Using the Experimenter

A Reference Manual and Applications Guide

Version 1.0 Rev. B
Edition

by
Ronald M. Jackson
Chief Engineer
Fascinating Electronics, Inc.

COPYRIGHT NOTICE:

IMPORTANT NOTICE:

Because of possible variations in the quality and condition of materials and workmanship, and variations in individual skill and prudence, Fascinating Electronics Inc. disclaims any responsibility for the safe and proper functioning of any projects or applications built using the Experimenter or any component or product sold by us. You as the builder of the application are responsible for the safe design and operation of what you have made. You are responsible for determining that the Experimenter is suitable and reliable enough for the task, and for meeting any applicable regulatory requirements.

The Experimenter is not to be used in any application where its failure could injure people or damage property. We especially do not want to hear of Experimenters running Grandma's pacemaker, cousin Edna's iron lung, or a nuclear power plant.

Fascinating Electronics Inc. disclaims any responsibility for incidental or consequential damage resulting from the assembly or operation of any component or product sold by us.

For more information about the Experimenter and our other products, please write, call, or email:

Fascinating Electronics Inc.
925 SW 83rd AVE
Portland, OR 97225-6307
Call Toll Free (US and Canada) 1.800.683.5487
Direct Dial 503.296.8579
Email Ron@FascinatingElectronics.com

Table of Contents

0

Introduction

Let me tell you a story: How the Experimenter came about…

There was a time, not very long ago, before computers were in every home, when people interested in science, technology, and electronics experimented with transistors and ICs. High school and college students built interesting semester projects. Grad students built equipment for their professors. Hobbyists built projects and kits—things like digital clocks and shortwave radios. People had fun, learned a lot, and their friends and associates were impressed with the projects they built.

Then the microprocessor came along, and things got more complicated. Yes, the personal computer offers incredible processing power so you can build far more amazing gadgets than ever before. But as computers and software became vastly more sophisticated it became a major project just connecting a sensor or motor to the computer. You would spend most of your time building the interface into the computer and creating driver software, and little time creating the amazing gadget itself. It was like inventing the wheel, over and over.

If you went out to buy a device that provided this interface to your computer, all you could find were industrial control boards. They are ridiculously expensive, and often perform just a single function. And an engineering degree is required to figure out how to use these things.

We thought that while there are many problems in the world, this is one problem we can most certainly cure. We set out to build a device that is easy to use, with built-in intelligence to handle all the details so that no special software drivers are required. A device that connects through a serial port, so that it will work with any PC, Mac, notebook—just about any type of computer. A device that provides a wide range of measurement and control capabilities: sensing voltages, counting pulses and measuring signal timing, providing many high current drivers for DC and stepping motor control, lots of digital I/O for controlling logic devices, and even a high current relay. And we priced it to be affordable for schools, hobbyists, and even industrial control engineers (it makes life easier even for those who have engineering degrees). We call it *the Experimenter*.

If you want to ***do something real*** with your computer, we have developed some fascinating applications for the Experimenter. Here you will find a chapter on an Ultrasonic RADAR (Application A), and a magnificent computerized meteorological station (Application B). If you are creating your own computerized gadget, the Experimenter gives you a great head start. The autonomous robot (Application C) was created by a team of high school students. Perseverance, research, and a significant head start from the Experimenter enabled them to build a very sophisticated robot.

Today around the world, in university labs and in the field, Experimenters are taking data. In factories Experimenters control industrial processes. In private homes and at professional sites Experimenters are monitoring the weather. At least one film studio is using Experimenters to control special effects.

This book will explain the Experimenter to you—how it works, how to connect it to your computer, the powerful commands it performs. You will learn how to make time and voltage measurements (necessary for many types of sensors). You'll learn how to control motors, both ordinary DC motors and stepping motors. How to output and read logic signals and how to flip the relay. The Experimenter is like a "Swiss Army Knife" with many tools in one compact package.

Ron

—Ronald M. Jackson
Chief Engineer

P.S. If you develop a nifty gadget that other Experimenter users may like to build, please let us know. We can all benefit by sharing information. You can send any comments you may have about the Experimenter directly to me at Fascinating Electronics Inc. And especially let me know if there is any way we can make the Experimenter more useful to you! Please email me at:

Ron@FascinatingElectronics.com

1

The Hardware

The Experimenter is a microcontroller with on-board software and special measurement and control hardware. In this chapter, we'll take a look at the hardware, and learn some of how the Experimenter performs its many functions.

A Tour of the Photograph with Functional Overlay

Figure 1-1 is a photograph of the Experimenter, with an overlay of the functions supported in each area. Let's take a quick tour of the photograph to give you a feel for the Experimenter. Starting in the upper-left corner of the board: the **logic supply** provides +5 volts to run all of the logic circuitry on the Experimenter. Below that is the "brains" of the Experimenter, the **microcontroller** with its support circuitry.

In the upper-right corner is the **RS-232C** circuitry, providing communication between your computer and the Experimenter. The next chapter will discuss communications in detail.

Below that is the **relay**, which you can use to switch high current loads. To the right is the **digital I/O** chip. This provides 24 bits of digital input or output, for binary sensing or low current control.

Just left of the digital I/O chip are two **driver** chips with heatsinks. These provide eight channels of high-current drive for DC and stepping motors, solenoids, additional relays, and other power devices.

Most of the connections to Experimenter functions are located in the **measurement and control I/O** area, below the microcontroller, drivers, and digital I/O. Exceptions are IO1, the backup battery input; IO2, IO3, IO4, the relay connections; and IO5 the ±10v boosted supplies.

