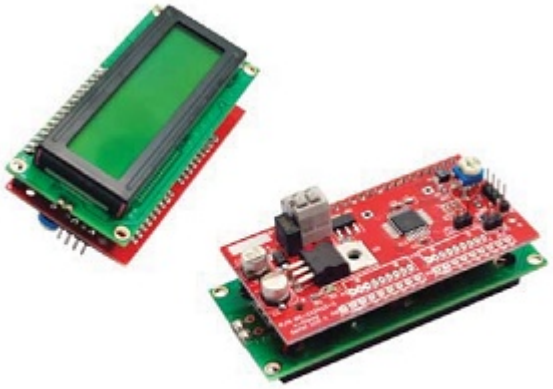
Serial LCD II
with Keypad Function

Technical Manual Rev 1r0

The Serial LCD II with Keypad Function is a general purpose display and inputs. It is designed interface to Serial LCD II rev2 with built-in microcontroller applications. A friendly-user display messages and data through its compact LCD (4x20) character display and get alphanumeric input by pressing the keypads.



Serial LCD II with 2x16 characters LCD display module.

Serial LCD II with 4x20 characters LCD display module.

FEATURES:

- Interface with gizDuino MCU boards.
- LCD 2 X16 or LCD 4 X 20 Display
- with 4x4 Matrix Keypad Module
- User adjustable contrast control.
- User adjustable (through user program code) backlight brightness.
- Fuse protected power input.
- On-board regulator allows the kit to work with 7.5V up to 10VDC power source.

Serial LCD II with 4x20 characters compact LCD display module. Available as an option is a acrylic mounting bracket.

