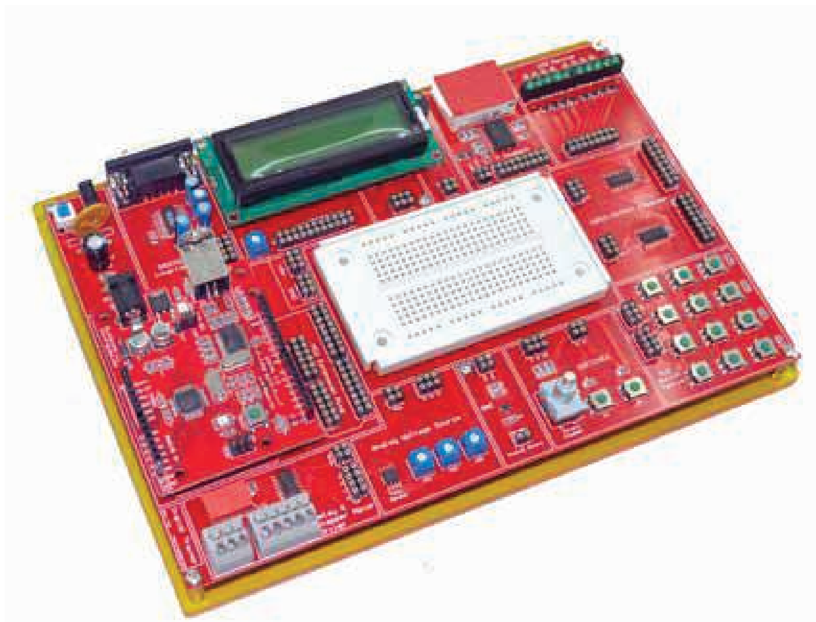
e-Gizmo

Universal MCU Trainer

Hardware Reference Manual
Rev 0r1 Preliminary



e-Gizmo Universal MCU Trainer is a convenient hardware peripheral platform designed to make your MCU learning experience a lot more enjoyable.

It is not committed to any single micro-controller; instead we took a novel approach by using and taking advantage of a gizDuino style MCU docking port. This allowed different e-Gizmo MCU boards to be used with the trainer. This is what set e-Gizmo Universal MCU Trainer apart from other run of the mill trainers. With the Universal MCU Trainer, you can immediately start your training with an MCU family of you choice using our currently available boards that include the gizDuino itself (atmega168 or atmega328 version), AVR atmega8L experimenter board, Zilog z8F042A MCU board, PIC18F2550, and STM32F100C8 ARM Cortex M3 MCU board. Take your pick. More MCU board options will become available in the future.

What's on Board

- Relay and Motor Driver
- Analog Voltage Sources
  - LM34 Temperature Sensor
  - 3 adjustable voltage sources
- Digital to Analog DAC ciruit
- Switch Devices
  - Rotary Encoder Switch
  - 2 push button switches
- 4x3 Keypad Matrix
- Input and Output Expander
- LED Monitor
- 2-Digit 7 segment LED Display
- Buzzer Circuit
- Real Time Clock RTC
- Alphanumeric LCD Display Unit
- RS232 Interface circuit
- Power Supply Entry
- Microcontroller Board Docking Port
- Breadboard

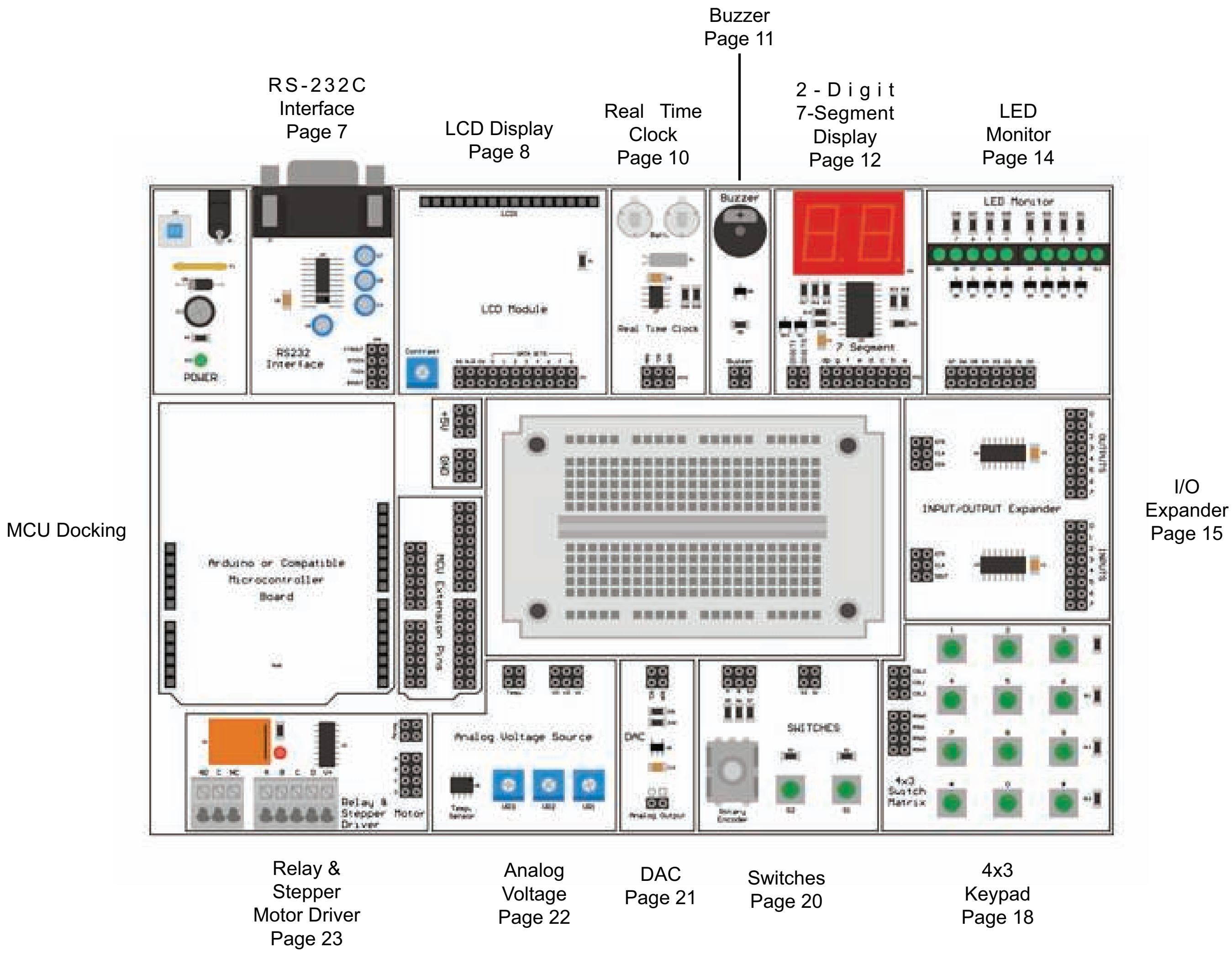


Figure 1. Universal MCU Trainer Board Layout. Clicking the text will take you to its corresponding discussion page.

GENERAL INFORMATION AND PRECAUTIONS

The Universal MCU Trainer Board requires an external well regulated +5V to power the whole trainer board. Use only the supplied AC/DC adapter unit. Using unqualified +5V power source may damage the trainer and void its service warranty.

All on-board experiment circuits that requires +5VDC power are all connected to the +5V power bus. There is no further need for a GND and +5V wiring points for each circuits.

Most components used in the trainer board, especially the MCUs, are sensitive to Electrostatic Discharge ESD and may be permanently damage if not handled properly. Readers are urged to read this article before working with the trainer.

http://www.e-gizmo.com/ARTICLES/ESD/ESD1.HTM

ESD precautions apply to all electronic components, not just with this trainer.

Wire Sockets

The Universal MCU Trainer is equipped with wire sockets that use a 2 column hi-rel socket connectors. Each pair in a row are connected together in the board, giving you at least two connect points for every port.

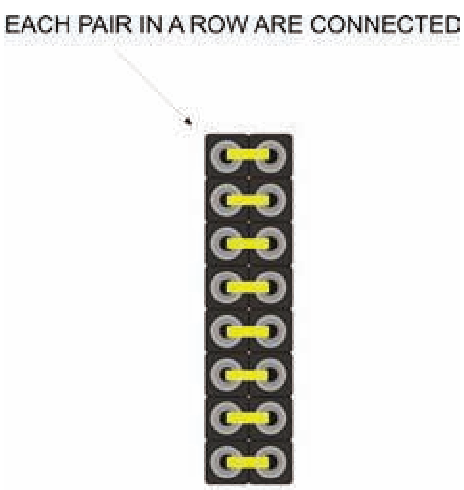


Figure 2. Every device terminal port provides two ways to connect. Each rows are connected together.

The socket connectors will work well with solid AWG25 or AWG26 wires. To connect a wire, simply insert the wire all the way in with moderate force. Each time you insert a wire for connection, you should always check afterwards if the wire is sufficiently grasped by the socket. You can do this by pulling the wire very slightly. You should feel the wire resisting the pull.

