Weeder Technologies
PO Box 2426, Ft Walton Beach, FL 32549
Pro-Kit
850-863-5723

RS-232 Digital I/O

By Terry J. Weeder

The potentiality to write/create computer programs to incorporate in your own custom applications is an asset which every computer owner has, though most seem to overlook or just plain take this for granted. Sure, all the bells and whistles and fancy graphics accompanying the heard of software packages which come with your computer keep you entertained at first, you still have not truly discovered the capability of this desktop tool until you've jumped in and started building your own programs. If you've never tried it before and don't know where to start, check out that easy-to-learn programming language called QBASIC which can be found in your DOS directory. Of course, in order to fully appreciate the power at hand, being able to link these programs with external switches and relays is a must, and can now be accomplished using your serial port and an I/O module costing as little as $32.

The RS-232 Digital I/O module as described in this text connects directly to the serial port of a PC and enables your home-brewed computer program to interact with its outside environment. 12 I/O pins are available and can be individually configured for input or output. The module is addressable using a DIP switch setting on the board, and is specifically designed to share the same RS-232 port with other units. Up to 16 units may be stacked in parallel on the same com port allowing a total of 192 I/O points.

Using an I/O pin for an input allows your program to sense button presses, change of state of switches, alarm trips, etc. Setting a pin for an output will let your program turn on/off electrical devices with the aid of a relay, triac or switching transistor. In addition, there are a few special commands which use eight of the twelve I/O pins as a group. A 4x4 matrix keypad can be decoded allowing 16 buttons to be monitored using only 8 I/O pins. These keypads come in handy to use for instance, on a security access panel activating an electro-mechanical lock, or a control panel for your computer based alarm system. You can also read or write an 8-bit word directly to these eight I/O pins.

The set of commands which can be sent from your PC's com port to configure and control the I/O module is shown in listing 1. These include:

HIGH/LOW - sets a specified I/O pin to a desired state.

TOGGLE - changes a specified I/O pin to the opposite state.

BUTTON - configures a specified I/O pin to sense a button press using automatic debounce and repeat if the button is held down.

SWITCH - configures a specified I/O pin to sense a switch transition using automatic debounce.

MATRIX - configures the first eight I/O pins to be used to decode a 4x4 matrix keypad using automatic debounce and repeat.

READ - reads the state of a specified I/O pin, or reads the first eight I/O pins as an 8-bit port.

WRITE - writes data to the first eight I/O pins as an 8-bit port. Data can be in the format of hex, binary, or an ascii character.

Circuit Theory

Refer to the schematic diagram shown in figure 2. The heart of the circuit is IC1 (part no. PIC16C55-XT/P), an EPROM-based 8-bit CMOS microcontroller manufactured by Microchip. This microcontroller has two 8-bit I/O ports, one 4-bit I/O port, and internal EPROM memory which holds the program used for encoding/decoding the data sent to and from the computer, reading and writing to the I/O pins, and reading the DIP switch setting (S1) which sets the board address. Crystal (XTAL1) sets the clock frequency. A detailed description of IC1's firmware is explained later.

The voltage levels used for serial communications on an RS-232 port are +3V to +25V for a logic 0, and -3V to -25V for a logic 1. Most RS-232 devices use +12V and -12V respectively. Bit 0 of port-A is used to send data to the serial port. A logic 1 is generated by placing bit 0 at a high level which turns off Q1, thus allowing the -12V from the TD (Transmit Data) pin to be applied to the RD (Receive Data) pin thru R2. Bit 0 is sent low to produce a logic 0 which turns on Q1, pulling the RD pin to +5V. Because the TD pin of an RS-232 port is normally at a marking level (-12V), it is possible to "steal" from it the negative voltage needed for communications at RS-232 levels and a separate supply is

not required. Bit 2 of port-A is tied to the DTR (Data Terminal Ready) pin thru R5 and determines when the unit is plugged into an active RS-232 port. Bit 1 of port-A is tied to the RD pin thru R4 and is used to verify an idle RS-232 state prior to sending any serial data. This will avoid a collision with the data sent from any other projects which are sharing the same RS-232 port.

Power for the circuit is supplied from a standard wall transformer with an output in the range of 7 to 15VDC. IC2, a 78L05 voltage regulator drops the input voltage to 5 volts which is required by the circuit. C1 and C2 stabilize the operation of the regulator and provide filtering. LED1 is used to indicate communications activity with the computer. Current limiting resistors R11 thru R22 protect IC1s I/O pins from excessive current flow during accidental shorts to 5V or ground. The DIP switch (S1) together with the pull-down resistors R7 thru R10 are used to set the address of the RS-232 Digital I/O module.