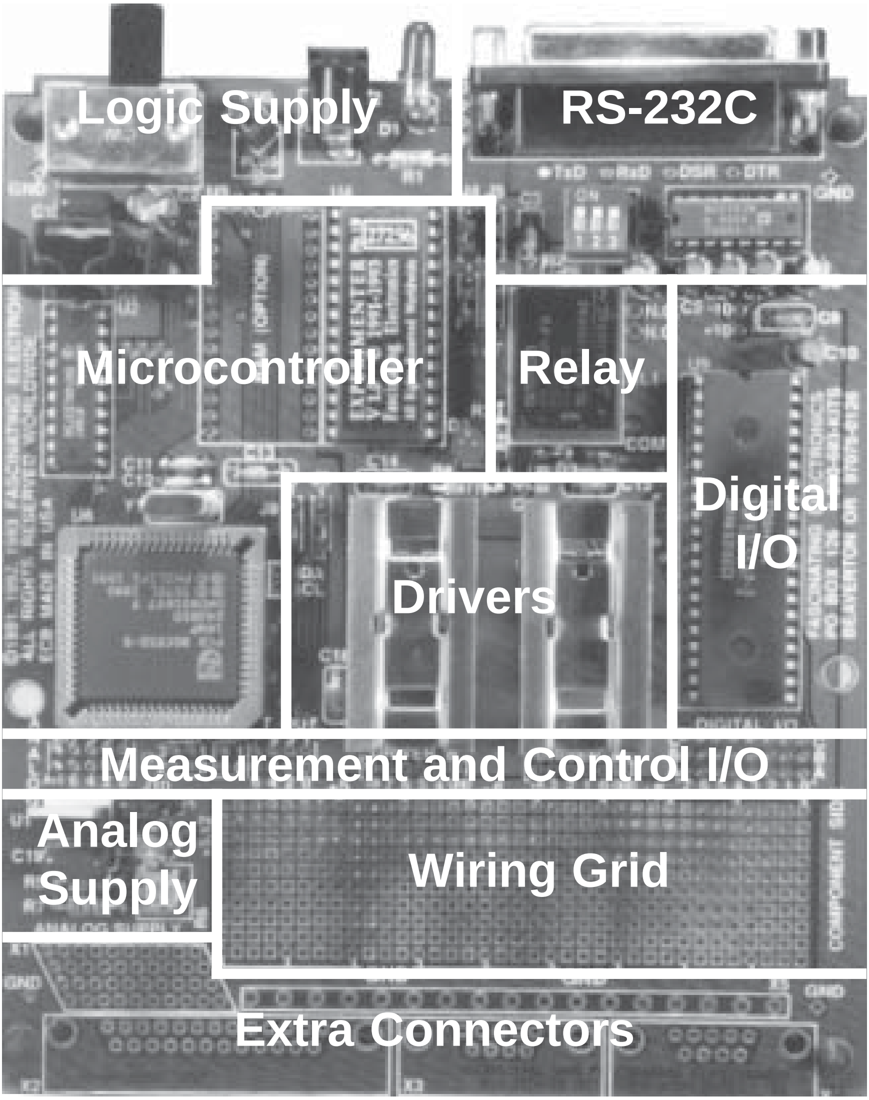


Figure 1-1: Experimenter with Functional Overlay
The Experimenter provides an intelligent interface between any computer with a serial port and a wide range of measurement and control functions. It is built around a microcomputer, with a variety of additional support and interfacing hardware. These include power supplies, RS-232C level shifters, high-current devices, and many channels of digital I/O. A large wiring grid and pads for extra connects enhance the ease-of-use in many projects.

At the far left of the **measurement and control I/O** area are eight analog inputs at IO6 ANALOG. These inputs can measure voltages with 5 mV resolution. These are useful for many types of sensors, like temperature sensors or photocells.

To the right of are four timer/counter inputs at IO8 T-IN. As the name implies, these can be used to measure time intervals or count pulses. These measurements have many applications. For example, counting raindrops in a weather station, counting rotations of a shaft, or measuring distance using echo ranging in an Ultrasonic RADAR.

Next are two pulse width modulator (PWM) outputs at IO9 PWM and eight timed outputs at IO10 T-OUT. The PWMs are used to control the duty cycle during which the drivers are enabled. This technique is often used to control the speed of DC motors. The timed outputs are just the unbuffered versions of the signals going to the drivers.

The driver outputs IO11 DRIVER-B AND IO12 DRIVER-A and IO13 DIGITAL-I/O are the remaining signals in the measurement and control I/O area.

The **Analog supply** provides an adjustable reference power source for the analog-to-digital converter on the Experimenter, and for use by any sensitive analog circuits you may add to the board. This is adjusted to 5.120 volts so that the 10-bit A/D step size is 5 mV. Jumper J12 lets you select between using this supply or the logic supply to power the A/D converter.

To the right is a large **wiring grid**. This is a great place to add your own projects. To make wiring even easier the logic supply and ground are provided to every fourth connection in the top and bottom rows in the grid. This corresponds to the standard power pin locations of upper right corner for V_{CC} and lower left for V_{SS} for most TTL logic ICs. Any chips that do not have this pinout should be installed to the right or left of these connections.

Below the wiring grid are pads for mounting extra connectors: one high density, one DB-25, two DB-9, and one 5 mm pitch terminal strip. You can add any of these connectors to suit your particular applications. These connector locations are marked **X1** through **X5**.

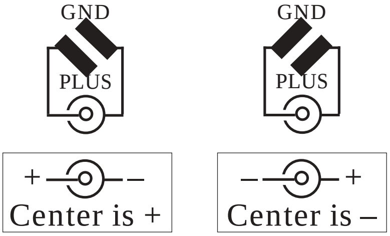


Figure 1-2: Polarity Selection
Jumper J1 is used to select the polarity of the power connector. Set the jumpers as shown for the power supply you are using with the Experimenter. Power supplies from Fascinating Electronics Inc. are center+, as shown at left.

A Tour of the Schematic

Though it may look intimidating at first, the Experimenter hardware is actually fairly straightforward. Let's go through the schematic, which is located at the centerfold (pages 38-39) of this book. DC power of 5.5 to 15 volts enters the Experimenter through connector P1 (schematic grid F-5). Since the center contact on the power connector is positive on some power supplies, and negative on others, jumper J1 provides a means to select between the types (**Figure 1-2**). IO1 lets you add a backup battery (make sure the power supply voltage is greater than the battery voltage). S1 provides a convenient power switch.

Voltage regulator U1 produces the logic supply. The logic supply provides +5 volts to all of the logic circuitry on the Experimenter, with power left over for additional circuitry. U1 is a low-dropout voltage regulator, permitting the Experimenter to run off of a 6 volt battery. U1 is rated at 1 amp.

