2 ezLCD-5x Command Reference

General Commands:

- CLS
- SET_COLORH
- SET_BG_COLORH
- SET_COLOR_RGB
- SET_BG_COLOR_RGB

Drawing Position Commands:

- RESTORE_POSITION
- SAVE_POSITION
- SET_XH
- SET_XHY
- SET_Y

Layer Commands:

- SET_COLORKEY
- SET_DRAWING_LAYER
- SET_LAYER_VISIBLITY

Backlight Commands:

- LIGHT_BRIGHT
- LIGHT_ON
- LIGHT_OFF

Point Commands:

- PLOT
- PLOT_XHY

Line Commands:

- H_LINEH
- V_LINE
- LINE_TO_XHY

Shape Commands:

- ARCH
- PIEH
- CIRCLE_RH
- CIRCLE_FH_FILL
- BOXH
- BOXH_FILL

Bitmap Commands:

Cache Management Commands:

Text & Font Commands:

Touch Screen Commands:

SD Flash Card Commands:

GPIO Commands:

ezNow Buzzer Commands:

System Commands:

Legacy Commands:

- **ARC_BOX**
- **BOX_FILL**
- **CIRCLE_R**
- **CIRCLE_R_FILL**
- **LINE_TO_XY**
- **PLOT_XY**
- **PUT_BITMAP**
- **SET_BG_COLOR**
- **SET_COLOR**
- **SET_XY**

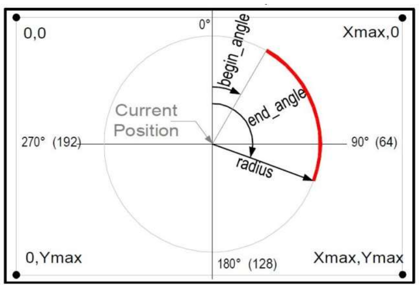
2.1 ARCH

Description: Draws an ARC in “Current Color”, with the center at “Current Position”, starting on “Begin Angle” and ending on “End Angle”.

Code: **8F**HEX, **143**DEC

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th></th></tr>
<tr><td colspan="8"><b>ARCH</b></td><td>Byte 0 (<b>Command</b>)</td></tr>
<tr><td colspan="8"><b>radius_MSB</b></td><td>Byte 1 (<b>Radius</b> MSB)</td></tr>
<tr><td colspan="8"><b>radius_LSB</b></td><td>Byte 2 (<b>Radius</b> LSB)</td></tr>
<tr><td colspan="8"><b>begin_angle_MSB</b></td><td>Byte 3 (<b>Arc Begin Angle</b> MSB)</td></tr>
<tr><td colspan="8"><b>begin_angle_LSB</b></td><td>Byte 4 (<b>Arc Begin Angle</b> LSB)</td></tr>
<tr><td colspan="8"><b>end_angle_MSB</b></td><td>Byte 5 (<b>Arc End Angle</b> MSB)</td></tr>
<tr><td colspan="8"><b>end_angle_LSB</b></td><td>Byte 6 (<b>Arc End Angle</b> LSB)</td></tr>
</table>

The angle is oriented clockwise with the zero positioned at the top of the screen, shown in the figure below.



See Also: PIEH, SET_XHY, SET_COLORH, CIRCLE_RH

Angle Coding: The full angle (360°) is equal to **4000**HEX (**16384**DEC).

Degrees to ARC Angle Units Equation: $Angle_{ARC} = Angle_{DEG} \cdot \frac{2048}{45}$

Angle Conversion Example:

2048$_{\text{DEC}}$ = **800**$_{\text{HEX}}$ = **45°**
4096$_{\text{DEC}}$ = **1000**$_{\text{HEX}}$ = **90°**
8192$_{\text{DEC}}$ = **2000**$_{\text{HEX}}$ = **180°**
12288$_{\text{DEC}}$ = **3000**$_{\text{HEX}}$ = **270°**
16384$_{\text{DEC}}$ = **4000**$_{\text{HEX}}$ = **360°** = **0°**

Drawing Example:

The following command sequence will draw a green arc from 45 to 225 degrees with the center positioned at (160, 117) and a radius of 80.

$$225 \cdot \frac{2048}{45} = 10240 \rightarrow 10240_{DEC} = 2800_{HEX}$$

SET_COLORH	84	HEX	
GREEN_LSB	11100000	BIN	
GREEN_MSB	00000111	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
0	0	DEC	(**Y** MSB)
117	117	DEC	(**Y** LSB)
ARCH	**8F**	HEX	
0	**0**	DEC	(**Radius** MSB)
80	**80**	DEC	(**Radius** LSB)
08	**08**	HEX	(**Begin_Angle** MSB)
00	**00**	HEX	(**Begin_Angle** LSB)
28	**28**	HEX	(**End_Angle** MSB)
00	**00**	HEX	(**End_Angle** LSB)

2.2 BOXH

NOTE: Legacy deprecated command. Recommended Replacement: **BOXHH**

Description: Draws a rectangle.

Code: **A2**HEX, **162**DEC

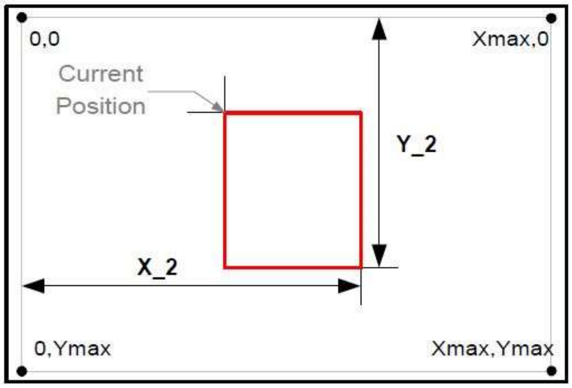
Command Format--Standard Mode:

7	6	5	4	3	2	1	0	
BOXH								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X_2** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X_2** LSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 3: **Y_2**

Command Format--Extended Mode:

7	6	5	4	3	2	1	0	
BOXH								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X_2** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X_2** LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 3: **Y_2** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y_2** LSB

The box is drawn starting at the **X** and **Y** co-ordinates of the "Current Position" until the **X_2** and **Y_2** co-ordinates, shown in the figure below.



See Also: SET_XHY, BOXH_FILL

Example:

The following sequence will draw a red rectangle with the top left corner positioned at (95, 10) and the bottom right corner at (180, 120).

SET_COLORH	84	HEX	
RED_MSB	00000000	BIN	
RED_LSB	11111000	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
95	95	DEC	(**X** LSB)
10	10	DEC	(**y**)
BOXH	**A2**	HEX	
180	**0**	DEC	(**X_2** MSB)
180	**180**	DEC	(**X_2** LSB)
120	**120**	DEC	(**Y_2**)

2.3 BOXH_FILL

NOTE: Legacy deprecated command. Recommended Replacement: **BOXHH_FILL**

Description: Draws a rectangle filled with “Current Color.”

Code: **A3**$_{HEX}$, **163**$_{DEC}$

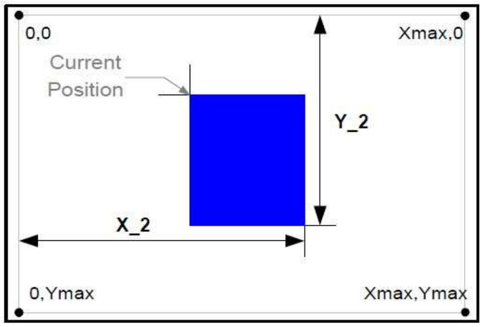
Command Format--Standard Mode:

7	6	5	4	3	2	1	0	
BOXH_FILL								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X_2** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X_2** LSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 3: **Y_2**

Command Format--Extended Mode:

7	6	5	4	3	2	1	0	
BOXH_FILL								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X_2** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X_2** LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 3: **Y_2** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y_2** LSB

The box is drawn starting at the **X** and **Y** co-ordinates of the “Current Position” until the **X_2** and **Y_2** co-ordinates, shown in the figure below.



See Also: SET_XHY, BOXH

Example:
The following sequence will draw a blue filled rectangle, with the top left corner positioned at (95, 10) and the bottom right corner at (180, 120).

SET_COLORH	84	HEX	
BLUE_LSB	00011111	BIN	
BLUE_MSB	00000000	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
95	95	DEC	(**X** LSB)
10	10	DEC	(**Y**)
BOXH_FILL	**A3**	HEX	
180	**0**	DEC	(**X_2** MSB)
180	**180**	DEC	(**X_2** LSB)
120	**120**	DEC	(**Y_2**)

2.4 BOXHH

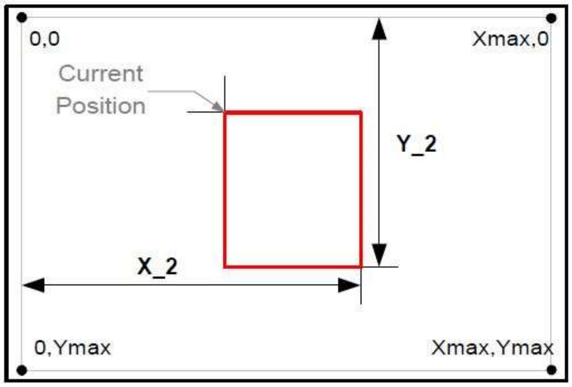
Description: Draws a rectangle.

Code: **A4**$_{\text{HEX}}$, **164**$_{\text{DEC}}$

Command Format:

7	6	5	4	3	2	1	0	
BOXHH								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X_2** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X_2** LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 3: **Y_2** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y_2** LSB

The box is drawn starting at the **X** and **Y** co-ordinates of the "Current Position" until the **X_2** and **Y_2** co-ordinates, shown in the figure below.



See Also: SET_XHYH, BOXHH_FILL

Example:

The following sequence will draw a red rectangle with the top left corner positioned at (95, 10) and the bottom right corner at (180, 120).

SET_COLOR_RGB	31	HEX	
RED	FF	HEX	
GREEN	0	HEX	
BLUE	00	HEX	
SET_XHYH	33	HEX	
0	0	DEC	(**X** MSB)
95	95	DEC	(**X** LSB)
0	0	DEC	(**Y** MSB)
10	10	DEC	(**Y**)
BOXHH	**A4**	HEX	
0	**0**	DEC	(**X_2** MSB)
180	**180**	DEC	(**X_2** LSB)
0	**0**	DEC	(**Y_2** MSB)
120	**120**	DEC	(**Y_2** LSB)

2.5 BOXHH_FILL

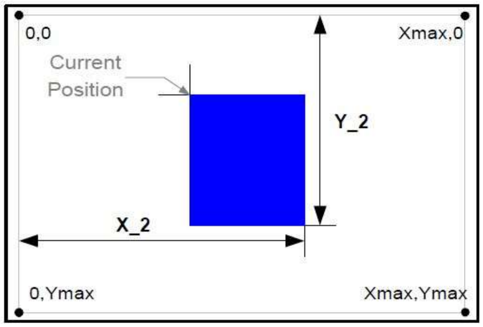
Description: Draws a rectangle filled with “Current Color.”

Code: **A5**HEX, **165**DEC

Command Format:

7	6	5	4	3	2	1	0	
BOXHH_FILL								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X_2** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X_2** LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 3: **Y_2** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y_2** LSB

The box is drawn starting at the **X** and **Y** co-ordinates of the “Current Position” until the **X_2** and **Y_2** co-ordinates, shown in the figure below.



See Also: **SET_XHYH**, **BOXHH**

Example:
The following sequence will draw a blue filled rectangle, with the top left corner positioned at (95, 10) and the bottom right corner at (180, 120).

SET_COLOR_RGB	31	HEX	
RED	0	HEX	
GREEN	0	HEX	
BLUE	FF	HEX	
SET_XHYH	33	HEX	
0	0	DEC	(**X** MSB)
95	95	DEC	(**X** LSB)
0	0	DEC	(**Y** MSB)
10	10	DEC	(**Y**)
BOXHH_FILL	**A5**	HEX	
0	**0**	DEC	(**X_2** MSB)
180	**180**	DEC	(**X_2** LSB)
0	**0**	DEC	(**Y_2** MSB)
120	**120**	DEC	(**Y_2** LSB)

2.6 BUTTON_DEF

NOTE: Legacy deprecated command. Recommended Replacement: BUTTON_DEF_EXT

Description: Defines and draws a touch button

Code: **B0**$_{HEX}$, **176**$_{DEC}$

Command Format--Standard Mode:

7	6	5	4	3	2	1	0	
BUTTON_DEF								Byte 0: **Command**
button_no								Byte 1: **Button No** (0 to 63)
state								Byte 2: **Initial State** (**1:** Up, **2:** Down, **3:** Disabled, **4:** Non-Visible)
button_up_image								Byte 3: Image No for Button Up (255 = none)
button_down_image								Byte 4: Image No for button Down (255 = none)
button_disabled_image								Byte 5: Image No for button Disabled (255 = none)
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 6: Button upper-left corner X-Coordinate MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 7: Button upper-left corner X-Coordinate LSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 8: Button upper-left corner Y-Coordinate
touch_zone_width								Byte 9: **Touch Zone Width**
touch_zone_height								Byte 10: **Touch Zone Height**

Command Format--Extended Mode:

7	6	5	4	3	2	1	0	
BUTTON_DEF								Byte 0: **Command**
button_no								Byte 1: **Button No** (0 to 63)
state								Byte 2: **Initial State** (**1:** Up, **2:** Down, **3:** Disabled, **4:** Non-Visible)
button_up_image								Byte 3: Image No for Button Up (255 = none)
button_down_image								Byte 4: Image No for button Down (255 = none)
button_disabled_image								Byte 5: Image No for button Disabled (255 = none)
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 6: Button upper-left corner X-Coordinate MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 7: Button upper-left corner X-Coordinate LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 8: Button upper-left corner Y-Coordinate MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 9: Button upper-left corner Y-Coordinate LSB
touch_zone_width								Byte 10: **Touch Zone Width**
touch_zone_height								Byte 11: **Touch Zone Height**

Images: The images used for buttons are stored on the SDCard and defined/numbered in *config.txt* or via the settings file created by the ezLCD Config. Utility

About the Touch Zone:

- Touch Zone is the active touch response area of the button. It is specified by **Width** (Byte 9) and **Height** (Byte 10).
- If the Button Up Icon is **defined** (Byte 3 is not 255), the Touch Zone is centered on it.
- If the Button Up Icon is **none** (Byte 3 = 255), the position of the upper-left corner of the Touch Zone is specified by **X** (Bytes: 6 and 7) and **Y** (Byte 8).

Both cases are shown on the figures below:

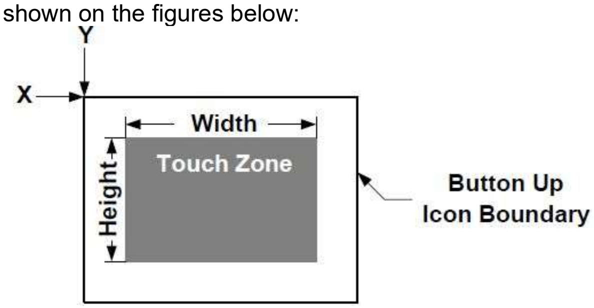


Figure 2 Button Up Icon is none (Byte 3 = 255)

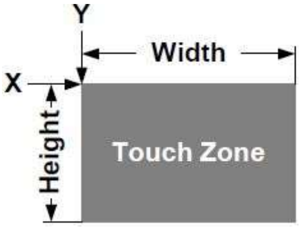


Figure 1. Button Up Icon is defined (Byte 3 is not 255)

About the Button State:

When a button is pressed or released, Button State is not automatically updated to reflect the button press. It is up to the enduser application to send the appropriate BUTTON_STATE commands to update the button state.

See Also: BUTTON_STATE, BUTTONS_ALL_UP, BUTTONS_DELETE_ALL, TOUCH_PROTOCOL

Note: Before using this command, please read the following chapters:

- Touch Screen
- ezButton
- cuButton

Example:

The following sequence will define the "Button 4" with the following bitmaps:

- **Button Up Icon from SD Card: 8**
- **Button Down Icon from SD Card: 9**
- **No Icon for Button Disabled state**

The button will be positioned at X = 260 and Y = 170. It's Touch Zone will have the width of 40 and the height of 30. The button will be initially drawn using Button Up icon.

BUTTON_DEF	**B0**	HEX	(**Command**)
4	**4**	DEC	(**Button No**)
1	**1**	DEC	(**Initial State:** Button Up)
8	**8**	DEC	(Button Up Icon No. in the serial flash)
9	**9**	DEC	(Button Down Icon No. in the serial flash)
255	**255**	DEC	(No Icon for Button Disabled)
1	**1**	DEC	(Upper-left corner X MSB)
4	**4**	DEC	(Upper-left corner X LSB)
170	**170**	DEC	(Upper-left corner Y)
40	**40**	DEC	(**Width** of the Touch Zone)
30	**30**	DEC	(**Height** of the Touch Zone)

2.7 BUTTON_DEF_LONG

NOTE: Legacy deprecated command. Recommended Replacement: **BUTTON_DEF_EXT**

Description: Defines and draws a touch button

Code: **B5**HEX, **181**DEC

Command Format:

7	6	5	4	3	2	1	0	
BUTTON_DEF_LONG								Byte 0: **Command**
button_no								Byte 1: **Button No** (0 to 63)
state								Byte 2: **Initial State** (**1:** Up, **2:** Down, **3:** Disabled, **4:** Non-Visible)
button_up_image_msb								Byte 3-4: Image No for Button Up (0xFFFF = none)
button_up_image_lsb								
button_down_image_msb								Byte 5-6: Image No for button Down (0xFFFF = none)
button_down_image_lsb								
button_disabled_image_msb								Byte 7-8: Image No for button Disabled (255 = none)
button_disabled_image_lsb								
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 9: Button upper-left corner X-Coordinate MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 10: Button upper-left corner X-Coordinate LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 11: Button upper-left corner Y-Coordinate MSB

y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 12: Button upper-left corner Y-Coordinate LSB
touch_zone_width								Byte 13: **Touch Zone Width**
touch_zone_height								Byte 14: **Touch Zone Height**

Images: The images used for buttons are stored on the SDCard and defined/numbered in *config.txt* or via the settings file created by the ezLCD Config. Utility

About the Touch Zone:

- Touch Zone is the active touch response area of the button. It is specified by **Width** (Byte 13) and **Height** (Byte 14).
- If the Button Up Icon is **defined** (Byte 3-4 is not 0xFFFF), the Touch Zone is centered on it.
- If the Button Up Icon is **none** (Byte 3-4 = 0xFFFF), the position of the upper-left corner of the Touch Zone is specified by **X** (Bytes: 9 and 10) and **Y** (Bytes 11 and 12).

Both cases are shown on the figures below:

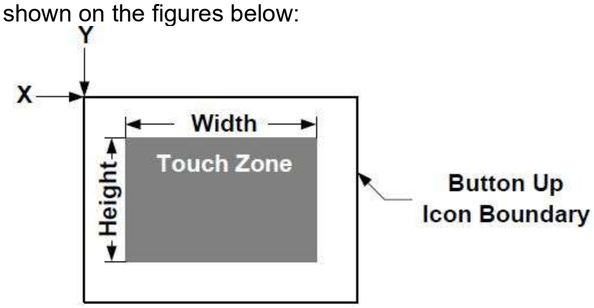


Figure 4 Button Up Icon is none (Byte 3 = 255)

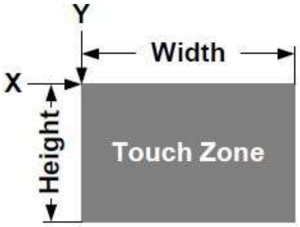


Figure 3. Button Up Icon is defined (Byte 3 is not 255)

About the Button State:

When a button is pressed or released, Button State is not automatically updated to reflect the button press. It is up to the enduser application to send the appropriate BUTTON_STATE commands to update the button state.

See Also: BUTTON_STATE, BUTTONS_ALL_UP, BUTTONS_DELETE_ALL, TOUCH_PROTOCOL

Note: Before using this command, please read the following chapters:

- Touch Screen
- ezButton
- cuButton

Example:

The following sequence will define the "Button 4" with the following bitmaps:

- **Button Up Icon from SD Card: 8**
- **Button Down Icon from SD Card: 9**
- **No Icon for Button Disabled state**

The button will be positioned at X = 260 and Y = 170. It's Touch Zone will have the width of 40 and the height of 30. The button will be initially drawn using Button Up icon.

