e-Gizmo

EZ HMI Display Terminal

Technical Manual and Communication Functions Reference Manual
Rev. 1r0

The EZ HMI (Human-Machine Interface) is a general purpose display and input terminal designed with microcontroller applications in mind. Users can display messages and data through its LCD display, and get alphanumeric input from users with the use of just a few simple commands. It even features a self contained menu driven function that allows users to set, edit, and save to its non volatile memory data- herein referred to as Parameters, for later retrieval and use by the user application programs.

A user interface function that carries these features usually requires lengthy codes (it does). Now that these functions are already built-in features of the EZ HMI, there goes one detail you no longer need to worry about- you can put all your programming effort on the application project.

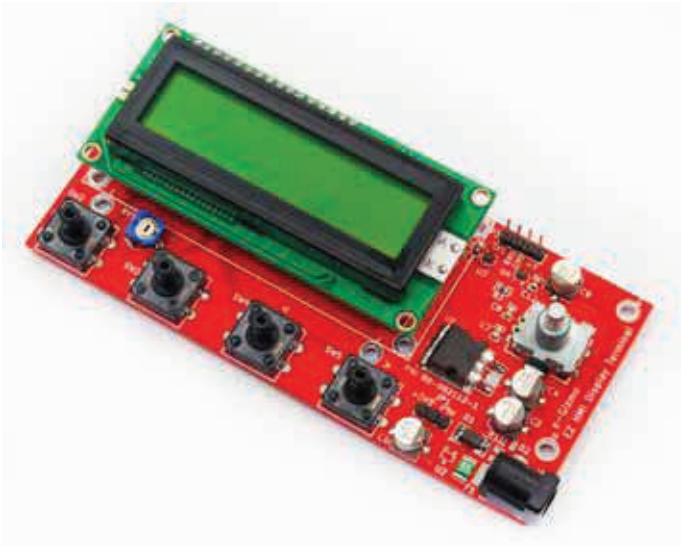


Figure 1. EZ HMI Display Terminal Module.

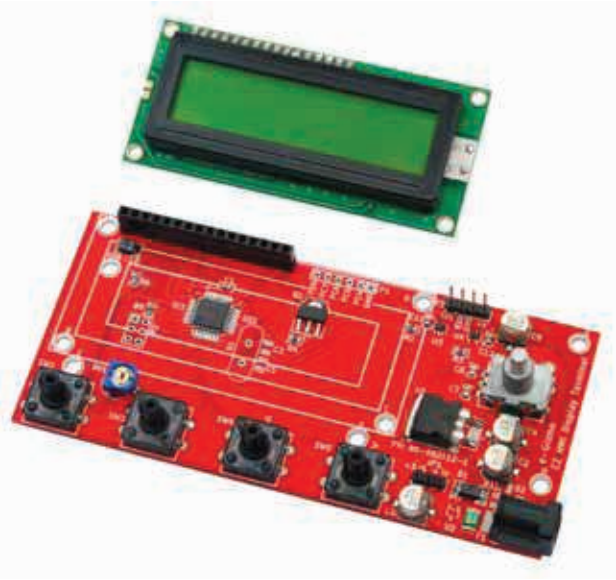


Figure 2. EZ HMI Display Terminal with LCD display module removed. You can use a VFD display module in place of the LCD if you want a high brightness- high contrast display.

- UART I/F, Requires only 2 I/O pins from host controller.
- UART port compatible to 3V-5V MCU Host, including Arduino & gizDuino
- Five user DIO (2 inputs + 3 outputs) provides additional user I/O
- Display user generated formatted message with relative ease.
- User Dialog function makes acquisition of user inputted data simple.
- Menu driven Parameter Entry function allows the user to enter non-volatile data for later retrieval and use by the host MCU using just a simple set of functions.
- Encoder switch allows entry of numeric and alphanumeric data, including selected symbols.
- 16 characters by 2 lines LCD or VFD display.
- Four push button keys used in parameter entry and dialog mode are free to use for other purposes in your program.

CONTROLS and TERMINALS

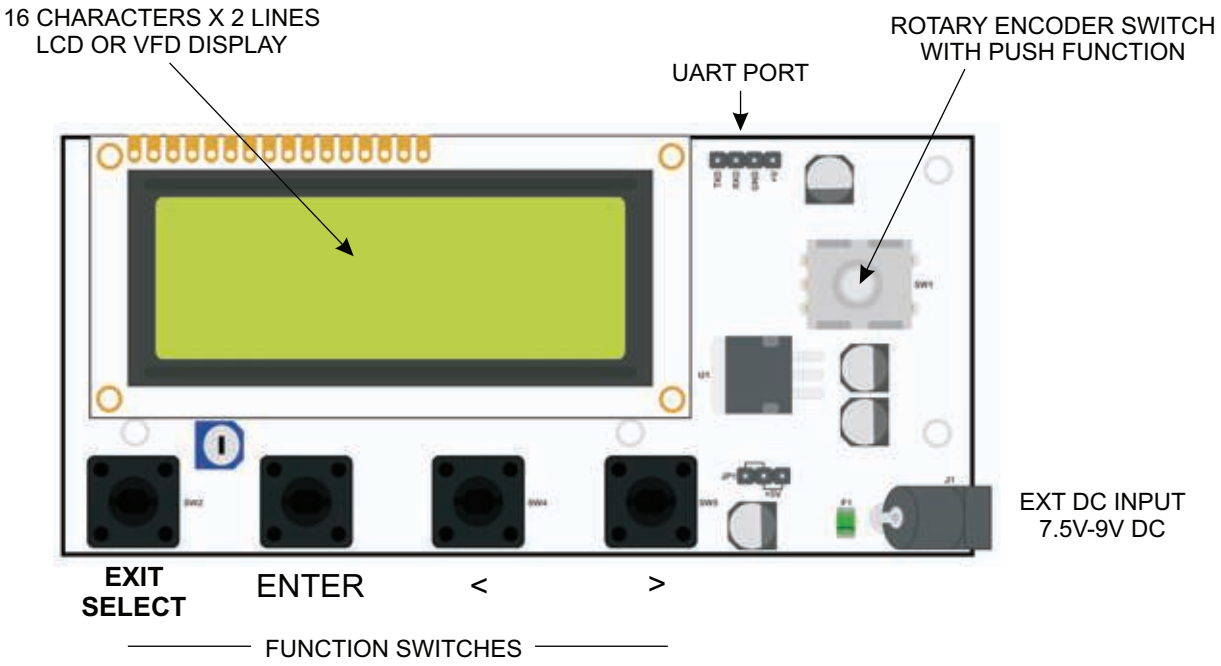


Figure 3. EZ HMI Controls and Port layout. The four push button controls and encoder switch data are shared between EZ HMI built-in functions and user defined functions.