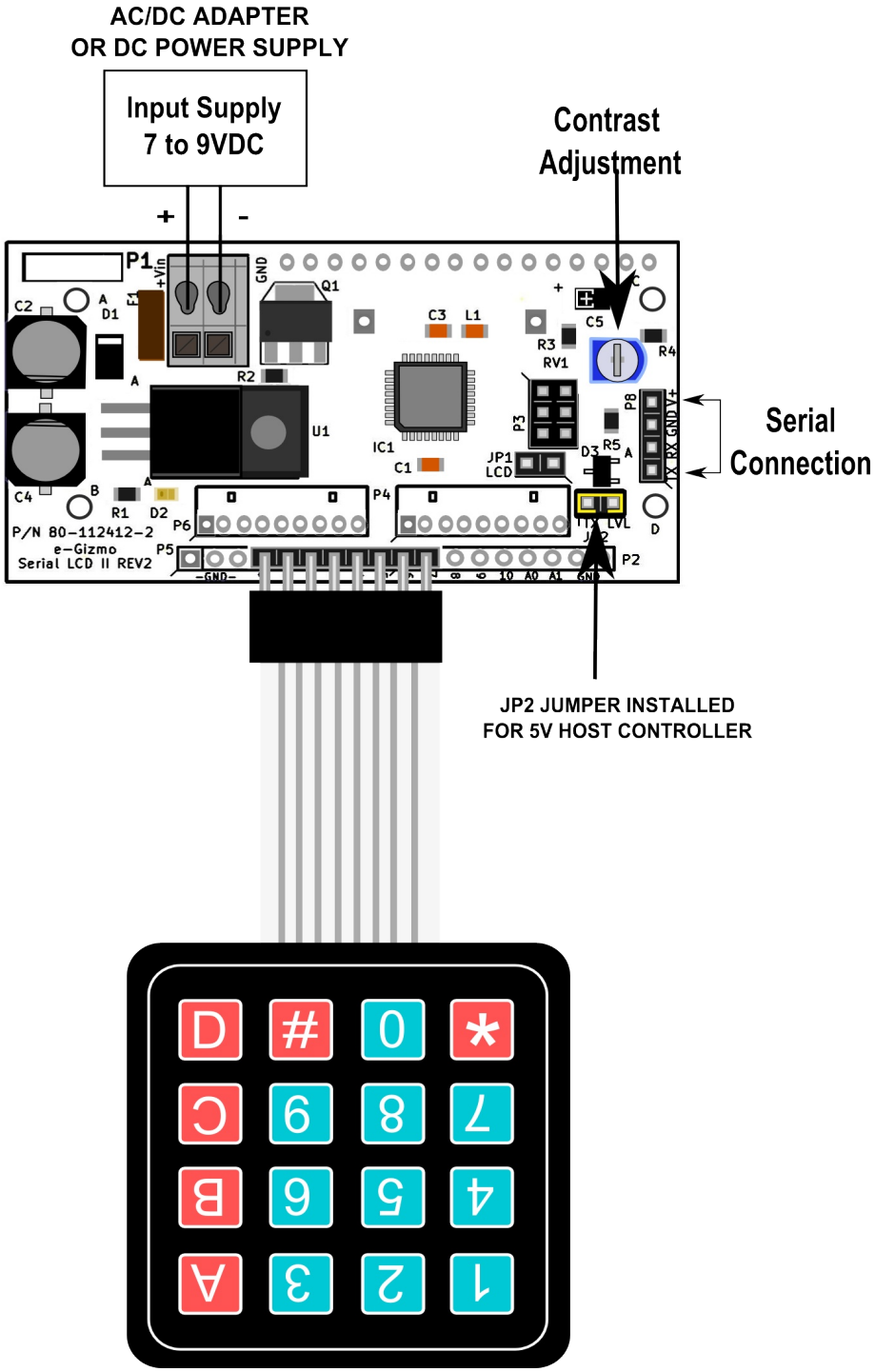


Figure 1. Major parts of Serial LCD II with Keypad 4x4 Module Function.

Important: The fuse will blow if the power supply is connected the wrong way. Replace the fuse to continue using the Serial LCD II kit.

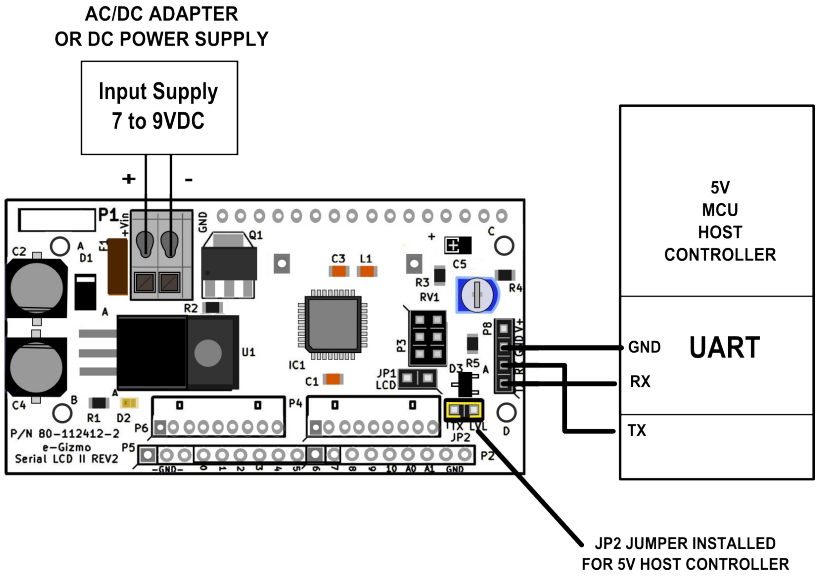


Figure 2. The Serial LCD II UART port works on a 5V logic, hence no level translation is required. Jumper JP2 may not be even required in most installations. If you are not sure, better install a jumper (5V logic only).

JUMPER SETTINGS

JP1: LCD Type

No Jumper - 2x16 or 4x20 LCD Display Hitachi Controller (default).

Jumper Installed - 4x20 LCD display using Samsung Controller.

JP2 : UART Logic Level

No Jumper - When used with a host controller running on 3.3VDC power and logic level.

Jumper Installed - When used with 5V host controller. e.g. gizDuino.

CONNECTORS

CON2 LCD PORT

The Serial LCD II will work with any 2x16 and 4x20 alphanumeric LCD display using or compatible with Hitachi HD44780 controller and Samsung S6A0073. That makes the Serial LCD II board compatible with probably at least 90% of LCD alphanumeric display modules being sold today.

CONN2 accepts both 14 and 16 pins (LCD with back light) LCD Modules with 0.1" (2.54mm) pitch single in line socket. Available as option is a 16-pin 1mm pitch FFC/FPC connector suited for the 4x20 MDLS20433 small size LCD display module.

P8 UART PORT

This is the serial I/O interface that connects to your host MCU UART port for control and data exchange. This port will work both with 5V and 3v3 logic by configuring jumper JP2 as previously discussed.