BUTTON_DEF_LONG	**B5**	HEX	(**Command**)
4	**4**	DEC	(**Button No**)
1	**1**	DEC	(**Initial State:** Button Up)
0	**0**	DEC	(Button Up Icon No. MSB)
8	**8**	DEC	(Button Up Icon No. LSB)
0	**0**	DEC	(Button Down Icon No. MSB)
9	**9**	DEC	(Button Down Icon No. LSB)
255	**255**	DEC	(Button Disabled Icon No MSB)
255	**255**	DEC	(Button Disabled Icon No LSB)
1	**1**	DEC	(Upper-left corner X MSB)
4	**4**	DEC	(Upper-left corner X LSB)
0	**0**	DEC	(Upper-left corner Y MSB)
170	**170**	DEC	(Upper-left corner Y LSB)
40	**40**	DEC	(**Width** of the Touch Zone)
30	**30**	DEC	(**Height** of the Touch Zone)

2.8 BUTTON_DEF_EXT

Description: Defines and draws a touch button

Code: **B7**HEX, **183**DEC

Command Format:

7	6	5	4	3	2	1	0	
BUTTON_DEF_EXT								Byte 0: **Command**
button_no								Byte 1: **Button No** (0 to 63)
state								Byte 2: **Initial State** (**1:** Up, **2:** Down, **3:** Disabled, **4:** Non-Visible)
button_up_image_msb								Byte 3-4: Image No for Button Up (0xFFFF = none)
button_up_image_lsb								
button_down_image_msb								Byte 5-6: Image No for button Down (0xFFFF = none)
button_down_image_lsb								
button_disabled_image_msb								Byte 7-8: Image No for button Disabled (255 = none)
button_disabled_image_lsb								
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 9: Button upper-left corner X-Coordinate MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 10: Button upper-left corner X-Coordinate LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 11: Button upper-left corner Y-Coordinate MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 12: Button upper-left corner Y-Coordinate LSB

w15	**w14**	**w13**	**w12**	**w11**	**w10**	**w9**	**w8**	Byte 13: **Touch Zone Width MSB**
w7	**w6**	**w5**	**w4**	**w3**	**w2**	**w1**	**w0**	Byte 14: **Touch Zone Width LSB**
h15	**h14**	**h13**	**h12**	**h11**	**h10**	**h9**	**h8**	Byte 15: **Touch Zone Height MSB**
h7	**h6**	**h5**	**h4**	**h3**	**h2**	**h1**	**h0**	Byte 16: **Touch Zone Width LSB**

Images: The images used for buttons are stored on the SDCard and defined/numbered in *config.txt* or via the settings file created by the ezLCD Config. Utility

About the Touch Zone:

- Touch Zone is the active touch response area of the button. It is specified by **Width** (Byte 13) and **Height** (Byte 14).
- If the Button Up Icon is **defined** (Byte 3-4 is not 0xFFFF), the Touch Zone is centered on it.
- If the Button Up Icon is **none** (Byte 3-4 = 0xFFFF), the position of the upper-left corner of the Touch Zone is specified by **X** (Bytes: 9 and 10) and **Y** (Bytes 11 and 12).

Both cases are shown on the figures below:

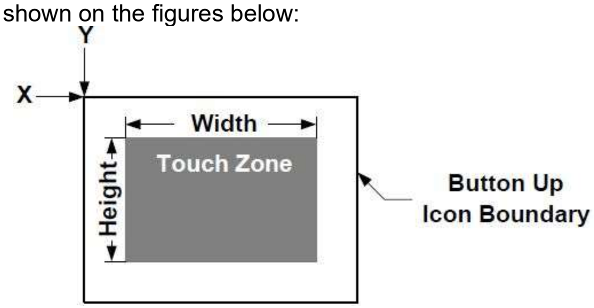


Figure 6 Button Up Icon is none (Byte 3 = 255)

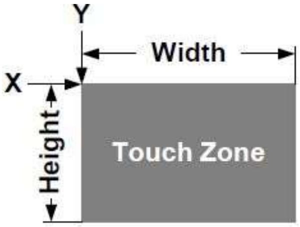


Figure 5. Button Up Icon is defined (Byte 3 is not 255)

About the Button State:

When a button is pressed or released, Button State is not automatically updated to reflect the button press. It is up to the enduser application to send the appropriate BUTTON_STATE commands to update the button state.

See Also: BUTTON_STATE, BUTTONS_ALL_UP, BUTTONS_DELETE_ALL, TOUCH_PROTOCOL

Note: Before using this command, please read the following chapters:

- Touch Screen
- ezButton
- cuButton

Example:

The following sequence will define the "Button 4" with the following bitmaps:

- **Button Up Icon from SD Card: 8**
- **Button Down Icon from SD Card: 9**
- **No Icon for Button Disabled state**

The button will be positioned at X = 260 and Y = 170. It's Touch Zone will have the width of 40 and the height of 30. The button will be initially drawn using Button Up icon.

BUTTON_DEF_EXT	**B7**	HEX	(**Command**)
4	**4**	DEC	(**Button No**)
1	**1**	DEC	(**Initial State:** Button Up)
0	**0**	DEC	(Button Up Icon No. MSB)
8	**8**	DEC	(Button Up Icon No. LSB)
0	**0**	DEC	(Button Down Icon No. MSB)
9	**9**	DEC	(Button Down Icon No. LSB)
255	**255**	DEC	(Button Disabled Icon No MSB)
255	**255**	DEC	(Button Disabled Icon No LSB)
1	**1**	DEC	(Upper-left corner X MSB)
4	**4**	DEC	(Upper-left corner X LSB)
0	**0**	DEC	(Upper-left corner Y MSB)
170	**170**	DEC	(Upper-left corner Y LSB)
0	**0**	DEC	(**Width** of the Touch Zone MSB)
40	**40**	DEC	(**Width** of the Touch Zone LSB)
0	**0**	DEC	(**Height** of the Touch Zone MSB)
30	**30**	DEC	(**Height** of the Touch Zone LSB)

2.9 BUTTON_STATE

Description: Changes the state of a previously defined touch button

Code: $\mathbf{B1}_{\mathrm{HEX}}$, $\mathbf{177}_{\mathrm{DEC}}$

7	6	5	4	3	2	1	0
BUTTON_STATE							
button_no state							

Byte 0: **Command**

Byte 1: **Button No** (0 to 63)

Byte 2: **Button State**

0 – Delete Permanently
1 – Up
2 – Down
3 – Disabled
4 – Non-Visible

About the Button State:

The button is automatically redrawn after its state has been changed, if the icon for the new state has been defined by the BUTTON_DEF command. Deleting the button (Byte 2 = 0) will not erase the button image from the screen. The ezLCD just stops reacting to the deleted button events. Changing the button state to Non-Visible (Byte 2 = 4) will also not erase the button image from the screen. The Non-Visible (Byte 2 = 4) state should mainly be used with the BUTTON_DEF command, if we do not wish the button to be initially drawn.

See Also: BUTTON_DEF, BUTTONS_ALL_UP, BUTTONS_DELETE_ALL, TOUCH_PROTOCOL

Commented [R1]: Add new button def

Note: Before using this command, please read the following chapters:

- Touch Screen
- ezButton
- cuButton

Example:

The following sequence will change the state of Button 4 to “Button Down”.
The button will be redrawn using the “Button Down” Icon.

BUTTON_STATE	**B1**	HEX	(**Command**)
4	**4**	DEC	(**Button No**)
2	**2**	DEC	(**Button Down**)

2.10 BUTTONS_ALL_UP

Description: Changes the state of all defined touch buttons to “Button Up”

Code: **B3**$_{HEX}$, **179**$_{DEC}$

7	6	5	4	3	2	1	0	
BUTTONS_ALL_UP								Byte 0: **Command**

Note: The button will be automatically redrawn if the icon for “Button Up” has been defined by the BUTTON_DEF command.

See Also: BUTTON_DEF, BUTTON_STATE, BUTTONS_DELETE_ALL, TOUCH_PROTOCOL

Commented [R2]: Add new button def

Note: Before using this command, please read the following chapters:

- Touch Screen
- ezButton
- cuButton

Example:
The following sequence will change the state of all Buttons to Up.

BUTTONS_ALL_UP **B3** HEX (**Command**)

2.11 BUTTONS_DELETE_ALL

Description: Deletes all touch buttons

Code: **B4**HEX, **180**DEC

7	6	5	4	3	2	1	0	
BUTTONS_DELETE_ALL								Byte 0: **Command**

Note: Deleting the buttons will not erase their image from the screen. The ezLCD-5x will just stop reacting to the button events.

See Also: BUTTON_DEF, BUTTON_STATE, BUTTONS_ALL_UP, TOUCH_PROTOCOL

Commented [R3]: Add new button def

Note: Before using this command, please read the following chapters:

- Touch Screen
- ezButton
- cuButton

Example:
The following sequence will delete all Buttons.

BUTTONS_DELETE_ALL **B4** HEX (**Command**)

2.12 CACHE_GET_FREE_SPACE

Description: Requests the remaining cache RAM available to load Icons for use with the PUT_SF_ICON command

Code: **68**HEX, **104**DEC

Min Firmware: **1.3.2**

7	6	5	4	3	2	1	0	
CACHE_GET_FREE_SPACE								Byte 0: Command

ezLCD Response

After receiving the CACHE_GET_FREE_SPACE command, the ezLCD responds with the following:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**0**	**1**	Byte 0: **3D**hex, **63**dec
Free Cache RAM Bvte 0								
Free Cache RAM Byte 1								
Free Cache RAM Byte 2								
Free Cache RAM Byte 3								

32-bit Free cache RAM Size

2.13 CACHE_ICON_LOAD

Description: Loads an image from the SD Card to the next location in Cache RAM so it can be used with the PUT_SF_ICON command

Code: **66**HEX, **102**DEC

Min Firmware: **1.3.2**

Path to the file on SD
(NULL-terminated string)

7	6	5	4	3	2	1	0	
CACHE_ICON_LOAD								Byte 0: Command
0	**0**	**0**	**0**	**0**	**0**	**0**	**DC**	Byte 1: If bit 0 set, store in cache decompressed
ASCII								Byte 2: First Character of the file path
ASCII								Byte 3: Second Character of the file path
⋮								
ASCII								Byte n: Last Character of file path
0								Byte n+1:NULL

ezLCD Response

After receiving the CACHE_ICON_LOAD command, the ezLCD responds with the following:

In case of the **success**:

0	**0**	**1**	**1**	**1**	**0**	**1**	**0**	Byte 0: **3A**HEX, **58**DEC
ICON ID								Byte 1: Assigned Icon ID

If the file could not be found or file is corrupt:

0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**HEX, **62**DEC

If there is not enough cache RAM available:

0	**0**	**1**	**1**	**1**	**0**	**0**	**1**	Byte 0: **39**HEX, **57**DEC

2.14 CACHE_ICON_SET_TRANSP

Description: Sets/Changes the Transparency Color and flag for an icon that has been loaded into cache RAM

Code: **67**$_{HEX}$, **103**$_{DEC}$

Min Firmware: **NOT YET IMPLEMENTED**

7	6	5	4	3	2	1	0		
CACHE_ICON_SET_TRANSP								Byte 0: Command	
ICON ID								Byte 1: ICON ID (0 to 254)	
0	**0**	**0**	**0**	**0**	**0**	**0**	**NT**	Byte 2: Bit 0=NonTransparent	
R7	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 3: Red	Transparent Color, ignored when Byte 2, Bit 0 is 1
G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 4: Green	
B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 5: Blue	

ezLCD Response

After receiving the CACHE_ICON_SET_TRANSP command, the ezLCD responds with the following:

In case of the **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**0**	**1**	**0**	Byte 0: **3A**$_{HEX}$, **58**$_{DEC}$

If the icon number is not cached:

0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**$_{HEX}$, **62**$_{DEC}$

Notes

This command will not affect icons that are already displayed. It only has an effect on new icons displayed with PUT_SF_ICON

2.15 CACHE_ICON_UNLOAD

Description: Removes the highest-numbered icon from the Cache and releases cache RAM for later re-use.

Code: $\mathbf{65}_{\text{HEX}}$, $\mathbf{101}_{\text{DEC}}$

Min Firmware: **1.3.2**

7	6	5	4	3	2	1	0	
CACHE_ICON_UNLOAD								Byte 0: Command

ezLCD Response

After receiving the CACHE_ICON_UNLOAD command, the ezLCD responds with the following:

In case of the **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**1**	Byte 0: $\mathbf{3F}_{\text{HEX}}$, $\mathbf{63}_{\text{DEC}}$
ICON COUNT								Byte 1: Number of Icons in Cache

If the no icons were cached:

0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: $\mathbf{3E}_{\text{HEX}}$, $\mathbf{62}_{\text{DEC}}$

Notes

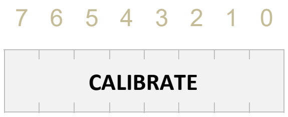
This command will not affect icons that are already displayed. It only has an effect on new icons displayed with PUT_SF_ICON

2.16 CALIBRATE

NOTE: Legacy deprecated command. Recommended Replacement: CALIBRATE_TOUCH

Description: Initiates the touch screen calibration sequence. Upon completion, the calibration data is stored on the SD card in the file */TouchCalibration.bin*

Code: **EF**HEX, 239DEC



Byte 0: **Command**

Note: Calibration is only supported on models with a Resistive touch screen

Calibration can also be triggered by grounding the PROG# pin on the CN1 connector.

See Also: BUTTON_DEF,

Commented [R4]: Add new button def

Example:
The following sequence will start the touch screen calibration sequence.

CALIBRATE	**EF**	HEX	**(Command)**

2.17 CALIBRATE_TOUCH

Description: Initiates the touch screen calibration sequence. Upon completion, the calibration data is stored on the SD card in the file */TouchCalibration.bin*

Code: **B6**$_{\text{HEX}}$, **182**$_{\text{DEC}}$

7	6	5	4	3	2	1	0	
CALIBRATE_TOUCH								Byte 0: **Command**

ezLCD Response

After the user completes the calibrate procedure (on the LCD), the ezLCD responds with the following:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**0**	**0**	**1**	**0**	Byte 0: **32**$_{\text{HEX}}$, **50**$_{\text{DEC}}$

Note: Calibration is only supported on models with a Resistive touch screen

Calibration can also be triggered by grounding the PROG# pin on the CN1 connector.

See Also: BUTTON_DEF,

Commented [R5]: Add new button def

Example:
The following sequence will start the touch screen calibration sequence.

CALIBRATE_TOUCH	**B6**	HEX	**(Command)**

2.18 CIRCLE_RH

Description: Draws a circle in “Current Color” centered at “Current Position”.

Code: **89**$_{HEX}$, **137**$_{DEC}$

7	6	5	4	3	2	1	0	
CIRCLE_RH								Byte 0: **Command**
r15	**r14**	**r13**	**r12**	**r11**	**r10**	**r9**	**r8**	Byte 1: **Radius** MSB
r7	**r6**	**r5**	**r4**	**r3**	**r2**	**r1**	**r0**	Byte 2: **Radius** LSB

See Also: SET_XHY, SET_COLORH

Example:
The following sequence will draw a green circle with the center positioned at (160, 117).

SET_COLORH	84	HEX	
GREEN_LSB	11100000	BIN	
GREEN_MSB	00000111	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
CIRCLE_RH	**89**	HEX	
0	**0**	DEC	(**Radius** MSB)
80	**80**	DEC	(**Radius** LSB)

2.19 CIRCLE_RH_FILL

Description: Draws a circle in “Current Color” centered at “Current Position”, filled with “Current Color”.

Code: **99**$_{\text{HEX}}$, **153**$_{\text{DEC}}$

7	6	5	4	3	2	1	0	
CIRCLE_RH_FILL								Byte 0: **Command**
r15	**r14**	**r13**	**r12**	**r11**	**r10**	**r9**	**r8**	Byte 1: **Radius** MSB
r7	**r6**	**r5**	**r4**	**r3**	**r2**	**r1**	**r0**	Byte 2: **Radius** LSB

See Also: SET_XHY, SET_COLORH

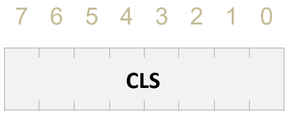
Example:
The following sequence will draw a red filled circle with the center positioned at (160, 117).

SET_COLORH	84	HEX	
RED_LSB	00000000	BIN	
RED_MSB	11111000	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
CIRCLE_RH_FILL	**99**	HEX	
0	**0**	DEC	(**Radius** MSB)
80	**80**	DEC	(**Radius** LSB)

2.20 CLS

Description: Clears the screen by filling it with the “Current Color”.

Code: **21**HEX, **33**DEC



Byte 0: **Command**

See Also: SET_COLORH

Example:
The following sequence will clear the screen (and fill it with white).

SET_COLORH	84	HEX
WHITE_LSB	11111111	BIN
WHITE_MSB	11111111	BIN
CLS	**21**	HEX

2.21 COPY_FRAME

Description: Copies all contents from the source frame to the destination frame

Code: **53**HEX, **83**DEC

7 6 5 4 3 2 1 0	
COPY_FRAME	Byte 0: **Command**
Destination Layer	Byte 1: **Destination Frame**
Source Layer	Byte 2: **Source Frame**

See Also: SET_LAYER_VISIBILITY, SET_DRAW_FRAME, SET_BG_DISP_FRAME, SET_FG_DISP_FRAME, MERGE_FRAME

2.22 DISABLE_COLOR_KEY

Description: Disables the color key function for the current drawing layer

Code: **3A**HEX, **58**DEC

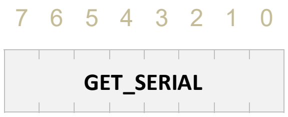
7 6 5 4 3 2 1 0

DISABLE_COLOR_KEY Byte 0: **Command**

2.20 GET_SERIAL

Description: Causes the ezLCD to reply with its permanent unique serial number

Code: **FD**HEX, **253**DEC



Byte 0: **Command**

See Also: GET_VERSION, SHOW_SETTINGS

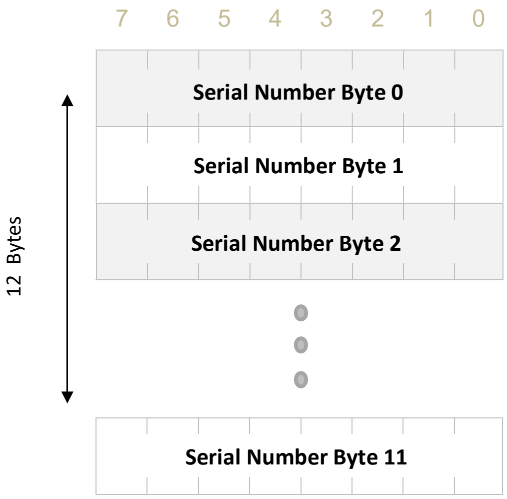
Commented [R6]: Add get serial short

Example:
The following sequence requests the ezLCD serial number.

GET_SERIAL **FD** HEX

ezLCD Response

After receiving the GET_SERIAL command, the ezLCD responds with the following sequence:



Byte 0: First byte of the serial number

Byte 1: Second byte of the serial number

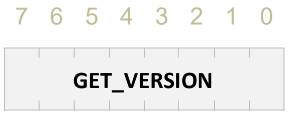
Byte 2: Third byte of the serial number

Byte 11: Last byte of the serial number

2.21 GET_VERSION

Description: Causes the ezLCD to reply with its firmware version, as a NULL-terminated ASCII string.

Code: **F9**HEX, **249**DEC



Byte 0: **Command**

See Also: GET_SERIAL, SHOW_SETTINGS

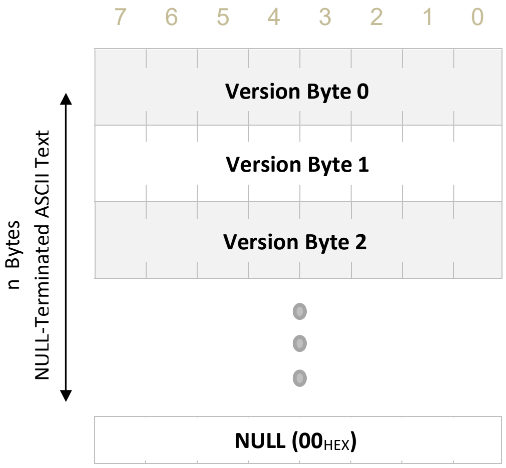
Commented [R7]: Add get serial short

Example:
The following sequence requests the ezLCD firmware version.