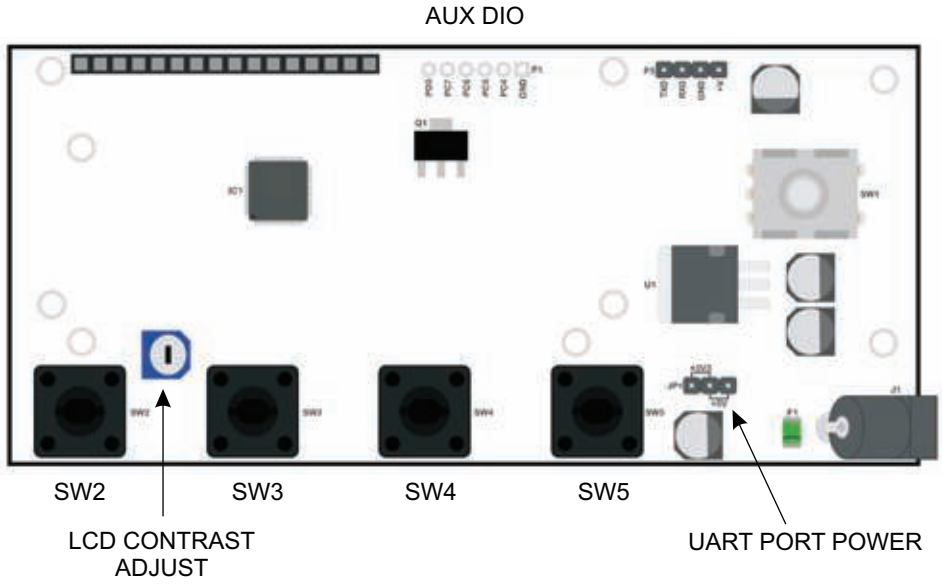


Figure 4. EZ HMI showing AUX DIO port location and LCD contrast adjustment.

PIN DESCRIPTION

Table 1. P1 - AUX DIO

The AUXiliary Digital Input Output DIO can be freely used for user applications. These can be controlled and monitored using the communication functions as described in section 4.12. The AUX DIO works on 5V digital logic levels.

Pin No:	MARKING	Description
1	GND	Circuit Ground
2	PC4	Digital Output 0
3	PC5	Digital Output 1
4	PC6	Digital Output 2
5	PC7	Digital Input 0
6	PD0	Digital Input 1

Table 2. P3 - UART PORT

The UART PORT TXD and RXD lines are provided with a level translator that allows it to be connect directly to any host controller working with logic level in 3.0 to 5.0V range. See Fig.6 to Fig. 9 for more details.

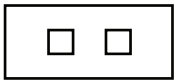
Pin No:	MARKING	Description
1	TXD	UART TX DATA OUTPUT
2	RXD	UART RX DATA INPUT
3	GND	Ground
4	+V	UART PORT Power**

**JP1 - UART POWER SOURCE

With JP1 installed, the +V UART pin act as +5V power input or output and is used as follows:

no J1 DC input:

- EZ HMI can be powered by supplying +5V to the +V to GND input of the UART PORT.
- Power applied through J1
- +V acts as a +5V power source that you can use to power other devices. Current drawn through +V pin must not exceed 250mA.



No jumper on JP1

Without JP1, +V pin is used for level UART I/O level translation. It must be supplied with a voltage input equal to the Vcc supply of the external host controller. This will ensure logic level compatibility between the host and EZ HMI.

ENCODER SWITCH

The encoder switch is a special rotary switch that provides for a simple but quick way of entering user inputs. Data is generally selected by rotating the shaft. Pushing the shaft will launch a predetermined set of actions, which may vary according to the functions using it.

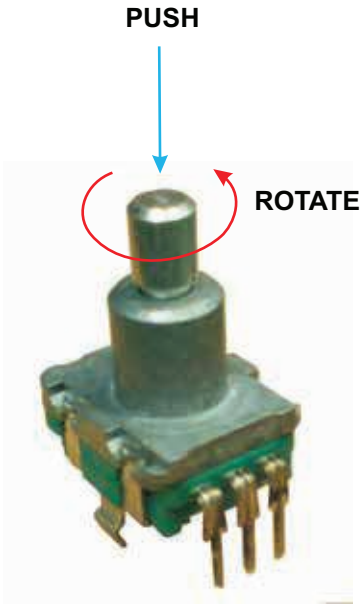


Figure 5. The encoder switch provides a user intuitive way of inputting data. Numeric and alphanumeric inputs is possible, replacing multikey ASCII keyboard.