If frequent usage wear a socket and could no longer hold the AWG26 wire, you could use a slightly larger wire, i.e. AWG25, to continue using the trainer. If the wear is too severe, the socket will have to be replaced.

Wire Stripping

Solid wires are notoriously easy to break, and the weak point is the area where your stripping left some mark in the copper wire itself. If you stripped the wire too short, the broken portion may slide deep inside the wire sockets. This will make it difficult, if not impossible, to extract the broken portion, and may require socket replacement to be able to use it again.

To prevent this, you should strip the wire starting at least 7mm from the tip, as Fig. 3 shows. This will ensure enough length of wire will protrude that will make extraction easy in case the wire is broken.

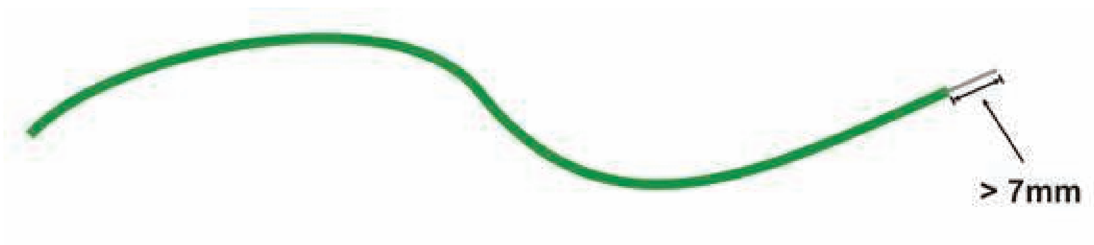


Figure 3. Wire Stripping Guide. The stripped portion should be at least 7mm long.

BreadBoard

The Breadboard will allow you to easily build small circuits that you may want to work on in addition to those already in the trainer module. The breadboard used in this trainer has two separate power bus. Located at the top is a single row connector that is all electrically connected with each other. Similarly, the single row located at the bottom is also connected together. Each socket remaining are connected together in groups of 5 in the column direction.

Avoid inserting components having leads measuring more than 0.8mm diameter or in the widest direction. These may damage the breadboard contact that may result in poor and intermittent connections.

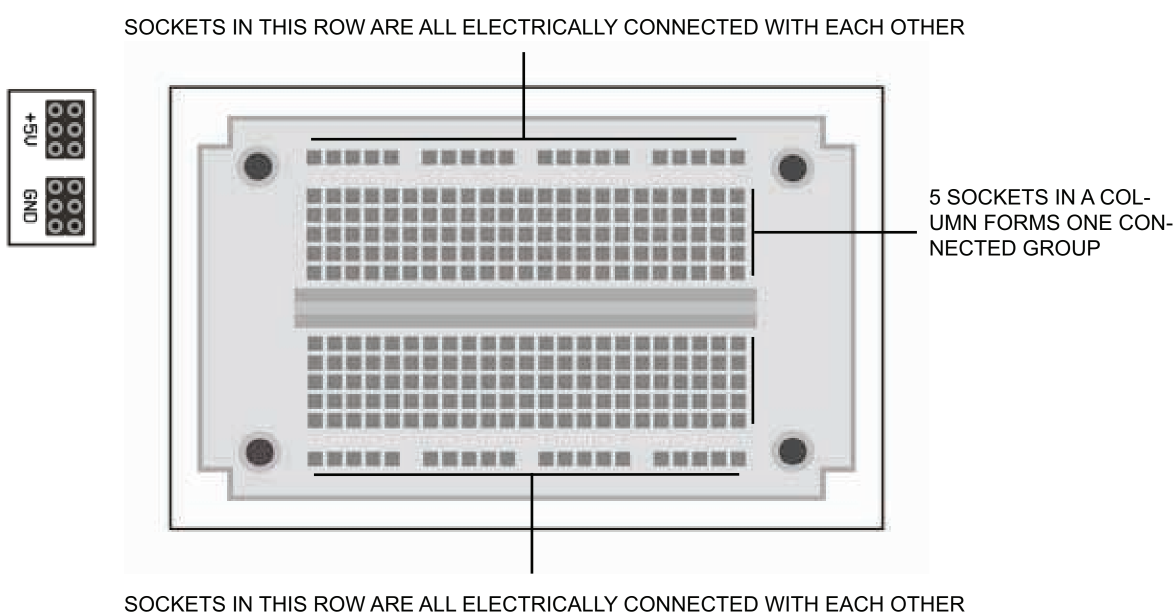


Figure 4. The breadboard consist of 270 tie points and is ideal for constructing temporary circuit during experiments.

MICROCONTROLLER BOARD DOCKING

This is where you install the MCU board of your choice. This docking port will accept any gizDuino style MCU board. Arduino boards, incidentally, do not use extended connectors. This, in effect, disallows any shields from being installed under it. To use the Arduino with the Universal MCU Trainer, a suitable adapter board is required.

List of current gizDuino Style MCU Boards you can use with the Universal Trainer:
- gizDuino168 and gizDuino328
- AVR ATMEGA8L experimenter board
- Zilog z8F042A MCU Board
- Pinguino PIC18F2550 MCU Board
- STM32F100C8 ARM Cortex M3 MCU Board

More will be added on this list in the near future. Note that each MCU family will require its own software developer kit SDK. Some may require a separate programming cable and/or debugging cable.

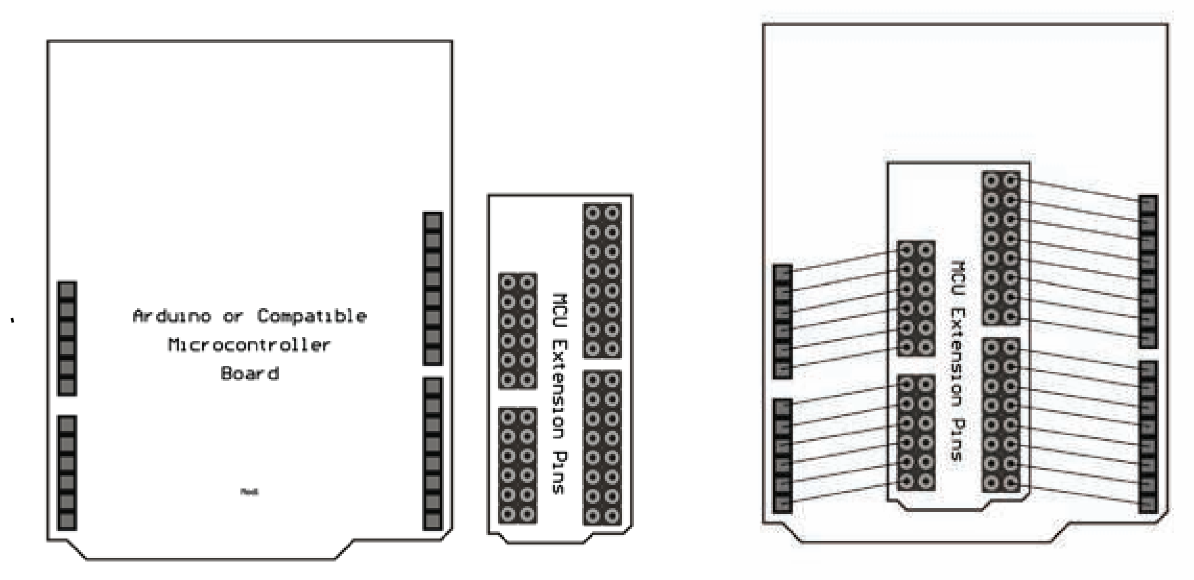


Figure 5. MCU Board Extension Pins equivalent wiring diagram.

MCU I/O Port

Once the MCU board is installed, it's I/O and special functions terminals are now available through the MCU Extension Pins connector. You connect the trainer's peripheral from this port. Alternatively, you can do the connections directly at the MCU board terminals, but it is not recommended. Keeping the MCU board terminals free will allow you to easily add extra shield or remove the MCU board. There some boards though with extra I/O pins connected to non-conforming terminals. In these cases, direct MCU board connections are the only way to go.

The MCU extension pins does what its name suggest, provide extension connection for the MCU board pins. An equivalent wiring diagram of these pins in relation to the MCU board is shown in figure 5.

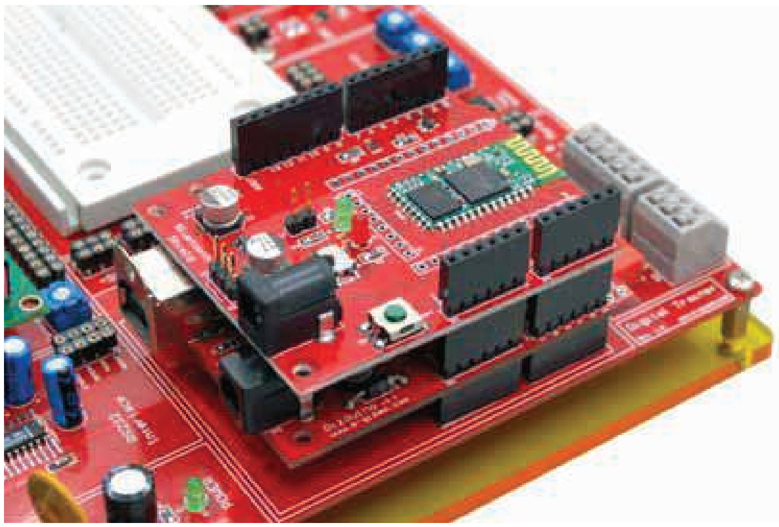


Figure 6. A shield can be added along with the MCU board to entend your learning experience. As shown in this photo, a Bluetooth shield will allow you to experiment with wireless Bluetooth enabled devices.