Low-dropout adjustable voltage regulator, U10, provides a separate analog supply. In addition to providing an adjustable reference for more accurate voltage measurements, the analog supply also provides "quiet" power for sensitive analog circuits — power that is isolated from the relatively noisy logic supply. If you build analog circuits, using this supply will help avoid many noise problems and ease your debugging. This voltage regulator is also rated at 1 amp.

The "brains" of the Experimenter is the microcontroller, U6 (schematic grid A-2). It is an eight bit microprocessor with additional measurement and control hardware built in. The microcontroller includes a serial port, baud rate generator, a clock oscillator, an analog multiplexer, analog-to-digital convertor, counters, timers, and pulse-width-modulators.

Your computer connects to the Experimenter through P2 (grid E-1), a standard 25-pin female connector. Since RS-232C uses larger voltage swings than standard logic, U5 is required to interface the RS-232C signals to the microcontroller. U5 also contains a voltage doubler and inverter that provides the unregulated plus and minus 10 volts required for RS-232C. These voltages are available for your use on IO5, but the current available is limited. The serial connection will be described in detail in the next chapter.

A 7.3728 MHz crystal, Y1, clocks the microcontroller. This peculiar frequency can be evenly divided into standard baud rates for RS-232C. Jumpers J6 allows you to select rates from 300 baud to 38.4 kilobaud.

The analog multiplexer in the microcontroller provides eight analog voltage measurement inputs (IO6, ANALOG 0:7). The analog-to-digital converter can measure these inputs with 5 millivolt resolution (10 bits) over the range from 0 to 5.115 volts. This is useful for many sensors, like thermistors for temperature measurement, or photocells for light sensing.

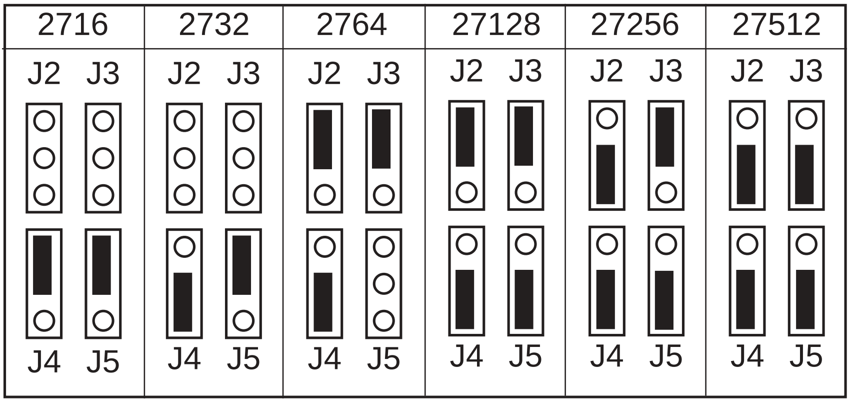


Figure 1-3: EPROM Type Selection Jumpers
The Experimenter can accommodate six different types of EPROMS, ranging in size from 2 kilobytes to 64 kilobytes. Configure jumpers J2-J5 as shown for the particular type of EPROM you are using.

Capacitor C13 (grid E-7) provides a reset signal to the microcontroller and to the parallel interface chip, U9. This causes the microcontroller and parallel interface chip to initialize their internal registers when the logic supply is switched on.

The lower eight bits of the address bus multiplex on the same lines as the data bus. The octal latch, U2 (grid A-4), holds the address value while the data is on the bus.

Software for the microcontroller is stored in an EPROM, U4 (grid A-6). The Experimenter can accommodate many different types of EPROM chips. Jumpers J2 through J5 select between the different types. These jumpers should be set for the type of EPROM that came with your Experimenter, usually a 27256 (**Figure 1-3**).

Though not currently used, the Experimenter has space for an additional memory. U3 (grid A-7) can accommodate an 128 kB static RAM, which may be accessed as four 32 kB banks.

The four counter/timer inputs (IO8, T-IN 0:3) connect directly to the microcontroller. As with the analog multiplexer and analog-to-digital converter, all counting and timing circuitry is within the microcontroller chip.

Eight precision timed outputs come directly from the microcontroller. Timed outputs are available unbuffered (IO10, T-OUT 0:7). They are also buffered to drive higher voltage and current applications by U7 and U8. Each of these driver outputs (IO12, DRIVER-A 0:3 and IO11 DRIVER-B 4:7) can source and sink up to 1 amp. That’s a lot of current! An external power source must be provided for these drivers, since the logic supply isn't up to that kind of load. Each driver has its own positive power input (+A and +B). These inputs must be connected to a power source of from +4.5 to +36 volts DC. Switched unregulated Experimenter power is available on the next pads to the left (SWP) and may be used for this purpose. Additional grounding points are provided on the DRIVER connections.

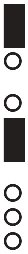
Pins 4, 5, 12 and 13 of the high current driver IC's (U7 and U8) solder to large metal heatsinks to pull the heat of these high-power devices. Because of the weight of these heatsinks the drivers should be secured to their sockets with a nylon cable tie. Otherwise they might fall out of their sockets!

The drivers have built-in protection against getting too hot — they will shut down if their internal temperature gets too high. They may, however, oscillate as this temperature is reached. **The drivers are not protected against short circuits!**

Using jumpers J8 through J10, you may either select to always enable each pair of driver outputs (no jumper), or to enable them under the control of the pulse width modulators (see **Figure 1-4**). The PWMs provide a continuous stream of pulses of controllable frequency and duty-cycle. When enabled by the varying duty-cycle from the PWMs, the driver outputs will provide varying amounts of power to a load. For example, by controlling the PWM duty-cycle the speed of a DC motor powered by the drivers can be varied.

The microcontroller controls the relay through a one transistor buffer, Q1. An LED, D3, lights when the relay’s coil is energized. Diode D4 prevents a damaging inductive voltage spike from occurring across the relay coil when the transistor turns off.

Jumper	Channels
J7	B7, B6
J8	B5, B4
J9	A3, A2
J10	A1, A0



Enable channels from PWM0.

Enable channels from PWM1.

Channels always enabled.

