parameters, serial programming mode ..OK
Entering programming mode.. OK
Erasing device.. OK
Programming FLASH using block mode..  100% OK
Leaving programming mode.. OK
```

## 1.6.5 How To

### 1.6.5.1 Add a new font to the ezLCD

**To create and add a new font to the ezLCD:**

1. Load the ezLCD Firmware from the disk, by pressing the Load button.
2. Specify font parameters in the **Font Lab**
3. Select the ASCII Range of the font by pressing Select button.
4. Press Process to convert the selected TTF font into ezLCD font. Upon successful conversion, the new font will be displayed on the **Scratchpad**.
5. You can save the font by pressing Save Font on the **Scratchpad**.
6. Rearrange the **ezlcd Font List**, if necessary.
7. Press copy to add the **Scratchpad** font to the **ezLCD Font List**.
8. Rearrange the **ezlcd Font List**, if necessary.
9. You can save the new ezLCD Firmware by pressing the Save button.
10. Program the ezLCD-001 with the new Firmware.

### 1.6.5.2 Rearrange the fonts

**To rearrange fonts on the ezLCD Font List:**

1. Make sure that the ezLCD Firmware is loaded
2. You can:
   - Rearrange the order of fonts by pressing one of  buttons.
   - Remove the font from the list by pressing  button.

### 1.6.5.3 Save a font from the ezLCD Font List

**To save a font from the ezLCD Font List:**

1. Make sure that the ezLCD Firmware is loaded
2. Select the font for saving from the **ezLCD Font List**.
3. Press 

to copy a font from the **ezLCD Font List** into the **Scratchpad**.
   **Caution:** This will replace the current Scratchpad font.
4. Save the font by pressing 

on the **Scratchpad**

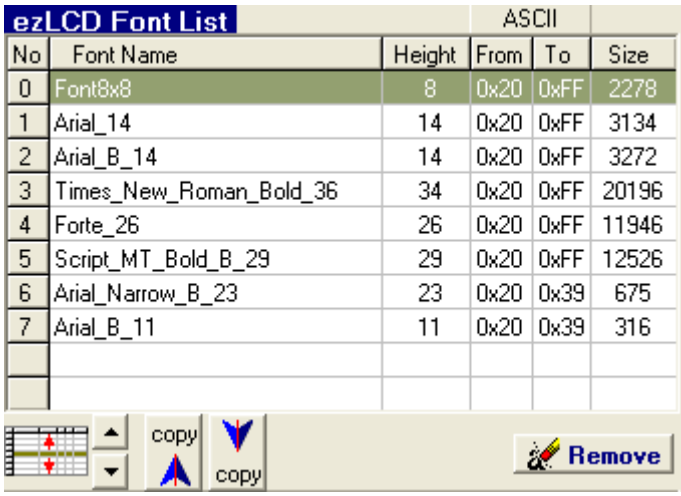
## 1.6.6 Fonts

### 1.6.6.1 ezLCD Font List

The **ezLCD Font List** is used to perform the following operations:

- Adding a new fonts to the Firmware
- Removing fonts from the Firmware.
- Rearranging the order of the Firmware fonts.

The **ezLCD Font List** shows the fonts of the loaded from the disk Firmware:




| No | Font Name | Height | From | To | Size |
|---|---|---|---|---|---|
| 0 | Font8x8 | 8 | 0x20 | 0xFF | 2278 |
| 1 | Arial_14 | 14 | 0x20 | 0xFF | 3134 |
| 2 | Arial_B_14 | 14 | 0x20 | 0xFF | 3272 |
| 3 | Times_New_Roman_Bold_36 | 34 | 0x20 | 0xFF | 20196 |
| 4 | Forte_26 | 26 | 0x20 | 0xFF | 11946 |
| 5 | Script_MT_Bold_B_29 | 29 | 0x20 | 0xFF | 12526 |
| 6 | Arial_Narrow_B_23 | 23 | 0x20 | 0x39 | 675 |
| 7 | Arial_B_11 | 11 | 0x20 | 0x39 | 316 |

**Where:**

No - Font Number (used in the command SELECT_FONT)

Font Name - Name of the Font (this is obvious)

Height - Distance (in ezLCD pixels) from the lowest point to the highest point of the font.