RS-232C INTERFACE

The RS232 Interface is a RS-232C to TTL logic level converter. It allows the MCU UART to interface with RS-232C enabled devices. Examples of RS-232C devices includes the PC (COM port), modem, serial printers, serial barcode scanners, etc..

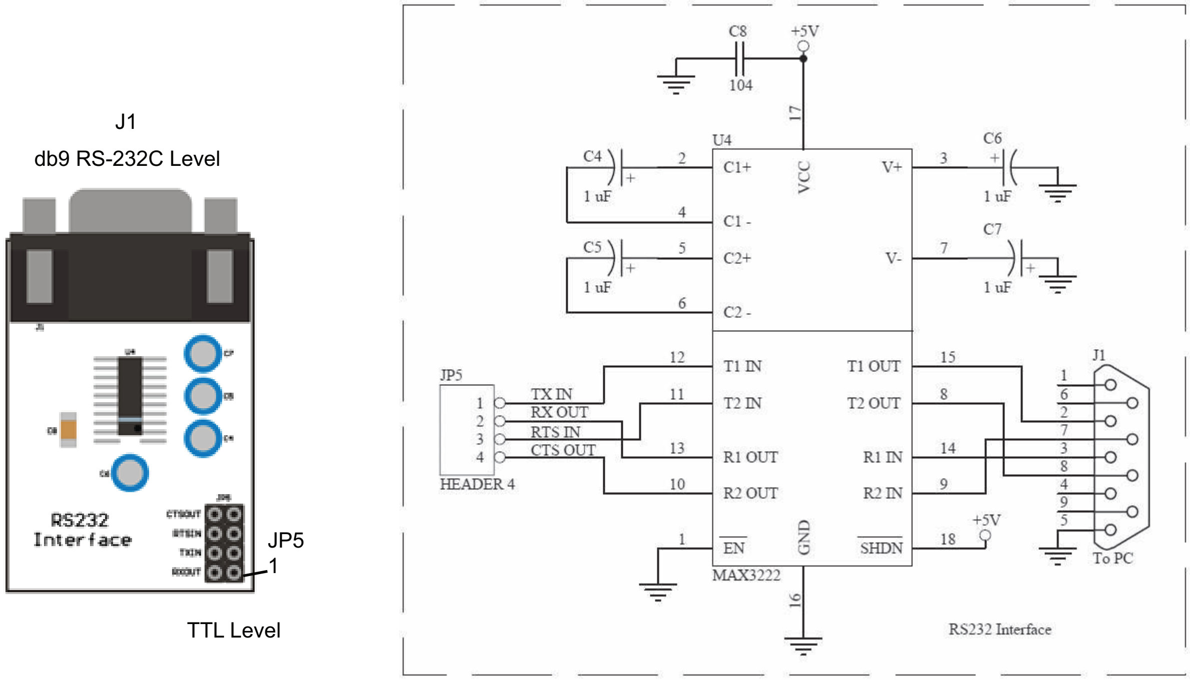


Figure 7. RS-232C Interface components layout and schematic diagram.

Table 1. JP5 RS-232 Interface TTL side

Pin No.	ID	Description
1	RX OUT	RX Data Output
2	TX IN	TX Data input
3	RTS IN	RTS Handshake Input
4	CTS OUT	CTS Handshake Output

Table 2. J1 RS-232 Interface DB9 side

Pin No.	ID	Description
2	TX OUT	TX Data output
3	RX IN	RX Data Input
8	RTS OUT	RTS Handshake Output
7	CTS IN	CTS Handshake Input
5	GND	Signal Ground

LCD MODULE

LCD Display Module is a tremendously popular low cost display device that can show user generated messages in 2 lines x 16 alphanumeric characters format. The 2x16 LCD display model included with the kit is based on Hitachi HD44780 LCD controller - the de facto industry standard for this class of displays. Having a standard, even if unofficial, is a good thing, since that means no hardware or firmware changes is necessary should you switch from one LCD brand to another.

Aside from 2 lines by 16 characters format, LCD modules are also available in 4 lines by 20 characters, plus more. In fact, there is a big collection of LCD modules out there each sporting a different character display configuration. Yet basically, they use the same HD44780 controller, the same LCD module pin out, and hence would work with common LCD routines. You only have to watch out for characters going outside of the display area.

Table 3. LCD Module Socket

Pin No.	ID	Description
1	GND	Ground
2	VDD	+5VDC Power
3	VLC	LCD Contrast Adjust
4	RS	Register Select
5	R/W	Read/Write
6	E	Enable
7	D0	Data Bit 0
8	D1	Data Bit 1
9	D2	Data Bit 2
10	D3	Data Bit 3
11	D4	Data Bit 4
12	D5	Data Bit 5
13	D6	Data Bit 6
14	D7	Data Bit 7
15	LED+	Backlight LED +
16	LED-	Backlight LED -

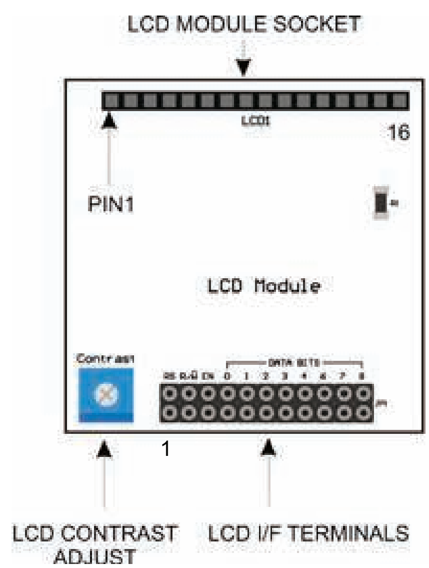


Figure 8. LCD Module port component layout.

Table 4. JP1 LCD Interface Port

Pin No.	ID	Description
1	RS	Register Select
2	R/W	Read/Write
3	EN	Enable
4	D0	Data Bit 0
5	D1	Data Bit 1
6	D2	Data Bit 2
7	D3	Data Bit 3
8	D4	Data Bit 4
9	D5	Data Bit 5
10	D6	Data Bit 6
11	D7	Data Bit 7

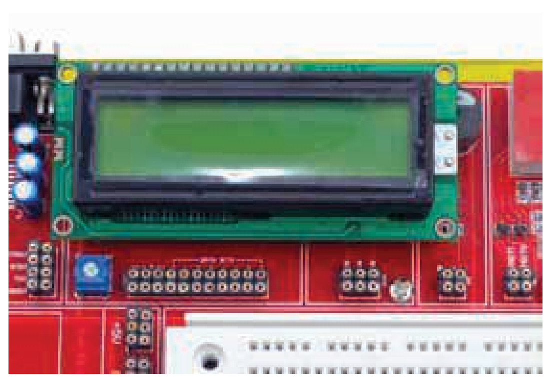


Figure 9. Photo showing a 2 lines x 16 characters LCD display installed in the LCD Module port.

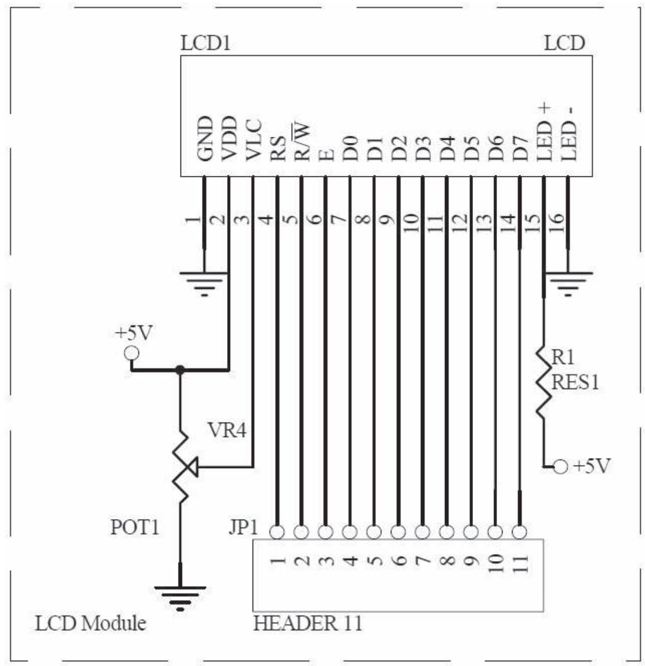


Figure 10. LCD Module port schematic diagram.