GET_VERSION **F9** HEX

ezLCD Response

After receiving the GET_VERSION command, the ezLCD responds with the following sequence:



Byte 0: First byte of the version, in ASCII

Byte 1: Second byte of the version, in ASCII

Byte 2: Third byte of the version, in ASCII

Byte n: Terminating NULL byte

2.22 GPIO_CONFIG

Description: Configure a GPIO Pin

Code: **BA**HEX, **186**DEC

7	6	5	4	3	2	1	0	
GPIO_CONFIG								Byte 0: **Command**
Pin Number								Byte 1: **Pin Number**
S2	**S1**	**0**	**OD**	**PD**	**PU**	**Out**	**In**	Byte 2: **Config Flags**

See Also: GPIO_READ, GPIO_WRITE

Pin Number: See Model Appendix for GPIO pin numbering

Config Flags:

Bit	Flag	If Set
0	In	Pin Configured as Input
1	Out	Pin Configured as Output
2	PU	Weak Pull-Up Resistor Applied
3	PD	Weak Pull-Down Resistor Applied
4	OD	Output: Open Drain Input: Ignored
5	--	Unused (Leave as 0)
6	S1	Output: Slew Rate (switching speed) Setting Input: Ignored
7	S2	

Slew Rate Values:

Value	Binary	Setting
0	00	Low
1	01	Medium

2	10	High
3	11	Very High

Note: If both **In** and **Out** are 0, pin is disabled (configured as High Impedance Analog Input per STM32 documentation)

ezLCD Response

After receiving the GPIO_CONFIG command, the ezLCD responds with the following:

In case of **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**0**	**0**	**1**	**0**	Byte 0: $\mathbf{32}_{HEX}$, $\mathbf{50}_{DEC}$

If there was an **error** (Invalid Configuration):

0	**0**	**1**	**1**	**0**	**1**	**1**	**0**	Byte 0: $\mathbf{36}_{HEX}$, $\mathbf{54}_{DEC}$

If there was an **error** (pin in use with ButtonDef):

0	**0**	**1**	**1**	**0**	**0**	**1**	**1**	Byte 0: $\mathbf{33}_{HEX}$, $\mathbf{51}_{DEC}$

If there was an **error** (Invalid Pin Number):

0	**0**	**1**	**1**	**0**	**1**	**0**	**1**	Byte 0: $\mathbf{35}_{HEX}$, $\mathbf{53}_{DEC}$

If there was an **error** (Pin in use by Alt Function):

0	**0**	**1**	**1**	**0**	**1**	**1**	**1**	Byte 0: $\mathbf{37}_{HEX}$, $\mathbf{55}_{DEC}$

2.23 GPIO_READ

Description: Read GPIO Pin(s)

Code: **BB**HEX, **187**DEC

7	6	5	4	3	2	1	0	
GPIO_READ								Byte 0: **Command**
Pin Number								Byte 1: **Pin Number**
0	**0**	**0**	**0**	**0**	**0**	**0**	**MU**	Byte 2: **Flags**

See Also: GPIO_CONFIG, GPIO_WRITE

Pin Number: See Model Appendix for GPIO pin numbering

Flags:

Bit	Flag	If Set
0	MU	Read 8 GPIO's starting at Pin Number vs only 1

ezLCD Response

After receiving the GPIO_READ command, the ezLCD responds with the following:

In case of **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**0**	**1**	**0**	**0**	Byte 0: **34**HEX, **52**DEC
b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**	Byte 1: Pin States

Note: If MU Flag==0, only b0 is valid; b1-b7=0

If there was an **error** (Invalid Pin Number):

0	**0**	**1**	**1**	**0**	**1**	**0**	**1**	Byte 0: **35**HEX, **53**DEC

If there was an **error** (Invalid Pin Config):

0	**0**	**1**	**1**	**0**	**1**	**1**	**0**

Byte 0: **36**$_{HEX}$, **54**$_{DEC}$

2.25 GPIO_WRITE

Description: Control a GPIO Output Pin

Code: **BC**$_{\text{HEX}}$, **188**$_{\text{DEC}}$

7	6	5	4	3	2	1	0	
GPIO_WRITE								Byte 0: **Command**
Pin Number								Byte 1: **Pin Number**
0	**0**	**0**	**0**	**TM**	**T**	**R**	**S**	Byte 2: **Function**
t15	**t14**	**t13**	**t12**	**t11**	**t10**	**t9**	**t8**	Byte 3: **Time Value MSB**
t7	**t6**	**t5**	**t4**	**t3**	**t2**	**t1**	**t0**	Byte 4: **Time Value LSB**

See Also: GPIO_CONFIG, GPIO_READ

Pin Number: See Model Appendix for GPIO pin numbering

Function:

Bit	**Flag**	**If Set**
0	S	Set Pin High
1	R	Reset Pin Low
2	T	Toggle Pin State
3	TM	Timed Operation

Time Value:

If TM bit=1, Time Value is the time in milliseconds for the operation to be performed (ie, set pin high for 100ms).

If TM bit=0, GPIO pin state is permanent until the next GPIO command for the respective Pin Number and Time Value is ignored

ezLCD Response

After receiving the GPIO_WRITE command, the ezLCD responds with the following:

In case of **success**:

7	6	5	4	3	2	1	0
0	**0**	**1**	**1**	**0**	**0**	**1**	**0**

Byte 0: **32**$_{HEX}$, **50**$_{DEC}$

If there was an **error** (Invalid Pin Number):

0	**0**	**1**	**1**	**0**	**1**	**0**	**1**

Byte 0: **35**$_{HEX}$, **53**$_{DEC}$

If there was an **error** (Not Configured as Output):

0	**0**	**1**	**1**	**0**	**1**	**1**	**0**

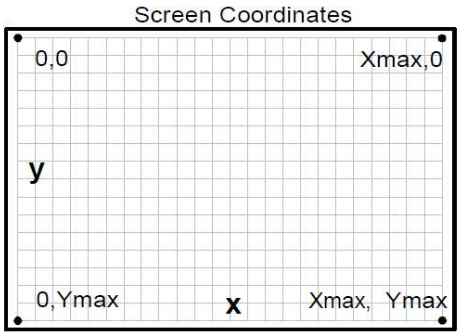
Byte 0: **36**$_{HEX}$, **54**$_{DEC}$

2.26 H_LINEH

Description: Quickly draws a horizontal line from the “Current Position” to the column specified by the parameter.

Code: **A0**$_{HEX}$, **160**$_{DEC}$

7	6	5	4	3	2	1	0	
H_LINEH								Byte 0: **Command**
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X** LSB



See Also: V_LINE, SET_XHY

Example:

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

SET_COLORH	84	HEX	
GREEN_LSB	11100000	BIN	
GREEN_MSB	00000111	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
20	20	DEC	(**X** LSB)
60	60	HEX	(**Y**)
H_LINEH	**A0**	HEX	
0	**0**	DEC	(**Y** MSB)
170	**170**	DEC	(**Y** LSB)

2.27 LIGHT_BRIGHT

Description: Sets the brightness of the screen backlight.

Code: **80**HEX, **128**DEC

7	6	5	4	3	2	1	0	
LIGHT_BRIGHT								Byte 0: **Command**
								Byte 1: **Brightness** (0 to 255)

Note: The default brightness is 255

See Also: LIGHT_ON, LIGHT_OFF

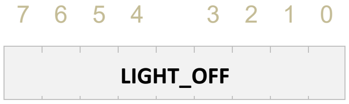
Example:
The following sequence will set the backlight to 25% of its full brightness.

LIGHT_BRIGHT	**80**	HEX
64	**64**	DEC

2.28 LIGHT_OFF

Description: Turns off the screen backlight.

Code: **23**$_{\text{HEX}}$, **35**$_{\text{DEC}}$



Byte 0: **Command**

See Also: LIGHT_ON, LIGHT_BRIGHT

Example:
The following sequence will turn off the screen backlight.

LIGHT_OFF 23 HEX

2.29 LIGHT_ON

Description: Turns on the screen backlight.

Code: **22**$_{HEX}$, **34**$_{DEC}$

7	6	5	4	3	2	1	0	
LIGHT_ON								Byte 0: **Command**

See Also: LIGHT_ON, LIGHT_BRIGHT

Example:
The following sequence will turn on the screen backlight.

LIGHT_ON **22** HEX

2.30 LINE_TO_XHY

NOTE: Legacy deprecated command. Recommended Replacement: **LINE_TO_XHYH**

Description: Draws a line in “Current Color”, from the “Current Position” to the specified position.

Code: **88**$_{HEX}$, **136**$_{DEC}$

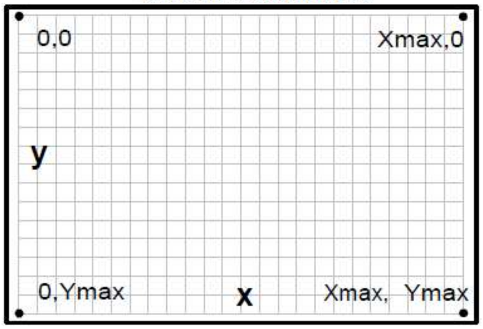
Command Format—Standard Mode:

7	6	5	4	3	2	1	0	
LINE_TO_XHY								Byte 0: **Command**
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X** LSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y**

Command Format—Extended Mode:

7	6	5	4	3	2	1	0	
LINE_TO_XHY								Byte 0: **Command**
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X** LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 3: **Y** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y** LSB

Screen Coordinates



See Also: SET_XHY, SET_COLORH, PLOT

Example:

The following sequence will draw a red line across the screen (using extended mode).

SET_MODE_EXTENDED	FB	HEX	
SET_COLORH	84	HEX	
RED_LSB	00000000	BIN	
RED_MSB	11111000	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X_0** MSB)
0	0	DEC	(**X_0** LSB)
0	0	DEC	(**Y_0** MSB)
0	0	DEC	(**Y_0** LSB)
LINE_TO_XHY	**88**	HEX	
1	**1**	DEC	(**X_1** MSB)
63	**63**	DEC	(**X_1** LSB)
1	**1**	DEC	(**Y_1** MSB)
233	**233**	DEC	(**Y_1** LSB)

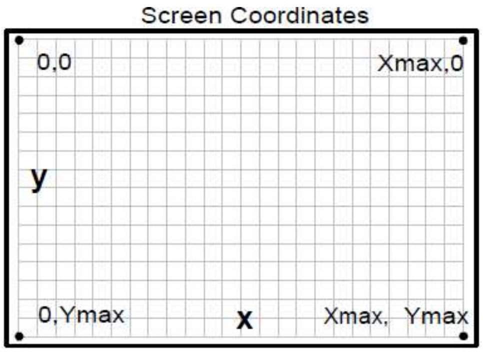
2.31 LINE_TO_XHYH

Description: Draws a line in “Current Color”, from the “Current Position” to the specified position.

Code: **3F**$_{\text{HEX}}$, **63**$_{\text{DEC}}$

Command Format:

7	6	5	4	3	2	1	0	
LINE_TO_XHYH								Byte 0: **Command**
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X** LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 3: **Y** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y** LSB



See Also: SET_XHYH, SET_COLOR_RGB, PLOT

Example:

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

SET_COLOR_RGB	31	HEX	
RED	FF	HEX	
GREEN	0	HEX	
BLUE	0	HEX	
0	0	DEC	(**X_0** MSB)
0	0	DEC	(**X_0** LSB)
0	0	DEC	(**Y_0** MSB)
0	0	DEC	(**Y_0** LSB)
LINE_TO_XHYH	**3F**	HEX	
0	**0**	DEC	(**X_1** MSB)
63	**63**	DEC	(**X_1** LSB)
0	**0**	DEC	(**Y_1** MSB)
233	**233**	DEC	(**Y_1** LSB)

2.32 MERGE_FRAME

Description: Merges the contents of the source frame with the destination frame according to the current value of Alpha. The resulting image is held in the destination frame.

Code: **54**HEX, **84**DEC

7 6 5 4 3 2 1 0	
MERGE_FRAME	Byte 0: **Command**
Destination Frame	Byte 1: **Destination Frame**
Source Frame	Byte 2: **Source Frame**

See Also: SET_ALPHA, SET_LAYER_VISIBILITY, SET_DRAW_FRAME, SET_BG_DISP_FRAME, SET_FG_DISP_FRAME, COPY_FRAME

Notes: When Alpha=255, COPY_FRAME and MERGE_FRAME yield identical results
Merging a frame with itself may yield unpredictable results

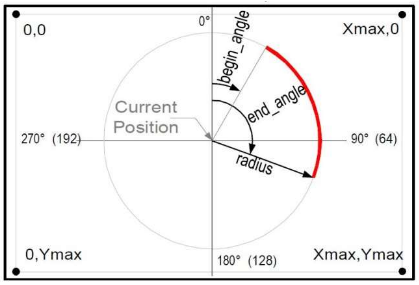
2.33 PIEH

Description: Draws a pie in “Current Color” with the center at “Current Position”, starting on “Begin Angle” and ending on “End Angle”.

Code: **90**HEX, **144**DEC

7	6	5	4	3	2	1	0	
PIEH								Byte 0: **Command**
radius_MSB								Byte 1: **Radius** MSB
radius_LSB								Byte 2: **Radius** LSB
begin_angle_MSB								Byte 3: **Pie Begin Angle** MSB
begin_angle_LSB								Byte 4: **Pie Begin Angle** LSB
end_angle_MSB								Byte 5: **Pie End Angle** MSB
end_angle_LSB								Byte 6: **Pie End Angle** LSB

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



See Also: PIEH, SET_XHY, SET_COLORH, CIRCLE_RH

Angle Coding: The full angle (360°) is equal to **4000**HEX (**16384**DEC).

Degrees to ARC Angle Units Equation: $Angle_{ARC} = Angle_{DEG} \cdot \frac{2048}{45}$

Angle Conversion Example:

2048$_{DEC}$ = **800**$_{HEX}$ = **45°**
4096$_{DEC}$ = **1000**$_{HEX}$ = **90°**
8192$_{DEC}$ = **2000**$_{HEX}$ = **180°**
12288$_{DEC}$ = **3000**$_{HEX}$ = **270°**
16384$_{DEC}$ = **4000**$_{HEX}$ = **360°** = **0°**

Drawing Example:

The following command sequence will draw a green arc from 45 to 225 degrees with the center positioned at (160, 117) and a radius of 80.

$$225 \cdot \frac{2048}{45} = 10240 \rightarrow 10240_{DEC} = 2800_{HEX}$$

SET_COLORH	84	HEX	
GREEN_LSB	11100000	BIN	
GREEN_MSB	00000111	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
PIEH	**90**	HEX	
0	**0**	DEC	(**Radius** MSB)
80	**80**	DEC	(**Radius** LSB)
08	**08**	HEX	(**Begin_Angle** MSB)
00	**00**	HEX	(**Begin_Angle** LSB)
28	**28**	HEX	(**End_Angle** MSB)
00	**00**	HEX	(**End_Angle** LSB)

2.34 PING

Description: Checks if the ezLCD-5x is connected and is ready to receive commands.

Code: $\mathbf{83}_{\text{HEX}}$, $\mathbf{131}_{\text{DEC}}$

7	6	5	4	3	2	1	0	
PING								Byte 0: **Command**

ezLCD Response: After receiving the PING command, the ezLCD responds with the PONG ($\mathbf{38}_{\text{HEX}}$, $\mathbf{56}_{\text{DEC}}$) byte:

7	6	5	4	3	2	1	0	
0	0	1	1	1	0	0	0	Byte 0: $\mathbf{38}_{\text{HEX}}$, $\mathbf{56}_{\text{DEC}}$ **(PONG)**

The response is sent through the same interface as the PING command.

Example:
The following sequence will check if the ezLCD is OK:

PING	**83**	HEX

If the ezLCD is connected and ready to receive commands, it responds with:

	38	HEX

2.35 PLOT

Description: Plots a point at “Current Position” in “Current Color”.

Code: **26**$_{\text{HEX}}$, **38**$_{\text{DEC}}$

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td></td></tr>
<tr><td colspan="8"><b>PLOT</b></td><td>Byte 0: <b>Command</b></td></tr>
</table>

See Also: SET_XHY, SET_COLORH

Example:
The following sequence will put a blue point at (160, 117).

SET_COLORH	84	HEX	
BLUE_LSB	00011111	BIN	
BLUE_MSB	00000000	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
PLOT	**26**	HEX	

2.36 PLOT_XHYH

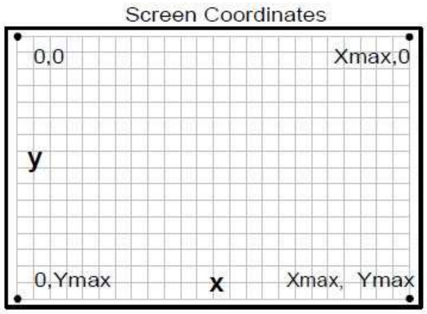
Description: Plots a point in “Current Color" at the specified position.

Code: **87**$_{HEX}$, **135**$_{DEC}$ (deprecated code)

3E$_{HEX}$, **62**$_{DEC}$

	6	5	4	3	2	1	0	
PLOT_XHY								Byte 0: **Command**
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X** LSB
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 3: **Y** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y** LSB

See Also: SET_XHY, SET_COLORH, PLOT



ezLCD 405 compatibility: Command format is the same as PLOT_XHY in extended mode. Standard mode in ezLCD405 has only 1 byte for Y coordinate

Example:

The following sequence will put a red point at (310, 117).

SET_COLORH	84	HEX	
RED_LSB	00000000	BIN	
RED_MSB	11111000	BIN	
PLOT_XHYH	**3E**	HEX	
1	**1**	DEC	(**X** MSB)
36	**36**	DEC	(**X** LSB 1*256+54=310)
0	0	DEC	(**Y** MSB)
117	**117**	DEC	(**Y** LSB)

Commented [R8]: Change example to RGB version

2.37 PRINT_CHAR

Description: Prints a character at the “Current Position”.

Code: **2C**$_{HEX}$, **44**$_{DEC}$

7 6 5 4 3 2 1 0		
PRINT_CHAR	Byte 0:	**Command**
ASCII	Byte 1:	**ASCII Character**

See Also: SELECT_FONT, PRINT_STRING

Example:
The following sequence will print a black character 'M' using Font 2.

SELECT_FONT	2B	HEX
2	2	DEC
SET_COLORH	84	HEX
BLACK_LSB	00000000	BIN
BLACK_MSB	00000000	BIN
PRINT_CHAR	**2C**	HEX
'M'	**4D**	HEX

Commented [R9]: Change example to RGB version

2.38 PRINT_CHAR_BG

Description: Prints a character at the “Current Position” on the background specified by the SET_BG_COLORH command.

Code: **3C**HEX, **60**DEC

7 6 5 4 3 2 1 0	
PRINT_CHAR_BG	Byte 0: **Command**
ASCII	Byte 1: **ASCII Character**

See Also: SELECT_FONT, SET_BG_COLORH, PRINT_STRING_BG

Example:
The following sequence will print the character 'M' in white on a black background using Font 2.

SELECT_FONT	2B	HEX
2	2	DEC
SET_BG_COLORH	94	HEX
BLACK_LSB	00000000	BIN
BLACK_MSB	00000000	BIN
SET_COLORH	84	HEX
WHITE_LSB	11111111	BIN
WHITE_MSB	11111111	BIN
PRINT_CHAR_BG	**3C**	HEX
'M'	**4D**	HEX

Commented [R10]: Change example to RGB

2.39 PRINT_STRING

Description: Prints a null-terminated string starting at “Current Position”.

Code: **2D**HEX, **45**DEC

7 6 5 4 3 2 1 0	
PRINT_STRING	Byte 0: **Command**
ASCII	Byte 1: **First Character**
ASCII	Byte 2: **First Character**
⋮	
ASCII	Byte n: **Last Character**
0	Byte n+1: **NULL**

See Also: SELECT_FONT, PRINT_CHAR

Example:
The following sequence will print "LCD" in purple using Font 1 at (160, 117).

SELECT_FONT	2B	HEX	
1	1	DEC	
SET_COLORH	84	HEX	
PURPLE_LSB	00010000	BIN	
PURPLE_MSB	10000000	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
PRINT_STRING	**2D**	HEX	
'L'	**4C**	HEX	
'C'	**43**	HEX	
'D'	**44**	HEX	
NULL	**0**	HEX	

Commented [R11]: Change example to RGB

2.40 PRINT_STRING_BG

Description: Prints null-terminated string starting at “Current Position” on the background specified by the SET_BG_COLORH command

Code: **3D**HEX, **61**DEC

7 6 5 4 3 2 1 0	
PRINT_STRING	Byte 0: **Command**
ASCII	Byte 1: **First Character**
ASCII	Byte 2: **First Character**
⋮	
ASCII	Byte n: **Last Character**
0	Byte n+1: **NULL**

See Also: SELECT_FONT, SET_BG_COLORH, PRINT_CHAR_BG

Example:
The following sequence print "LCD" in yellow on a navy background in the middle of the screen using Font 0.

SET_BG_COLORH	94	HEX	
NAVY_LSB	00010000	BIN	
NAVY_MSB	00000000	BIN	
SET_COLORH	84	HEX	
YELLOW_LSB	11100000	BIN	
YELLOW_MSB	11111111	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
SELECT_FONT	2B	HEX	
0	0	DEC	
PRINT_STRING_BG	**3D**	HEX	
'L'	**4C**	HEX	
'C'	**43**	HEX	
'D'	**44**	HEX	
NULL	**0**	HEX	

Commented [R12]: Change example to RGB

2.41 PUT_BITMAP_RGB

Description: Displays a Bitmap on the screen starting at Current Position.