PIN	ID	Description
1	TX	UART Transmit OUTPUT
2	RX	UART Receive INPUT
3	GND	Ground
4	V+	+3v3 or Open circuit for +5V Logic

REMOTE DIO

Serial LCD II has 11 Digital I/Os (DIO) that can be configured, under user program control, to work either as input or output independently. These remote DIO essentially gives your host MCU extra ports you can use to drive, for example, LEDs, buzzers, or read switches, sensors, etc.

Two types of terminals are available as options for use with the DIO. A 2-mm pitch ZIF may be installed as P5 and P2, or a 2.54mm pitch SIL header may be used for P6 and P4. In fact, you can install both terminal connectors, if you want to,

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

Summary of Functions

F - Initialize as 4x20 LCD Display
c - Clear Display
1 - Display to line 0
2 - Display to line 1
3 - Display to line 2
4 - Display to line 3
5 - Clear & Display to line 0
6 - Clear & Display to line 1
7 - Clear & Display to line 2
8 - Clear & Display to line 3
> - cursor position
m - scroll display to line 0
M - scroll display to line 1
B - backlight Brightness 0-19
T - Test
S - set Aux Output
R - Reset Aux Output
I - Read input
i - configure as input
o - configure as output

Important:

Communications Format
Every packet of data transmission 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 as shown in the following example:

Example 1: Clear LCD Display

Transmission Format: Format: [STX]c[ETX]

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:
‘   correct way to send [STX] & ETX marker
Serial1.print(chr(2)+”c”+chr(3))

Wrong:
Serial1.print(“[STX]c[ETX]”) ‘WRONG!
```

Arduino:

```
Correct:
Serial.write(0x02); // correct way to send [STX]
Serial.print(“c”); // ASCII equivalent of ‘c’ 0x63
Serial.write(0x03); // [ETX] marker

Wrong:
Serial.print(“[STX]c[ETX]”); // WRONG!
```

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

Transmission Format: [STX]1HelloWorld![ETX]

ACII Symbol	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

Visual Basic:

```
Serial1.print(chr(2)+”1Hello 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”); // “\002” = STX, “\003” = ETX
```

Notice that in both example, only the STX and ETX marker need to be manually converted to their ASCII code, for the simple reason that they have no equivalent printable characters. The three line implementation (long 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 convert and code it in short format

1. **F** – Initialize as 4x20 LCD display

Initialize a Hitachi Controller based LCD for a 4x20 display (standard size). This function is not required for Samsung Controller based 4x20 LCD display (mini size). To ensure user firmware compatibility to both 4x20 LCDs, it is recommended that this initialization function be carried out regardless of the controller.

Format: [STX]F[ETX]

Response:
OK – Function exe

2**. c** - Clear LCD

Clear LCD display

Format: [STX]c[ETX]

Response:
OK – Ready

Example (Arduino):
Serial.write(0x02); //STX code
Serial.print("c"); //c – Clear LCD function
Serial.write(0x03); //ETX code

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

Format:[STX]>nn[ETX]

Response:
OK – Ready
ERR - Print Start position invalid

Where:

nn = start print position, 0-15(0-20)
0- left most position
15(20)– rightmost position
Default value=0

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

(Arduino):

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

4. Printing/Display Functions

1 – Print to Line 0 (Top Row)
2 – Print to Line 1 (Bottom Row)
3 – Print to Line 2 (4x20 LCD only)
4 – Print to Line 3 (4x20 LCD only)

5 – Clear and then Print message to Line 0
6 - Clear and then Print message to Line 1
7 - Clear and then Print message to Line 2 (4x20 LCD only)
8 - Clear and then Print message to Line 2 (4x20 LCD only)

Print user message to specified line. Message can contain any printable characters up to 16(20) characters long. Long messages will be automatically truncated to 16 (20) characters.

Functions 1 to 4 will simply write over any prior existing message on a line, taking the space of only those needed by the message. For example, if your program displayed "String Theory" on line 1 , and then later on print another message, say "123" on the same line, the printed result will be "123ing Theory". This feature is useful for programs that need to refresh just a small portion of the displayed message everytime. It helps to reduce display flicker.

Use functions 5 to 8 if you want to erase all pre-existing message on a line before printing the latest specified message.
Use > Print Start Position function before calling any one of the printing functions if you want to start printing on any valid position along the line.

Format: [STX]1wETX]

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

Example 1 (Arduino): Display "J3J3mon" to line

```
Serial.write(0x02); //STX code
Serial.print("1"); // 1 – print to line 0 func
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 to
Serial.write(0x02); //STX code
Serial.print(">4"); // Start on 4th character position
Serial.write(0x03); //ETX

// clear line print message

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

5. Scrolling Display Functions

m – display a scrolling message to line 0
M – display a scrolling message to line 1

Long messages can be displayed on a line using the Scrolling Display Function. Messages up to 79 characters long can be displayed on each 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]mw[ETX], [STX]**M**w[ETX]

Response:
OK = Ready

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

Example 1 (Arduino): Scrolling Display "Programmers have more fun!" to line 1

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

6. **T** – Test Communication Link

Use to test the communications link and determine device status.