WIRING CONFIGURATIONS

Important: Wire lenghts must not exceed 1 Meter**.**

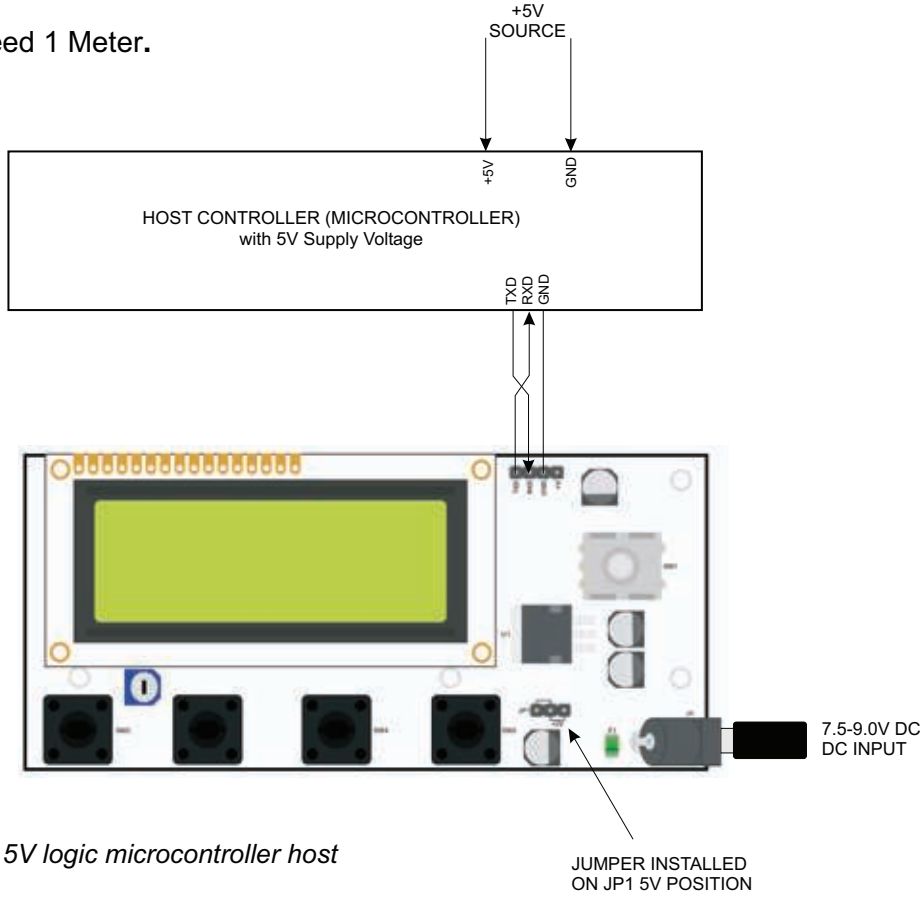


Figure 6. UART wiring interface with a 5V logic microcontroller host with separate power source.

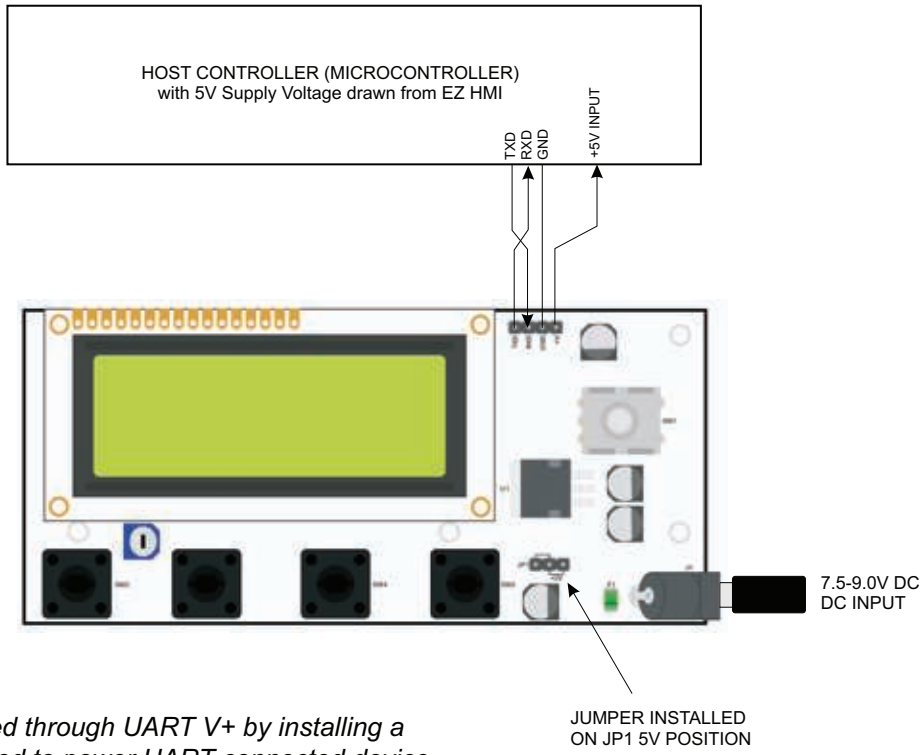


Figure 7. The EZ HMI +5V, when routed through UART V+ by installing a jumper at JP1 +5V position, can be used to power UART connected device, such as the host MCU, or even wireless UART devices.

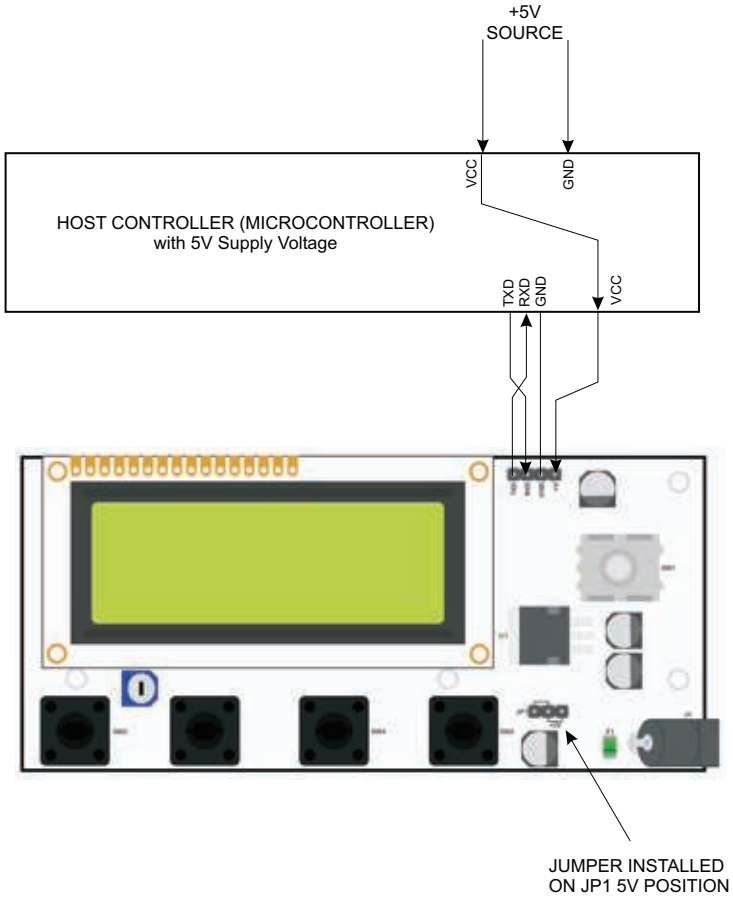


Figure 8. The EZ HMI, conversely, can be powered from a +5V sourced by the host controller circuit.