Command Set

TITLE	COMMAND	DESCRIPTION
HIGH	H pin	Make pin an output and set it high (pin = A-L).
LOW	L pin	Make pin an output and set it low (pin = A-L).
TOGGLE	T pin	Make pin an output and toggle state (pin = A-L).
BUTTON	B pin	Make pin an input and respond to button press. Returns "pinL" when pressed (pin = A-L).
SWITCH	S pin	Make pin an input and respond to switch transition. Returns "pinL" or "pinH" when toggled (pin = A-L).
MATRIX	M	Make pins A-D an output, pins E-H an input, and decodes 4x4 matrix keypad presses. Returns "Mrow,col" (row = A-D, col = E-H).
READ	R pin	If pin = A-L, read state. Returns "pinH" or "pinL" (pin = A-L). If pin = P read pins A-H as 8-bit port.
WRITE	W f d	Make pins A-H an output and writes data(d) to pins as 8-bit port; format(f) can be "h" for hex, "b" for binary or "$" for ascii character.

Note, all commands must be preceeded by the WTDIO header charcter D, and address character A-P. All responses from the WTDIO will also contain this preface.

Listing 1

The PIC Firmware

A PIC16C55 programmed with the "RS-232 Digital I/O" firmware is available from the supplier mentioned in the parts list.

At power-up, the program stored in IC1's eprom enters a loop which monitors the com port and any of the I/O pins which have been configured for input. If a start bit is detected at the com port, the program branches to a routine which begins by reading the first character sent from the PC. This is the header character and determines if the data to follow is meant for the RS-232 Digital I/O kit, or used by some other kit sharing the same com port. Immediately following this header character is the address character (A-P). IC1 reads the DIP switch setting to determine if there is an address match and in turn either fetches or discards the remainder of the data stream.

IC1 reacts to any valid command received from the com port by updating the state and/or configuration of any specified I/O pin(s) and returning any required data. When data is to be returned, IC1 first listens to the com port to determine if it is available, then sends the header and address characters followed by the response as depicted in listing 1. The data stream is terminated by a carriage return and the program returns to the main loop.

If an action is sensed at any of the I/O pins which have been set up for a switch, button, or matrix keypad, the action is verified after a 65ms delay - used for debounce purposes, then IC1 sends the appropriate indication to the com port in the same manner mentioned above. If configured for button or matrix, this response will be repeated for as long as the button is held down. The matrix decoding process is accomplished simply by individually setting each one of the row pins (A-D) low, then scanning the column pins (E-H) to see if any has been shorted to the row pin through a pressed button on the keypad.

Construction

The complete circuit fits nicely on a PC board measuring just under 3" x 2.5". Refer to the parts placement diagram shown in figure 11, identify the component side of the PC board which is marked and begin by soldering in the 28-pin IC socket. The resistors can be mounted next. Notice the extra holes/pads near resistors R11 thru R22. These are for optional 10K pull-up resistors mounted vertically and should be left blank for now. Solder in the capacitors paying particular attention to the orientation of the polarized caps C1 and C2. When mounting the LED, identify the anode which is the long lead; this should

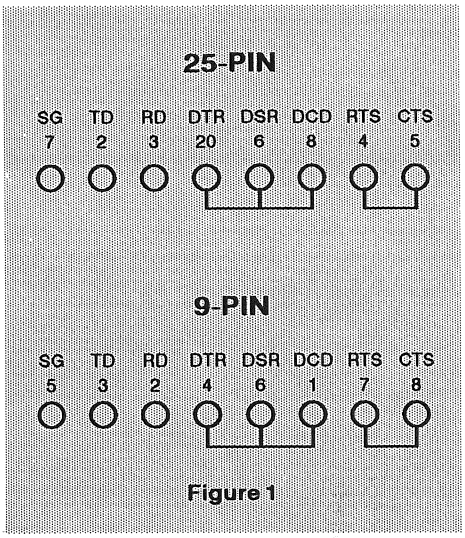


Figure 1

correspond with the pad labeled "A" on the parts placement diagram.