LCD Contrast Adjustment

If upon the application of power the LCD does not display anything, the LCD contrast adjustment may have been set way off. Simply adjust the LCD contrast trimmer until the characters appears and are displayed with the desired contrast. Too much contrast will make the characters appear as line of blocks.

Application Hints

- Most applications does not require a data read from the LCD Module. R/W line can be permanently connected to GND in these cases.
- The LCD Module can be configured to work with either 8-bit or 4-bit data bus. This must be set during the initialization routine. 4-bit data transfer uses only D4-D7, D0-D3 lines can be left unconnected.

Initialization

The LCD's HD44780 is a very slow controller, it does not want to be rushed into doing things. This is especially true during the initialization. If you did not get the initialization right, you will end up with an application that crashes the LCD often or one that will not function at all.

A good and detailed article about LCD initialization can be downloaded from this site:

http://web.alfredstate.edu/weimandn/lcd/lcd_initialization/lcd_initialization_index.html

REAL TIME CLOCK RTC

A Real Time Clock RTC circuit is an electronic subsystem that keeps an accurate time and calendar. It is in fact an electronic watch. Based on DS1307 chip, this module has battery back up so that it can continue its time keeping functions even when the main power is turned OFF.

Other features includes an on-chip RAM (56 bytes, you can use these to store non permanent data that stays put even after power is removed from the circuit) and I2C data interface.

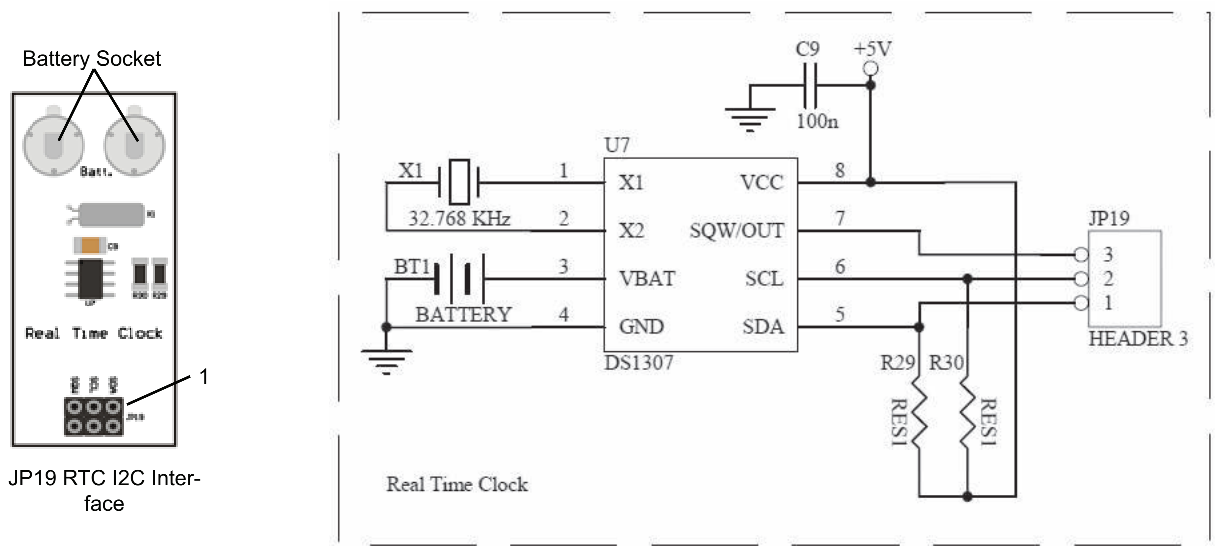


Figure 11. RTC section component layout and schematic diagram. A Real Time Clock RTC is an electronic watch that can keep an accurate time even with power removed.

Table 5. JP19 RTC I2C Interface Port

Pin No.	ID	Description
1	SDA	I2C Serial Data I/O
2	SCL	I2C Serial Clock Input
3	SQW OUT	Square Wave Output

For a detailed description of the DS1307, please refer to its corresponding device datasheet attached in the appendix section of this document.

Important: The RTC may not function properly if the back-up battery is not installed.

BUZZER

A buzzer, when used as annunciator, provides cost effective audible feedback between the machine and the user. Use it to indicate a warning or an alarm, a key closure, a machine function, etc.

The buzzer used in this trainer is essentially a small loudspeaker. It is capable of converting to sound input frequencies from 300Hz to about 5Khz, and is loudest at around 2000-2400Hz. With it, you can write codes that generates sound effects and musical tones.

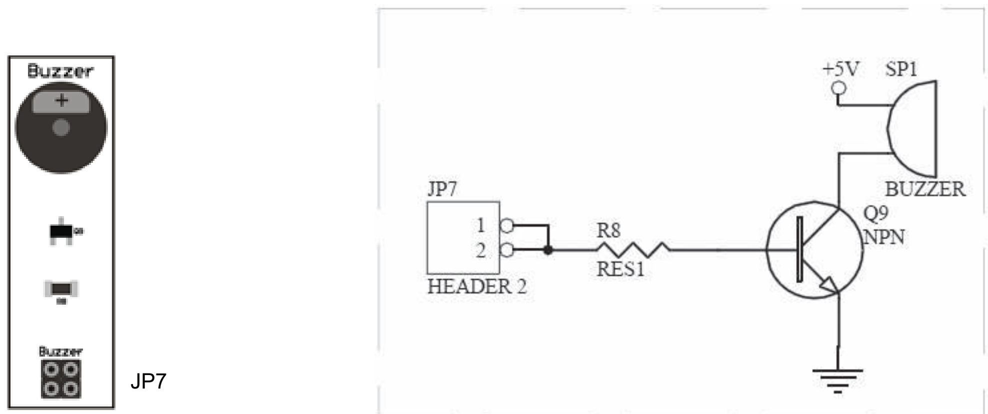


Figure 12. The Buzzer section component layout and schematic diagram. You can program your MCU to produce musical and sound effects through the buzzer.

Table 6. JP7 Buzzer Input Port

Pin No.	ID	Description
1	IN	Buzzer Input
2	IN	Buzzer Input

2-DIGIT 7-SEGMENT LED DISPLAY

Even with the growing popularity of LCD displays, there is still a huge number of applications where 7 segment LEDs display simply has the edge. To name a few examples, LEDs still rules on clock display, timers, queue counter, score boards.

This 2-digit multiplex 7-segment LED display circuitry will enable you to study hands on the principles involved in multiplexing LED displays, and implement it in codes. Once you got a good grasp, adding more digits will be an easy task.

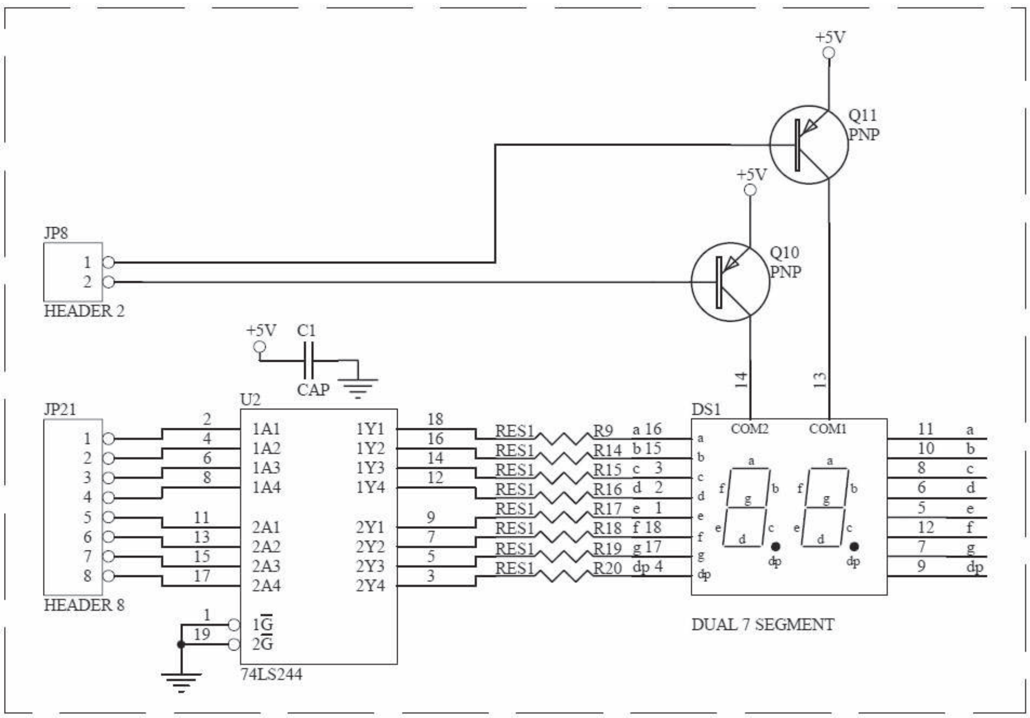


Figure 13. The 2-digit 7-segment LED display circuit. The LEDs are wired in common anode multiplex configuration.