For example: 


ASCII From and ASCII To - Limits of the ASCII Range. Letters outside the ASCII Range will not be drawn by the ezLCD. Minimizing the ASCII Range saves ezLCD ROM space.

Size - Number of bytes occupied by font

 - Rearrange the order of the fonts, by moving the selected font up or down

- Add the Scratchpad font to the end of the list.

- Copy the selected font to the Scratchpad, where it can be saved to the disk.

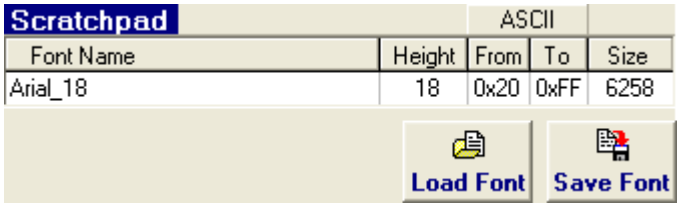
- Remove (delete, erase) the selected font from the list

### 1.6.6.2 Scratchpad

**Scratchpad** is used as an interfacing buffer between the disk, the **ezLCD Font List** and the **Font Lab**

**Scratchpad Output:**
- Adding the **Scratchpad** font to the **ezLCD Font List**
- Saving the **Scratchpad** font on the disk

**Scratchpad Input:**
- **Font Lab** puts newly generated font on the **Scratchpad**
- Adding the **Scratchpad** font to the **Font List**
- Loading an ezLCD font from the disk




**Where:**

Font Name - Name of the Scratchpad font (this is obvious)

Height - Distance (in ezLCD pixels) from the lowest point to the highest point of the font.

For example: 


ASCII From and ASCII To - Limits of the ASCII Range. Letters outside the ASCII Range will not be drawn by the ezLCD. Minimizing the ASCII Range saves ezLCD ROM space.

Size - Number of bytes occupied by font

- Load a font from the disk

- Save the Scratchpad font on the disk

**ezLCD Font List Scratchpad Operations:**

- Add the Scratchpad font to the end of the ezLCD Font List

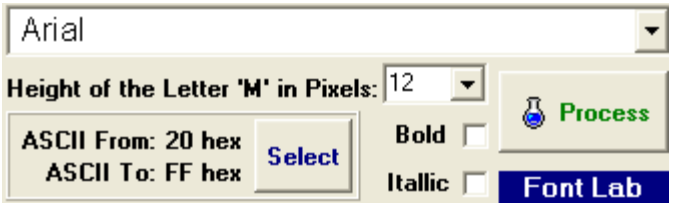
- Copy the selected font to the Scratchpad, where it can be saved to the disk.

**Font Lab Scratchpad Operations:**

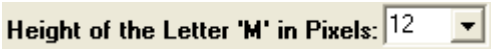
- Generate a new font and put it on the Scratchpad

### 1.6.6.3 Font Lab

Font Lab is used to convert TTF fonts into ezLCD fonts.
Created font is moved to the **Scratchpad**.




**Where:**




Letter 'M' is used as a common reference to specify the font height.
Usually the font height will be bigger then letter M, since it is defined as the distance (in ezLCD pixels) from the lowest point to the highest point of the font, as t is shown on the example below.


However, for example, if the particular font contains only capital letters (ASCII Range: 41 to 5A hex), it's height will be equal to the height of the letter 'M'.


This panel is used to specify the ASCII range of the font.
Letters outside the ASCII Range will not be drawn by the ezLCD. Minimizing the ASCII Range saves ezLCD ROM space.