Care should be taken when soldering in the transistor (Q1) and the regulator (IC2) to avoid a solder bridge between the closely spaced pads. The crystal (XTAL1) should be mounted leaving a small gap between the bottom of its case and the PC board to avoid the chance of its metal case shorting the two pads together. Finish by installing the DIP switch (S1).

Obtain a piece of 4-conductor telephone cord to be used as your RS-232 cable, and a connector (both available from Radio Shack). Figure 1 shows the hook-up diagram for both a 9-pin and a 25-pin RS-232 connector. Match the "SG", "DTR", "TD" and "RD" connections on the PC board with their corresponding pins on the connector you are using as shown in the diagram. Solder a jumper wire on the RS-232 connector between pins "RTS" and "CTS", and between pins "DTR", "DSR" and "DCD" as shown.

Use a DC wall transformer (also available from Radio Shack) with an output voltage in the range of 7 to 15 VDC. Cut off the connector at the end of the wires, use a voltmeter to determine positive and negative, and solder those wires to the "POS" and "NEG" connections on the PC board. After all components and wires have been soldered, closely examine both sides of the PC board for solder bridges and/or cold solder joints and re-solder if necessary. Carefully plug IC1 into its socket using the orientation as shown in the parts placement diagram.

Operation

The RS-232 Digital I/O kit can share the same serial port as other kits of its kind simply by wiring each kit in parallel to the same RS-232 connector. You must however, remove R1 and R2 on any subsequent kit (ie: of all the kits parallelled on the same port, only one kit should have R1 and R2 installed). Doing this will allow you to piggyback up to 16 I/O modules and use them on the same port which currently supports any of the other RS-232 kits offered by **Pro-Kit.**

Solder a solid-conductor wire onto each I/O pad (A thru L) and run the opposite ends to a solderless breadboard. Use the set-up shown in figure 3 along with the simple QBASIC program shown in listing 2 to test out your circuit. Plug the wall transformer into an electrical outlet and plug the RS-232 connector into your PC. Note, always apply power to your kit before plugging into an active RS-232 port or the oscillator may fail to start.

Start the program then hit "C" on your keyboard. A prompt will appear asking you to type in a command. Type "HA" - using caps - then hit enter. LED1 will flash as the unit receives the data from the serial port, then the I/O pin labeled "A" will change to an output and be set high which will turn on its LED on the breadboard. Hit "C" again and type in "LA". This will set I/O pin "A" back to low thus turning off the LED. Try this command using each I/O pin label and watch as the corresponding LEDs turn on and off. If LED1 on the I/O board does not flash when sending data out the com port, there is either a wiring error in your serial cable or your program is not accessing the correct port. Closely examine your cable including the jumpers on the RS-232 connector, and verify that the serial port you are plugging into is the same one you are opening in your QBASIC program. With correct hook-up to the serial port, and the QBASIC program not running, the DTR, TD and RD pads on the board will all be a negative voltage. After starting the program, the DTR pad will change to a positive voltage.

The WRITE command can be used to send an 8-bit word to pins A thru H, and can be of three different formats selected by the character which precedes the data. For example, decimal value 74 can be sent as a hex character pair using the syntax "Wh4A", as a binary string using "Wb01001010", or as an ascii character using "W$J".

```
CLS
OPEN "COM1:1200,N,8,1" FOR RANDOM AS #1
ON COM(1) GOSUB RECEIVE
COM(1) ON

DO
    KEY$ = INKEY$
    IF UCASE$(KEY$) = "C" THEN
        INPUT "Enter Command ", OUT$
        GOSUB TRANSMIT
    END IF
LOOP UNTIL KEY$ = CHR$(27)
CLOSE #1
END

RECEIVE:
    COM(1) OFF
    IF INPUT$(2, #1) = "DA" THEN
        LINE INPUT #1, IN$
        PRINT IN$
    ELSE LINE INPUT #1, DISCARD$
    END IF
    COM(1) ON
    RETURN

TRANSMIT:
    COM(1) OFF
    OUT$ = "DA" + OUT$
    PRINT #1, OUT$
    LINE INPUT #1, DISCARD$
    COM(1) ON
    RETURN
```

Listing 2

Creating Your Own Program

The RS-232 Digital I/O module communicates at 1200 baud, no parity, 8 data bits and 1 stop bit. Your program should contain the line OPEN "COM1:1200,N,8,1" FOR RANDOM AS #1, or similar. Also the ON COM GOSUB statement should be used as shown in the sample program to handle branching to a subroutine when data is received from the module. All data sent by the I/O module to the serial port is preceded by the header character (D) and the board address character (A-P), then ending with a carriage return. All commands sent by the PC to the I/O module must also be preceded by these header and address characters.