About the Circuit

Multiplexing is a method used to rapidly switch two (or more) digits in synchronization with their corresponding segment drives (a-b-c-d-e-f-g). This rapid switching will appear to us as if both digit are permanently on displaying two numbers. For example, if segment b & c are turned ON with Q10 ON, the leftmost digit will display a "1". With Q11 ON (and Q10 OFF), the "1" will be displayed instead to the rightmost digit. If you do this in rapid succession, your eyes will see an "11" displayed. If a different segment is turned ON during Q11 switch ON, say a-b-c-d-e-f-g segments are all on for an "8" display, what your eyes will perceived will be "18".

The circuit is a two-digit 7 segment common anode display, and can be made to work just as depicted. Q10 and Q11 are the common multiplexing drive. Driving the gate of these transistors to near GND level or Logic 0 will switch them ON, turning ON the corresponding digit display. It is important to note that these transistors are equipped with internal resistor bias; hence external base current limiting transistors are not needed.

A fairly moderate amount of current is needed to drive individual led segments to a desired brightness. This current drive cannot be provided by the MCU itself; hence U2 buffer is added in the circuit to act segment drive. Resistors are put in series with the segment drives R9-R17 to limit the drive current to a level safe for the LED segments.

Table 7. JP21 7-segment Drive Input

Pin No.	ID	Description
1	a	a drive input, active low
2	b	b drive input, active low
3	c	c drive input, active low
4	d	d drive input, active low
5	e	e drive input, active low
6	f	f drive input, active low
7	g	g drive input, active low
8	dp	dp drive input, active low

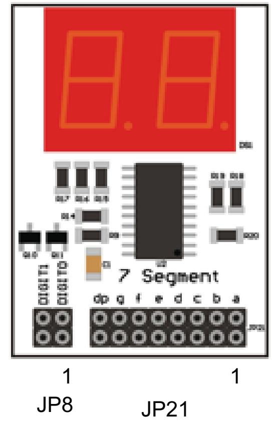


Figure 14. The 2-digit 7-segment LED display component layout.

Table 8. JP8 Common Anode MPX Drive Input

Pin No.	ID	Description
1	digit0	Least significant digit, active low
2	digit1	Most significant digit, active low

Displaying Numbers

One important thing to note in this display subsystem is that all control inputs are Active Low, meaning, they will turn ON an LED segment when the corresponding control input is set to Logic 0. Unlit segment inputs, hence, must remain in logic 1 state.

You can display a numeric character by turning ON a set of segment, as shown in Fig. 15. As the illustrations shows, a “2” can be displayed by pulling to logic 0 inputs a-b-d-e-g. An “8” will be displayed if all segment inputs are pulled to Logic 0. The decimal point can be turn ON by pulling dp input to Logic 0.

The 7-segment circuit did not use any decoder to generate a display character. The user program has full control to what can be displayed. Hence, you can generate pseudo alphanumeric characters, and whatever 7-segment pattern you wish to display, in fact.

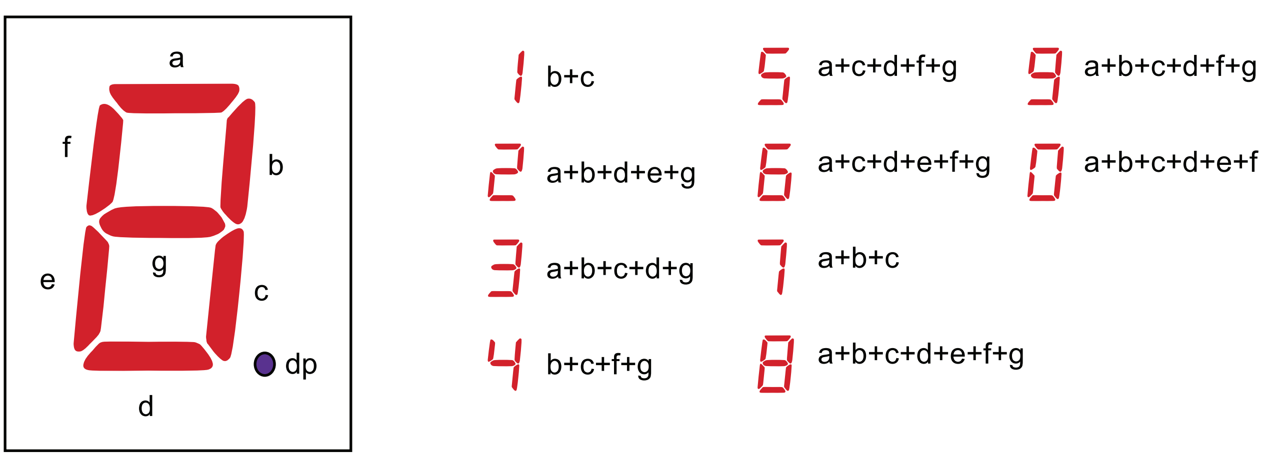


Figure 15. 7-segment LED display map. A numerical character can be displayed by turning ON a set of segment as shown in the illustration. You can even map your own character pattern.

LED MONITOR

The LED monitor is a set of 8 LEDs indicator than can conveniently display the state of its corresponding monitored I/O. It is, as a matter of fact, a simple 8-channel logic probe. It is useful as a troubleshooting tool, and can be used as well as visual indicators.

Each leds are buffered by a transistor buffer. This presents a high impedance load to the port it monitors, hence, will generally not affect the port it is monitoring.

The transistors used in this circuit has built-in bias resistor (digital transistor) and does not require an external base current limiting resistor. R21-R28 limits the current flowing through their corresponding LEDs to a safe value.

To turn ON an LED, apply a Logic 1 to its corresponding input port.

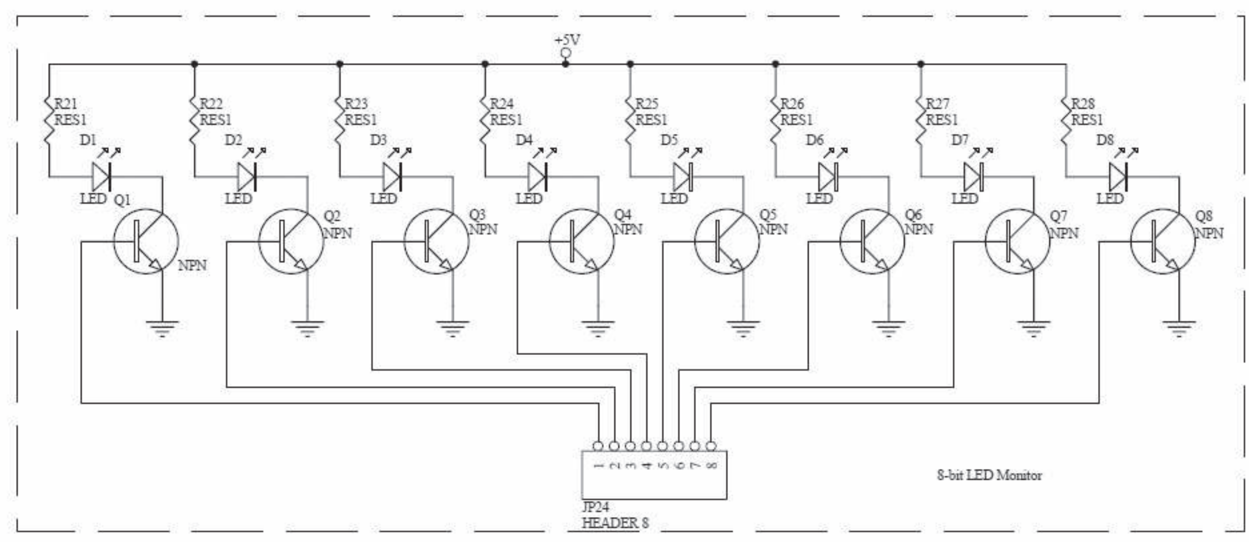


Figure 16. The LED monitor are all buffered by transistors with built-in bias resistors. The buffers presents a high impedance load to the I/O it is monitoring, hence, loading will not be a problem.

Table 9. JP24 LED Monitor Drive Input

Pin No.	ID	Description
1	D0	LED D1, active high
2	D1	LED D2, active high
3	D2	LED D3, active high
4	D3	LED D4, active high
5	D4	LED D5, active high
6	D5	LED D6, active high
7	D6	LED D7, active high
8	D7	LED D8, active high

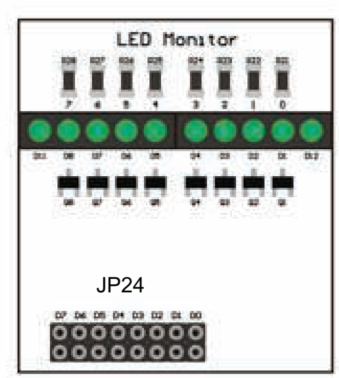


Figure 17. The LED Monitor Component layout. Only eight of them are used for I/O monitoring. The two extra LEDs are uncommited and may be used for other purpose.

INPUT/OUTPUT EXPANDER

As you progress your skill with microcontrollers and start building circuits, you may sometimes find the need for more number of I/Os than the device can provide. You could either use a microcontroller with more I/O pins, or simply use a serial Input and Output expander chip.

The input and output chip you will use in this trainer are actually run of the mill shift registers, hence, you will learn a cost effective way of increasing the number I/O without getting your microcontroller replaced.