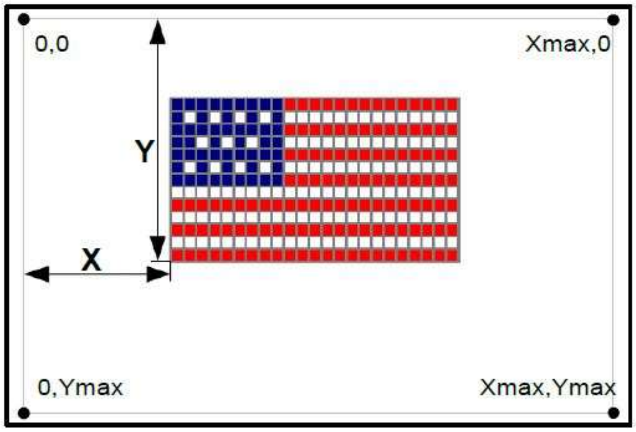
Code: **9F**$_{\text{HEX}}$, **159**$_{\text{DEC}}$

Min Firmware: **1.3.2**

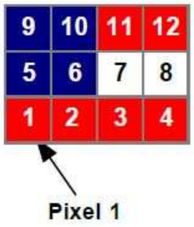
	7	6	5	4	3	2	1	0		
	PUT_BITMAP_RGB								Byte 0: Command	
header	**WIDTH_MSB**								Byte 1:	Width
	WIDTH_LSB								Byte 2:	
	HEIGHT_LSB								Byte 3:	Height
	HEIGHT_MSB								Byte 4:	
	0	**0**	**0**	**0**	**0**	**0**	**0**	**NT**	Byte 5: Bit 0=NonTransparent	
	R7	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 6: Red	Transparent Color, ignored when Byte 4, Bit 0 is 1
	G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 7: Green	
	B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 8: Blue	
pixels	**R7**	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 9: Red	Pixel$_1$ at (X, Y)
	G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 10: Green	
	B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 11: Blue	
	R7	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 12: Red	Pixel$_2$ at (X+1, Y)
	G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 13: Green	
	B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 14: Blue	
	⋮									
	R7	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 6+ (3 * Width * Height)	Pixel at (X + Width – 1, Y – Height +1)
	G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 7+ (3 * Width * Height)	
	B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 8+ (3 * Width * Height)	

Notes:

- The total number of bytes is: $3 \cdot Width \cdot Height + 9$
- When Byte 5, Bit 0 = 0, Bytes 6-8 specify the Transparent Color.
- Pixels equal to the Transparent Color are ignored during bitmap drawing.
- All pixels are drawn when Byte 5, Bit 0 is not 0.
- Image is transmitted from the bottom up (Y decrements from the current position)



Example of the order of the pixels in case of the 4x3 bitmap.



See Also: SET_XHY, SET_COLORH

2.42 PUT_BITMAPH

NOTE: Legacy deprecated command. Recommended Replacement: PUT_BITMAP_RGB

Description: Displays a Bitmap on the screen starting at Current Position.

Code: **9E**HEX, **158**DEC

Command Format—Standard Mode:

	7	6	5	4	3	2	1	0	
	PUT_BITMAPH								Byte 0: Command
header	width_MSB								Byte 1: Width MSB
	width_LSB								Byte 2: Width LSB
	height_LSB								Byte 3: Height
	0	0	0	0	0	0	0	NT	Byte 4: Bit 0=NonTransparent
	G2	G1	G0	B4	B3	B2	B1	B0	Byte 5: Transparent Color LSB, ignored when Byte 4 is not 0
	R4	R3	R2	R1	R0	G5	G4	G3	Byte 6: Transparent Color MSB, ignored when Byte 4 is not 0
pixels	G2	G1	G0	B4	B3	B2	B1	B0	Byte 7: Pixel$_1$ at (X, Y)
	R4	R3	R2	R1	R0	G5	G4	G3	Byte 8: Pixel$_1$ at (X, Y)
	G2	G1	G0	B4	B3	B2	B1	B0	Byte 9: Pixel$_2$ at (X+1, Y)
	R4	R3	R2	R1	R0	G5	G4	G3	Byte 10: Pixel$_2$ at (X+1, Y)
				⋮					
	G2	G1	G0	B4	B3	B2	B1	B0	Byte $4+2\cdot \boldsymbol{width}\cdot \boldsymbol{height}$: Pixel at $(X+Width-1, Y-Height+1)$
	R4	R3	R2	R1	R0	G5	G4	G3	Byte $5+2\cdot \boldsymbol{width}\cdot \boldsymbol{height}$: Pixel at

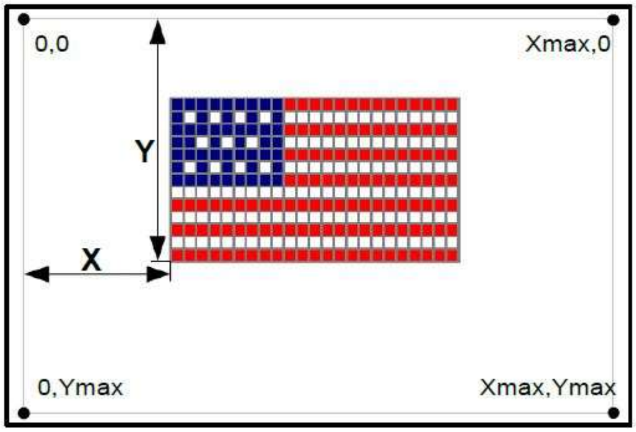
🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟩	$(\boldsymbol{X+Width-1, Y-Height+1})$

Command Format—Extended Mode:

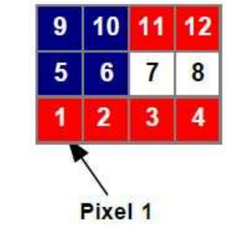
7	6	5	4	3	2	1	0	
PUT_BITMAPH								Byte 0: Command
width_MSB								Byte 1: Width MSB
width_LSB								Byte 2: Width LSB
height_MSB								Byte 3: Height MSB
height_LSB								Byte 4: Height LSB
non_transparent								Byte 5: 0 = Transparent
G2	**G1**	**G0**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 6: Transparent Color LSB, ignored when Byte 4 is not 0
R4	**R3**	**R2**	**R1**	**R0**	**G5**	**G4**	**G3**	Byte 7: Transparent Color MSB, ignored when Byte 4 is not 0
G2	**G1**	**G0**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 8: Pixel$_1$ at (X, Y)
R4	**R3**	**R2**	**R1**	**R0**	**G5**	**G4**	**G3**	Byte 9: Pixel$_1$ at (X, Y)
G2	**G1**	**G0**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 10: Pixel$_2$ at (X+1, Y)
R4	**R3**	**R2**	**R1**	**R0**	**G5**	**G4**	**G3**	Byte 11: Pixel$_2$ at (X+1, Y)
			⋮					
G2	**G1**	**G0**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte $\boldsymbol{5+2\cdot width\cdot\ height}$: Pixel at $(\boldsymbol{X+Width-1, Y-Height+1})$
R4	**R3**	**R2**	**R1**	**R0**	**G5**	**G4**	**G3**	Byte $6+2\cdot width\cdot\ height$: Pixel at $(\boldsymbol{X+Width-1, Y-Height+1})$

Notes:

- The total number of bytes is: $2 \cdot Width \cdot Height + 7$
- When Byte 4 = 0, Bytes 5 and 6 specify the Transparent Color.
- Pixels equal to the Transparent Color are ignored during bitmap drawing.
- All pixels are drawn when Byte 4 is not 0.
- Image is transmitted from the bottom up (Y decrements from the current position)



Example of the order of the pixels in case of the 4x3 bitmap.



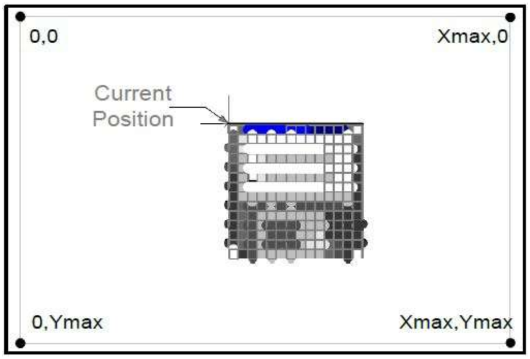
9	10	11	12
5	6	7	8
1	2	3	4

See Also: SET_XHY, SET_COLORH

2.43 PUT_SF_ICON

Description: Displays an image with its upper-left corner positioned at the “Current Position”. The images used are stored on the SDCard and defined/numbered in *config.txt* or via the settings file created by the ezLCD Config. Utility.

Code: **58**$_{HEX}$, **88**$_{DEC}$



7 6 5 4 3 2 1 0	
PUT_SF_ICON	Byte 0: **Command**
ICON ID	Byte 1: **ICON ID** (0 to 254)

Note: Maximum number of icons is 255 (**ID**’s range from 0 to 254)

ezLCD004/005 Compatibility Note:

The images used for this command are stored on the SD card, as the ezLCD5x does not have a serial flash chip.

See Also: SET_XHY

Example:
The following sequence will display Icon 3 with its upper-left corner positioned at (60, 43).

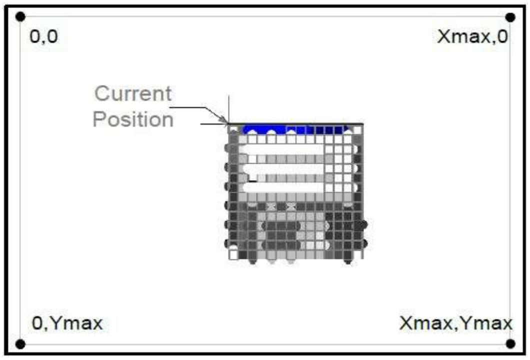
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
60	60	DEC	(**X** LSB)
43	43	DEC	(**Y**)
PUT_SF_ICON	**58**	HEX	
3	**3**	DEC	

2.44 PUT_PICT_NO

NOTE: Legacy deprecated command. Recommended Replacement: PUT_SF_ICON

Description: Displays an image with its upper-left corner positioned at the "Current Position". The images used are stored on the SDCard and defined/numbered in *config.txt* or via the settings file created by the ezLCD Config. Utility.

Code: **59**HEX, **89**DEC



7 6 5 4 3 2 1 0	
PUT_SF_ICON	Byte 0: **Command**
ICON ID MSB	Byte 1: **ICON ID** MSB
ICON ID LSB	Byte 2: **ICON ID** LSB

Note: Maximum number of icons is 255 (**ID**'s range from 0 to 254)

See Also: SET_XHY

Example:
The following sequence will display Icon 3 with its upper-left corner positioned at (60, 43).

SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
60	60	DEC	(**X** LSB)
43	43	DEC	(**Y**)
PUT_SF_ICON	**59**	HEX	
0	**0**	DEC	
3	**3**	DEC	

2.44 REPLACE_COLOR

Description: Replaces color in the Edit Rectangle set by the SET_EDIT_RECT command

Code: **5D**HEX, **93**DEC

Min Firmware: **2.0**

7	6	5	4	3	2	1	0		
REPLACE_COLOR								Byte 0: Command	
R7	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 1: Red	**Old Color**
G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 2: Green	
B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 3: Blue	
R7	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 4: Red	**New Color**
G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 5: Green	
B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 6: Blue	

See Also: SET_EDIT_RECT

Example:
The following sequence will replace color red with green inside the rectangle size of 100x50 and positioned at (320, 240)

SET_EDIT_RECT	**5C**	HEX	
1	1	DEC	(X MSB)
64	64	DEC	(X LSB)
0	0	DEC	(Y MSB)
240	240	DEC	(Y LSB)
0	0	DEC	(Width MSB)
100	100	DEC	(Width LSB)
0	0	DEC	(Height MSB)
50	0	DEC	(Height LSB)
REPLACE_COLOR	**5D**	HEX	
RED	FF	HEX	(Old Color Red)
GREEN	00	HEX	(Old Color Green)
BLUE	00	HEX	(Old Color Blue)
RED	00	HEX	(New Color Red)
GREEN	FF	HEX	(New Color Green)
BLUE	00	HEX	(New Color Blue)

2.45 RESET

Description: Reboots the ezLCD CPU

Code: **FA**$_{HEX}$, **250**$_{DEC}$

7	6	5	4	3	2	1	0	
RESET								Byte 0: **Command**

Note: After the RESET is completed, the ezLCD will transmit EZLCD_READY (**EA**$_{HEX}$) on the default transmitter interface. It is recommended that you wait until receiving this prior to sending any additional commands to the ezLCD.

Sending this command via USB will cause the USB device to disconnect

Example:
The following sequence will reboot the ezLCD

RESET	**FA**	HEX	**(Command)**

After the ezLCD has completed rebooting, the following will be sent on the default transmitter interface:

EZLCD_READY	**EA**	HEX

2.46 RESTORE_POSITION

Description: Restores the “Current Position” saved by the SAVE_POSITION command.

Code: **36**$_{\text{HEX}}$, **54**$_{\textbf{DEC}}$

7 6 5 4 3 2 1 0	
RESTORE_POSITION	Byte 0: **Command**
POSITION ID	Byte 1: **POSITION ID** (0 to 255)

See Also: SAVE_POSITION, SET_XHY

Example:
The following sequence will draw 3 lines with the common starting point: (160, 117)

SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
SAVE_POSITION	35	HEX	
12	12	DEC	(**Position ID**)
LINE_TO_XHY	88	HEX	
0	0	DEC	(**X1** MSB)
247	247	DEC	(**X1** LSB)
67	67	DEC	(**Y1**)
RESTORE_POSITION	**36**	HEX	
12	**12**	DEC	(**Position ID**)
LINE_TO_XHY	88	HEX	
0	0	DEC	(**X1** MSB)
73	73	DEC	(**X1** LSB)
67	67	DEC	(**Y1**)
RESTORE_POSITION	**36**	HEX	
12	**12**	DEC	(**Position ID**)
V_LINE	41	HEX	
217	217	DEC	

2.47 SAVE_POSITION

Description: Stores the “Current Position” to the “Position ID”. The saved position may be later restored by the RESTORE_POSITION command.

Code: **35**$_{\text{HEX}}$, **53**$_{\text{DEC}}$

7 6 5 4 3 2 1 0	
SAVE_POSITION	Byte 0: **Command**
POSITION ID	Byte 1: **POSITION ID** (0 to 255)

See Also: RESTORE_POSITION, SET_XHY

Example:
The following sequence will draw 3 lines with the common starting point: (160, 117)

SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
SAVE_POSITION	**35**	HEX	
12	**12**	DEC	(**Position ID**)
LINE_TO_XHY	88	HEX	
0	0	DEC	(**X1** MSB)
247	247	DEC	(**X1** LSB)
67	67	DEC	(**Y1**)
RESTORE_POSITION	36	HEX	
12	12	DEC	(**Position ID**)
LINE_TO_XHY	88	HEX	
0	0	DEC	(**X1** MSB)
73	73	DEC	(**X1** LSB)
67	67	DEC	(**Y1**)
RESTORE_POSITION	36	HEX	
12	12	DEC	(**Position ID**)
V_LINE	41	HEX	
217	217	DEC	

2.48 SD_FILE_CLOSE

Description: Closes SD Flash file. Re-enables the touch screen if no other SD files are opened.

Code: **72**HEX, **114**DEC

7 6 5 4 3 2 1 0	
SD_FILE_CLOSE	Byte 0: **Command**
FILE ID	Byte 1: **File ID**

Notes: SD card has to be formatted in the supported file system.

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_OPEN, SD_FILE_CREATE, SD_FILE_CLOSE_ALL

About the File ID: File ID is returned in the response to the SD_FILE_OPEN command. It identifies the file after it has been opened. Since a maximum 2 files may be concurrently opened, the File ID should be: 1 or 2. Values higher than 2 are interpreted as 2, and 0 is interpreted as 1.

Example:
The following sequence will close SD file 1.

SD_FILE_CLOSE	**72**	HEX	
1	**1**	DEC	(**File ID**)

2.49 SD_FILE_CLOSE_ALL

Description: Closes all opened SD Flash files and re-enables the touch screen.

Code: **73**HEX, **115**DEC

7	6	5	4	3	2	1	0	
SD_FILE_CLOSE_ALL								Byte 0: **Command**

Notes: SD card has to be formatted in the supported file system.

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_OPEN, SD_FILE_CREATE, SD_FILE_CLOSE

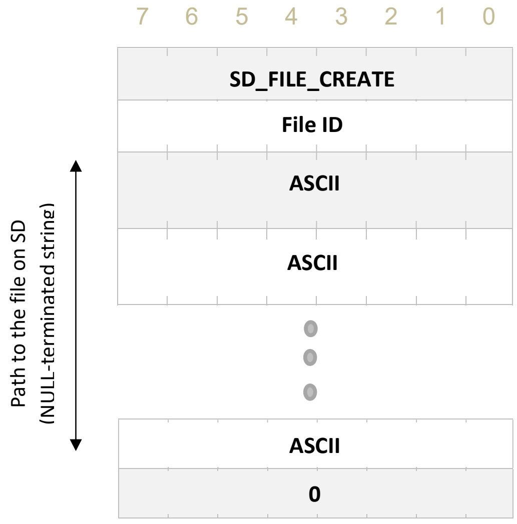
Example:
The following sequence will close all opened SD files and re-enable the touch screen.

SD_FILE_CLOSE_ALL	**73**	HEX

2.50 SD_FILE_CREATE

Description: Creates a new SD Flash file and opens it for writing. File Position Index is set to 0.

Code: **76**HEX, **118**DEC



Byte 0: **Command**

Byte 1: **File ID**

Byte 2: First Character of the file path

Byte 3: Second Character of the file path

Byte n: **Last Character**

Byte n+1: **NULL**

Notes: SD card has to be formatted in the supported file system.

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_OPEN, SD_FILE_CLOSE, SD_FILE_CLOSE_ALL

About the File ID:

File ID identifies the file after it has been opened. Since maximum 2 files may be concurrently opened, the File ID should be: 1 or 2. Values higher than 2 are interpreted as 2 and 0 is interpreted as 1. If the file ID was previously opened, the previous file will be automatically closed before the file is created.

About the File Path:

- File Path specifies the full path to the file on SD including directory, filename and extension
- Directories should be separated by: / (**not by:** \ like in Windows and DOS).
- File Path is not case-sensitive. The drive and root directory do not have to be indicated, for example, both: A:/Cat/Jumped/Over.txt and cat/jumped/over.TXT specify the same file.
- Long file names are supported, however the File Path (directory + filename + extension + NULL) may not exceed 64 bytes.

ezLCD Response

After receiving the SD_FILE_CREATE command, the ezLCD responds with the following sequence:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**1**	Byte 0: **3F**$_{\text{HEX}}$, **63**$_{\text{DEC}}$
File ID / Error								Byte 1: **File ID** (Success), 0 (Error)

The ezLCD response is sent through the same interface, which received the SD_FILE_CREATE command.

Touch Screen Processing

SD_FILE_CREATE command temporary disables the touch screen.

The touch screen will be automatically re-enabled when all files are closed. This can be done by issuing the SD_FILE_CLOSE or SD_FILE_CLOSE_ALL command.

Note: The touch screen is temporary disabled, even if due to error no file is created. If this is the case, issuing "dummy" SD_FILE_CLOSE or SD_FILE_CLOSE_ALL command will re-enable the touch screen.

Example:
The following sequence will create and open file MyFile.dat

SD_FILE_CREATE	**76**	HEX	
1	**1**	DEC	(**File ID**)
'M'	**4D**	HEX	
'y'	**79**	HEX	
'F'	**46**	HEX	
'i'	**69**	HEX	
'l'	**6C**	HEX	
'e'	**65**	HEX	
'.'	**2E**	HEX	
'd'	**64**	HEX	
'a'	**63**	HEX	
't'	**74**	HEX	
NULL	**0**	HEX	

If the file has successfully been created, the ezLCD responds with the following sequence:

3F	HEX
1	DEC (file ID)

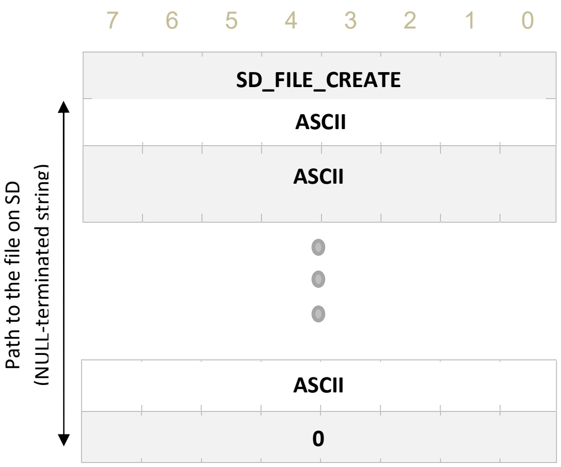
In case of the failure, the following sequence will be sent by the ezLCD:

3F	HEX
0	DEC (indicates error)

2.51 SD_FILE_DELETE

Description: Deletes the SD file

Code: **7D**HEX, **125**DEC



Byte 0: **Command**

Byte 2: First Character of the file path

Byte 3: Second Character of the file path

Byte n: **Last Character**

Byte n+1: **NULL**

Notes: SD card has to be formatted in the supported file system.