Two important notes here: Because of the structure of the RS-232 interface used by the I/O module, all characters that are sent to the module are also echoed back to the PC. Therefore your program must use the COM(1) OFF statement prior to using the PRINT #1, and then a LINE INPUT #1 statement to dump all echoed characters in the buffer before issuing a COM(1) ON statement. Also, always use the COM(1) OFF statement at the beginning of your subroutine branched to by the ON COM GOSUB statement, then a COM(1) ON statement at the end after all characters have been received. Failure to turn off event trapping as mentioned above will cause communications errors between the PC and the I/O module.

Figure 4 demonstrates the ease at which this module can be wired to buttons and switches. Pull-up resistors should be installed on the I/O board for each; these can be mounted vertically in the extra holes provided near the I/O pads. Normally-Closed alarm zones can be wired to each of the 12 inputs. Configuring these for switches will return a response to your PC whenever any loop is broken. The automatic debounce incorporated in the firmware will work well preventing false trips due to electrical noise or short intermittences. To further protect against false alarms, your program could be set up to ignore trips which return to their original state within a minimal amount of time.

Figure 5 shows the module set up to decode a 4x4 matrix keypad. Be sure to install pull-up resistors on I/O pins E thru H before issuing the matrix command. When a button is pressed, the row and column characters are returned to the PC. Example, pressing button 6 will return "MBG", pressing button 8 will return "MCF", etc. A DIP switch is used in figure 6 to remotely set or change a configuration or variable within a program which issues the READ command periodically. Again, use pull-up resistors on the input pins.

Figures 7 thru 9 give examples of using an I/O pin with the HIGH, LOW or TOGGLE commands to turn on/off electrical devices. These circuits can be copied on each I/O pin. A standard TV remote transmitter can be used to send commands to your PC using the IR Remote Control Receiver kit offered by **Pro-Kit** and the set-up shown in figure 10. Configure these I/O pins for switches and have your program respond to the transitions.

Keep in mind that each time a command is sent to the I/O module acting upon a particular I/O pin, any previous configuration for that pin will be canceled. For instance, setting a pin high or low which was being used to read a switch or button will disable it from responding to that action. Also, issuing a command which acts upon any of the I/O pins used for decoding a matrix keypad will disable the matrix function.

Parts List

Resistors (All are 1/4-watt, 10% units)

R1, R7, R8, R9, R10 - 10,000 ohm

R2 - 1,000 ohm

R3, R4, R5 - 47,000 ohm

R6, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22 - 620 ohm

Capacitors

C1 - 47 uF, 35-WVDC, electrolytic

C2 - 10 uF, 35-WVDC, electrolytic

C3, C4 - 15 pF, ceramic disc

C5 - 0.1 uF, Mylar

Semiconductors

IC1 - PIC16C55-XT/P (pre-programmed) 8-bit microcontroller (Microchip)

IC2 - 78L05 low power 5-volt regulator

LED1 - light emitting diode, red

Q1 - 2N4403, general-purpose PNP silicon transistor

Other Components

S1 - DIP switch, 4-pole XTAL1 - 4 MHz crystal

Miscellaneous: PC board, 28-pin IC socket, DC wall transformer, RS-232 connector & cable, 10K pull-up resistors, solder, etc.

The following items are available from Weeder Technologies, PO Box 2426, Ft Walton Beach, FL 32549. 850-863-5723.

- Double-sided etched and drilled PC board (WTDIO-B), $9.50
- All board mounting components including pre-programmed PIC16C55 (WTDIO-C), $22.50
- A pre-programmed PIC16C55 only (PIC-DIO), $18.00

All orders must include an additional $4.00 for shipping and handling. Florida residents add sales tax. Visa and MasterCard welcome.