Figure 1-4: Pulse Width Modulator Selection Jumpers
Output drivers U7 and U8 may be enabled under pulse width modulator control, or may be always enabled. Jumpers J7 and J8 control IO11 DRIVER-B (U7). Jumpers J9 and J10 control IO12 DRIVER-A (U8).

2

The Serial Connection

The Experimenter may be connected to any type of computer equipped with a serial port (RS-232C). On PCs serial ports are called COM ports. On Macs the external modem port and printer port are both serial ports. **Note that USB is not compatible with RS-232C.** If your computer does not come with an RS-232C port you will need to add one to use the Experimenter. This chapter will help you establish the serial link between your computer and the Experimenter. Connecting your Experimenter to a serial port on your computer is usually easy, but occasionally there are complications.

The Physical Serial Connection

The Experimenter uses the same type of cable that would be used to connect an external modem to your computer. The Experimenter is not a modem, of course, but that type of cable is fairly common, so that is what we use.

The connector on the computer's end of the cable is not so well standardized. Some PCs use 9-pin D-sub connectors, some use 25 pins. Macs use an 8-pin Mini Din connector. Standard PC and Mac cables are available from Fascinating Electronics.

On a PC, you may have a choice of serial ports. The example programs in this book were written in BASIC, and many versions of BASIC can only access COM1 and COM2. If the your computer language is able to use other COM ports, use the port you prefer.

On a Macintosh, use either the modem port (if your Mac does not have an internal modem) or the printer port. Both of these are standard serial ports.

Some Nitty-Gritty Details

For those of you who like getting the nitty-gritty details, here is the technical information. The Experimenter is configured as **data communications equipment**, DCE. It expects to be sent **transmitted data** (TxD) on pin 2, and sends out **received data** (RxD) on pin 3. It holds off sending data if **data terminal ready** (DTR, pin 20) is false (or not connected). The Experimenter asserts **data set ready** (DSR, pin 6), **clear to send** (CTS, pin 5), and **carrier detect** (CD, pin 8) immediately when it is switched on.

Some computers and serial cables are manufactured with reduced signal counts on the serial ports — as few as 3 wires (TxD, RxD, signal ground). If you are using a three wire serial link you must add a wire from CTS (pin 5) to DTR (pin 20) on the serial connector (J2). Otherwise the Experimenter believes that your computer is sending an “I’m not ready” signal because it isn’t driving DTR true. Sometimes serial cables and 9-pin to 25-pin adapters only connect the TxD, RxD and signal ground pins. This can produce an “I’m not ready” problem on both ends: both the Experimenter and the computer believing that the other device is signaling that it is not ready to receive data. I have found tracing out the adapter wiring with an ohm meter helps in identifying what is going wrong.

Direct Communication

The easiest way to begin using your Experimenter is with a communications program like HyperTerminal. To use HyperTerminal create a New Connection for the Experimenter. In the Connection Description menu name the connection Experimenter and select an attractive icon. In the Connect To menu at the Connect Using field select Direct To Com1 (or whichever com port you are using). In the Com1 Properties menu set the baud rate to 9600 (make sure the Experimenter is set to the same rate, see below), 8 data bits, no parity, 1 stop bit and hardware flow control. You should now be able to type directly to your Experimenter and see its responses in the Hyperterminal window. You can save this connection and use it later (at the click of the new Experimenter.ht icon).

If you don’t have a communications program, **Listing 2-1: ECHO-EXP.BAS** is a simple communications program written in BASIC, suitable for use with the Experimenter. This particular version was written for a PC in Microsoft’s QuickBASIC. You may need to modify this program to work on other types of computers, and translate it for different dialects of BASIC. This program will prompt you for which serial port to use. Other than that, ECHO-EXP.BAS has all communications parameters preset for the Experimenter.

The Experimenter supports a variety of baud rates. The rate is set by J6. **Figure 2-1: Baud Rates** gives the correspondence between baud rates and J6 jumper settings. The Experimenter checks the baud rate only once, at power-up.

If you change the setting while the Experimenter is running it will not notice the change and will continue to use the old baud rate. So, if you change the setting you must power-off the Experimenter. Be sure the Experimenter and your communications program are using the same baud rate. The baud rate in ECHO-EXP.BAS is preset for 9600 baud.

If everything is working, when you switch-on the Experimenter you will see:

```
Experimenter Copyright 1991-93 Fascinating Electronics.  Ver 1.0

Exp>
```

You may now start sending commands to the Experimenter. We will discuss those commands in the following chapters.

At power-on, the Experimenter runs some diagnostic checks to make sure it is working properly. If it finds any problems, it will report that to you. If you did not get the above message, verify that the Experimenter's power light (D1) is on. If it is, try running the Experimenter in test mode. To select test mode, install all three J6 jumpers. Cycle the power on the Experimenter, and if the Experimenter is working properly the relay will repeatedly click on and off. If it does then the problem is most likely not with the Experimenter, but with the serial connection to your computer. Verify that you are using the correct serial port, and the correct type of cable.

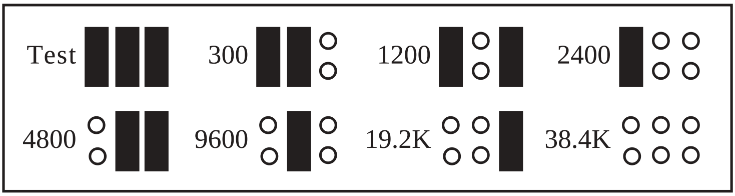


Figure 2-1: Baud Rates
The Experimenter sets its baud rate from the jumper settings on J6. The value is read only at power-on. The TEST setting (On-On-On) causes the relay to click repeatedly. This is a simple diagnostic test to assure that the microcontroller is working properly.