Format: [STX]T[ETX]

Response:

"OK" = Ready

7. **B**- Backlight Brightness

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]Bnn[ETX]

Where:
nn – Brightness level, 0-18

Response:
- ERR = invalid level
- OK = Ready

Example (Arduino): Set LCD backlight brightness to 5

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

8. **S,R,I,i,o** Serial LCD Auxiliary I/O functions

S – Set an output (set to Logic 1)
R – Reset an output (reset to Logic 0)
I – Read an input
i – configure as input
o – configure as output

The Serial LCD II has 11 digital I/O port that you can use as slow speed I/O in your applications.

I/O are configured as inputs by default. You can configure any number of available I/O as outputs if it is so desired. Generally, configuration is done only once during program initialization, but you can configure any I/O to act as input or output at any time.

Format: [STX]Sn[ETX]
[STX]Rn[ETX]
[STX]In[ETX]
[STX]in[ETX]
[STX]on[ETX]

Where:
n – I/O number 0-10

Response:

S,R,i,o function
"OK"
"ERR" - Invalid I/O number

I function
"0" - input at logic 0
"1" - input at logic "1"
"ERR" - Invalid I/O number

Example (Arduino): Set I/O 5 as an Output

```
Serial.write(0x02); //STX code
Serial.print("o"); // Set an output
Serial.print("5"); // output 5
Serial.write(0x03); //ETX
```

Example (Arduino): Set I/O 5 to logic "1"

```
Serial.write(0x02); //STX code
Serial.print("S"); // Set an output
Serial.print("5"); // output 5
Serial.write(0x03); //ETX
```

Example (Arduino): Read state I/O 10

```
Serial.write(0x02); //STX code
Serial.print("I"); // Read an INPUT
Serial.print("10"); // input 10
Serial.write(0x03); //ETX
```

Tip: You can enable the built-in pull up of any input
port by writing logic "1" on it (using the "S" function)