RS-232 DIGITAL I/O

WTDIO

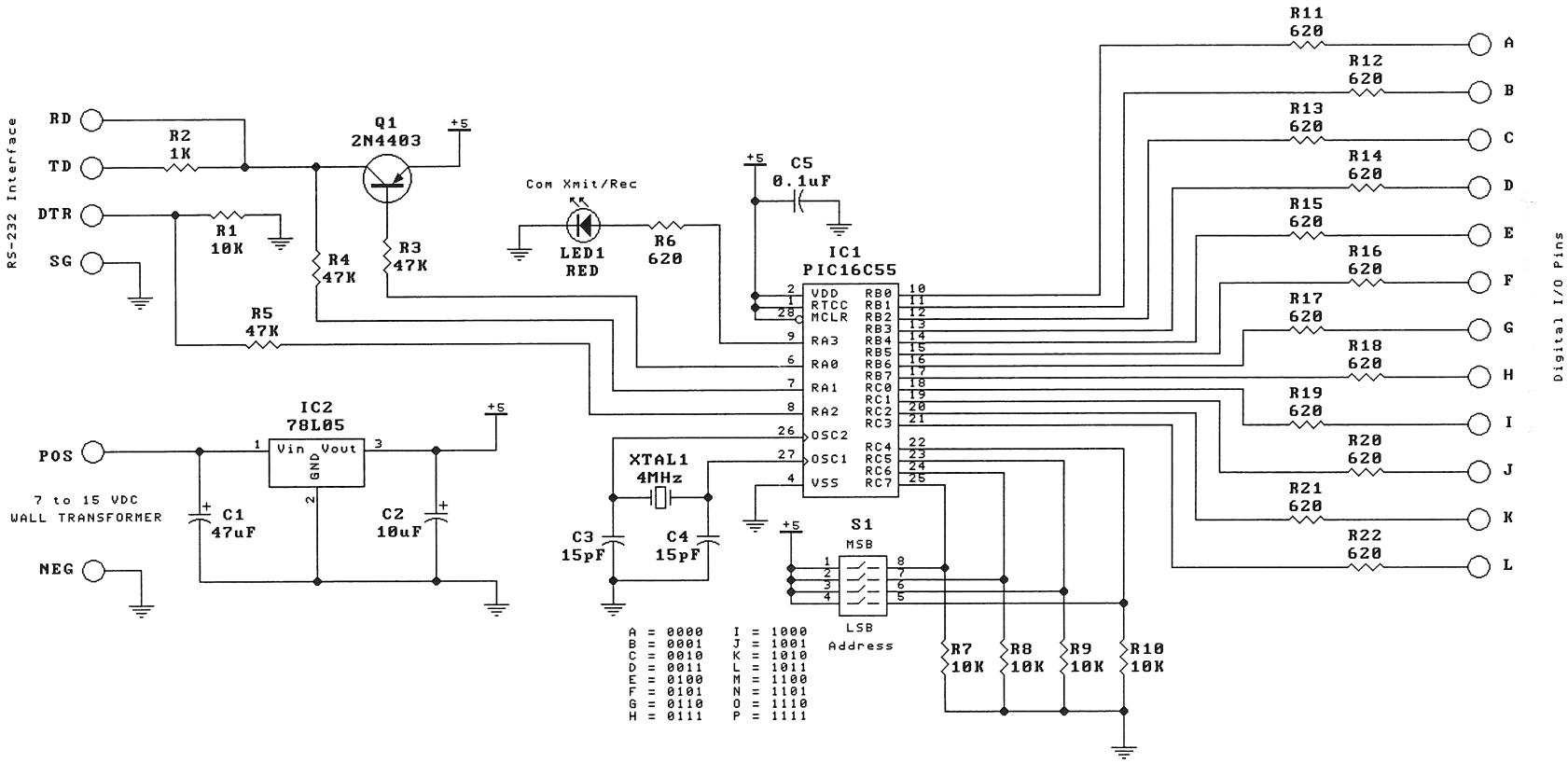


Figure 2

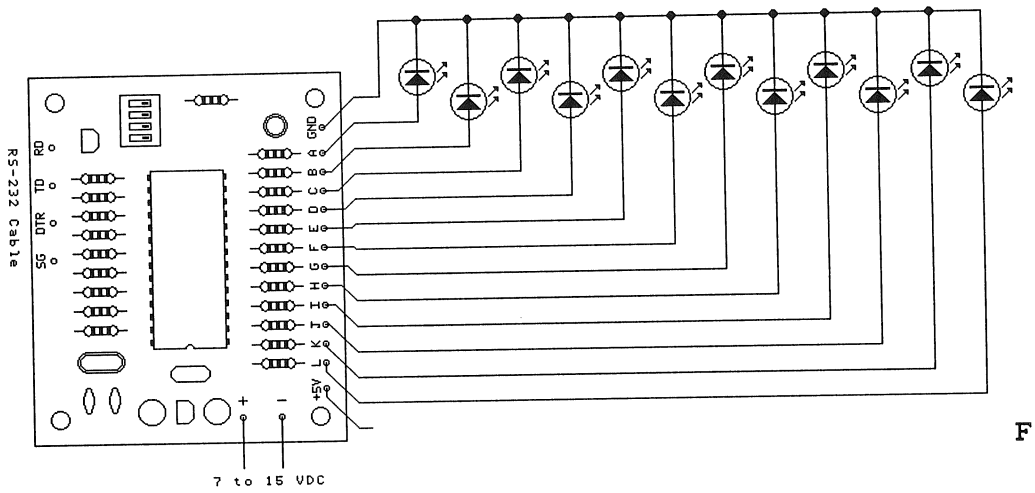


Figure 3