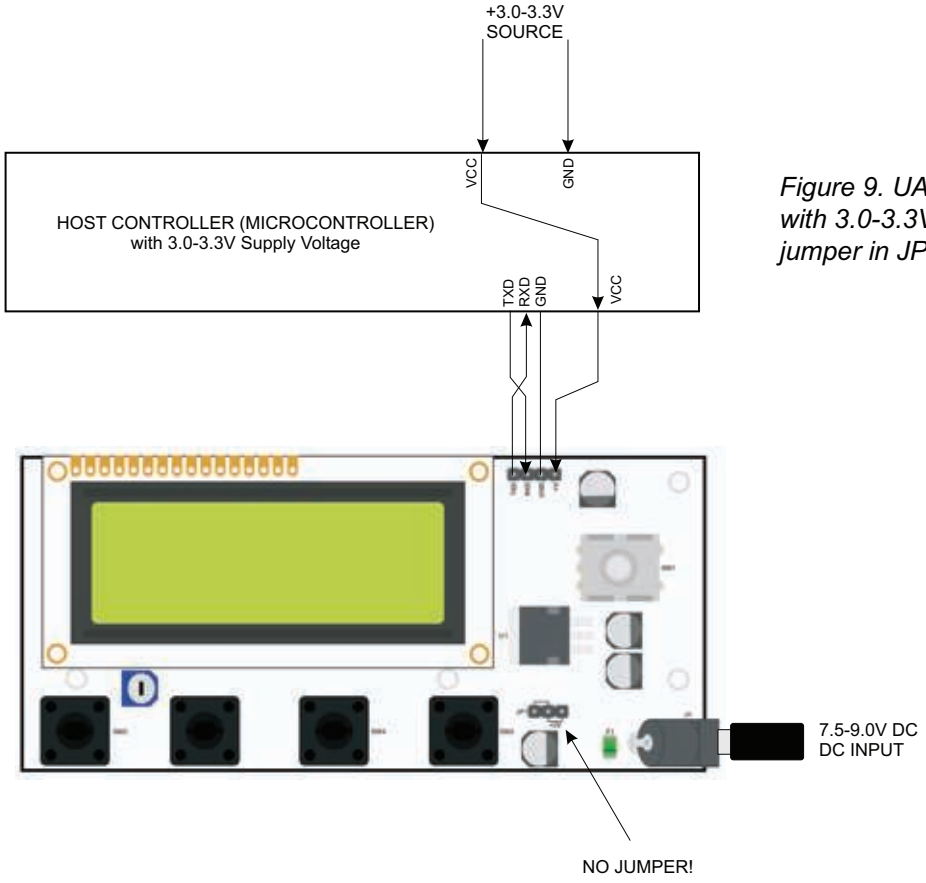


Figure 9. UART wiring interface to host controllers with 3.0-3.3V working logic levels. Make sure the jumper in JP1 is removed.

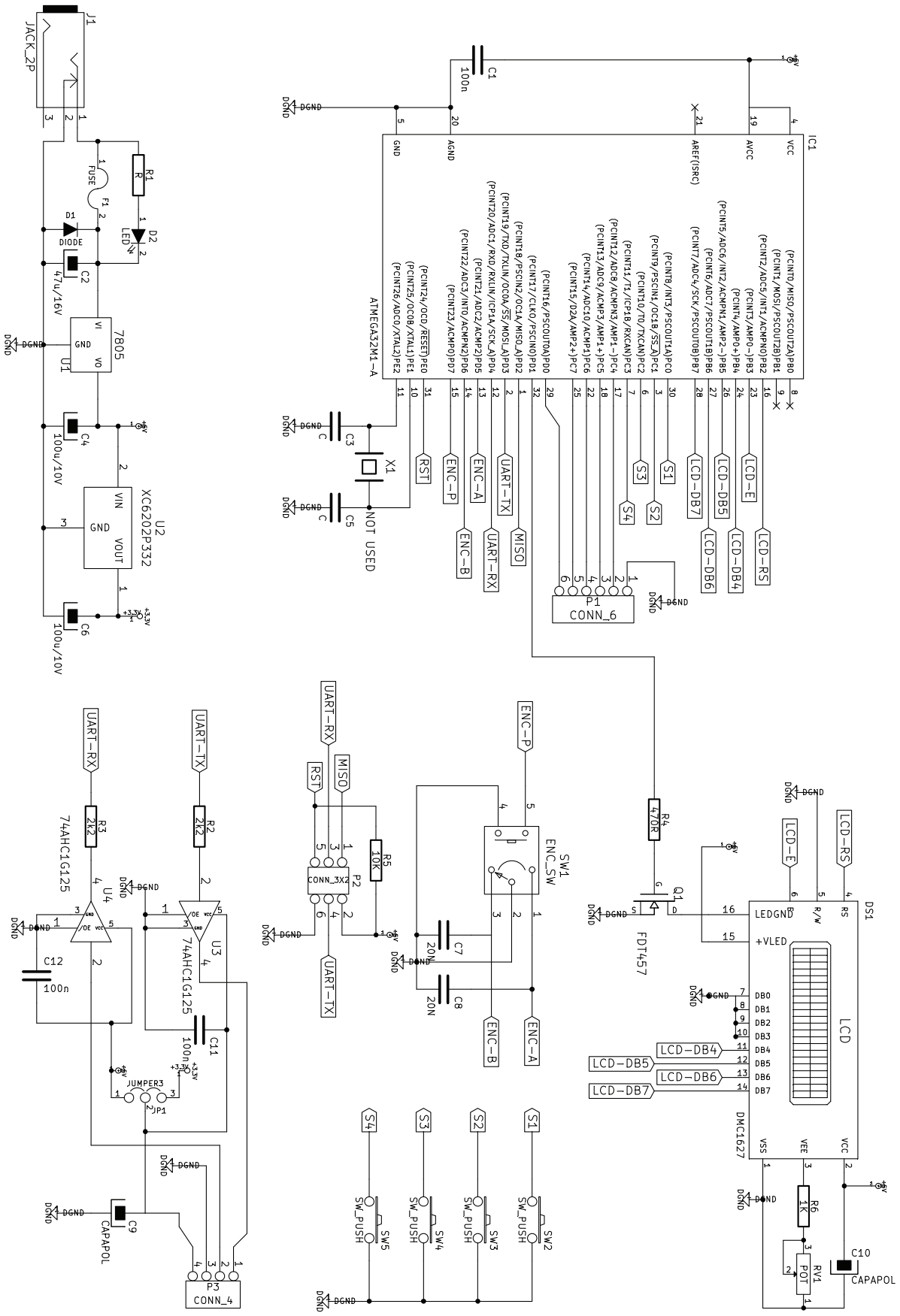


Figure 10. EZ HMI Schematic Diagram

1. DISPLAYING MESSAGE

The LCD used in the EZ HMI has a display area of 16 characters x 2 lines. With the use of just half dozen or so functions, users can effortlessly display messages and other info.