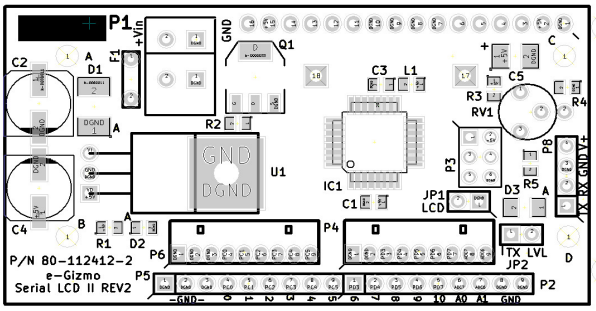


Figure 3. Silkscreen Guide

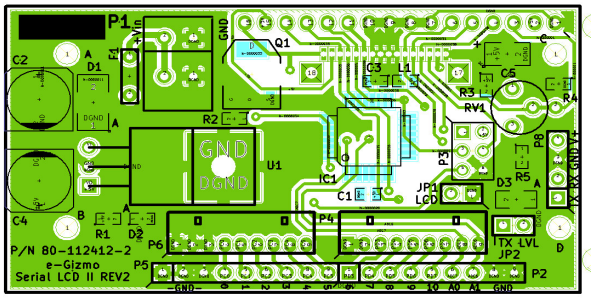


Figure 4. Bottom Copper Guide

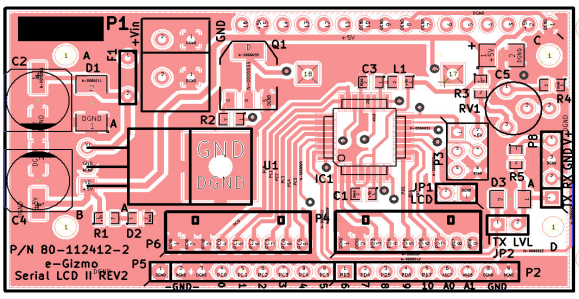


Figure 5. Top Copper Guide

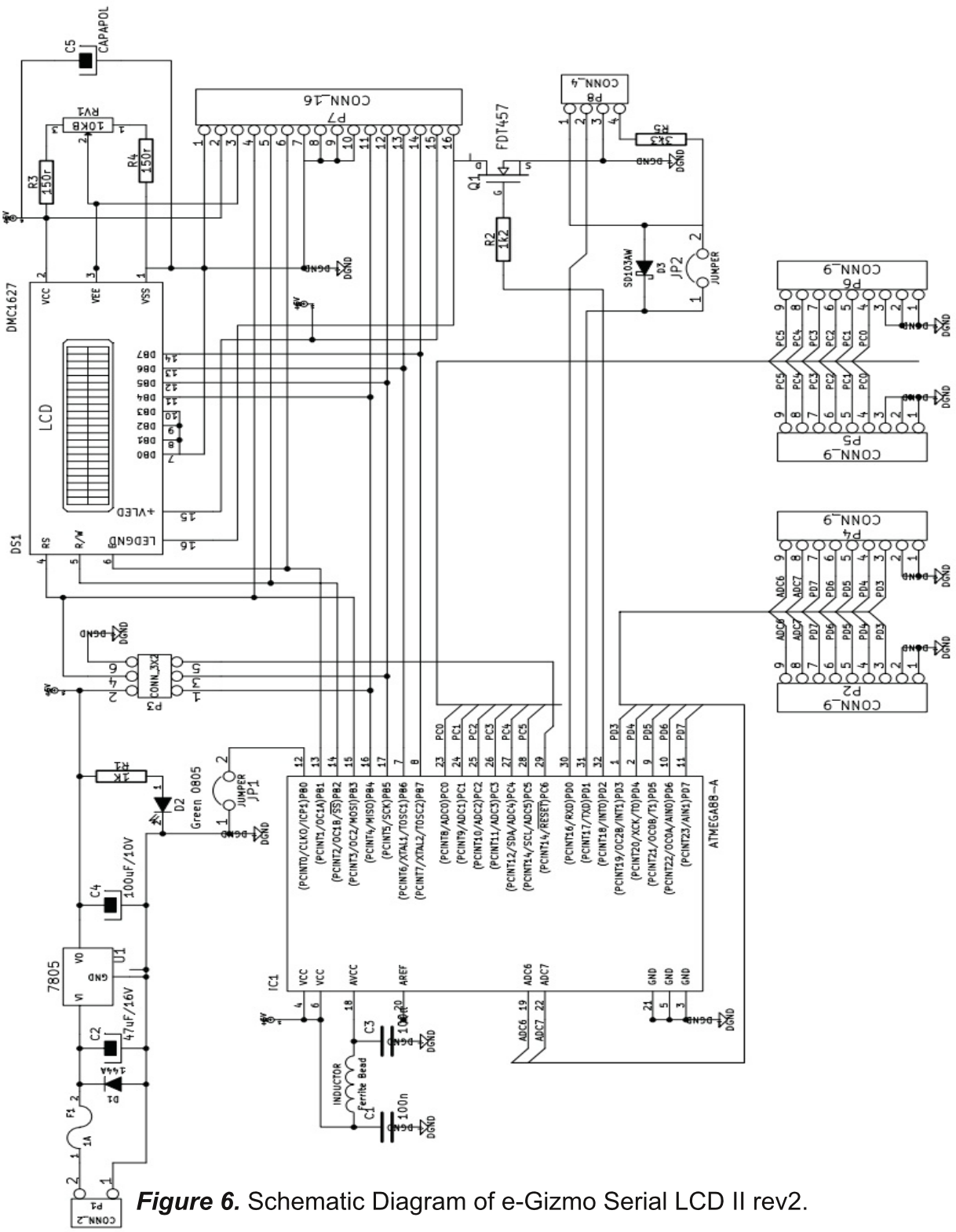


Figure 6. Schematic Diagram of e-Gizmo Serial LCD II rev2.

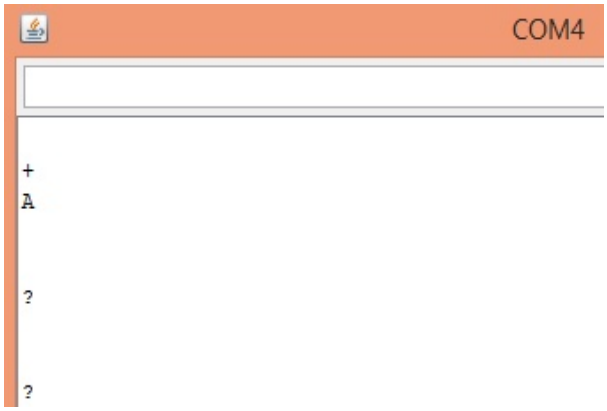
Sample Serial Output

SerialEvent example

Open the SerialEvent sample modified.