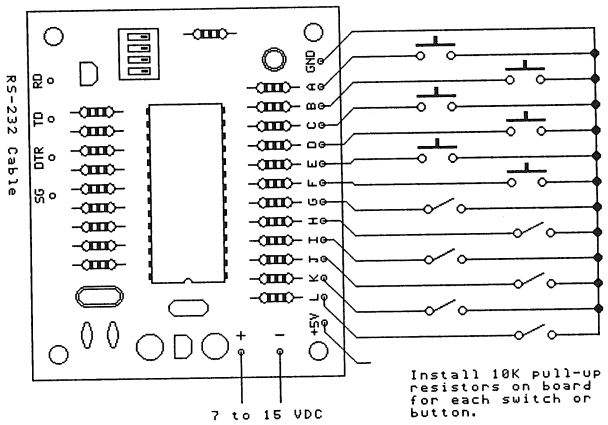


Figure 4

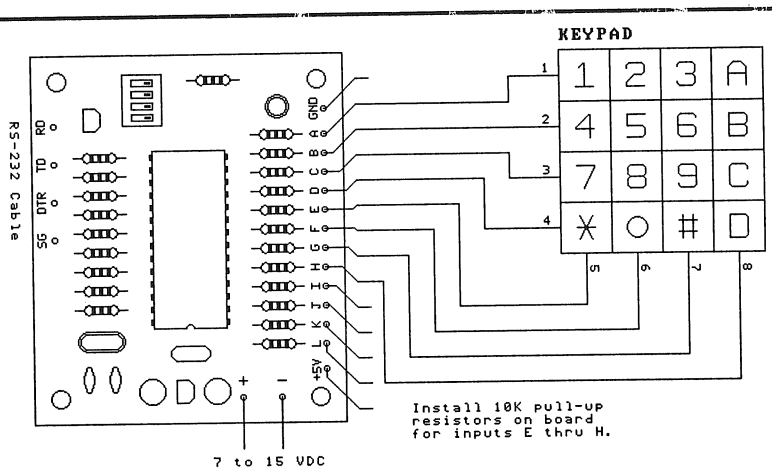


Figure 5

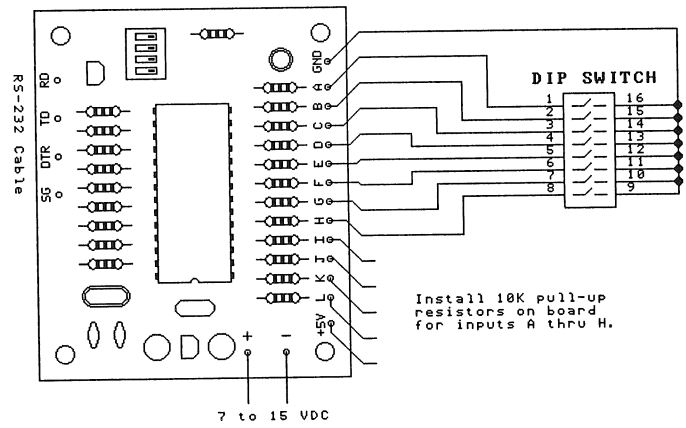