ASCII From: - Displays the bottom of the ASCII Range

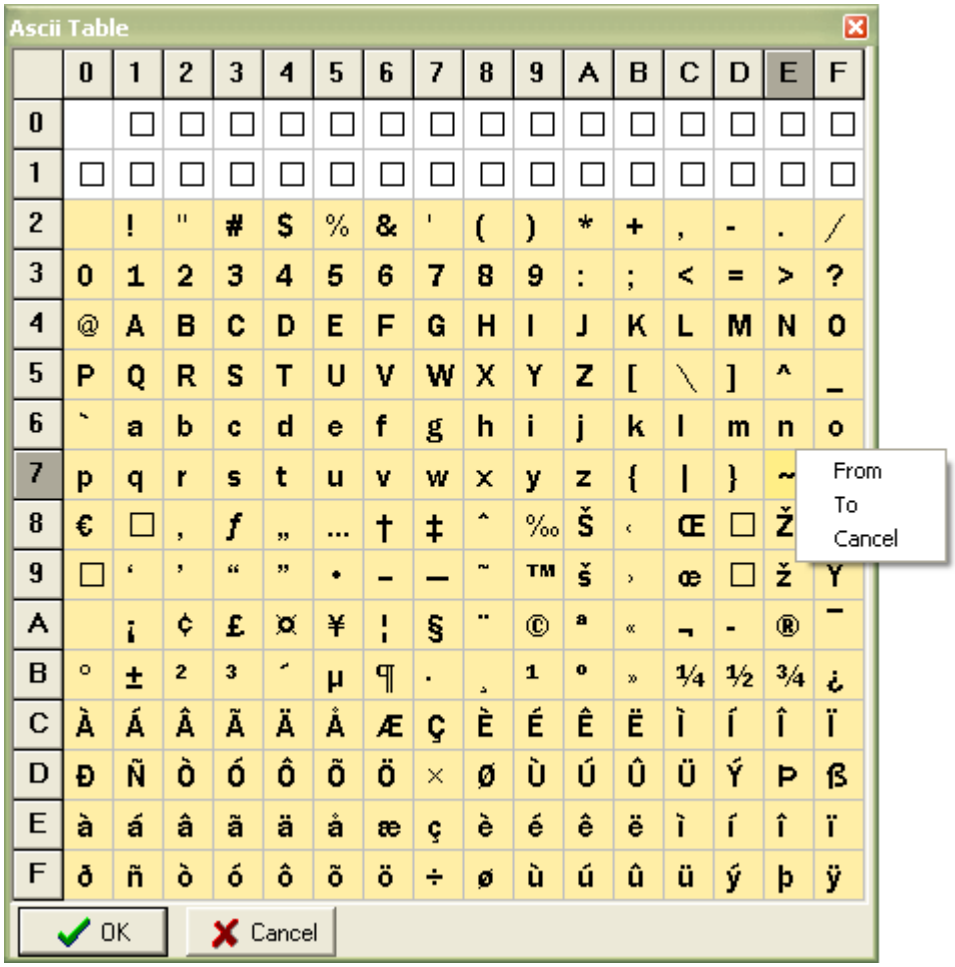
ASCII To: - Displays the top of the ASCII Range


- Selects the ASCII Range.
Described in Selecting ASCII Range


This button is used to start converting a TTF font into the ezLCD Font. Created font is moved to the **Scratchpad**.

1.6.6.3.1 Selecting ASCII Range


When the **Select** button is pressed, the following form pop-ups:




The above form displays the ASCII Table of the selected font.

The currently selected ASCII Range has a background color:
The limits of the ASCII Range may be modified by clicking on the table cell.

In case of doubt, ezLCDrom will display the following pop-up menu: From To Cancel

Press OK to confirm the new ASCII Range,

or Cancel to return without any modifications.

## 1.7 Document History

| DATE | WHO | WHAT |
|---|---|---|
| 17-MAR-2004 | Michal | Initial Creation |
| 21-MAR-2004 | Michal | **Added:**<br>• Quick Start Chapter: Av232 Utility<br>• ezLCD Board Dimensions<br>**Modified:**<br>• Quick Start<br>• Hardware Description |
| 20-AUG-2004 | Michal | Started work on a final version (not ready yet) |

# Index

# - P -



# - R -



# - S -



# - T -



# - U -



# - V -



# - W -