• Function "**c**" erases the whole content of LCD display.

• Function "**>**" is use to set the cursor to any location within the displayable area. This allows you to start printing anywhere on the LCD display screen.

• Function "**1**" and "**2**" allows you to display message on the top line and bottom line respectively. Portions of LCD display not overwritten with the new message will remain unaffected.

• Function "**3**" and "**4**", allows you to display message as in "**1**" and "**2**", except the line is cleared first of its old display before putting in the new message.

Scrolling Message

If your message is too long to fit inside the 16 characters display area, you can use the scrolling message functions to display your long messages. Scrolling from right to left, function "**m**" displays your scrolling message at the top line, while function "**M**" use the bottom line to display. You can display messages up to 79 characters long per line using these functions.

Tip: The LCD display does not respond well with fast changing display. Scrolling message may result in display that appears to overlap on top of each other, hence, may be hard to read. To minimize this, adjust the LCD contrast until an acceptable scrolling display is obtained. This may result in lower overall contrast. This limitation does not apply to VFD displays.

See the Section 4.1 to 4.4 for more details on the usage of the display functions

2. PARAMETERS AND PARAMETER ENTRY MODE

The EZ HMI Parameter Entry Mode is a built-in function that allows you to set, modify, and save to EZ HMI non-volatile memory up to 48 data sets; we will refer to these data sets as parameters here on. This function will be useful for applications where parameters need to be adjusted by the user from time to time. Example applications includes (and not limited to) temperature control, process control, timer, instrumentation - to name just a few.

Using the same function, each parameter can be named by the user for easy identification. Parameters are grouped according to the type of data it can hold. These are listed in the following table 3.

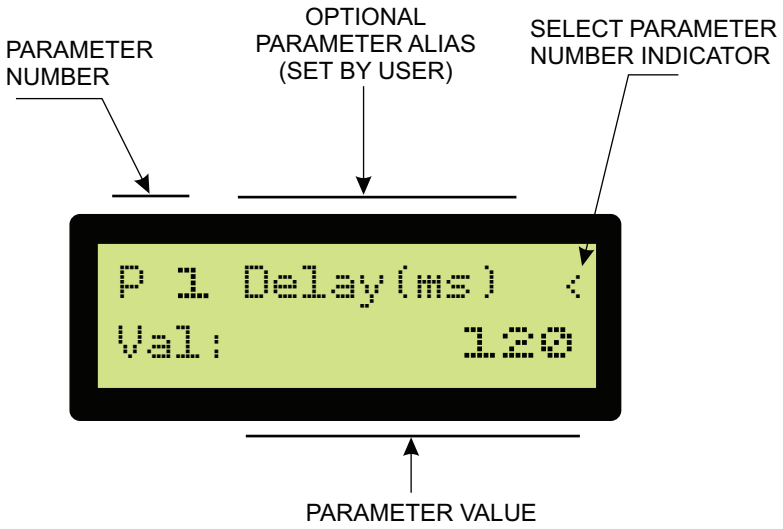
2.1 Entering Parameter Entry Mode

Parameter entry mode can be initiated by program control (see section 4.7) or by pressing the encoder switch push switch as follows:

2.1.1 Push the encoder switch knob.
2.1.2 Hold it in push position until "Adjust Parameter Mode" is displayed on LCD screen. This should take place within 3 seconds.
2.1.3 Release the encoder push switch.

Table 3. EZ HMI Non Volatile Data Storage

Parameter No	Capacity	Data Type	Value Range
0-15	16	16 bit Signed Integer	-32768 to +32767
16-31	16	16 bit Unsigned Integer	0-65535
32-47	16	32 bit Signed Integer	- 2,147,483,648 to + 2,147,483,647



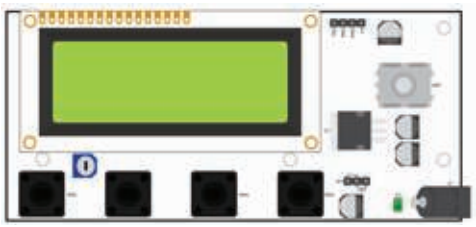
2.2 Parameter Entry Mode display Screen

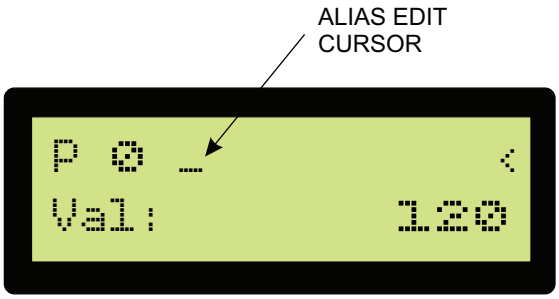
To select and display parameter value:

2.2.1 Push the encoder switch as may be necessary until the “<” sign appears on the rightmost top line position,
2.2.2 Rotate the encoder switch until the desired parameter number is displayed.
2.2.3 The parameter alias and its current value are displayed.

To change parameter alias:

2,2,4 Select the parameter number as in 2.2.1-2.2.3.
2.2.5 Momentarily push SW5 “>” switch. A cursor should appear at the first character position of the alias.
2.2.6 Rotate the encoder switch until the desired character is displayed.
2.2.7 Momentarily push SW5 “>” switch. The cursor should jump to the next character.
2.2.8 Repeat 3 and 4 until the desired alias is complete.