Figure 6

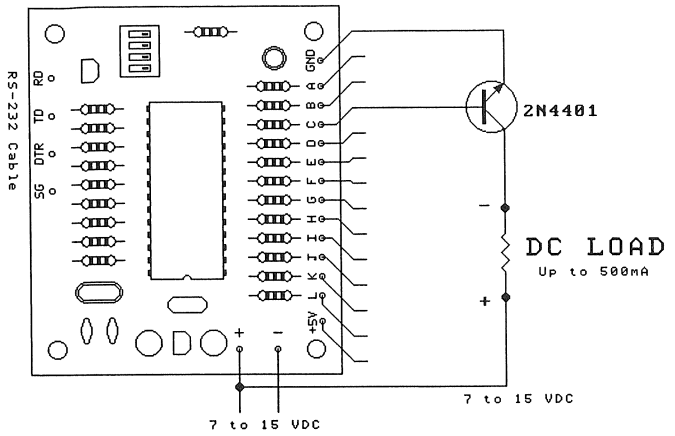


Figure 7

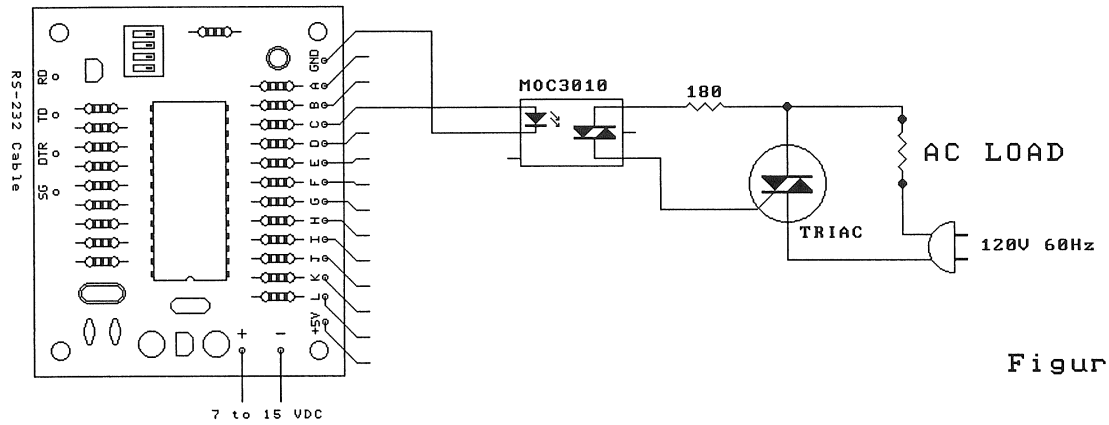


Figure 8

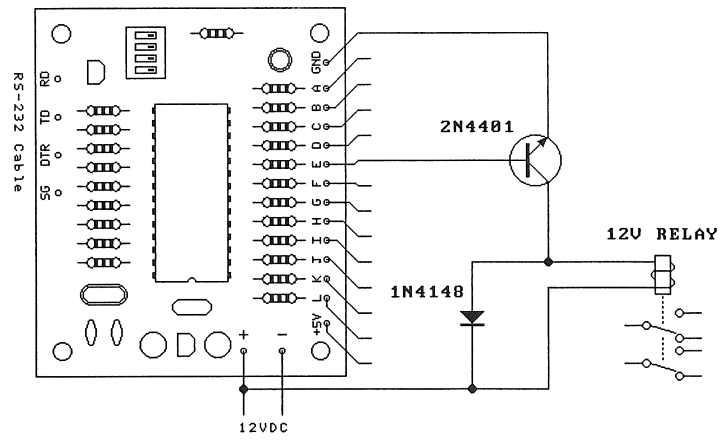