A read-only or opened file cannot be deleted.

Supported File Systems: FAT12, FAT16, FAT32

About the File Path:

- File Path specifies the full path to the file on SD including directory, filename and extension
- Directories should be separated by: / (**not by:** \ like in Windows and DOS).
- File Path is not case-sensitive. The drive and root directory do not have to be indicated, for example, both: A:/Cat/Jumped/Over.txt and cat/jumped/over.TXT specify the same file.
- Long file names are supported, however the File Path (directory + filename + extension + NULL) may not exceed 64 bytes.
- Wildcards are not allowed.

ezLCD Response

After receiving the SD_FILE_DELETE command, the ezLCD responds with either of the following sequences:

In case of the **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**0**	**1**	**0**	Byte 0: **3A**$_{HEX}$, **58**$_{DEC}$

In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**$_{HEX}$, **62**$_{DEC}$

The ezLCD response is sent through the same interface, which received the SD_FILE_DELETE command.

Example:
The following sequence will delete file MyFile.dat

SD_FILE_DELETE	**7D**	HEX
'M'	**4D**	HEX
'y'	**79**	HEX
'F'	**46**	HEX
'i'	**69**	HEX
'l'	**6C**	HEX
'e'	**65**	HEX
'.'	**2E**	HEX
'd'	**64**	HEX
'a'	**63**	HEX
't'	**74**	HEX
NULL	**0**	HEX

If the file has successfully been deleted, the ezLCD responds with the following sequence:

3A	HEX

In case of the failure, the following sequence will be sent by the ezLCD:

3E	HEX

2.52 SD_FILE_GET_SIZE

Description: Gets the size (in bytes) of the opened SD Flash file.

Code: **74**HEX, **116**DEC

7 6 5 4 3 2 1 0	
SD_FILE_GET_SIZE	Byte 0: **Command**
FILE ID	Byte 1: **File ID**

Notes: SD card has to be formatted in the supported file system.

This command works only if the file is already opened by the SD_FILE_OPEN command

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_OPEN, SD_FILE_CLOSE, SD_FILE_CLOSE_ALL

About the File ID:

File ID is returned in the response to the SD_FILE_OPEN command. It identifies the file after it has been opened. Since maximum 2 files may be concurrently opened, the File ID should be: 1 or 2.

Values higher than 2 are interpreted as 2 and 0 is interpreted as 1.

ezLCD Response

After receiving the SD_FILE_GET_SIZE command, the ezLCD responds with either of the following sequences:

In case of the **success**:

32-bit File Size

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**0**	**1**	Byte 0: **3D**hex, **63**dec
File Size 0								
File Size 1								
File Size 2								
File Size 3								

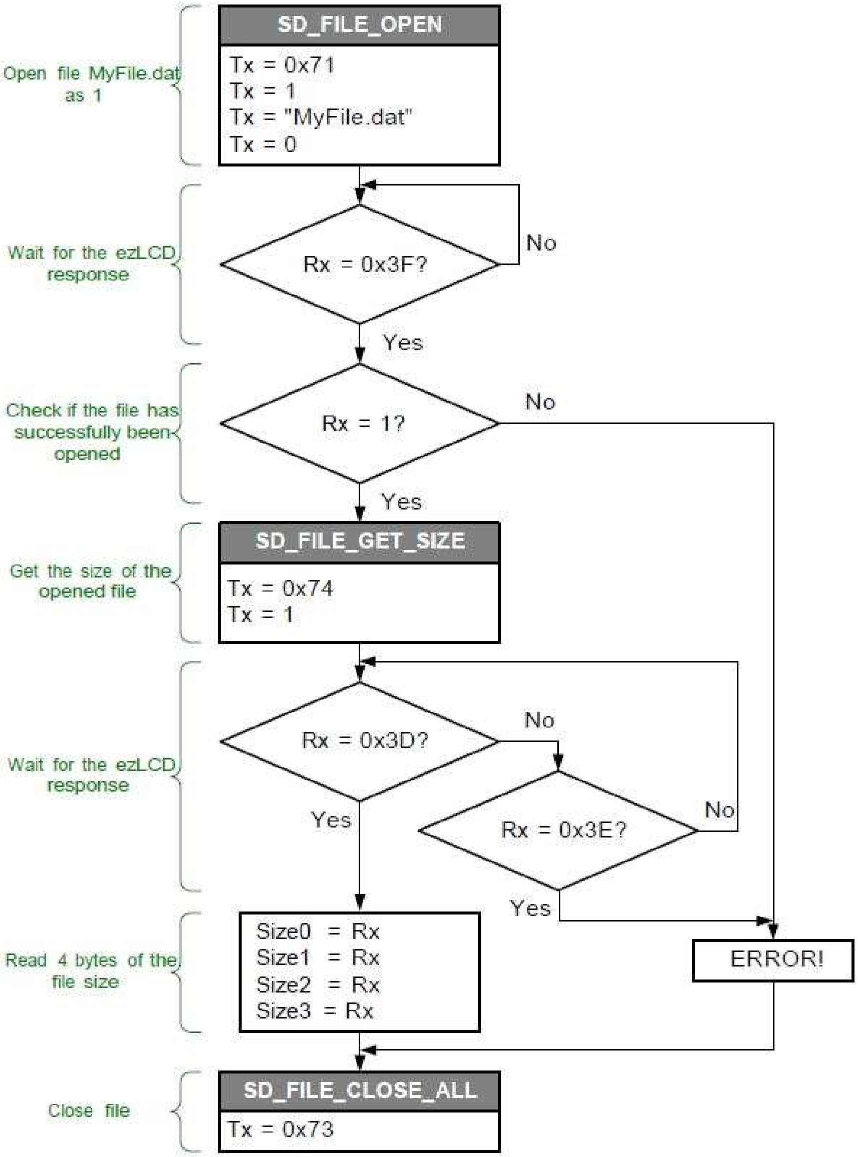
In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**HEX, **62**DEC

The ezLCD response is sent through the same interface, which received the SD_FILE_GET_SIZE command.

Example:

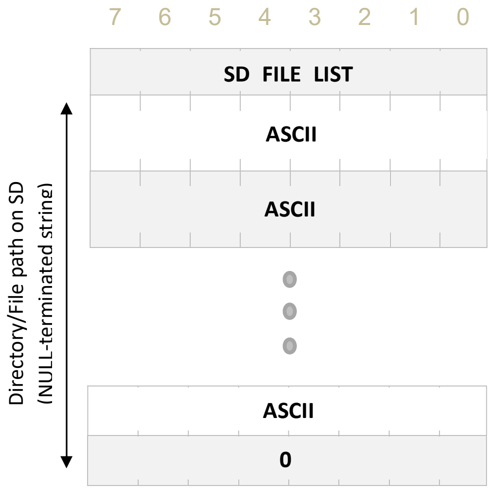
The following flow chart on the next page shows an example of getting the size of the file MyFile.dat



2.53 SD_FILE_LIST

Description: Gets the list of files and sub-directories which reside in the specified SD Directory. This command is similar to the DOS "dir" command.

Code: **79**HEX, **121**DEC



Byte 0: Command

Byte 2: First Character of the file path

Byte 3: Second Character of the file path

Byte n: Last Character

Byte n+1: NULL

Notes: SD card has to be formatted in the supported file system.

Supported File Systems: FAT12, FAT16, FAT32

About the SD Directory/File Path:

- Directory Path specifies the path to the SD directory, SD file or group of files and sub-directories.
- Wildcards: '*' and '?' are supported
- Directories should be separated by: / (**not by:** \ like in Windows and DOS).
- Directory Path is not case-sensitive. The drive and root directory do not have to be indicated, for example: A:/Cat/Jumped/Over, CAT/juMped/OvEr/ and cat/jumped/over specify the same.
- Long directory names are supported; however, the Directory Path NULL may not exceed 64 bytes.

ezLCD Response

After receiving the SD_FILE_LIST command, the ezLCD responds with either of the following sequences:

In case of the **success**:

(**3A**$_{\text{HEX}}$, **58**$_{\text{DEC}}$), followed by the NULL-terminated string containing the directory files and sub-directories list:

- Entries (files or sub-directories) are separated by the Line Feed character (**0A**$_{\text{HEX}}$ or **10**$_{\text{DEC}}$)
- Entries are sent in no particular order
- Sub-directories have '/' as their last character

For example:

3A$_{\text{HEX}}$	*Start*
whatever.txt	*file*
Pictures/	*directory*
Cat.doc	*file*
ezLCD.bin	*file*
SOURCES/	*directory*
0	*End (NULL)*

In case of an **error**:

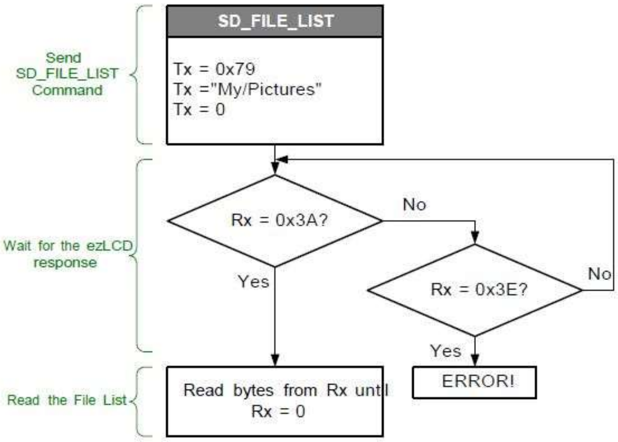
7	6	5	4	3	2	1	0
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**

Byte 0: **3E**$_{\text{HEX}}$, **62**$_{\text{DEC}}$

The ezLCD response is sent through the same interface, which received the SD_FILE_LIST command

Example:

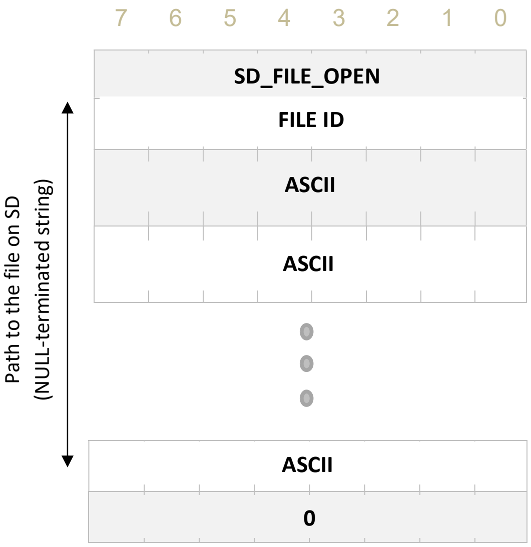
The following flow chart shows an example of reading the file list from the directory



2.54 SD_FILE_OPEN

Description: Opens an **existing** SD Flash file for reading or writing. File Position Index is set to 0. In order to open non-existing, new file, use the command SD_FILE_CREATE

Code: **71**HEX, **113**DEC



Byte 0: Command

Byte 1: File ID

Byte 2: First Character of the file path

Byte 3: Second Character of the file path

Byte n: Last Character

Byte n+1: NULL

Notes: SD card has to be formatted in the supported file system.

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_CREATE, SD_FILE_CLOSE, SD_FILE_CLOSE_ALL

About the File ID:

File ID identifies the file after it has been opened. Since maximum 2 files may be concurrently opened, the File ID should be: 1 or 2. Values higher than 2 are interpreted as 2 and 0 is interpreted as 1.

About the File Path:

- File Path specifies the full path to the file on SD including directory, filename and extension
- Directories should be separated by: / (**not by:** \ like in Windows and DOS).
- File Path is not case-sensitive. The drive and root directory do not have to be indicated, for example, both: A:/Cat/Jumped/Over.txt and cat/jumped/over.TXT specify the same file.
- Long file names are supported, however the File Path (directory + filename + extension + NULL) may not exceed 64 bytes.

ezLCD Response

After receiving the SD_FILE_OPEN command, the ezLCD responds with the following sequence:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**1**	Byte 0: **3F**$_{HEX}$, **63**$_{DEC}$
File ID / Error								Byte 1: **File ID** (Success), 0 (Error)

The ezLCD response is sent through the same interface, which received the SD_FILE_OPEN command.

Touch Screen Processing:

SD_FILE_OPEN command temporary disables the touch screen.

The touch screen will be automatically re-enabled when all files are closed. This can be done by issuing the SD_FILE_CLOSE or SD_FILE_CLOSE_ALL command.

Note: The touch screen is temporary disabled, even if due to error no file is opened. If this is the case, issuing "dummy" SD_FILE_CLOSE or SD_FILE_CLOSE_ALL command will re-enable the touch screen.

Example:
The following sequence will open file MyFile.dat

SD_FILE_OPEN	**71**	HEX	
1	1	DEC	(**File ID**)
'M'	4D	HEX	
'y'	79	HEX	
'F'	46	HEX	
'i'	69	HEX	
'l'	6C	HEX	
'e'	65	HEX	
'.'	2E	HEX	
'd'	64	HEX	
'a'	63	HEX	
't'	74	HEX	
NULL	0	HEX	

If the file has successfully been opened, the ezLCD responds with the following sequence:

3F	HEX
1	DEC (file ID)

In case of the failure, the following sequence will be sent by the ezLCD:

3F	HEX
0	DEC (indicates error)

2.55 SD_FILE_READ

Description: Reads the specified number of bytes from the opened SD Flash file, starting from File Position Index. File Position Index is incremented by the number of the bytes read, however it will not exceed (FILE_SIZE – 1).

Code: **75**$_{\text{HEX}}$, **117**$_{\text{DEC}}$

7 6 5 4 3 2 1 0

32-bit No of Bytes to read

SD_FILE_READ	Byte 0: **Command**
FILE ID	Byte 1: **File ID**
Number of Bytes to Read 0	Byte 2: Number of Bytes to Read 0 (LSB)
Number of Bytes to Read 1	Byte 3: Number of Bytes to Read 1
Number of Bytes to Read 2	Byte 4: Number of Bytes to Read 2
Number of Bytes to Read 3	Byte 5: Number of Bytes to Read 3 (MSB)

Notes: SD card has to be formatted in the supported file system. This command works only if the file is already opened by the SD_FILE_OPEN command

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_OPEN, SD_FILE_CLOSE, SD_FILE_CLOSE_ALL

About the File ID:

File ID is returned in the response to the SD_FILE_OPEN command. It identifies the file after it has been opened. Since maximum 2 files may be concurrently opened, the File ID should be: 1 or 2. Values higher than 2 are interpreted as 2 and 0 is interpreted as 1.

ezLCD Response

After receiving the SD_FILE_READ command, the ezLCD responds with either of the following sequences:

In case of the **success**:

Read Data Bytes

7 6 5 4 3 2 1 0	
$3C_{HEX}$	Byte 0: Indicates Success
Data Byte 0	Byte 1: Data Byte 0
Data Byte 1	Byte 2: Data Byte 1
⋮	
Data Byte n	Byte n+1: Data Byte n

Note: If the Number of Bytes to Read is greater than the number of bytes left in the file, all of the extra bytes will be preempted by 0.

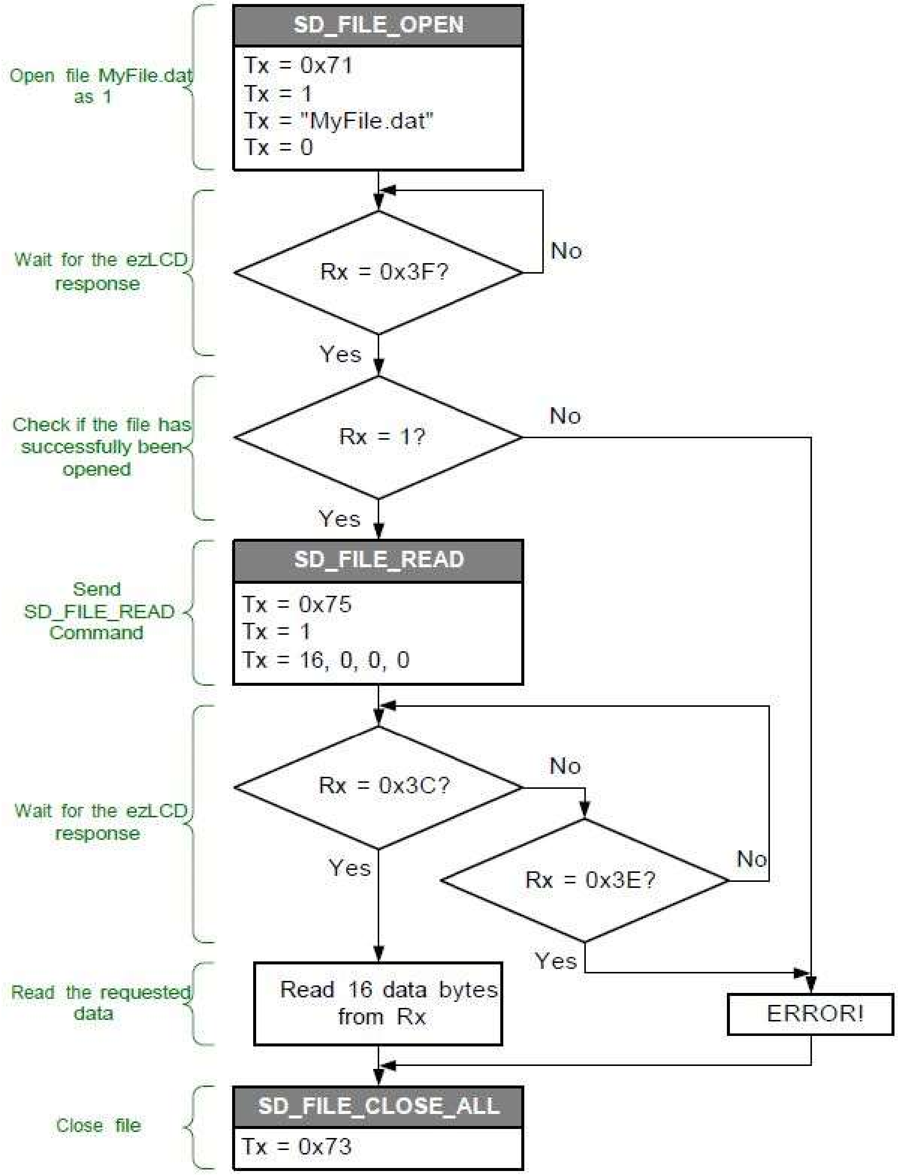
In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **$3E_{HEX}$**, **62_{DEC}**

The ezLCD response is sent through the same interface, which received the SD_FILE_READ command.

Example:

The following flow chart shows an example of reading first 16 bytes from the file MyFile.dat



2.56 SD_FILE_REWIND [Currently NOT Supported]

Description: Moves the File Position Index to the beginning of the opened SD Flash file.

Code: **7A**HEX, **122**DEC

7 6 5 4 3 2 1 0	
SD_FILE_REWIND	Byte 0: Command
FILE ID	Byte 1: File ID

Notes: SD card has to be formatted in the supported file system.

This command works only if the file is already opened by the SD_FILE_OPEN command, or created and opened by the SD_FILE_CREATE command.

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_OPEN, SD_FILE_SEEK, SD_FILE_READ, SD_FILE_WRITE, SD_FILE_CLOSE, SD_FILE_CLOSE_ALL

About the File ID:

File ID is returned in the response to the SD_FILE_OPEN command. It identifies the file after it has been opened. Since maximum 2 files may be concurrently opened, the File ID should be: 1 or 2. Values higher than 2 are interpreted as 2 and 0 is interpreted as 1.

About the File Position Index

The File Position Index specifies the Read/Write position offset (in bytes) from the beginning of the file. Upon opening of the file, the File Position Index is set to 0. The File Position Index is incremented by the subsequent read or write operations on the opened file.

ezLCD Response

After receiving the SD_FILE_REWIND command, the ezLCD responds with either of the following sequences:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**0**	**0**	**1**	Byte 0: **39**$_{HEX}$, **57**$_{DEC}$

In case of the **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**$_{HEX}$, **62**$_{DEC}$

In case of an **error**:

The ezLCD response is sent through the same interface, which received the SD_FILE_REWIND command.

Example:

The following sequence will set the File Position Index at the beginning of the file.