This is a good test to try it.

serial print when new serial data arrives the loop prints the string and clears it. (See the Figure)



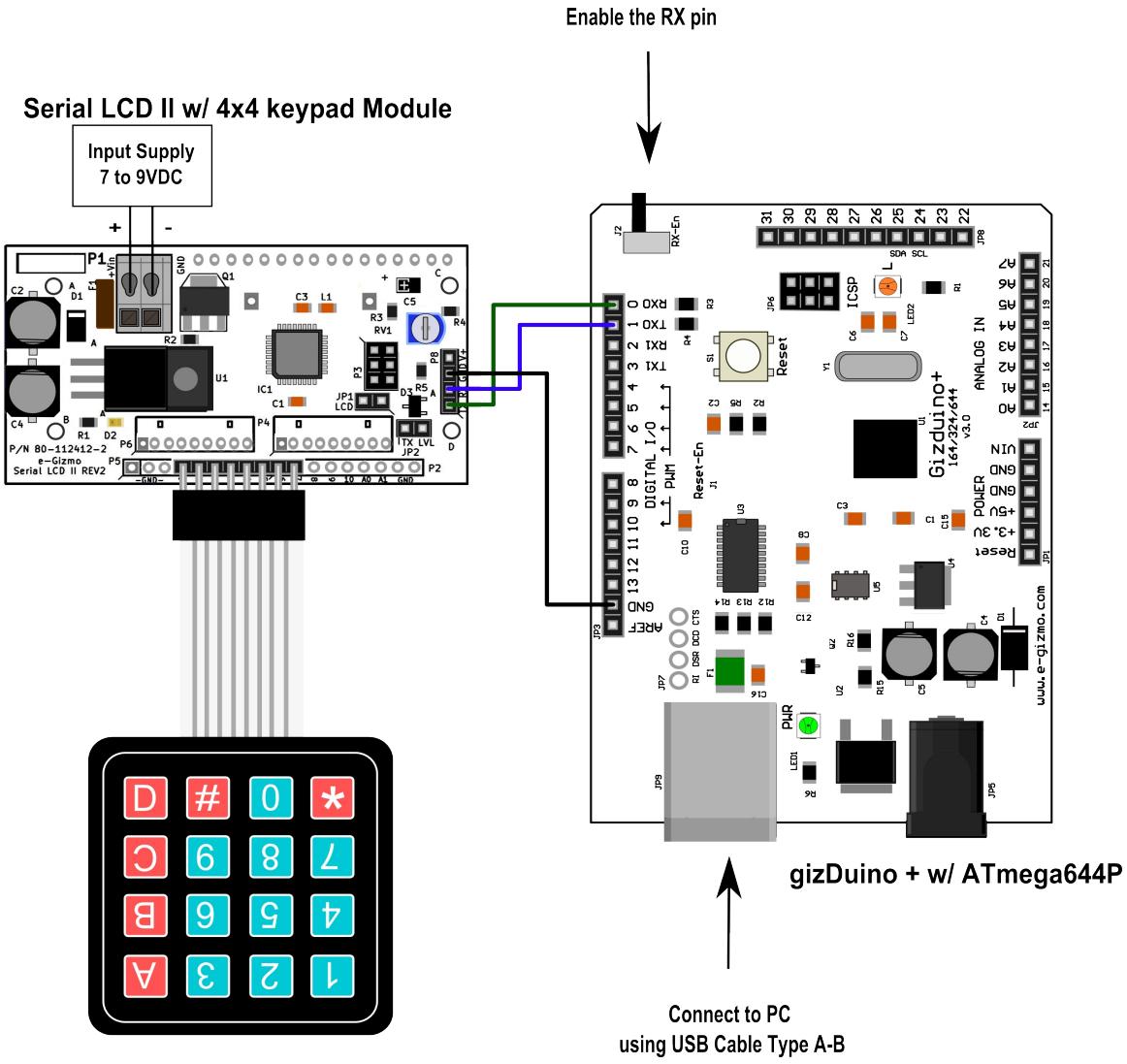


Figure 7. Sample Application of e-Gizmo Serial LCD II rev2 with 4x4 Keypad Module.