Figure 9

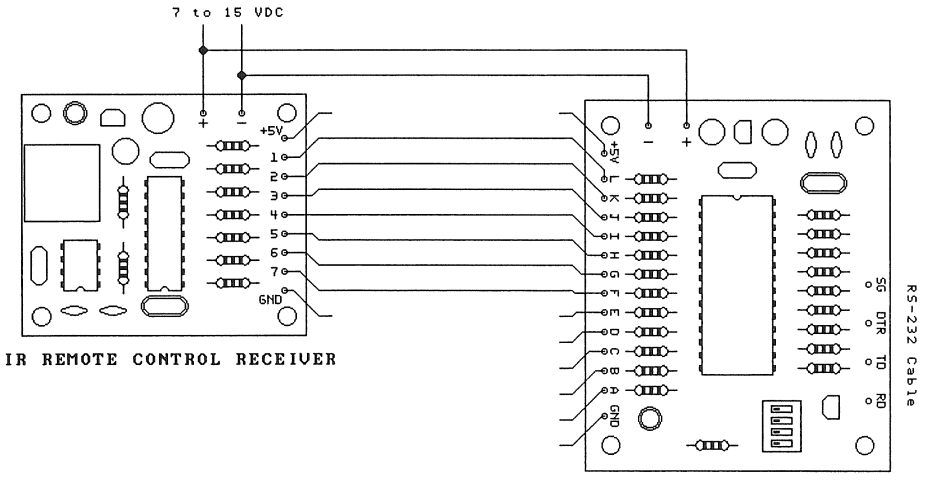
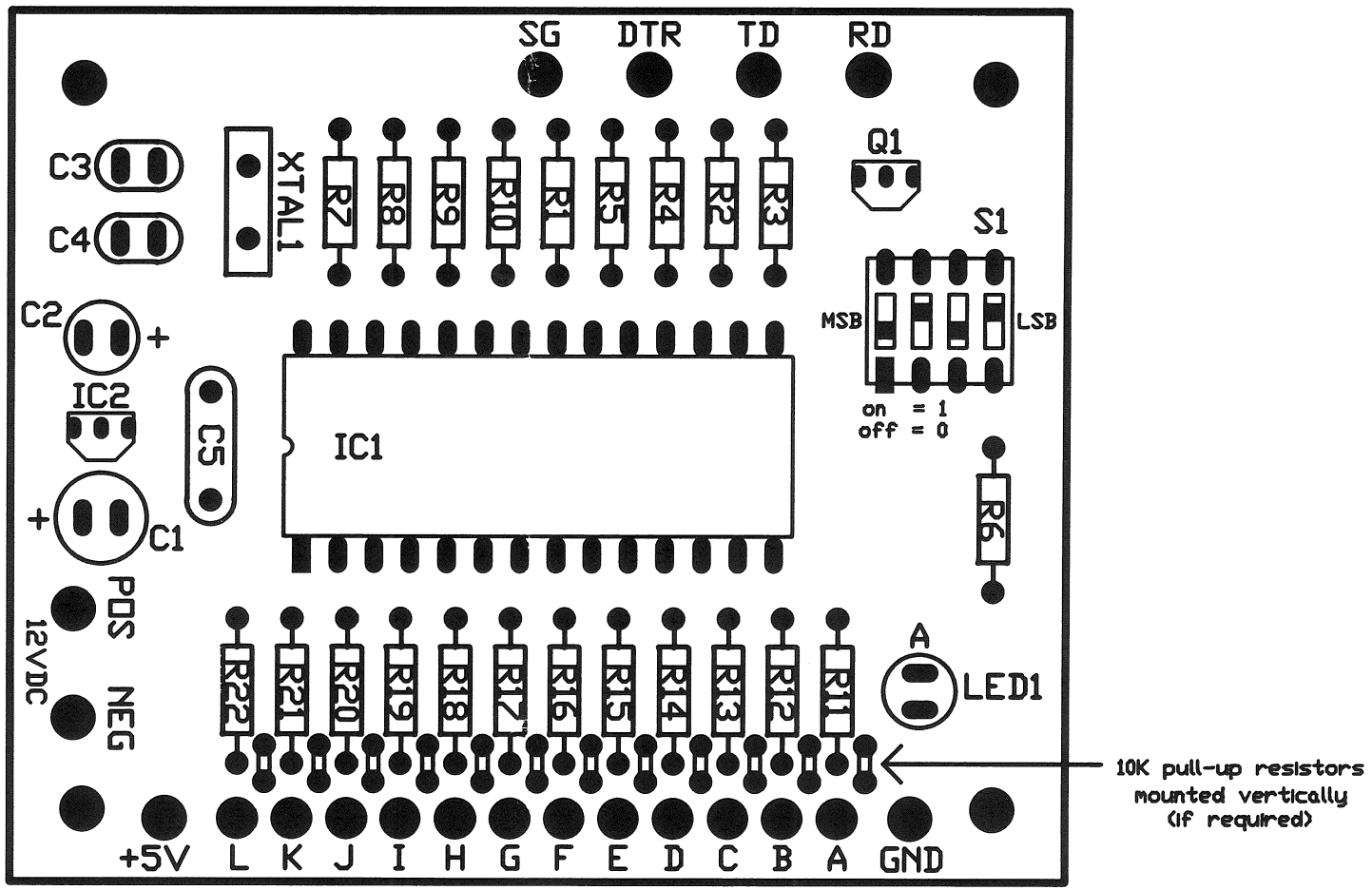


Figure 10



WTDIO

Figure 11