SW5



§ Tips:
• The cursor switch SW4 “<” and SW5 “>” can be pressed freely to locate the cursor back and forth to the desired character position. Change character as desired by rotating the encoder switch.
• Alias names can take up to 10 characters.

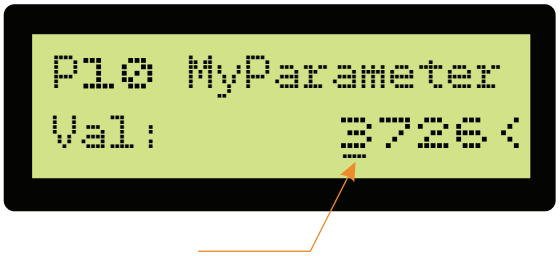
To change/enter parameter value

2.2.9 Select the parameter number as described in 2.2.1-2.2.3.

2.2.10 Momentarily push the encoder switch until the “<” marker appears on the bottom line rightmost position.

2.2.11 Press the cursor position switch SW4 “<” or SW5 “>” to position the cursor to the desired digit position.

2.2.12 Rotate the encoder until the desired value is displayed.



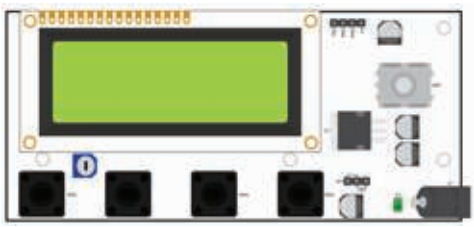
SW4 “<” OR SW5 “>”
TO MOVE CURSOR POSITION

To Exit Parameter Entry Mode

2.2.13 Momentarily press SW2 until “Save Changes?” message appears. The Yes/No/Cancel selection should also appear on the bottom line.

2.2.14 Press cursor switch SW4 “<” or SW5 “>” until the cursor coincides with the desired response.

2.2.15 Press SW3 to enter the choice and exit.



SW2 SW3 < SW4 SW5 >

Exit Selection:

Yes – To save all changes to non-volatile memory

No – To exit without saving to non-volatile memory. The edited data will remain in effect but will be reset to the last saved values once the power is cycled OFF to ON.

Cancel – Re-enter parameter edit mode.

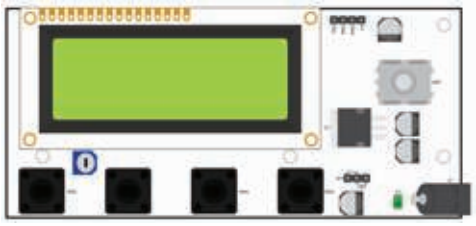
Important: The edit parameter mode exits with a cleared LCD display. User application program must reload the display to its previous condition upon exiting this function. User application program can detect this condition by using the **T** test function as described in the communications section.

§ Tips:
• The non-volatile memory can be rewritten up to 100,000 times.
• Your host controller can read the parameter even while it is being adjusted. See section 4.8 for more details.

3. DIALOG/USER INPUT MODE

The Dialog function provides a quick way for your applications programs to collect user input data. You can launch it with any prompt message you like, limited only by the 16 characters the LCD can display. It can accept numeric and alphanumeric user input up to 14 characters in length.

The Dialog function can only be launched under a user application program control. It has two operating modes, mode 0 is the simpler of the two, limiting the user to a Yes/No/Cancel response. Mode 1 allows the user to enter numeric or alphanumeric inputs. See section 4.9 for more details. This section will discuss the function at the user interaction level only.

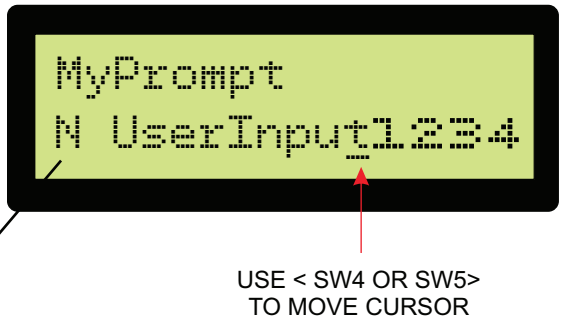


3.1 Mode 0: Yes/No/Cancel response

3.1.1 Press cursor switch SW4 “<” or SW5 “>” until the cursor coincides with the desired response.
3.1.2 Press SW3 to send the selected response and exit Dialog mode.

3.2 Mode 1: Numeric/Alphanumeric

In this mode, the input character set is divided into three groups to make message composition task easier. The active input character set selection is indicated by a single character on the leftmost position of the bottom line:



N - Number and Symbols
A - Capital letters and “^’[]”
a - lowercase letters and “{}|”

3.2.1. Press SW2 repeatedly as may be necessary until the desired character input set is selected.
3.2.2. Rotate the encoder switch until the desired character is displayed.
3.2.3 Press cursor switch SW5 “>” to point the cursor to the next character position. You can use the cursor switch SW4 “<” or SW5 “>” to skip or go back to any position within the input display area.
3.2.4 Repeat steps 1-3 until the desired input is completed.
3.2.5 Momentarily press SW3 to send input message and exit Dialog mode

Important: The Dialog Function exits with a cleared LCD display. User application program must reload the display to its previous condition upon exiting this function. User application program can detect this condition by using the **T** test function as described in section 4.5.

4. COMMUNICATIONS FUNCTIONS

Communications settings:

Baud Rate: 9600
Data: 8 Bit
Parity: none
Handshake: none

Summary of Functions

c - Clear Display
1 - Display to line 0 (Top Row)
2 - Display to line 1 (Bottom Row)
3 - Clear & Display to line 0
4 - Clear & Display to line 1
> - Set Print Start position
m - Scroll Display to line 0
M - Scroll Display to line 1
p - Goto Edit Parameter Mode
r - Read Parameters
k - Read a Key (one shot mode)
K - Read a key (level mode)
e - Read user encoder
E - Write user encoder value
B - Set Backlight Brightness 0-19
T - Test
D - Dialog Box/User Input
S - Set Aux Output
R - Reset Aux Output
I - Read Aux Input

Important:

Communications Format

Every packet of data transmission (and response) are wrapped inside an [STX] and [ETX] marker.

[STX] – Start of transmission marker, ASCII value = 0x02
[ETX] – End of transmission marker, ASCII value = 0x03

The first character after the [STX] marker is a single character function specifier. Each transmission may contain just a function specifier only, or may contain a series of data in addition to the function specifier. End of transmission is signaled by the [ETX]marker.