SD_FILE_REWIND	**7A**	HEX
1	**1**	DEC (File ID)

If the File Position Index has successfully been moved, the ezLCD responds with the following sequence:

39	HEX

In case of the failure, the following sequence will be sent by the ezLCD:

3E	HEX

2.57 SD_FILE_SEEK [Currently NOT Supported]

Description: Moves the File Position Index of the opened SD Flash file by the specified number of bytes, from the position specified by the 'Whence' parameter.

Code: **7C**HEX, **124**DEC

32-bit Signed Offset

7	6	5	4	3	2	1	0	
SD_FILE_READ								Byte 0: Command
FILE ID Offset 0								Byte 1: File ID
Offset 1								Byte 2: Offset 0 (LSB)
Offset 2								Byte 3: Offset 1
Offset 3								Byte 4: Offset 2
								Byte 5: Offset 3 (MSB)
Whence								Byte 6: 0 = from File Beginning (SD_SEEK_SET) 1 = From Current File Position Index (SD_SEEK_CUR) 2 = From File End (SD_SEEK_END)

Notes: SD card has to be formatted in the supported file system. This command works only if the file is already opened by the SD_FILE_OPEN command, or created and opened by the SD_FILE_CREATE command.

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_OPEN, SD_FILE_REWIND, SD_FILE_READ, SD_FILE_WRITE, SD_FILE_CLOSE, SD_FILE_CLOSE_ALL

About the Offset:

Offset is specified by the 32-bit signed integer. When the offset is negative, the File Position Index is moved backwards.

About the File Position Index

The File Position Index specifies the Read/Write position offset (in bytes) from the beginning of the file. Upon opening of the file, the File Position Index is set to 0. The File Position Index is incremented by the subsequent read or write operations on the opened file.

About the File ID:

File ID is returned in the response to the SD_FILE_OPEN command. It identifies the file after it has been opened. Since maximum 2 files may be concurrently opened, the File ID should be: 1 or 2. Values higher than 2 are interpreted as 2 and 0 is interpreted as 1.

ezLCD Response

After receiving the SD_FILE_SEEK command, the ezLCD responds with either of the following sequences:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**0**	**0**	**1**	Byte 0: **39**$_{HEX}$, **57**$_{DEC}$

In case of the **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**$_{HEX}$, **62**$_{DEC}$

In case of an **error**:

The ezLCD response is sent through the same interface, which received the SD_FILE_SEEK command.

Example:
The following sequence will advance the File Position Index by 23 bytes.

SD_FILE_SEEK	**7C**	HEX	
1	**1**	DEC	(**File ID**)
23	**23**	DEC	(**Offset** LSB)
0	**0**	DEC	
0	**0**	DEC	
0	**0**	DEC	(**Offset** MSB)
Whence	**1**	DEC	(from the current File Position Index)

If the File Position Index has successfully been moved, the ezLCD responds with the following sequence:
39 HEX

In case of the failure, the following sequence will be sent by the ezLCD:
3E HEX

2.58 SD_FILE_TELL

Description: Gets the File Position Index of the opened SD Flash file.

Code: **7B**$_{HEX}$, **123**$_{DEC}$

7 6 5 4 3 2 1 0	
SD_FILE_TELL	Byte 0: Command
FILE ID	Byte 1: File ID

Notes: SD card has to be formatted in the supported file system. This command works only if the file is already opened by the SD_FILE_OPEN command, or created and opened by the SD_FILE_CREATE command.

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_OPEN, SD_FILE_SEEK, SD_FILE_CLOSE, SD_FILE_CLOSE_ALL

About the File ID:

File ID is returned in the response to the SD_FILE_OPEN command. It identifies the file after it has been opened. Since maximum 2 files may be concurrently opened, the File ID should be: 1 or 2.

Values higher than 2 are interpreted as 2 and 0 is interpreted as 1.

About the File Position Index

The File Position Index specifies the Read/Write position offset (in bytes) from the beginning of the file. Upon opening of the file, the File Position Index is set to 0. The File Position Index is incremented by the subsequent read or write operations on the opened file.

ezLCD Response

After receiving the SD_FILE_TELL command, the ezLCD responds with either of the following sequences:

In case of the **success**:

32-bit File Size

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**0**	**1**	Byte 0: **3D**HEX, **61**DEC
File Position Index 0								Byte 1: File Position Index 0 (LSB)
File Position Index 1								Byte 2: File Position Index 1
File Position Index 2								Byte 3: File Position Index 2
File Position Index 3								Byte 4: File Position Index 3 (MSB)

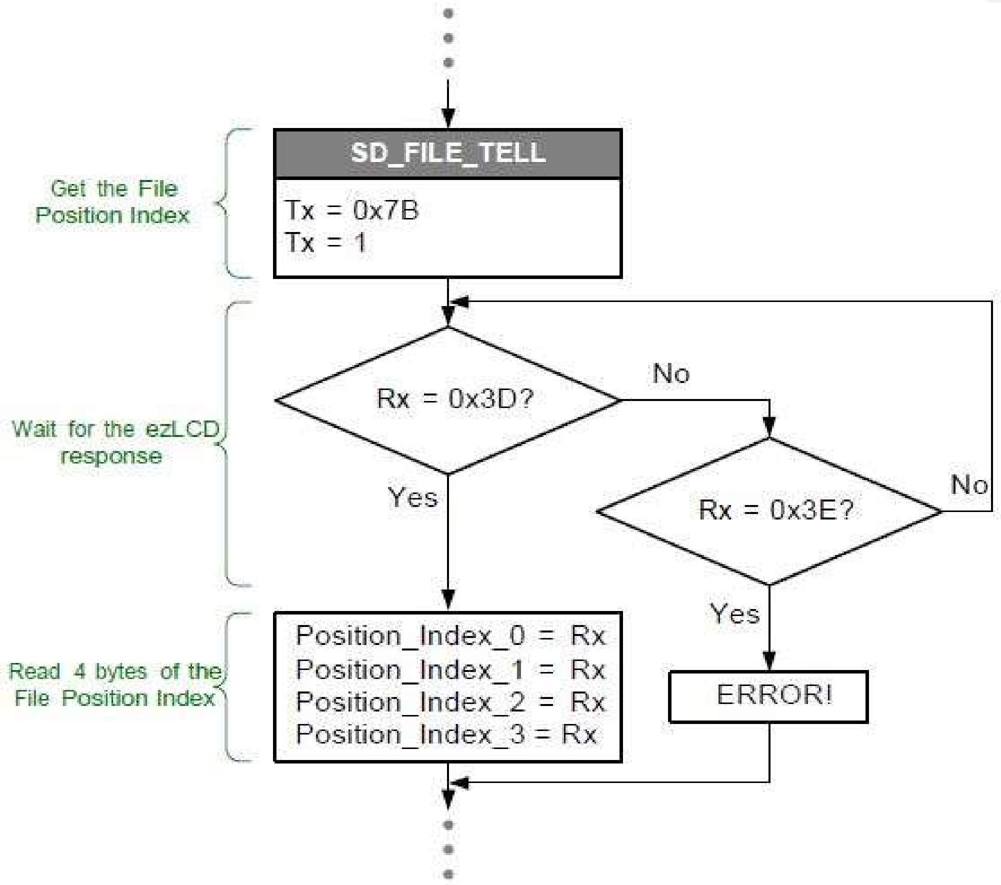
In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**HEX, **62**DEC

The ezLCD response is sent through the same interface, which received the SD_FILE_TELL command.

Example:

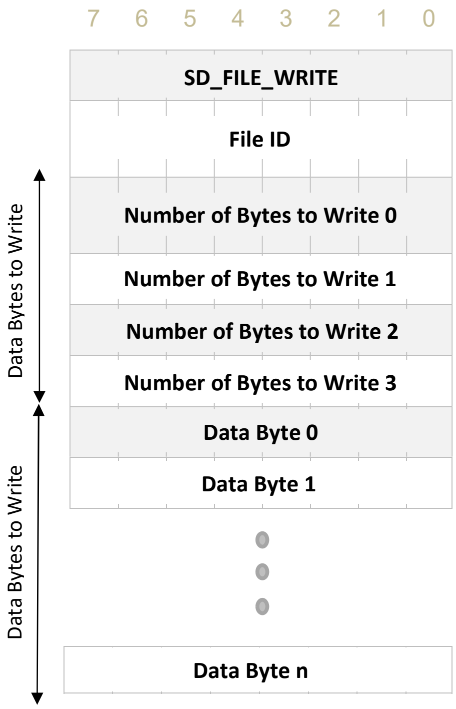
The following flow chart shows an example of getting the File Position Index of the opened file with File Id = 1.



2.59 SD_FILE_WRITE

Description: Writes the specified number of bytes to the opened SD Flash file, starting from File Position Index. File Position Index is incremented by the number of the bytes written.

Code: **77**HEX, **119**DEC



Byte 0: Command

Byte 2: First Character of the file path

Byte 3: Second Character of the file path

Byte n+1: Data Byte n

Notes: SD card has to be formatted in the supported file system.

This command works only if the file is already opened by the SD_FILE_OPEN command, or created and opened by the SD_FILE_CREATE command.

Supported File Systems: FAT12, FAT16, FAT32

See Also: SD_FILE_CREATE, SD_FILE_OPEN, SD_FILE_CLOSE, SD_FILE_CLOSE_ALL

About the File ID:

File ID is returned in the response to the SD_FILE_CREATE or SD_FILE_OPEN command. It identifies the file after it has been opened. Since maximum 2 files may be concurrently opened, the File ID should be: 1 or 2. Values higher than 2 are interpreted as 2 and 0 is interpreted as 1.

ezLCD Response

After receiving the SD_FILE_WRITE command, the ezLCD responds with either of the following sequences:

In case of the **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**0**	**1**	**1**	Byte 0: **3B**$_{\text{HEX}}$, **59**$_{\text{DEC}}$

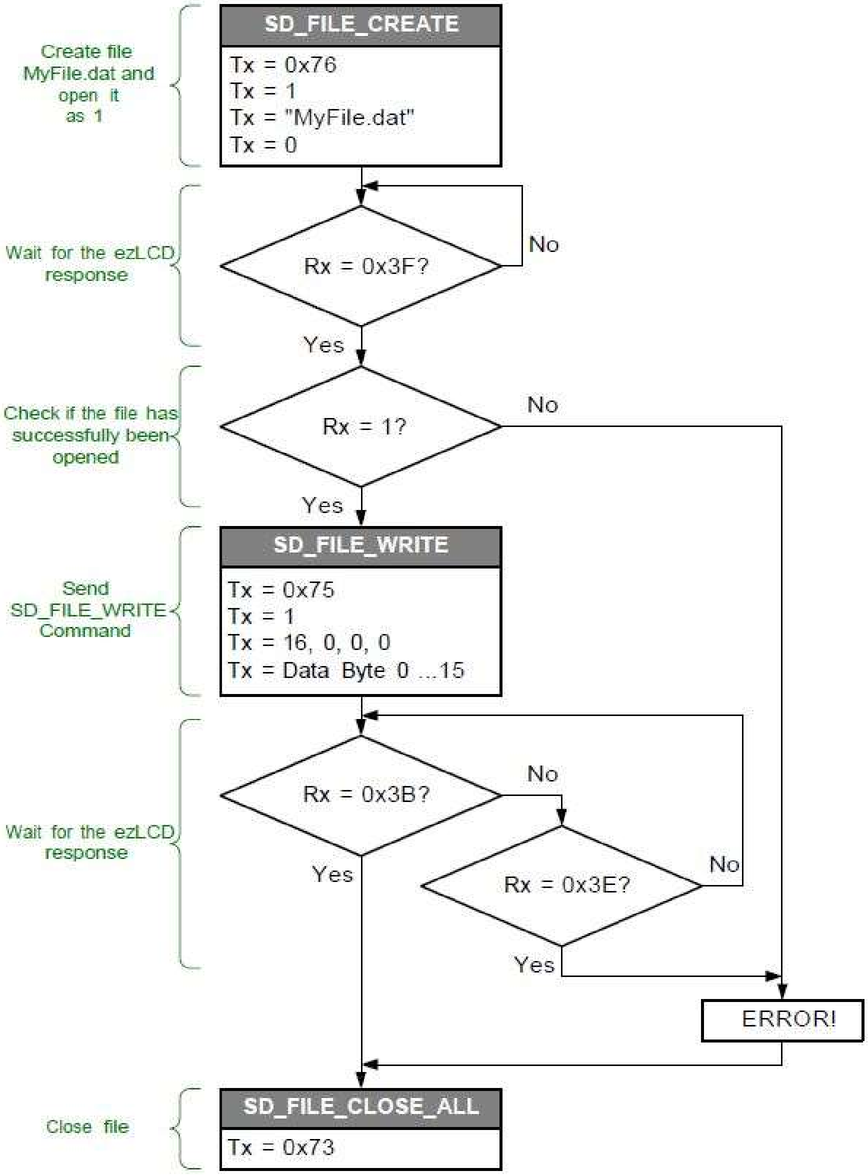
In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**$_{\text{HEX}}$, **62**$_{\text{DEC}}$

The ezLCD response is sent through the same interface, which received the SD_FILE_WRITE command.

Example:

The following flow chart on the next page shows an example of writing 16 bytes into the created file MyFile.dat



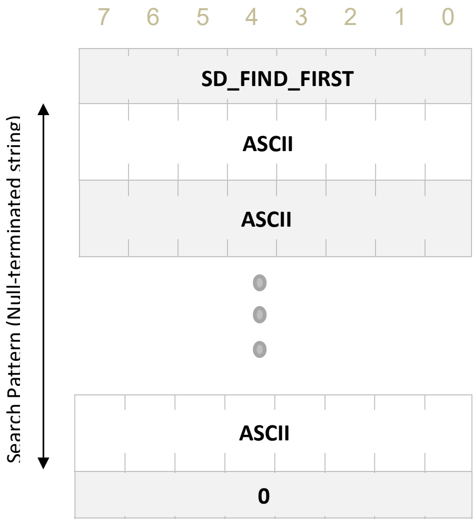
2.60 SD_FIND_FIRST and SD_FIND_NEXT [Currently NOT Supported]

Description: Obtain the list of SD files and sub-directories (one by one), which match the specified search pattern.

Code: **SD_FIND_FIRST**: **4A**HEX, **74**DEC

SD_FIND_NEXT: **4B**HEX, **75**DEC

Wildcards: '*' and '?' are supported for the ASCII



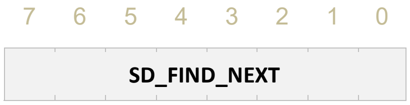
Byte 0: Command

Byte 2: First Character of the pattern

Byte 3: Second Character of the pattern

Byte n: Last Character of the folder path

Byte n+1: NULL



Byte 0: Command

SD_FIND_FIRST gets only the first found file or directory, which matches the search pattern.

Each time, the SD_FIND_NEXT is issued, it finds the next file or directory, which matches the search pattern specified in the last SD_FIND_FIRST command.

The difference between the above described mechanism and SD_FILE_LIST command is that it obtains the files and directories one by one, while SD_FILE_LIST obtains them all at once.

Note: SD card has to be formatted in the supported file system.

Supported File Systems: FAT12, FAT16, FAT32

About the Search Pattern:

- Specifies the path to the SD directory, SD file or group of files and sub-directories.
- Wildcards: '*' and '?' are supported
- Directories should be separated by: / (**not by:** \ like in Windows and DOS).
- Search Pattern is not case-sensitive. The drive and root directory do not have to be indicated, for example: A:/Cat/Jumped/Over, CAT/juMped/OvEr/ and cat/jumped/over specify the same.
- Long directory and file names are supported, however the Search Pattern + NULL may not exceed 64 bytes.

See Also: SD_FILE_LIST

ezLCD Response

After receiving any of the described commands, the ezLCD responds with either of the following sequences:

In case of the **success**:

(**3A**$_{\text{HEX}}$, **58**$_{\text{DEC}}$), followed by the NULL-terminated string containing file or directory name. Directories have '/' as their last character

Examples:

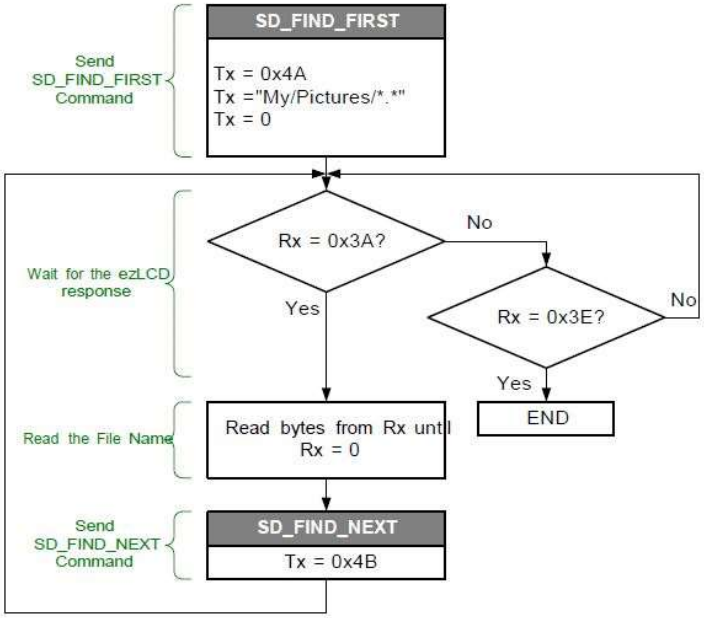
3A$_{\text{HEX}}$	*Start*
whatever.txt	*file*
0	*End (NULL)*
or	
3A$_{\text{HEX}}$	*Start*
Pictures/	*directory*
0	*End (NULL)*

In case no files were found or in case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**$_{\text{HEX}}$, **62**$_{\text{DEC}}$

The ezLCD response is sent through the same interface, which received the command

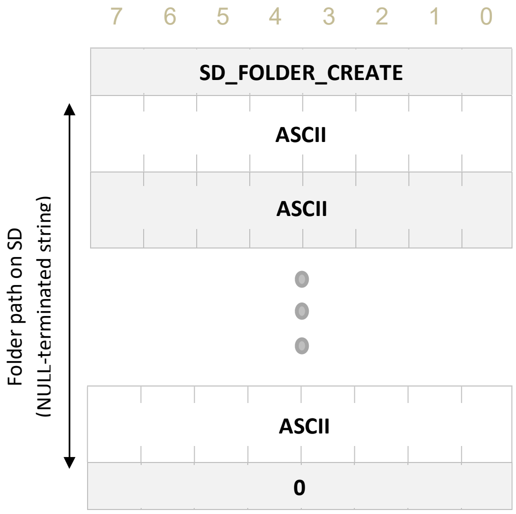
Example:
The following flow chart shows an example of reading the file list from the directory "My/Pictures".



2.61 SD_FOLDER_CREATE [Currently NOT Supported]

Description: Creates a new folder (directory) on the SD. This command is similar to the DOS "mkdir" command.

Code: **46**HEX, **70**DEC



Byte 0: Command

Byte 2: First Character of the folder path

Byte 3: Second Character of the folder path

Byte n: Last Character of the folder path

Byte n+1: NULL

Notes: SD card has to be formatted in the supported file system. Parent directory (folder) has to exist.

Supported File Systems: FAT12, FAT16, FAT32

About the Folder Path:

- Folder Path specifies the full path to the directory on the SD.
- Directories (folders) should be separated by: / (**not by:** \ like in Windows and DOS).
- Long names are supported, however the Folder Path (+ NULL) may not exceed 64 bytes.

ezLCD Response

After receiving the SD_FOLDER_CREATE command, the ezLCD responds with either of the following sequences:

In case of the **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**0**	**1**	**1**	Byte 0: **3B**$_{\text{HEX}}$, **59**$_{\text{DEC}}$

In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**$_{\text{HEX}}$, **62**$_{\text{DEC}}$

The ezLCD response is sent through the same interface, which received the SD_FOLDER_CREATE command.

Example:
The following sequence will create a folder named “MyDir” in the root directory

SD_FOLDER_CREATE	**46**	HEX
'M'	4D	HEX
'y'	79	HEX
'D'	44	HEX
'i'	69	HEX
'r'	72	HEX
NULL	0	HEX

If the folder has successfully been created, the ezLCD responds with the following sequence:

3B	HEX

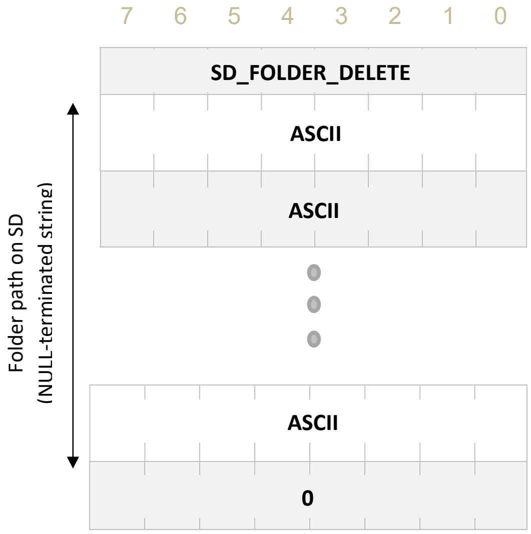
In case of the failure, the following sequence will be sent by the ezLCD:

3E	HEX

2.62 SD_FOLDER_DELETE [Currently NOT Supported]

Description: Deletes an empty folder (directory) on the SD. This command is similar to the DOS "rmdir" command.