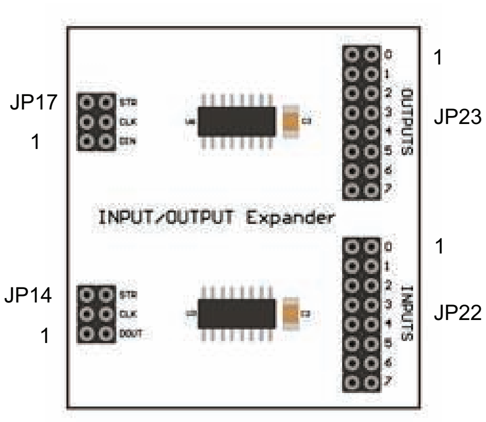


Figure 18. Input and Output expander component layout. This I/O expander can serve 8 input + 8 output more using 5 or 6 MCU I/O.

Table 10. JP17 Expanded Output Control Port

Pin No.	ID	Description
1	DIN	Data Input
2	CLK	Clock Input, rising edge
3	STR	Strobe Input,active low

Table 11. JP23 Expanded Output Port

Pin No.	ID	Description
1	0	Q1 Output
2	1	Q2 Output
3	2	Q3 Output
4	3	Q4 Output
5	4	Q5 Output
6	5	Q6 Output
7	6	Q7 Output
8	7	Q8 Output

Table 12. JP14 Expanded Input Control Port

Pin No.	ID	Description
1	DOUT	Data Output
2	CLK	Clock Input, rising edge
3	STR	Strobe Input,active low

Table 13. JP22 Expanded Input Port

Pin No.	ID	Description
1	0	P0 input
2	1	P1 input
3	2	P2 input
4	3	P3 input
5	4	P4 input
6	5	P5 input
7	6	P6 input
8	7	P7 input

About The Circuit

The complete schematic of the Expanded I/O subsection is shown in Fig. 19. Taking a quick look at the control inputs JP14 and JP17, you will notice the control inputs are nearly identical. Both have STR- strobe, CLK- clock, and Data pins, differing only with the direction of the data line. The input expander has DOUT – Data Out, since your host controller will be reading from U3 input port JP22. The output expander, on the other hand, has DIN- Data In, since your host controller will be writing to U6 the bit pattern you want to appear at its output port JP23.

Each data port requires only three I/O pins from the MCU the read 8 bits input or write 8 bits output. In fact, it is possible for both devices to share common CLK pin source, saving one more MCU I/O pin. And there is more. Although it is not implement in this trainer, these chips actually has extra pins that will allow you to daisy chain more chips without the burden of additional I/O pins from he host controller. For example, adding one more 4094 chip daisy chained to U6 will add 8 more outputs; two will add 16 more, and so on. So now, you are using only three MCU I/O pins to control 16, 24, maybe more, outputs. The same principle applies to the input expander circuit U3. Here now you can see the tremendous advantage when you apply this technique to your circuit.

But there is a caveat. Serial access tends to slow things down, input or output access won't be as fast compared to direct MCU I/O access. This will not be a problem at all if your input or output devices does not require fast actions. Examples these I/O devices includes switches, LED display, relay. Examples of I/O devices that may not work well with expanded I/Os are high speed pulse or PWM input and output devices.

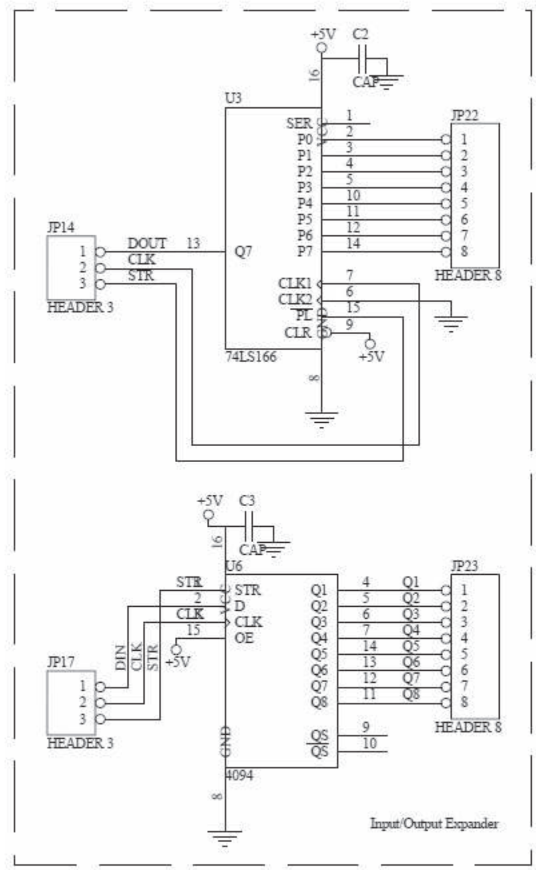


Figure 19. Input and Output expander schematic diagram.

Expanded input

The expanded input circuitry takes an 8 bit input and , under a host MCU control, transfer these serially via a single I/O port. The Input expander is based on 74LS166 parallel input to serial output shift registers. Input read and serial transfer is done as follows:

1. Start with STR=1 and CLK=0
2. Reset and then set STR (STR=0 then STR=1). This will load the current external input port states to U3 internal registers in preparation for transport. P7 state is now available for reading at DOUT
3. Pulse CLK (CLK=1 and then CLK=0). P6 state is now available at DOUT.
4. Repeat step 3 six more times to flush out and read the remaining 6 bits.

Expanded Output

The expanded output circuit works in a similar way as described above, but now the data is moving in the opposite direction. Data is send serial by the host MCU and is assembled as a parallel 8 bit data by U6 MC14094 shift/store register.

1. Start with STR=0 and CLK=0
2. Write to DIN the desired output state of Q8.
3. Pulse CLK (CLK=1 and then CLK=0). Q8 state is now copied in the internal registers of U6.
4. Repeat steps 2 to 3 seven more times to transfer Q7 to Q1, each time replacing DIN states correspondingly. During this process, the output Q8-Q1 still retains their old states, and will be changed at the same time with the latest entered state once step 5 is executed.
5. Set and the reset STR (STR=1 and then STR=0) to move and latch the latest transferred bit data pattern to their corresponding port JP23.

For a detailed description of the 74LS166 and MC14094 chips, please refer to their corresponding device datasheet attached in the appendix section of this document

4X3 KEYPAD MATRIX

A keypad is a small group of keys used to manually enter a data or command while the MCU is running. To save I/O pins, keypads are usually wired in matrix connection. The keypad used in this trainer consists of 12 push switches. If connected as direct inputs, these keys will correspondingly require 12 input pins. Instead, these keys are wired in a 4x3 matrix arrangement as depicted in Fig. 20. This technique would require only 4 + 3 = 7 I/Os to read the keypad instead of the usual 12, saving 5 I/Os that can be used for other purpose.

Because of the advantage thus described, majority of keypads are connected in matrix arrangement. Learning how to use and program for one is an essential skill every MCU programmer should have.

The keypad switches are wirred in 4 rows x 3 column arrangement. As such, it has 4 bit port JP4 connecting its rows and 3 bit port JP18 connecting its columns. For proper operation, the rows must connect to 4 MCU inputs and the columns must connect to 3 MCU outputs. The rows lines are provided with pull up resistors R11-R13, hence, the row inputs are on normally high state. A key closure is detected if any one of the row input goes to logic 0. Hence, the keypad matrix must be scanned by pulling to logic low a column pin one at a time and then reading the row pins each time for any key closure. The program can then resolve which key is pressed (if any) and then perform the desired action.

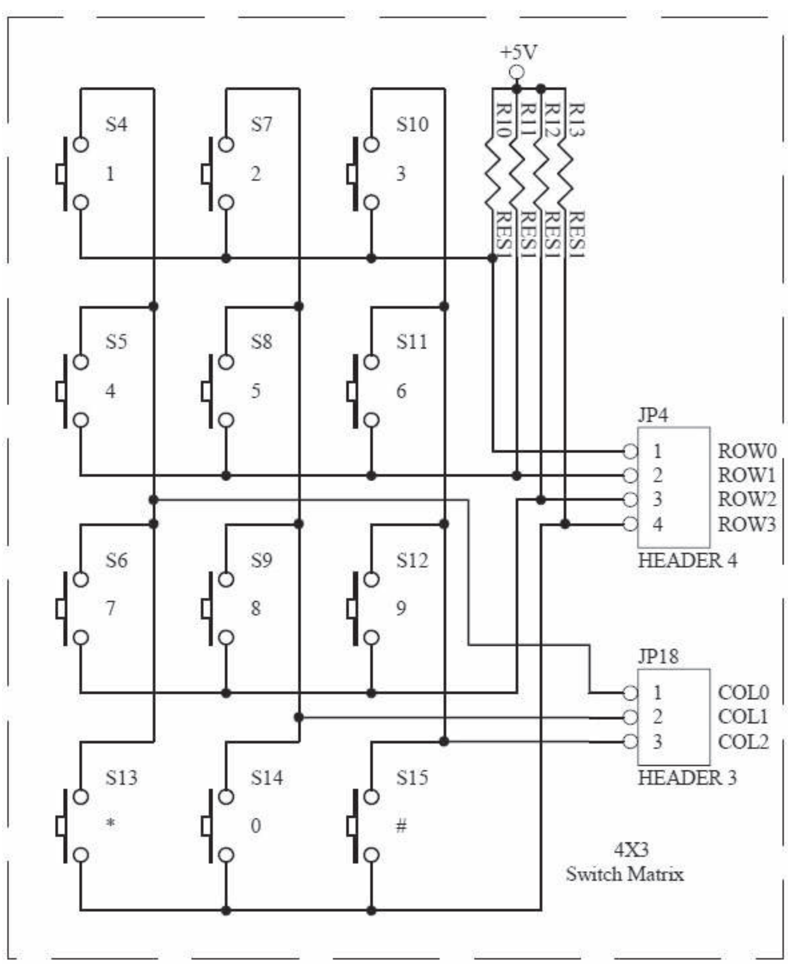


Figure 20. The 4x3 Keypad Schematic Diagram. The MCU output scans the keyboard by driving each col pins to logic 0 one at a time and then reading the rows pin. A key closure will cause at least one row pin to read a logic 0. User code can then decode which key is pressed.