Listing 2-1: ECHO-EXP.BAS (next page)
This QuickBASIC program provides 9600 baud communication between a PC and an Experimenter. Both source code and executable versions of this program are available, along with the other programs in this book, on Application Disk #1 (SWP-AP1), available from Fascinating Electronics, Inc.

```
' This program provides 9600 baud communications with an Experimenter.

backspace$ = CHR$(8)            'Backspace is ASCII character 8.
linefeed$ = CHR$(10)            'Linefeed is ASCII character 10.

' Clear the screen and select which COM port to use.
CLS
DO
   INPUT "Which COM port will the Experimenter use (1 or 2)"; comPort%
   PRINT
LOOP UNTIL comPort% = 1 OR comPort% = 2
LOCATE , , 0                    'Turn off the cursor.
PRINT USING "Connecting to the Experimenter through COM#."; comPort%
PRINT

' Open the specified COM port as file #1.
ON ERROR GOTO ioError1          'On an error, give a helpful message.
IF comPort% = 1 then
   OPEN "com1: 9600,n,8,1" FOR RANDOM AS #1
ELSE
   OPEN "com2: 9600,n,8,1" FOR RANDOM AS #1
END IF
ON ERROR GOTO ioError2          'Ignore further errors.
CLS
PLAY "MB L32 C D E F G A B > C D E F G A B"
PRINT "Connected to the EXPERIMENTER!"

' Print message and activate cursor.
PRINT "------------------------------"
PRINT "Type Q to Quit and return to BASIC."
PRINT "Type ? for help with Experimenter commandss."
PRINT
LOCATE , , 1                    'Activate the cursor.

' Check for keyboard input.  Quit if the user types Q, otherwise send
the
' keystroke to the Experimenter. Backspaces erase the previous
character
' on the line. Linefeeds are ignored.
key$ = INKEY$           'Get any keystroke from the keyboard.
DO UNTIL UCASE$(key$) = "Q"
   IF LEN(key$) > 0 THEN PRINT #1, key$;
   IF NOT EOF(1) THEN
      expKey$ = INPUT$(1, #1)
      IF expKey$ = backspace$ then
         LOCATE , POS(0) - 1
         PRINT " ";
         LOCATE , POS(0) -1
      ELSEIF NOT expKey$ = linefeed$ THEN PRINT expKey$;
      END IF
   END IF
   key$ = INKEY$
LOOP
CLS
END

' ******************** ERROR HANDLING ROUTINES *********************
' Error occurred while opening the COM port. Give a helpful message.
ioError1:
   PLAY "MB L16 A B C"
   PRINT "Be sure the Experimenter is powered-on, 9600 baud."
   PRINT "Verify the Experimenter is connected to the correct COM
port."
   PRINT
   ON ERROR GOTO ioError2
RESUME
' Ignore all further errors.
ioError2:
RESUME
```

3

The Firmware

The Experimenter has its own microprocessor running a built-in program. Built-in software is called firmware. This firmware lets the Experimenter perform many complex measurement and control activities without intervention by your computer. Just send the Experimenter a command and off it goes.

When you request a measurement, or when you ask for the status of an ongoing command, the Experimenter responds in plain ASCII text (no binary numbers to convert). This makes it easy to communicate directly with the Experimenter using a communication program (as suggested in the preceding chapter), or to control the Experimenter through programs running on your computer.

Help Features

The Experimenter has built-in help features. When you type a question mark (**`?`**) at the **`Exp>`** prompt, the Experimenter responds with a table of the commands and their parameters (see **Figure 3-1**). You can get help on individual commands by giving the command name, followed by a question mark. All Experimenter commands may be abbreviated by giving only the first letter of the command.

Command Syntax

Version 1.0 firmware includes the commands **`A`** through **`I`**, with the exception of **`B`**, which is reserved for a later firmware release. All other commands will result in an error message. The Experimenter automatically converts lowercase letters into uppercase, so you may enter either.

```
ANALOG  channel
B-command  reserved for later future use.
COUNTER/TIMER  channel  function  wait
DIGITAL-I/O  port  out.1/mode  dur.1  out.2  dur.2  out.3
ENABLE-PWM  channel  duty-cycle  rate
FLIP-RELAY  relay.number  state
GENERAL-INFORMATION/CONTROL  selection  off/on
H-BRIDGE  group  direction  duration  speed  type
INDIVIDUAL-OUTPUT channel state duration comp.duration cycles

**  To abbreviate, use only the first letter of the name  **
```

Figure 3-1: Help Text
The Experimenter prints this help text in response to a **?***-command.*

Each command may be followed by several parameters. Parameters are integers, between **0** and a maximum of **65535**. Values greater than **65535** will result in an error message. Parameters may be separated from each other by just about any other non-numeric character, but spaces are recommended. Do not put commas in your numbers. That is, enter one-thousand as **1000** not as **1,000**.

It is not always necessary to give every last parameter to a command every time. For example, once given the ***speed*** and ***type*** parameters for an **H**-command, the Experimenter remembers them for all subsequent **H**-commands for that group. However, it is not possible to skip parameters. If you later want to issue an **H**-command leaving all parameters the same except for ***type***, you must again provide all the parameters to the left of ***type***.

Special Characters

Processing your command begins after you send a ***carriage return*** (enter).

The Experimenter stores the characters you send it in an input buffer. If you reach the end of the input buffer before typing a ***carriage return***, the Experimenter will not accept any further characters (and will not echo them). It will also signal that this has happened by sending you a ***bell character*** (beep).

It is possible for the Experimenter to generate responses faster than they can be communicated. The Experimenter maintains an output buffer for this text. If the output buffer overflows, text will be lost. To inform you that this has happened the Experimenter puts a ***pound sign*** (#) in the output buffer.

If you make a typing mistake, the ***backspace key*** will delete the previous character from the Experimenter's input buffer. It may not erase the character from your computer screen, depending your communications program.

If you want to start over, the ***escape key*** causes the Experimenter to dump the whole contents of the input buffer. This is also handy when initializing computer control mode (described later), to get rid of any garbage characters that may have come from noise on the communications line during power-on.

Xon/Xoff handshaking is supported by the Experimenter. In this type of handshaking, your computer sends the Experimenter a ***control-S character*** when it is running out of room in its input buffer. This causes the Experimenter to stop sending characters. After your computer has digested the input, it sends out a ***control-Q character*** causing the Experimenter to resume sending.