Code: **4D**HEX, **77**DEC



Byte 0: Command

Byte 2: First Character of the folder path

Byte 3: Second Character of the folder path

Byte n: Last Character of the folder path

Byte n+1: NULL

Notes: SD card has to be formatted in the supported file system. Folder (directory) has to be empty

Supported File Systems: FAT12, FAT16, FAT32

About the Folder Path:

- Folder Path specifies the full path to the directory on the SD.
- Directories (folders) should be separated by: / (**not by:** \ like in Windows and DOS).
- Long names are supported, however the Folder Path (+ NULL) may not exceed 64 bytes.
- Wildcards are not allowed.

ezLCD Response

After receiving the SD_FOLDER_DELETE command, the ezLCD responds with either of the following sequences:

In case of the **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**0**	**1**	**0**	Byte 0: **3A**$_{\text{HEX}}$, **58**$_{\text{DEC}}$

In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**$_{\text{HEX}}$, **62**$_{\text{DEC}}$

The ezLCD response is sent through the same interface, which received the SD_FOLDER_DELETE command.

Example:
The following sequence will delete a folder named “MyDir” from the root directory

SD_FOLDER_DELETE	**4D**	HEX
'M'	4D	HEX
'y'	79	HEX
'D'	44	HEX
'i'	69	HEX
'r'	72	HEX
NULL	0	HEX

If the folder has successfully been deleted, the ezLCD responds with the following sequence:

3A	HEX

In case of the failure, the following sequence will be sent by the ezLCD:

3E	HEX

2.63 SD_FORMAT [Currently NOT Supported]

Description: Formats the SD in the specified file system.

Code: **4F**HEX, **79**DEC

7 6 5 4 3 2 1 0	
SD_FORMAT	Byte 0: Command
ASCII	Byte 1: First Character of the File System Specification
ASCII	Byte 2: Second Character of the File System Specification
ASCII	Byte 3: Third Character of the File System Specification
ASCII	Byte 4: Fourth Character of the File System Specification
ASCII	Byte 5: Fifth Character of the File System Specification

File System Specification

Warning: This command will erase all files on the SD

About the File System Specification:

- Sets the file system in which the SD will be formatted.
- 5 ASCII characters
- ASCII characters only. For example: the code of '1' is 31hex.
- Supported file systems: FAT12, FAT16, FAT32

About the supported File Systems:

	FAT12	FAT16	FAT32
Full Name	File Allocation Table		
	12-bit version	16-bit version	32-bit version
Introduced	1977	July 1988	August 1996
Max file size	32 MB	2 GB	4 GB
Max number of files	4,077	65,517	268,435,437
Max Volume size	32 MB	2 GB	8 TB

ezLCD Response

After receiving the SD_FORMAT command, the ezLCD responds with either of the following sequences:

In case of the **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**0**	**1**	**0**	Byte 0: **3A**$_{HEX}$, **58**$_{DEC}$

In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**$_{HEX}$, **62**$_{DEC}$

The ezLCD response is sent through the same interface, which received the SD_FORMAT command.

Example:

The following sequence will format the SD in FAT16

SD_FORMAT	**4F**	HEX
'F'	**46**	HEX
'A'	**41**	HEX
'T'	**54**	HEX
'1'	**31**	HEX
'6'	**36**	HEX

If the folder has successfully been deleted, the ezLCD responds with the following sequence:

	3A	HEX

In case of the failure, the following sequence will be sent by the ezLCD:

	3E	HEX

2.64 SD_INSERTED

Description: Checks if the SD card is inserted

Code: **49**$_{HEX}$, **73**$_{DEC}$

7	6	5	4	3	2	1	0
SD_INSERTED							

Byte 0: Command

ezLCD Response

After receiving the SD_INSERTED command, the ezLCD responds with either of the following sequences:

If there **is a card inserted** in the SD slot:

7	6	5	4	3	2	1	0
0	**0**	**1**	**1**	**1**	**1**	**0**	**1**

Byte 0: **3D**$_{HEX}$, **61**$_{DEC}$
(SD is inserted)

If there is **no card inserted** in the SD slot:

7	6	5	4	3	2	1	0
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**

Byte 0: **3E**$_{HEX}$, **62**$_{DEC}$
(SD not inserted/error)

The ezLCD response is sent through the same interface, which received the SD_INSERTED command.

Example:

The following sequence will check if the SD card is present in the SD slot

SD_INSERTED	**49**	HEX

If an SD card is present in the SD slot:

	3D	HEX

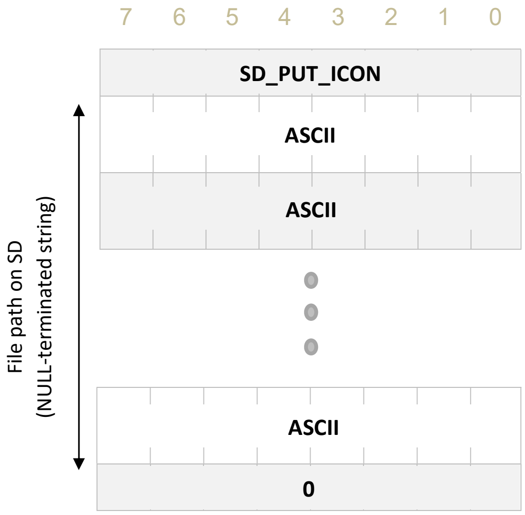
If there is no card inserted in the SD slot:

	3E	HEX

2.65 SD_PUT_ICON

Description: Displays an icon with its upper-left corner positioned at the Current Position. The icon is read from the file on the SD Flash card attached to the SD/MMC interface.

Code: **70**HEX, **112**DEC



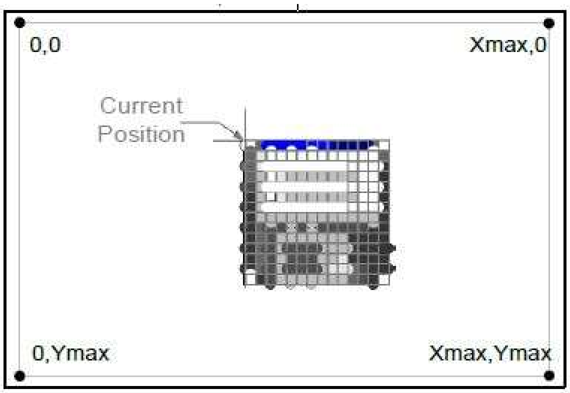
Byte 0: Command

Byte 1: First Character of the file path

Byte 3: Second Character of the file path

Byte n: Last Character of the file path

Byte n+1: NULL



Notes: SD card has to be formatted in the supported file system.

Supported File Systems: FAT12, FAT16, FAT32

Supported Formats: .jpg, 24-bit .bmp, .ezh, and ".ezp" (legacy)

About the File Path:

- File Path specifies the full path to the file on SD including directory, filename and extension
- Directories should be separated by: / (**not by:** \ like in Windows and DOS).
- File Path is not case-sensitive. The drive and root directory do not have to be indicated, for example, both: A:/Cat/Jumped/Over.txt and cat/jumped/over.TXT specify the same file.
- Long file names are supported, however the File Path (directory + filename + extension + NULL) may not exceed 64 bytes.

Recommended Formats:

It is strongly recommended to use jpg format due to the reduced storage requirements and faster display speed. .ezp and .ezh formats can be created using the ezLCD-5x Utility.

About the ".ezh" and ".ezp" (legacy) files:

The ".ezp" and ".ezh" files contain pre-processed bitmaps. They are displayed much faster than .bmp files. Also, these files support transparency. Files with many different image formats (such as .bmp and .png) can be converted to ".ezp" files using the ezLCD-5x Configuration Utility.

2.66 SD_SCREEN_CAPTURE

Description: Saves an image of the displayed screen to the SD as “.bmp” file.

Code: **44**HEX, **68**DEC

7	6	5	4	3	2	1	0	
SD_SCREEN_CAPTURE								Byte 0: Command

This command is helpful when writing the documentation of your ezLCD project, because the captured screen images may be used as examples. Screen capture files have names "Scr_xxxx.bmp", where xxxx is a consecutive number. For example: Scr_0001.bmp, Scr_0002.bmp, etc. The files are created in the "Scr_Cap" SD folder. If the SD does not have the "Scr_Cap" folder, it will be created automatically.

Notes: SD card has to be formatted in the supported file system. This command may take up to 2 seconds to execute.

Supported File Systems: FAT12, FAT16, FAT32

ezLCD Response:

After execution of the SD_SCREEN_CAPTURE command, the ezLCD responds with either of the following sequences:

In case of **success**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**0**	**1**	**1**	Byte 0: **3B**HEX, **59**DEC

In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**HEX, **62**DEC

The ezLCD response is sent through the same interface, which received the SD_SCREEN_CAPTURE command.

Example:

The following sequence will save the image of the displayed screen to the SD file.

SD_SCREEN_CAPTURE **44** HEX

If the screen image has been written to the .bmp file, the ezLCD responds with:

3B HEX

In case of a failure, the following byte will be sent by the ezLCD:

3E HEX

2.67 SD_SIZE [Currently NOT Supported]

Description: Gets the physical size (in bytes) of the SD Card.

Code: **78**HEX, **120**DEC

7	6	5	4	3	2	1	0	
SD_SIZE								Byte 0: Command

ezLCD Response:

After receiving the SD_SIZE command, the ezLCD responds with either of the following sequences:

In case of **success**:

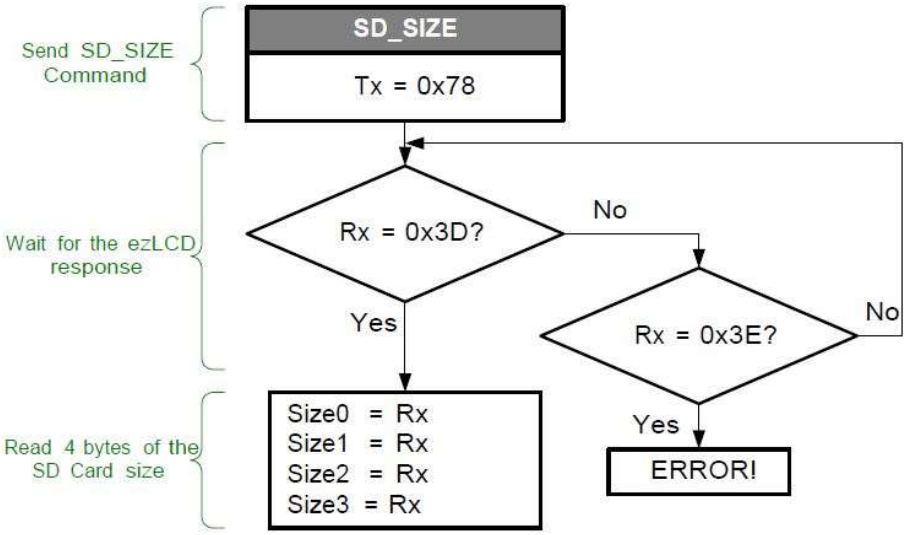
	7	6	5	4	3	2	1	0	
	0	0	1	1	1	1	0	1	Byte 0: Command
32-bit SD Card Size	**SD Card Size 0**								Byte 1: SD Card Size 0 (LSB)
	SD Card Size 1								Byte 2: SD Card Size 1
	SD Card Size 2								Byte 3: SD Card Size 2
	SD Card Size 3								Byte 4: SD Card Size 3

In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**HEX, **62**DEC

The ezLCD response is sent through the same interface, which received the SD_SIZE command.

Example:
The following flow chart shows an example of getting the size of the SD Card.



2.68 SD_SPACE_INFO

Description: Gets the information about the space usage (in bytes) of the formatted SD Card.

Code: **48**HEX, **72**DEC

7	6	5	4	3	2	1	0	
SD_SPACE_INFO								Byte 0: Command
What								Byte 1: 1 = Get Free Space, 2 = Get Used Space, Other = Total Formatted Space

Notes: SD card has to be formatted in the supported file system.

Supported File Systems: FAT12, FAT16, FAT32

ezLCD Response:

After receiving the SD_SPACE_INFO command, the ezLCD responds with either of the following sequences:

In case of **success**:

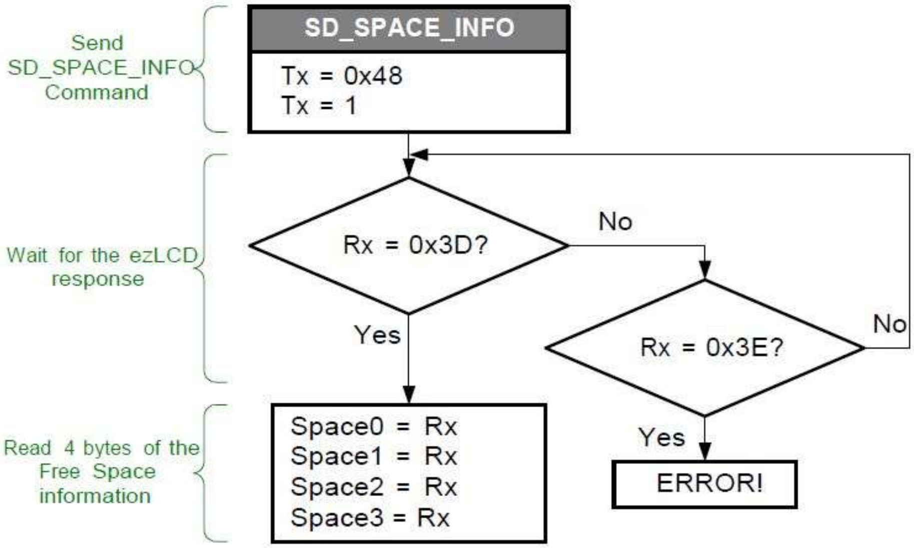
	7	6	5	4	3	2	1	0	
	0	0	1	1	1	1	0	1	Byte 0: **3D**HEX, **61**DEC
32-bit Space Info	**Space 0**								Byte 1: Space 0 (LSB)
	Space 1								Byte 2: Space 1
	Space 2								Byte 3: Space 2
	Space 3								Byte 4: Space 3 (MSB)

In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	Byte 0: **3E**HEX, **62**DEC

The ezLCD response is sent through the same interface, which received the SD_SPACE_INFO command.

Example:
The following flow chart shows an example of getting the number of the available bytes on the formatted SD card.



2.69 SELECT_FONT

Description: Sets the Current Font.

Code: **2B**HEX, **43**DEC

7 6 5 4 3 2 1 0	
SELECT_FONT	Byte 0: Command
font number	Byte 1: Font Number

Note: Fonts (in the form of TTF or EZF) are stored on the SD Card. Font numbering is defined using either the ezLCD 5x Configuration Utility or via the config.txt file. In the event that no fonts are installed (the factory default), a single 8x8 font is available as font number 0.

Commented [R13]: Check if Factory font still avail as font 0

See Also: PRINT_STRING, PRINT_CHAR

Example:
The following sequence will print a black character 'M' in the middle of the screen using Font 2.

SELECT_FONT	**2B**	HEX
2	**2**	DEC
SET_COLORH	84	HEX
BLACK_LSB	00000000	BIN
BLACK_MSB	00000000	BIN
PRINT_CHAR	2C	HEX
'M'	4D	HEX

Commented [R14]: Modify with SET_COLOR_RGB for example

2.70 SET_ALPHA

Description: Sets the Alpha (transparency) value.

Code: **20**HEX, **32**DEC

7 6 5 4 3 2 1 0	
SELECT_ALPHA	Byte 0: Command
alpha	Byte 1: Alpha Value

See Also: SET_COLOR_RGB

Commented [R15]: Add Transparency reference and section from 10x manual

Example:
The following sequence will draw a semi-transparent color filled circle

SET_ALPHA	**20**	HEX
	128	DEC
CIRCLE_RH_FILL	**99**	HEX
	0	DEC (Radius MSB)
	80	DEC (Radius LSB)

2.71 SET_BG_COLOR_RGB

Description: Sets the Background Color using RGB888 for the following instructions: PRINT_CHAR_BG, PRINT_STRING_BG

Code: **32**HEX, **50**DEC

Min Firmware: **1.3.2**

7	6	5	4	3	2	1	0	
SET_BG_COLOR_RGB								Byte 0: Command
R7	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 1: Red
G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 2: Green
B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 3: Blue

See Also: PRINT_CHAR_BG, PRINT_STRING_BG

Example:
The following sequence will print "LCD" in yellow on a navy background, using Font 0.

SET_BG_COLOR_RGB	**32**	HEX	
NAVY_R	**00000000**	BIN	
NAVY_G	**00000000**	BIN	
NAVY_B	**01000000**	BIN	
SET_COLORH	84	HEX	
YELLOW_LSB	11100000	BIN	
YELLOW_MSB	11111111	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
SELECT_FONT	2B	HEX	
0	0	DEC	
PRINT_STRING_BG	3D	HEX	
'L'	4C	HEX	
'C'	43	HEX	
'D'	44	HEX	
NULL	0	HEX	

Commented [R16]: Update with SET_COLOR_RGB for example

2.72 SET_BG_COLORH

NOTE: Legacy deprecated command. Recommended Replacement: SET_BG_COLOR_RGB

Description: Sets the Background Color for the following instructions:
PRINT_CHAR_BG, PRINT_STRING_BG

Code: $\mathbf{94}_{\text{HEX}}$, $\mathbf{148}_{\text{DEC}}$

7	6	5	4	3	2	1	0	
SET_BG_COLORH								Byte 0: Command
G2	**G1**	**G0**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 1: Color LSB
R4	**R3**	**R2**	**R1**	**R0**	**G5**	**G4**	**G3**	Byte 2: Color MSB

See Also: PRINT_CHAR_BG, PRINT_STRING_BG

Example:
The following sequence will print "LCD" in yellow on a navy background, using Font 0.

SET_BG_COLORH	**94**	HEX	
NAVY_LSB	**00010000**	BIN	
NAVY_MSB	**00000000**	BIN	
SET_COLORH	84	HEX	
YELLOW_LSB	11100000	BIN	
YELLOW_MSB	11111111	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
160	160	DEC	(**X** LSB)
117	117	DEC	(**Y**)
SELECT_FONT	2B	HEX	
0	0	DEC	
PRINT_STRING_BG	3D	HEX	
'L'	4C	HEX	
'C'	43	HEX	
'D'	44	HEX	
NULL	0	HEX	

2.73 SET_COLOR_KEY

Description: TBD

Commented [R17]: TBD after review of H757 manual

Code: **37**$_{\text{HEX}}$, **55**$_{\text{DEC}}$

7	6	5	4	3	2	1	0	
SET_COLOR_KEY								Byte 0: Command
R7	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 1: Red
G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 2: Green
B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 3: Blue

2.74 SET_COLOR_RGB

Description: Sets the Current Color using RGB888.

Code: **31**$_{HEX}$, **49**$_{DEC}$

Min Firmware: **1.3.2**

7	6	5	4	3	2	1	0	
SET_COLOR_RGB								Byte 0: Command
R7	**R6**	**R5**	**R4**	**R3**	**R2**	**R1**	**R0**	Byte 1: Red
G7	**G6**	**G5**	**G4**	**G3**	**G2**	**G1**	**G0**	Byte 2: Green
B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 3: Blue

See Also: CLS, PLOT

Example:
The following sequence will fill the whole screen with green.

SET_COLORH	**31**	HEX
GREEN_R	**00000000**	BIN
GREEN_G	**11111111**	BIN
GREEN_B	**00000000**	BIN
CLS	21	HEX

2.75 SET_COLORH

NOTE: Legacy deprecated command. Recommended Replacement: SET_COLOR_RGB

Description: Sets the Current Color.

Code: **84**HEX, **132**DEC

7	6	5	4	3	2	1	0	
SET_COLORH								Byte 0: Command
G2	**G1**	**G0**	**B4**	**B3**	**B2**	**B1**	**B0**	Byte 1: Color LSB
R4	**R3**	**R2**	**R1**	**R0**	**G5**	**G4**	**G3**	Byte 2: Color MSB

See Also: CLS, PLOT

Example:
The following sequence will fill the whole screen with green.