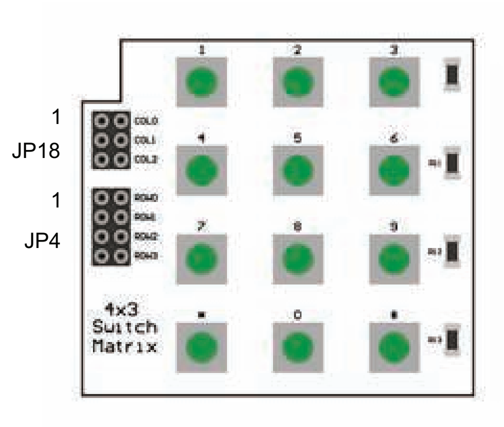


Figure 21. The 4x3 Keypad switch layout. The switch are abitrarily labelled with numbers and symbols commnonly used with telephone sets. You can label the keys in anyway that will suit your application.

Table 14. JP18 Column Scan

Pin No.	ID	Description
1	COL0	Connect to an MCU Output
2	COL1	Connect to an MCU Output
3	COL2	Connect to an MCU Output

Table 15. JP4 Row Output

Pin No.	ID	Description
1	ROW0	Connect to an MCU input
2	ROW1	Connect to an MCU input
3	ROW2	Connect to an MCU input
4	ROW3	Connect to an MCU input

ENCODER SWITCH AND PUSH BUTTON SWITCHES

The switch subsection consists of a pair of normally open push button switch, and a rotary encoder switch.

A rotary quadrature encoder switch, or simply encoder switch, is a pure digital device that has the feel of an analog potentiometer. In fact, it is used much in the same way. You rotate it clockwise or counterclockwise to increase or decrease something the device is doing.

A short article detailing the operation and usage of this device can be downloaded from

http://e-gizmo.com/wordpress/?p=139

A copy of this article is also included in the appendix of this document as a reference.

Push button switches, S1 and S2, are held to logic 1 state by R2 and R3 when not pressed. Pushing a push button switch will pull to logic 0 the corresponding MCU input pins it is connected to.

The rotary encoder switch is biased in a much the same way. All its terminals are normally at logic 1 and pulled to logic 0 upon contact closure.

The rotary encoder switch in this trainer also has a push button switch S3 that can be activated by a gentle downward push of the encoder shaft.

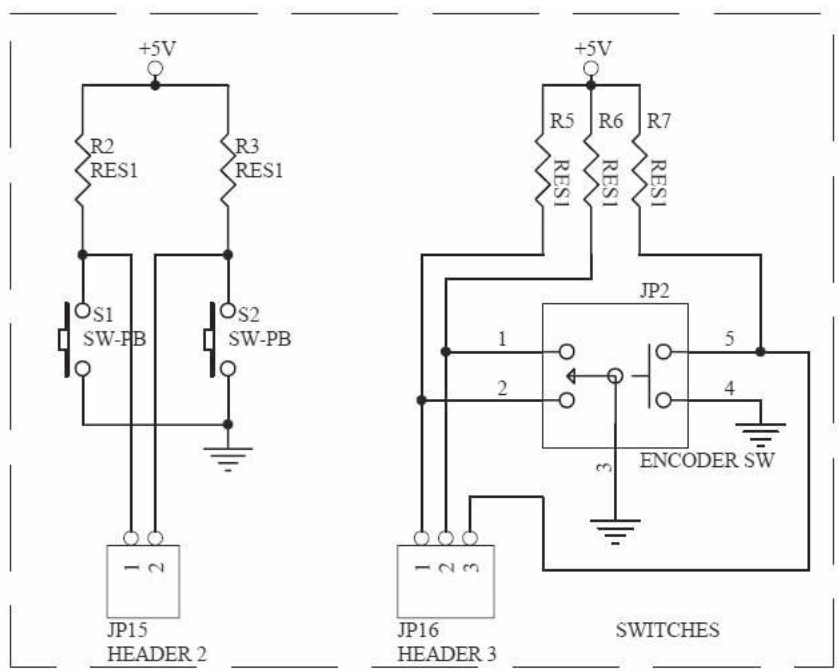


Figure 22. Encoder and Push button switch schematic diagram. S1 and S2 are normally open push button switches. The encoder switch is represented by JP2 block.

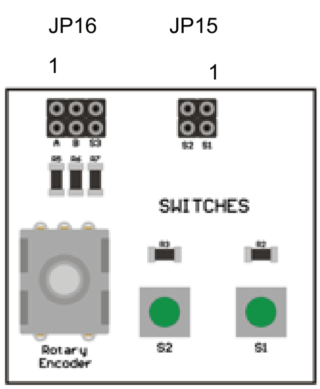


Figure 23. Encoder and Push button switch component arrangement.

Table 15. JP15 Push Button Switch

Pin No.	ID	Description
1	S1	S1 output, normally high
2	S2	S2 output, normally high

Table 16. JP16 Encoder Switch

Pin No.	ID	Description
1	A	Phase A output
2	B	Phase B output
3	S3	Push Switch output, normally high

DIGITAL TO ANALOG CONVERTER

Currently, most microcontroller have Analog to Digital ADC converter on chip, but few have Digital to Analog DAC converters. Through this trainer, you will learn how to interface and use an external serial DAC with your MCU.

A DAC is an interface device that will allow you to output an analog voltage based on a digital value you put in. The MAX5382 is an 8-bit DAC, it converts an 8-bit wide data into a corresponding analog voltage in 256 steps. This device is housed in a tiny SOT-23 chip, and requires only two pins to interface with a host MCU. This two input serial interface is better known as the I2C interface. Recall the RTC also uses an I2C serial interface. This interface, together with SPI, are arguably the most popular chip level serial interfaces used by tens of thousands of devices.

For a detailed description of the MAX5382, please refer to its corresponding device datasheet attached in the appendix section of this document.

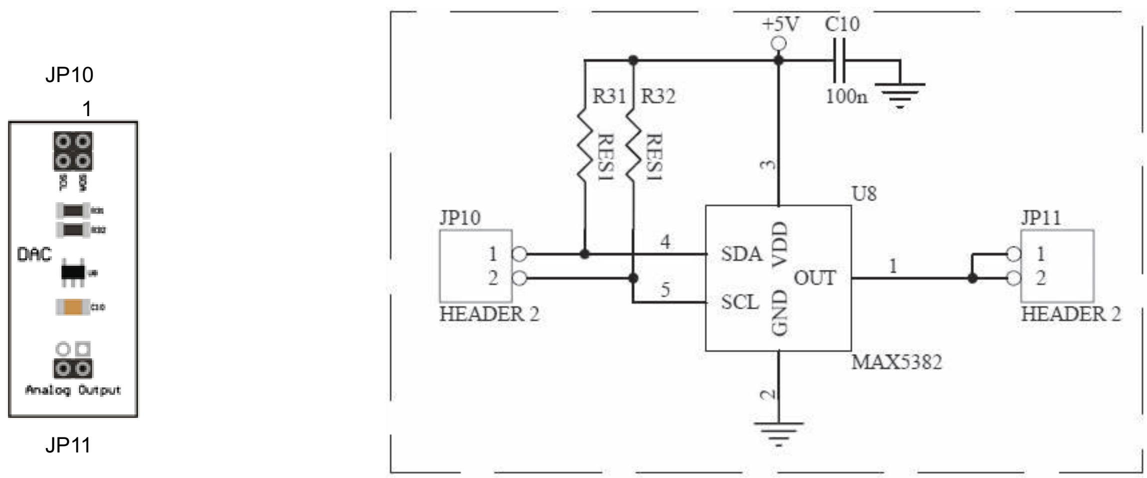


Figure 24. The DAC section component layout and schematic diagram. The DAC ICMAX5382 is an 8-bit DAC chip house in a tiny sot-23-5 package.