Software Controlled Communication

Software development for personal computers gets easier all the time. New versions of old programming languages like BASIC have been given structured programming constructs (no more line numbers!), high resolution graphics support, and include nice editors and debugging capabilities. It is fun and immensely satisfying to write a program that links physical world events to graphics on your computer screen. It is also easy to do. Programming languages have good support for the serial ports, so no special hardware driver is needed for the Experimenter.

The Experimenter powers-on in **manual communication mode**. This is for the convenience of folks using communication programs as described in the previous chapter. But the communication mode is easy to change. The **`GENERAL-INFORMATION/CONTROL`** command lets you switch the Experimenter to **computer control mode**. Just send the command **`G 0 1`**. This does several things:

1. ***It disables the echo of characters.*** Most communication programs expect the system with which they connect to echo characters as they are typed. Naturally, in computer control mode, echoing characters is unnecessary and would just clutter up the Experimenter's replies with the echoes of commands. So, echo gets disabled.

2. ***It disables appending a linefeed after each carriage return.*** Again, most communication programs expect a ***linefeed character*** after each carriage return. But most programming languages looking for input from a serial port are only expecting a carriage return. So, linefeed gets disabled.

3. ***It disables appending a `0` after each timer measurement.*** The timers in the Experimenter measure in 10 microsecond steps. For example, a value of 56 corresponds to **`560`** microseconds. However, people think more clearly when we stick with common units like milliseconds and microseconds, rather than unusual units like 10's of microseconds and quarters of fortnights. So, at power-

on the Experimenter defaults to adding a **0** to all timer measurements (except **0** itself, which is still **0** not **00**). But your computer program can deal with any old measurement choice—it's all just numbers to your program. So, appending a **0** gets disabled.

There is one other thing your program should do before you get into the thick of commands and responses. It should clear any pending characters out of its input buffer. Otherwise, when your program goes to read its first response from the Experimenter, the first thing it will get back is the Experimenter's power-on message. Which is very confusing if, for example, your program had just sent a command and was expecting a number in response.

Computer Program Template

To give you some idea how to write a control program, **Listing 3-2** provides a template for you to use when writing programs that control the Experimenter. Add your own statements in place of the comment block where it says "YOUR CODE GOES HERE." Notice how the program handles possible communications errors, initializes the communications port, and clears the input buffer. If you get tired of the melody played every time the program successfully connects with the Experimenter, just delete the PLAY statement after the CLS command.

As an example using this template, put the statements in **Listing 3-1** in place of the "YOUR CODE GOES HERE" block in **Listing 3-2**. This program will scan the analog inputs as fast as your computer will go, presenting the measured voltages on your computer. These values will be random voltages, unless you connect something to the analog inputs. In the next chapter we will look at all of the commands, and in the section on the ANALOG command we will discuss providing voltages to the analog inputs.

```
' Print a header on the screen.
CLS
PRINT "Analog Voltage Scan, type Q to quit."
PRINT
PRINT "Channel   Millivolts"
PRINT "-------   ----------"
DO
   LOCATE 5, 1, 0               'Position for voltage readings.
   FOR channel% = 0 TO 7
      PRINT #1, "A"; channel%   'Request an analog measurement.
      INPUT #1, value%          'Get measurement result.
      PRINT USING "   #         ####   "; channel%; value%
   NEXT
LOOP UNTIL UCASE$(INKEY$) = "Q" 'Quit when 'Q' key is pressed.
```

Listing 3-1: ANALOG-8.BAS, Scans Analog Inputs
This program will scan the voltage measurement inputs (ANALOG 0:7) and present the measurements on the display. Insert this code in place of the "YOUR CODE GOES HERE" block in the TEMPLATE.BAS program.

```
' This is a template for you to use when writing computer programs
' for an Experimenter connected to COM1 or COM2 at 9600 baud.
escape% = 27                    '<escape> is ASCII character 27.

' Clear the screen and select which COM port to use.
CLS
DO
   INPUT "Which COM port will the Experimenter use (1 or 2)"; comPort%
   PRINT
LOOP UNTIL comPort% = 1 OR comPort% = 2
LOCATE , , 0                    'Turn off the cursor.
PRINT USING "Connecting to the Experimenter through COM#."; comPort%
PRINT

' Open the specified COM port as file #1.
ON ERROR GOTO ioError1          'On an error, give a helpful message.
IF comPort% = 1 then
   OPEN "com1: 9600,n,8,1" FOR RANDOM AS #1
ELSE
   OPEN "com2: 9600,n,8,1" FOR RANDOM AS #1
END IF
ON ERROR GOTO ioError2          'Ignore further errors.
CLS
PLAY "MB L32 C D E F G A B > C D E F G A B"
PRINT "Connected to the EXPERIMENTER!"

' Put the Experimenter in computer controlled mode, clear input buffer.
SLEEP 1                         'Allow Experimenter time to power up.
PRINT #1, CHR$(escape%)         'Clear Experimenter of any characters.
SLEEP 1                         ‘Allow Experimenter time to execute
clear
PRINT #1, "G 0 1"               'Put in computer controlled mode.
SLEEP 1                         'Allow time for transmit of command.
DO UNTIL EOF(1)                 'Clear input buffer on your computer.
   dummy$ = INPUT$(1, #1)
LOOP

' ******************************************************************
' *       The Experimenter is now ready.  YOUR CODE GOES HERE!     *
' ******************************************************************
END

' ******************** ERROR HANDLING ROUTINES ********************
' Error occurred while opening the COM port. Give a helpful message.
ioError1:
   PLAY "MB L16 A B C"
   PRINT "Be sure the Experimenter is powered-on, 9600 baud."
   PRINT "Verify the Experimenter is connected to the correct COM
port."
   PRINT
   ON ERROR GOTO ioError2
RESUME
' Ignore all further errors.
ioError2:
RESUME
```

Listing 3-2: TEMPLATE.BAS for Initializing the Experimenter

This listing is common to many programs that control the Experimenter. It sets up communication between your computer and an Experimenter connected to COM1 or COM2 at 9600 baud. It also initializes both your computer and the Experimenter for computer controlled communications. By substituting in chunks of code, such as ***Listing 3-1****, your computer can command the Experimenter.*

4

Command Tutorial

This chapter gives you a little background on each of the Experimenter commands. The next chapter is a reference guide to the commands, giving detail on each of the parameters. You may wish to refer to that chapter as you go through this tutorial. While performing these exercises, you may communicate with the Experimenter using HyperTerminal or ECHO-EXP.BAS (**Listing 2-1**).