SET_COLORH	**84**	HEX
GREEN_LSB	**11100000**	BIN
GREEN_MSB	**00000111**	BIN
CLS	21	HEX

2.76 SET_BG_DISP_FRAME

Description: Selects background (layer 0) frame number to display

Code: **52**HEX, **82**DEC

7 6 5 4 3 2 1 0	
SET_BG_DISP_FRAME	Byte 0: Command
Frame No	Byte 1: Frame Number

Startup Value: At boot. The background layer is set to Frame 0

See Also: SET_LAYER_VISIBILITY, SET_DRAW_FRAME, COPY_FRAME, MERGE_FRAME

2.77 SET_FG_DISP_FRAME

Description: Selects foreground (layer 1) frame number to display

Code: **52**$_{\text{HEX}}$, **82**$_{\text{DEC}}$

7 6 5 4 3 2 1 0	
SET_FG_DISP_FRAME	Byte 0: Command
Frame No	Byte 1: Frame Number

Startup Value: At boot. The foreground layer is set to Frame 1

See Also: SET_LAYER_VISIBILITY, SET_DRAW_FRAME, COPY_FRAME, MERGE_FRAME

Commented [R18]: Set links

2.78 SET_DRAW_FRAME

Description: Selects the current frame used for all commands

Code: **51**HEX, **81**DEC

7 6 5 4 3 2 1 0	
SET_DRAW_FRAME	Byte 0: Command
Frame No	Byte 1: Frame Number

Startup Value: At boot. Frame 1 is the drawing frame

See Also: SET_LAYER_VISIBILITY, SET_BG_DISP_FRAME, SET_FG_DISP_FRAME, COPY_FRAME, MERGE_FRAME

Commented [R19]: Set links

2.79 SET_EDIT_RECT

Description: Sets the rectangle for editing (replacing colors, etc)

Code: **$5C_{HEX}$, 92_{DEC}**

Min Firmware: **2.0**

7	6	5	4	3	2	1	0		
SET_EDIT_RECT								Byte 0: Command	
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1:	**Upper-Left X Coordinate**
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2:	
y15	**y14**	**y13**	**y12**	**y11**	**y10**	**y9**	**y8**	Byte 3:	**Upper-Left Y Coordinate**
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4:	
Width MSB								Byte 5:	**Width**
Width LSB								Byte 6:	
Height MSB								Byte 7:	**Height**
Height LSB								Byte 8:	

See Also: REPLACE_COLOR

Example:
The following sequence will replace color red with green inside the rectangle size of 100x50 and positioned at (320, 240)

SET_EDIT_RECT	**5C**	HEX	
1	1	DEC	(X MSB)
64	64	DEC	(X LSB)
0	0	DEC	(Y MSB)
240	240	DEC	(Y LSB)
0	0	DEC	(Width MSB)
100	100	DEC	(Width LSB)
0	0	DEC	(Height MSB)
50	0	DEC	(Height LSB)
REPLACE_COLOR	**5D**	HEX	
RED	FF	HEX	(Old Color Red)
GREEN	00	HEX	(Old Color Green)
BLUE	00	HEX	(Old Color Blue)
RED	00	HEX	(New Color Red)
GREEN	FF	HEX	(New Color Green)
BLUE	00	HEX	(New Color Blue)

2.80 SET_LAYER_VISIBILITY

Description: Controls the visibility of a layer. Other layer(s) are unaffected

Code: **38**HEX, **56**DEC

7 6 5 4 3 2 1 0	
SET_LAYER_VISIBILITY	Byte 0: Command
layer	Byte 1: Layer Number (0-1)
visibility	Byte 2: Visibility (0-1)

Visibility: 0 = Invisible; 1=Visible

Startup Value: At boot. Layer 1 (Foreground) is visible and Layer 0 (Background) is invisible

See Also: SET_DRAW_FRAME, SET_BG_DISP_FRAME, SET_FG_DISP_FRAME, COPY_FRAME, MERGE_FRAME

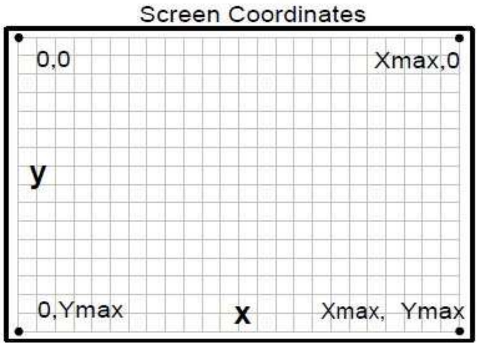
Commented [R20]: Set links

2.81 SET_XH

Description: Sets only the X-coordinate of the Current Position. Y coordinate remains unchanged.

Code: **6E**$_{\text{HEX}}$, **110**$_{\text{DEC}}$

7	6	5	4	3	2	1	0	
SET_XH								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X** LSB



See Also: SET_Y, SET_XHY

Example:
The following sequence will put a 2 blue points in the same row.

SET_COLORH	84	HEX	
BLUE_LSB	00011111	BIN	
BLUE_MSB	00000000	BIN	
SET_XH	**6E**	HEX	
0	**0**	DEC	(**X** MSB)
160	**160**	DEC	(**X** LSB)
PLOT	26	HEX	
SET_XH	**6E**	HEX	
0	**0**	DEC	(**X** MSB)
170	**170**	DEC	(**X** LSB)
PLOT	26	HEX	

2.82 SET_XHY

NOTE: Legacy deprecated command. Recommended Replacement: **SET_XHYH**

Description: Sets the Current Position.

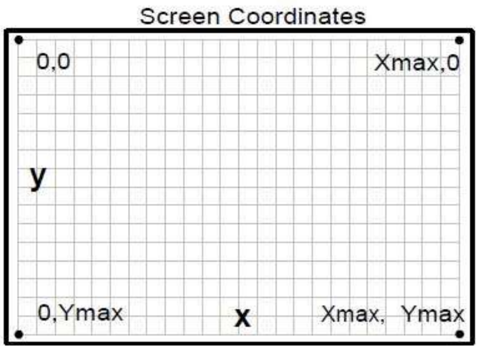
Code: **85**HEX, **133**DEC

Command Format—Standard Mode:

7	6	5	4	3	2	1	0	
SET_XHY								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X** LSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 3: **Y**

Command Format—Extended Mode:

7	6	5	4	3	2	1	0	
SET_XHY								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X** LSB
y15	**y14**	**y13**	**y12**	**y11**	**x10**	**y9**	**y8**	Byte 3: **Y** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y** LSB



See Also: PLOT, LINE_TO_XHY, CIRCLE_RH

Example:
The following sequence will put a blue point at (160, 117).

SET_COLORH	84	HEX	
BLUE_LSB	00011111	BIN	
BLUE_MSB	00000000	BIN	
SET_XHY	**85**	HEX	
0	**0**	DEC	(**X** MSB)
160	**160**	DEC	(**X** LSB)
117	**117**	DEC	(**Y**)
PLOT	26	HEX	

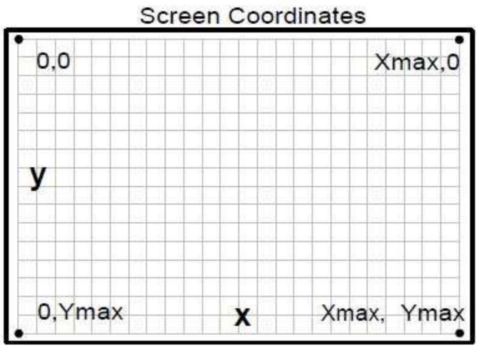
2.83 SET_XHYH

Description: Sets the Current Position.

Code: **33**$_{\text{HEX}}$, **51**$_{\text{DEC}}$

Command Format:

7	6	5	4	3	2	1	0	
SET_XHYH								Byte 0: Command
x15	**x14**	**x13**	**x12**	**x11**	**x10**	**x9**	**x8**	Byte 1: **X** MSB
x7	**x6**	**x5**	**x4**	**x3**	**x2**	**x1**	**x0**	Byte 2: **X** LSB
y15	**y14**	**y13**	**y12**	**y11**	**x10**	**y9**	**y8**	Byte 3: **Y** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y** LSB



See Also: PLOT, LINE_TO_XHY, CIRCLE_RH

Example:
The following sequence will put a blue point at (160, 117).

SET_COLOR_RGB	31	HEX	
RED	0	HEX	
GREEN	0	HEX	
BLUE	FF	HEX	
SET_XHYH	**33**	HEX	
0	**0**	DEC	(**X** MSB)
160	**160**	DEC	(**X** LSB)
0	**0**	DEC	(**Y** MSB)
117	**117**	DEC	(**Y** LSB)
PLOT	26	HEX	

2.84 SET_Y

NOTE: Legacy deprecated command. Recommended Replacement: **SET_YH**

Description: Sets only the Y-coordinate of the Current Position. X coordinate remains unchanged

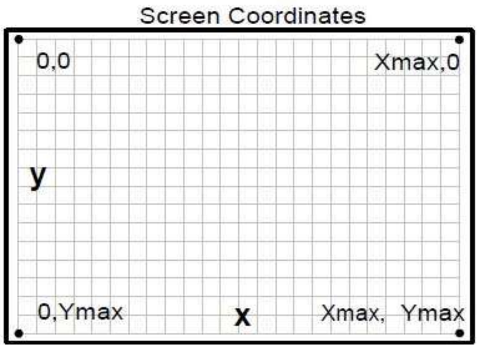
Code: **5F**$_{\text{HEX}}$, **95**$_{\text{DEC}}$

Command Format—Standard Mode:

7	6	5	4	3	2	1	0	
SET_Y								Byte 0: Command
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y**

Command Format—Extended Mode:

7	6	5	4	3	2	1	0	
SET_Y								Byte 0: Command
y15	**y14**	**y13**	**y12**	**y11**	**x10**	**y9**	**y8**	Byte 3: **Y** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y** LSB



See Also: SET_XH, SET_XHY

Example:
The following sequence will put a 2 blue points in the same column.

SET_COLORH	84	hex	
BLUE_LSB	00011111	bin	
BLUE_MSB	00000000	bin	
SET_Y	**5F**	**hex**	
70	**0**	**dec**	(y)
PLOT	26	hex	
SET_Y	**5F**	**hex**	
75	**75**	**dec**	(y)
PLOT	26	hex	

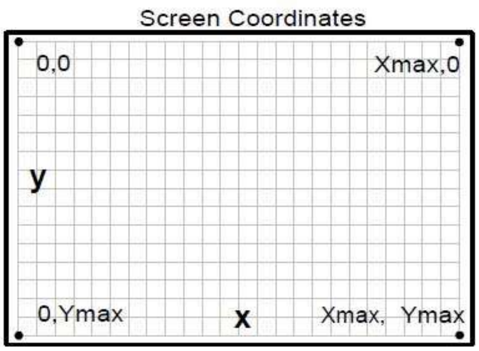
2.85 SET_YH

Description: Sets only the Y-coordinate of the Current Position. X coordinate remains unchanged

Code: **6F**HEX, **111**DEC

Command Format:

7	6	5	4	3	2	1	0	
SET_YH								Byte 0: Command
y15	**y14**	**y13**	**y12**	**y11**	**x10**	**y9**	**y8**	Byte 3: **Y** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y** LSB



See Also: SET_XH, SET_XHY

Example:

The following sequence will put a 2 blue points in the same column.

SET_COLOR_RGB	31	HEX	
RED	0	HEX	
GREEN	0	HEX	
BLUE	FF	HEX	
SET_YH	**6F**	**hex**	
0	**0**	**dec**	(y MSB)
70	**70**	**dec**	(y LSB)
PLOT	26	hex	
SET_YH	**6F**	**hex**	
0	**0**	**dec**	(y MSB)
75	**75**	**dec**	(y LSB)
PLOT	26	hex	

2.86 SHOW_SETTINGS

Description: Displays the current LCD settings on the ezLCD screen

Code: **FE**HEX, **254**DEC

7	6	5	4	3	2	1	0	
SHOW_SETTNIGS								Byte 0: **Command**

See Also: GET_SETTINGS

Example:
The following sequence will display the current ezLCD settings on the LCD

SHOW_SETTINGS FE HEX

2.87 FONT_direction

Description: Sets the orientation of the text, as shown on the picture below.

Code:	**FONT_NORTH**:	**60**HEX, **96**DEC
	FONT_EAST:	**61**HEX, **97**DEC
	FONT_SOUTH:	**62**HEX, **98**DEC
	FONT_WEST:	**2F**HEX, **99**DEC

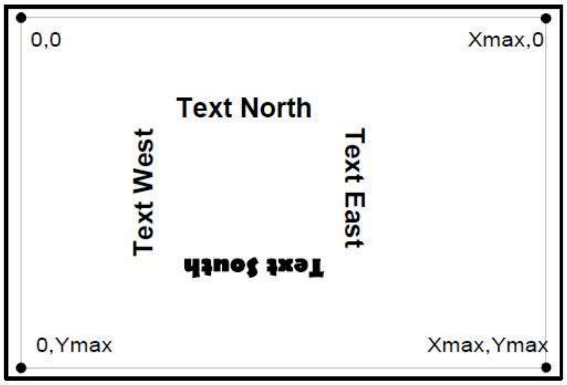
7	6	5	4	3	2	1	0	
FONT_NORTH								Byte 0: Command

7	6	5	4	3	2	1	0	
FONT_EAST								Byte 0: Command

7	6	5	4	3	2	1	0	
FONT_SOUTH								Byte 0: Command

7	6	5	4	3	2	1	0	
FONT_WEST								Byte 0: Command

Note: FONT_NORTH is the default text orientation



See Also: PRINT_CHAR, PRINT_STRING, SELECT_FONT

Example:
The following sequence will print a text pattern similar to the one pictured above.

SET_XHY		85	HEX	
0		0	DEC	(**X** MSB)
60		60	DEC	(**X** LSB)
10		10	DEC	(**Y**)
SELECT_FONT		2B	HEX	
0		0	DEC	
FONT_NORTH		**60**	HEX	
PRINT_STRING	"Text North"	2D	HEX	
NULL		0	HEX	
FONT_EAST		**61**	HEX	
PRINT_STRING	"Text East"	2D	HEX	
NULL		0	HEX	
FONT_SOUTH		**62**	HEX	
PRINT_STRING	"Text South"	2D	HEX	
NULL		0	HEX	
FONT_WEST		**63**	HEX	
PRINT_STRING	"Text West"	2D	HEX	
NULL		0	HEX	

2.88 TOUCH_PROTOCOL

Description: Changes the default behavior of the ezLCD touch control function

Code: **B2**$_{HEX}$, **178**$_{DEC}$

7 6 5 4 3 2 1 0	
TOUCH_PROTOCOL	Byte 0: Command
protocol	Byte 1: Protocol 0=None, 1=ezButton, 2=cuButton, 64=CalibratedXY

About the Touch Protocols:

Currently, the following touch protocols are implemented:

1. None
   - No touch screen events are sent regardless of what buttons are defined on the screen. **This is the power-on default unless the defaults were modified with the ezLCD 5x Utility or within *config.txt***
2. ezButton
   - Touch screen buttons can be defined BUTTON_DEF command.
   - ezLCD sends Button Down and Button Up events for the buttons defined by the BUTTON_DEF command.
   - Easy protocol. Button IDs and events are coded in 1 byte.
   - Events are sent only once per button state change.
3. cuButton
   - Similar to the ezButton, however the button states are sent continuously, 5 to 20 times per second.
4. CalibratedXY
   - ezLCD sends TOUCH_X and TOUCH_Y packets (X and Y coordinates), when the screen is pressed
   - ezLCD sends PEN_UP packets when the touch screen is not pressed.
   - Multi-byte packed oriented protocol.
   - Packets are sent continuously, 5 to 50 times per second.

See Also: BUTTON_DEF, BUTTON_STATE, BUTTONS_ALL_UP, BUTTONS_DELETE_ALL

Important: Before using this command, please read the following chapters:

- Touch Screen
- ezButton
- cuButton
- CalibratedXY

Example:
The following sequence will change the Touch Protocol to ezButton.

TOUCH_PROTOCOL	**B2**	HEX	(Command)
1	**1**	DEC	(ezButton Protocol)

2.89 V_LINE

NOTE: Legacy deprecated command. Recommended Replacement: V_LINEH

Description: Quickly draws a vertical line from Current Position, to the row specified by the parameter.

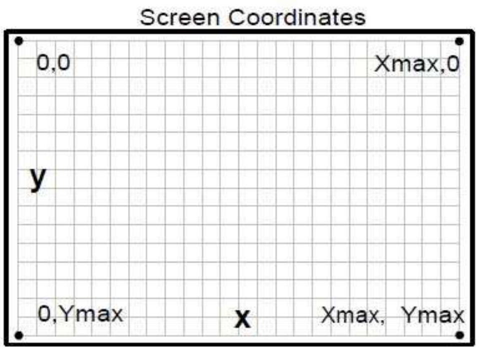
Code: **41**HEX, **65**DEC

Command Format—Standard Mode:

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

Command Format—Extended Mode:

7	6	5	4	3	2	1	0	
V_LINE								Byte 0: Command
y15	**y14**	**y13**	**y12**	**y11**	**x10**	**y9**	**y8**	Byte 3: **Y** MSB
y7	**y6**	**y5**	**y4**	**y3**	**y2**	**y1**	**y0**	Byte 4: **Y** LSB



See Also: H_LINEH, SET_XHY

Example:

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

SET_COLORH	84	HEX	
BLUE_LSB	00011111	BIN	
BLUE_MSB	00000000	BIN	
SET_XHY	85	HEX	
0	0	DEC	(**X** MSB)
95	95	DEC	(**X** LSB)
10	10	DEC	(**Y**)
V_LINE	**41**	HEX	
110	**110**	DEC	

2.90 GET_RTC

Description: Gets the current time from the battery-backed Real Time Clock

Code: **C8**$_{HEX}$, **200**$_{DEC}$

Command Format:

7	6	5	4	3	2	1	0	
GET_RTC								Byte 0: **Command**

ezLCD Response

7	6	5	4	3	2	1	0	
0	**0**	**1**	**0**	**1**	**1**	**1**	**0**	Byte 0: **2E**$_{HEX}$, **46**$_{DEC}$
Hour								Byte 1: **Hour (0-23)**
Minutes								Byte 2: **Minutes (0-59)**
Seconds								Byte 3: **Seconds (0-59)**
Month								Byte 4: **Month (1-12)**
Day								Byte 5: **Day (1-31)**
Year								Byte 6: **Year (0-99)**

See Also: **GET_RTC_EPOCH, SET_RTC, SET_RTC_EPOCH**

2.91 GET_RTC_EPOCH

Description: Gets the current time from the battery-backed Real Time Clock in UNIX epoch format

Code: **C9**$_{\text{HEX}}$, **201**$_{\text{DEC}}$

Command Format:

7	6	5	4	3	2	1	0	
GET_RTC_EPOCH								Byte 0: **Command**

ezLCD Response

7	6	5	4	3	2	1	0	
0	**0**	**1**	**0**	**1**	**1**	**1**	**0**	Byte 0: **2E**$_{\text{HEX}}$, **46**$_{\text{DEC}}$
LSB								Byte 1: **LSB**
Byte 2								Byte 2:
Byte 3								Byte 3:
MSB								Byte 4: **MSB**

See Also: GET_RTC, SET_RTC, SET_RTC_EPOCH

Reference: https://www.epochconverter.com/

2.92 SET_RTC

Description: Sets the current time from the battery-backed Real Time Clock

Code: **C6**$_{\text{HEX}}$, **198**$_{\text{DEC}}$

Command Format:

7 6 5 4 3 2 1 0	
SET_RTC_EPOCH	Byte 0: **Command**
Hour	Byte 1: **Hour (0-23)**
Minutes	Byte 2: **Minutes (0-59)**
Seconds	Byte 3: **Seconds (0-59)**
Month	Byte 4: **Month (1-12)**
Day	Byte 5: **Day (1-31)**
Year	Byte 6: **Year (0-99)**

ezLCD Response

In case of success:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**0**	**0**	**0**	**0**	Byte 0: **30**$_{\text{HEX}}$, **48**$_{\text{DEC}}$

In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**0**	**1**	**1**	**1**	**1**	Byte 0: **2F**$_{\text{HEX}}$, **47**$_{\text{DEC}}$

See Also: **GET_RTC, GET_RTC_EPOCH, SET_RTC_EPOCH**

2.93 SET_RTC_EPOCH

Description: Sets the current time from the battery-backed Real Time Clock using UNIX epoch format

Code: **C7**HEX, **199**DEC

Command Format:

7 6 5 4 3 2 1 0	
SET_RTC_EPOCH	Byte 0: **Command**
LSB	
Byte 2	
Byte 3	
MSB	

ezLCD Response

In case of success:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**1**	**0**	**0**	**0**	**0**	Byte 0: **30**HEX, **48**DEC

In case of an **error**:

7	6	5	4	3	2	1	0	
0	**0**	**1**	**0**	**1**	**1**	**1**	**1**	Byte 0: **2F**HEX, **47**DEC

See Also: **GET_RTC, GET_RTC_EPOCH, SET_RTC**

Reference: **https://www.epochconverter.com/**