Table 17. JP10 DAC Control Inputs

Pin No.	ID	Description
1	SDA	Serial Data Input
2	SCL	Clock Input

Table 18. JP11 Analog Out

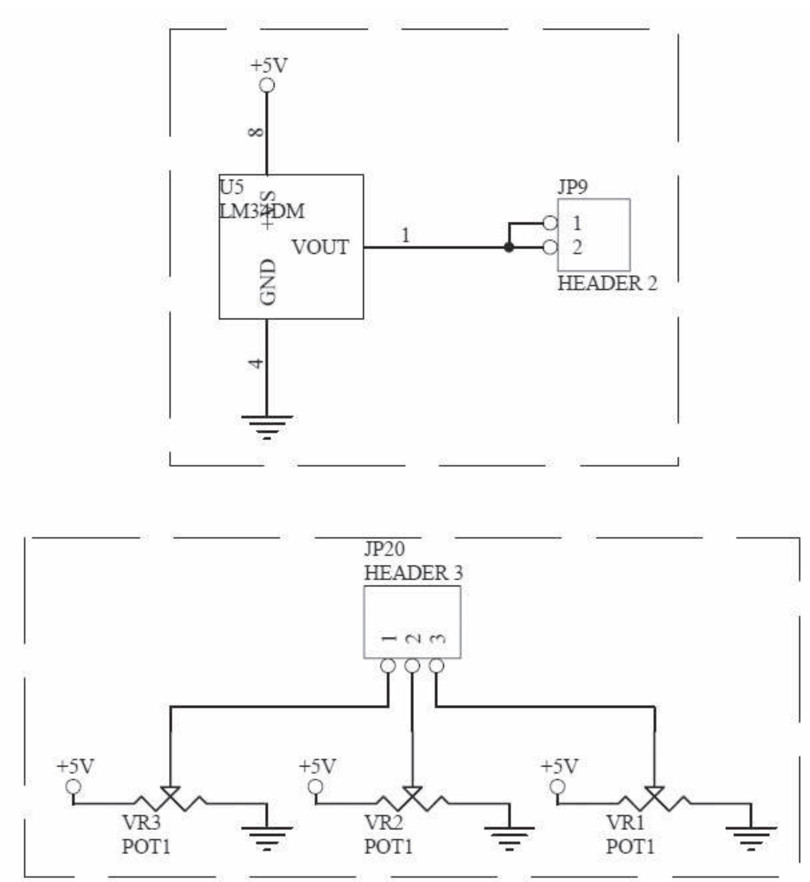
Pin No.	ID	Description
1	AOUT	Analog Voltage Output
2	AOUT	Analog Voltage Output

ANALOG VOLTAGE SOURCE

Most MCUs of today have the ADC circuit already on chip, and these functions can be accessed by configuring via the user program the proper pin. The Analog voltage source subsystem provides both simulated and real analog voltage source that you use to practice with your on-chip Analog to Digital Converter ADC.

Three trimmer resistors allow you to set and read analog voltages from 0-5V DC each for your practice session. And then you can try your skill later with one real world device, an LM34 temperature sensor U5. This temperature sensor will give you the ambient temperature reading in analog Fahrenheit scale. The added challenge is converting the acquired readings to the more familiar (in the Philippines) centigrade readout, not a terribly hard thing to do.

For a detailed description of the LM34, please refer to its corresponding device datasheet attached in the appendix section of this document.



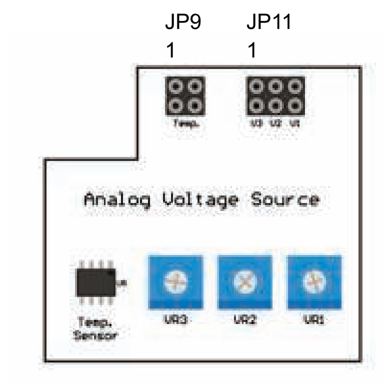


Table 19. JP9 Temperature Sensor Output

Pin No.	ID	Description
1	TEMP	Analog Out
2	TEMP	Analog Out

Table 20. JP11 Trimmer Analog Out

Pin No.	ID	Description
1	V3	VR3 Analog Out
2	V2	VR2 Analog Out
3	V1	VR1 Analog Out

Figure 25. The ADC experiment circuit consists of three simulated analog sources VR1-VR3 and a LM34 Temperature sensor.

STEPPER MOTOR AND RELAY DRIVER

The Stepper Motor Drive subsection is can be used to study and control stepper motor operations via user code. It consists of 4 sections of uln2003 driver U1, this driver can be used to work with a small 6 or five wires unipolar stepper motor. It can drive with up to 500mA sink current, but owing to its location in the PCB layout, it will work best if the current is held below 250mA. A fixed 5V source is available for the stepper motor common through pin 5 of JP12. Generally, stepper motors rated up to 12V will still work for practice purpose.

Important: This driver has open collector output. Also it is not designed for heavy duty and heavy load applications.

Each leg of the motor driver circuits are actually independent from each other, and can be useful for other purpose other than as stepper motor drive. For example, you can use a section to drive a solenoid, another to drive a power LED, small DC motor, etc.

A Relay circuit is also included for experiments that may require an isolated switch. The relay contact can handle switching current of up to 1A, and is safe to use for up to 30VDC, 30W load.

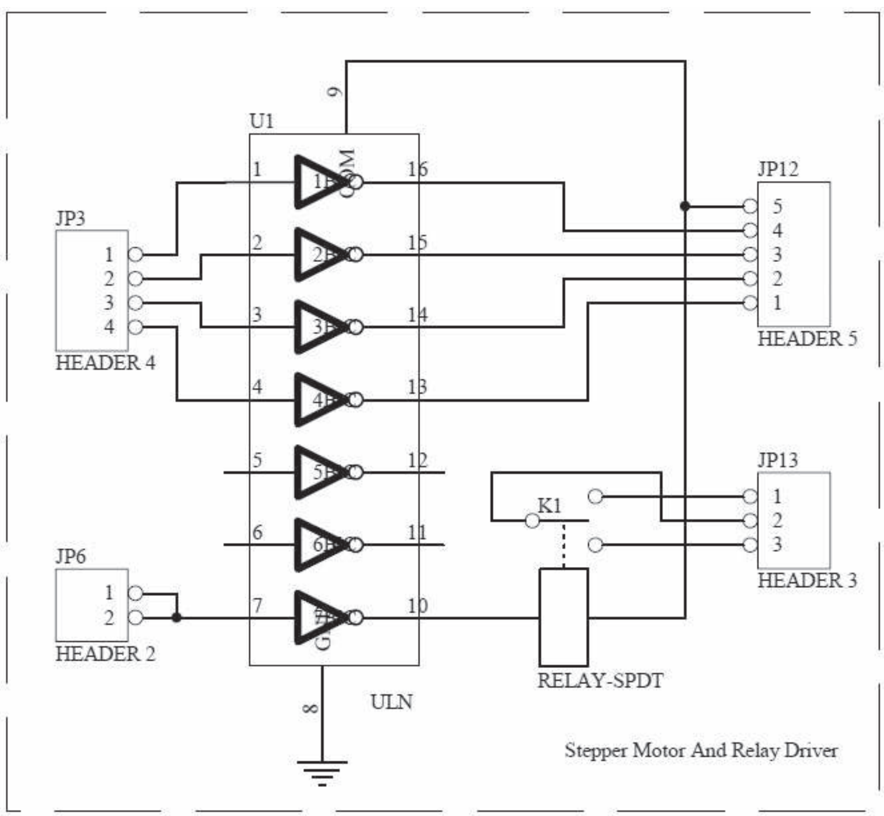


Figure 26. Stepper Motor Driver and Relay schematic diagram.

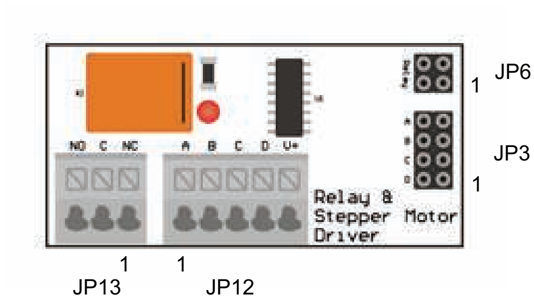


Figure 27. Stepper Motor Driver and Relay circuit component layout.

Table 21. JP6 Relay Control Input

Pin No.	ID	Description
1	Relay	Relay ON Active High
2	Relay	Relay ON Active High

Table 22. JP13 Relay Contacts

Pin No.	ID	Description
1	NC	Normally Close Contact
2	C	Common Connection
3	NO	Normally Open Contact

Table 23. JP3 Driver Control Input

Pin No.	ID	Description
1	D	Phase D In Active High
2	C	Phase C In Active High
3	B	Phase B In Active High
4	A	Phase A In Active High

Table 24. JP12 Driver Open Collector Outputs (O.C.)

Pin No.	ID	Description
1	A	Phase A Output O.C.
2	B	Phase B Output O.C.
3	C	Phase C Output O.C.
4	D	Phase D Output O.C.
5	V+	Common +5V