ANALOG Voltage Measurement

Use the **ANALOG** command to measure and report the voltage on any of the eight analog inputs (ANALOG 0:7). When you issue an **A**-command, the Experimenter selects the channel you specified, makes the voltage measurement, and reports the value to you in millivolts.

The analog inputs are designed for monitoring relatively slowly varying signals. If the slew rate on an analog input is greater than 10 volts/millisecond the analog to digital converter may become confused and output an incorrect value. This is usually the result of inadequate filtering or digital noise spikes coupling into the analog signal. Isolating sensitive analog signals from digital electronics should help. Adding a capacitor from the analog input to analog ground (AG, part of IO6 ANALOG) may also prove helpful. A good value to try is 0.1 µF.

A simple way of experimenting with this input is to connect up a potentiometer (pot) between the A+5 and AG pads. The value of the pot is not critical, use any convenient value from 1 KΩ to 100 KΩ. Connect the wiper of the pot to analog measurement input 0 (ANALOG 0). Put a 0.1 µF capacitor from the wiper to the analog ground. See **Figure 4-1**.

Adjust the pot so that the wiper is about centered. Now give the command:

```
Exp> A 0

2500
```

The Experimenter measures the voltage on the wiper and should respond with a value of about 2500 (millivolts). Now, adjust the pot so the wiper is all the way to the A+5 end of the pot. Give the command:

```
Exp> A 0

5115
```

This time the Experimenter measured 5115 (millivolts). This is the maximum measurement voltage. Adjust the pot so that the wiper is all the way to the AG end of the pot. Give the command:

```
Exp> A 0

0
```

Now the Experimenter has measured 0 (millivolts) on this input. This is the minimum measurement voltage. When you connect analog sensors to these inputs you must provide suitable scaling circuitry so that the voltage is within the range from 0 to 5.12 volts. You will probably want to scale the input voltage to take up most of this range for maximum resolution.

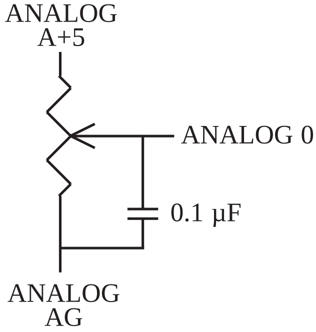


Figure 4-1: Creating a Voltage with a Pot *A potentiometer provides a convenient way to make an adjustable voltage for experimenting with analog voltage measurement. The values of the pot and cap are not critical—use any convenient value from 1 KΩ to 100 KΩ, and around 0.1 µF. By the way, this is how analog joysticks work.*

B-COMMAND

This command is reserved for a future firmware release.

COUNTER/TIMER Pulse Counting / Time Measurement

The Experimenter has four counter/timer inputs (IO8 T-IN 0:3). These inputs expect a logic level voltage swing, such as from a TTL or CMOS gate, and can accept signals from mechanical devices like push-button switches.

Counter/timers can be used to count pulses or measure time intervals. To explore some of these capabilities, you may wish to attach two push-button switches to the counter/timer inputs as shown in **Figure 4-2**.

In the figure, switches are connected to inputs T-IN 0 and T-IN 1. When pushed, the switches short the inputs to GND (logic low). Otherwise the pull-up resistors bring the inputs up to the 5 volt logic supply (logic high). So, T-IN 0 and 1 should normally be high, should go low when you press the switch, then should return high when you release the switch.

Lets try this. Issue the command:

```
Exp> C 0 1

0
```

This command tells the Experimenter to report the number of counts accumulated on T IN 0, then to reset that counter, and set up the counter to count rising edges (low-to-high transitions). Press and release the switch connected to T IN 0 once. Now issue the command:

```
Exp> C 0 1

1
```

The counter should have counted to one. Did yours? You may have found that the Experimenter counted some number greater than one. If it did, this is due to a phenomenon called “contact bounce.” This happens because the electrical contacts in mechanical switches (especially cheap mechanical switches) may

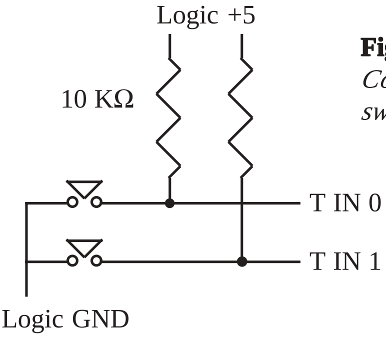


Figure 4-2: Switches for Counter/Timers
Connect two normally open push-button switches to the T IN 0 and T IN 1 inputs as shown. The values of pull-up resistors are not critical, but a value around 10 KΩ would be appropriate. Using these switches you can investigate the Experimenter’s ability to make time measurements of real world events.

not snap closed perfectly and stay together. Roughness in the surface of the contacts, dirt, and the bouncing as the contacts strike each other may cause the electrical signal produced by the switch to oscillate. The Experimenter is sensitive to contact bounces lasting longer than a hundred microseconds or so.

One way to minimize contact bounce is to add a capacitor across the switch. This damps out the oscillations, resulting in a gradual rise (and abrupt fall) in voltage as the switch is opened and closed.

The first time we gave the command **`C 0 1`**, we started the counting. From then on it would count each rising edge it saw on T-IN 0. Whenever the Experimenter gets a count command, it reports the current value of that counter. Since the counter was idle when we gave the first count command it reported **`0`**.

When you pushed the button, one (or more) pulses were produced. The Experimenter counted them. Then when you again issued the command **`C 0 1`**, the Experimenter reported to you the number of pulses it had counted, reset the counter to 0, and resumed looking for rising edges to count.

If you don’t want to reset the counter, you simply want to see what the accumulated count is so far, then use the command **`C 0 0`**. If you want to stop the counter after reporting a reading, you can use the command **`C 0 4`**. Any pulses produced after this command will be ignored. You may wish to experiment with these commands.