[STX] and [ETX] are data packet markers and should not be transmitted as literal string. They should be send in their ASCII representation. The correct way of transmitting the [STX] and [ETX] markers are shown in the following example:

Example 1: Clear LCD Display

Transmission Format: [STX]**c**[STX]

This should be transmitted in their ASCII code representation as shown in the following table:

Symbol	STX	c	ETX
Hex	0x02	0x63	0x03

Visual Basic:

Correct:

```
Serial1.print(chr(2)+"c"+chr(3))
```

Wrong:

```
Serial1.print("[STX]c[ETX]")
```

Arduino:

Correct:

```
Serial.write(0x02);
Serial.print("c");
Serial.write(0x03);
```

Wrong:

```
Serial.print("[STX]c[ETX]");
```

Example 2: Display “Hello World!” on top line

Transmission Format: [STX]**1**Hello World![ETX]
Visual Basic:

ASCII	STX	1	H	e	l	l	o		W	o	r	l	d	!	ETX
Hex	0x02	0x31	0x48	0x65	0x6c	0c6c	0x6f	0x20	0x57	0x6f	0x72	0x6c	0x64	0x21	0x03

Serial1.print(chr(2)+”**1**Hello World!”+chr(3))

Arduino:

```
Serial.write(0x02);              //[STX]
Serial.print(“1Hello World!”);   //1Hello World!
Serial.write(0x03);              //[ETX]
```

Alternately, you can use the C/Arduino “\” operator to send the ASCII code of [STX] and [ETX], together with the function and data:

```
Serial.print(“\0021Hello World!\003”);
```

where: “\002” = [STX], “\003” = [ETX]

Notice that in both example, only the [STX] and [ETX] marker need to be converted manually to their ASCII code, for the simple reason that they have no equivalent printable characters. The three line implementation (long hand format) may make your program longer, but is more human readable. Hence, for clarity, all example codes given are shown in the long format. We leave it up to you if you want to code it in short format.

Function Description

4.1. **c** - Clear LCD

Clear LCD display

Note: This function is not available while the EZ HMI is in Parameter Entry / Edit Mode or while in Dialog mode.

Format: [STX]**c**[ETX]

Response: (Note: Wrapped with [STX] & [ETX])

OK – Ready

Example (Arduino):

```
Serial.write(0x02);      //STX code
Serial.print(“c”);       //c – Clear LCD function
Serial.write(0x03);      //ETX code
```

4.2. **>** - Set Printing Start Position

Note: This function is not available while the EZ HMI is in Parameter Entry / Edit Mode or while in Dialog mode.

Format: [STX]**>**nn[ETX]

Where:
nn = start print position, 0-15
0 - left most position
15 – rightmost position
Default value=0

Response: (Note: Wrapped with [STX] & [ETX])
OK – Ready
ERR - Print Start position invalid

Example:
Set print start position to 10th character in a line

(Arduino):

```
Serial.print(0x02);      //[STX]
Serial.print(“>”);
Serial.print(“10”);
Serial.print(0x03);      //[ETX]
```

4.3. Printing/Display Functions

1 – Print to Line 0 (Top Row)
2 – Print to Line 1 (Bottom Row)
3 – Clear and then Print message to Line 0
4 - Clear and then Print message to Line 1

Note: These functions are not available while the EZ HMI is in Parameter Entry / Edit Mode or while in Dialog mode.

Message can contain any printable characters up to 16 characters long. Long messages will be automatically truncated to 16 characters.

Functions **1** and **2** will simply write over any pre-existing message on a line, taking the space of only those needed by the latest message. For example, if your program displayed “String Theory” on line 1 beforehand, and you print another message as “123”, the result will be “123ing Theory”. This feature is useful for programs that need to refresh just a small portion of the displayed message from time to time. It helps reduce screen refresh flicker.

Use functions **3** and **4** if you want to erase all pre-existing message on a line before printing the new message.

Use **>** Print Start Position function before calling any one of the printing functions if you want to start printing on any arbitrary position along the line.

Format: [STX]**1**w[ETX]

Where:
w= user message, up to 16 characters long

Example 1 (Arduino):

Display “J3J3mon” to line 0

```
Serial.write(0x02);      //STX code
Serial.print("1");       // 1 – print to line 0 function
Serial.print("J3J3mon"); // w
Serial.write(0x03);      //ETX

Alternatively:

Serial.print("\0021J3J3mon\003");
```

Example 2 (Arduino):

Display “e-Gizmo” to line 1 near center position

```
// Set start position
Serial.write(0x02);      //STX code
// Start printing on 4th character position
Serial.print(">");
Serial.print("4");
Serial.write(0x03);      //ETX

// clear line & print message

Serial.write(0x02);              //STX code
// 4 – clear print to line 1 function
Serial.print("4");
Serial.print("e-Gizmo");         // m=e-Gizmo
Serial.write(0x03);              //ETX
```

4.4 Scrolling Display Functions
m – display a scrolling message to line 0
M – display a scrolling message to line 1

Note: This function is not available while the EZ HMI is in Parameter Entry / Edit Mode or while in Dialog mode.

Long messages can be displayed on a line using the Scrolling Display Function. Messages up to 79 characters long can be displayed on a line.

Tips:
• Scrolling message is more readable if preceded by 15 blank spaces.
• Lower LCD contrast if the characters appear overlapping as they are scrolled.

Format: [STX]**m**w[ETX]
[STX]**M**w[ETX]

Where:
w= user message, up to 79 characters long

Response: (Note: Wrapped with [STX] & [ETX])
OK - Ready

Example 1 (Arduino):

Scrolling Display “Programmers have more fun!” to line 1

```
Serial.write(0x02);      //STX code
// M – scrolling display to line 1 function
Serial.print(“M”);
Serial.print(“Programmers have more fun!”);
Serial.write(0x03);      //ETX
```

4.5. **T** – Test Communication Link

Use to test the communications link and determine device status.

Format: [STX]**T**[ETX]

Response: (Note: Wrapped with [STX] & [ETX])
EDIT = currently in edit mode
DIA = currently in dialog mode
REF = display refresh request (see explanation below)
OK = Ready

Tip: The Parameter and Dialog function exits with a blank LCD screen. All data displayed prior to the function request are erased. The REF response reminds the user of this condition, and user code must reload the LCD with the desired display. Any display function request will clear the REF condition.

4.6. **B**- Backlight Brigthness

The LCD backlight brightness can be adjusted in 18 steps, with 1 setting it to the dimmest and 19 (and 0) to the brightest setting.

Format: [STX]**B**nn[ETX]

Where:
nn – Brightness level, 0-18

Response: (Note: Wrapped with [STX] & [ETX])
ERR = invalid level
OK = Ready

Example (Arduino):

Set LCD backlight brightness to 5

```
Serial.write(0x02);      //STX code
Serial.print(“B”);       // LCD Backlight function
Serial.print(“5”);       // nn brightness level
Serial.write(0x03);      //ETX
```

4.7. **p** - Edit Parameter Mode

Note: This function is not available while the EZ HMI is in Dialog mode.

EZ HMI has a built-in function that allows you to enter and save on its non-volatile memory up to 48 system parameters (data). See section 2 “PARAMETERS and PARAMETER ENTRY MODE” for usage and details.

Tips:
• You can read any parameter while it is being modified in the Edit Parameter Mode.
• Parameters values are exclusive for user specified applications. No stored parameter value has any influence on the operation of EZ HMI.

Format: [STX]**p[**ETX]

Response: (Note: Wrapped with [STX] & [ETX])
EDIT - Currently in Edit Parameter Mode

4.8. **r** – Read a parameter

Note: This function is not available while the EZ HMI is in Dialog mode.

Read a parameter data from EZ-HMI.

Format: [STX]**r**nn[ETX]

Where:
nn – Parameter index, 0-47.

Response: (Note: Wrapped with [STX] & [ETX])
parameter value

4.9. **D** – Open a Dialog (User Input)

Note: This function is not available while the EZ HMI is in Parameter Entry / Edit Mode.

The Open a Dialog Function provides an easy way for a user to interact with a host controller. Using this function, the user can easily display a prompt message, and then get the user respond, which may be a simple Yes/No/Cancel response, or a Numeric or Alphanumeric user input. See section 3 “DIALOG/ USER INPUT” for usage and details.

Format: [STX]**D**nw[ETX]

Where:
- n – Dialog type:
  - 0 – Yes/No/Cancel,
  - 1 – Numeric/Alphanumeric (up to 14 characters)
- w- User Prompt (16 character maximum)

Response: (Note: Wrapped with [STX] & [ETX])
- User entered numeric/alphanumeric data
- DIA - Entered Dialog mode

Example (Arduino):

Create a Dialog that returns a Y/N/C user response

```
Serial.write(0x02);      //STX code
// Dialog with Yes/No/Cancel selection
Serial.print(“D”);
Serial.print(“0”);
Serial.print(“Test Input:”);  // “Test Input:” prompt
Serial.write(0x03);      //ETX
```

4.10. **k, K** - Read Key

k – Read function key state (one shot mode)
K – Read function key state (level mode)

Note: This function is not available while the EZ HMI is in Parameter Entry / Edit Mode or while in Dialog mode.

Read the state of any one of the four functions keys. The key state can be read in two ways- one shot mode **k** and level mode **K**.

In one shot mode, a pressed key will be flagged as “1” only if it has been detected as not pressed during a previous Read Key request. In other words, whenever a key is pressed, it will return a “1” on the first Read Key k request, subsequent Read Key **k** request will return a “0” even if the addressed key remained pressed. User must release the key and Read Key k must see the key in released state before it can report it to be in pressed “1” state again. This feature is useful in preventing the host controller from unintentionally repeating the execution of some task in response to a key closure.

Use the level mode K if you want to read the current state of the addressed key. That is, as long as the addressed key is pressed, Read Key **K** will report it in “1” state at every request. As always, if the key is not pressed, it will be reported in “0” state.

Format: [STX]**k**n[ETX]
[STX]**K**n[ETX]

Where:
- n – Key Number
  - 0 – SW2
  - 1 – SW3
  - 2 –SW4 “<”
  - 3 – SW5 “>”

Response: (Note: Wrapped with [STX] & [ETX])
- 1 – if n key is pressed
- 0 – **K** if n key is not pressed
- 0 - **k** if n key is already detected as pressed and user keep pressing it.
- ERR - invalid key number

11. **e**, **E** - Read from/Write to Encoder

Note: This function is not available while the EZ HMI is in Parameter Entry / Edit Mode or while in Dialog mode.

A user register keeping a copy of encoder value can be read and loaded with a value by the user using this function. The encoder value is a 16-bit signed integer register which can assume any value with +32767/-32768 range. Value rolls to opposite sign each time the encoder is adjusted past it minimum and maximum value.

Format: [STX]**e**[ETX]
[STX]**E**[ETX]

Response: (Note: Wrapped with [STX] & [ETX])
- **e** - Signed encoder value
- **E** - OK

4.12. **S**,**R**,**I** EZ HMI Auxiliary I/O functions

S – Set an EZ HMI output (Logic 1)
R – Reset an EZ HMI output (Logic 0)
I – Read an EZ HMI input

The EZ HMI has an auxiliary digital I/O port (connector P1)- consisting of 3 output ports and 2 input ports, that you can control and use in your target applications.

Format: [STX]**S**n[ETX]
[STX]**R**n[ETX]
[STX]**I**n[ETX]

Where:
n – I/O number
Output Port
0 – PC4
1 –PC5
2– PC6
Input Port
(With weak pull up, normally "1")
0 – PC7
1 – PD0

Response: (Note: Wrapped with [STX] & [ETX])
S,**R** function
OK - Ready
ERR -Invalid I/O number
I function
0 - input reads logic 0
1 - input reads logic "1"
ERR - Invalid I/O number

Example (Arduino):

Set PC5 to logic "1"

```
Serial.write(0x02);     //STX code
Serial.print("S");       // Set an output
Serial.print("1");       // output 1 = PC5
Serial.write(0x03);     //ETX
```

Example (Arduino): Read state of PC7

```
Serial.write(0x02);     //STX code
Serial.print("I");       // Read an INPUT
Serial.print("0");       // input 0 = PC7
Serial.write(0x03);     //ETX
```