Now lets look at a timer command. Issue the following command (but don’t push either of the switches):

```
Exp> C 0 12

0
```

Counter/timer function 12 causes the Experimenter to measure the time interval from a falling-edge on T-IN 0 to a falling-edge on T-IN 1. The Experimenter waits for a couple of seconds for the first edge to occur. If it doesn't see any edge it gives up waiting and returns a value of 0. This prevents the Experimenter from “hanging” if no edge is present. You can program the length of time for the Experimenter to wait, in 0.1 second increments. The command **`C 0 12 100`** will cause the Experimenter to wait for 10 seconds before giving up and reporting 0. If you give the command **`C 0 12 0`** the Experimenter will wait forever for that first edge. The power-up default is about 2 seconds.

Now let’s issue a command, followed by immediately pushing the switch connected to T IN 0:

```
Exp> C 0 12

655350
```

About a half-second after the first switch was pressed, the Experimenter printed the value 655350. The Experimenter started timing when the first switch was pressed, measuring with a resolution of 10 microseconds. When the time interval reached the maximum value of 655350 microseconds the counter was stopped, and the Experimenter reported the value.

Lets try making a measurement. This will require quick reflexes. After you issue the command **`C 0 12 100`**, you will have 10 seconds in which to press the first button (going to T-IN 0). Then, as quickly as you can, press the second button (going to T-IN 1). The Experimenter will measure and report the time between button presses. The minimum measurement interval the Experimenter can detect is under 250 microseconds.

DIGITAL I/O Input and Output through Digital Ports

The Experimenter provides 24 bits of digital I/O. Before using the digital I/O, you must select which ports you want to use as inputs and which ports you want to use as outputs. Ports A and B are each eight bits wide (7:0), and port C can be divided into two ports, each four bits wide (7:4, 3:0). Bit seven is the most-significant bit (left-most), bit 0 is the least-significant (one's) bit.

Let's consider an example. Suppose you want port A and the high nibble of port C to be outputs, port B and the low nibble of port C to be inputs. Look up the DIGITAL I/O in the Command Reference chapter. In the table, we find that the mode number for this configuration would be 131. So, in order to configure the digital I/0 as desired, we must send the command **`D 3 131`**.

Now let's suppose you want to send out the byte 10001000_b to the A port for 25 seconds. The binary value 10001000_b corresponds to 136 decimal. So, we would send the command **`D 0 136 25000 0`**, meaning send to port A (number 0) the value 136 for a duration of 25000 milliseconds, then after that time expires output the value 0.

If you wanted to check the progress of the timer, you could type:

```
Exp> D 0

136, 17531
```

This tells you that port A still has a value of 136, and that the duration remaining for this output is 17,531 milliseconds. If you waited a while longer, then typed:

```
Exp> D 0

0, 0
```

This shows that the timer has expired, and the value of the output port is now 0. The digital I/O command will let you sequence up to three bytes on each port, with durations of from 1 to 65535 milliseconds between them. This is useful for creating control signals and strobes. With additional buffering, these outputs can be used to drive motors, solenoids, pneumatic and hydraulic valves, and other devices.

Now let's look at using these ports as inputs. Suppose you want to read port B, which we had set up to be an input port. The command **`D 1`** would read the value of port B, and would respond with two numbers as above. The first number is the value read from the port. The second number is always 0 in Version 1.0 firmware.

Input ports are useful for sensing switches and monitoring digital signals. Suppose you were building a sophisticated burglar alarm for your home. You could install magnetic switches in the doors and windows that trigger when opened. In the figure below, several switches are used on windows in the family room, one is used on the front door. When the door or any of the three windows are opened, the input to the Experimenter goes high. Your computer could then take appropriate action. Using the relay on the Experimenter, your computer could switch-on a loud alarm horn, or switch-on a tape recording of a vicious barking dog.

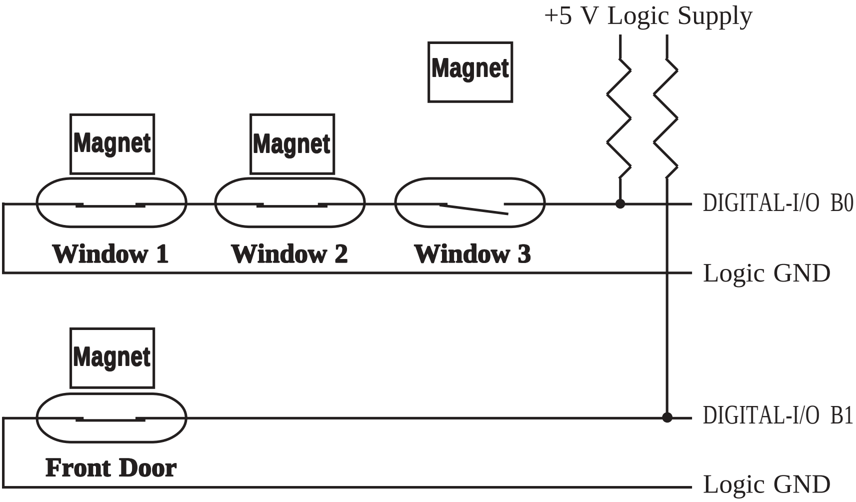


Figure 4-3: Magnetic Switches for a Burglar Alarm
These magnetic switches are closed when a magnet is near them, and open when the magnet moves away. The pull-up resistors cause the digital I/O lines to go to logic 1 when the switches open. Resistor values are not critical, but a value in the 1 KΩ to 10 KΩ range would be appropriate.

ENABLE-PWM Enable Drivers with PWM

Pulse width modulation is a technique for achieving analog-like voltage control using only digital logic. The PWM controls the proportion of time that an output is driven. When the output is driven a large proportion of the time, the effect is as if a large analog voltage is present. When the output is driven a small proportion of the time, the effect is as if a small analog voltage is present.

There are two pulse width modulators on the Experimenter (0 and 1). Each is independently controlled. You can control both the ***duty-cycle*** and the ***rate***. The ***duty-cycle*** is the proportion of enabled to disabled time. The ***rate*** determines how many cycles are produced each second. **Figure 4-4** shows the enable waveform available at the PWM outputs (IO9 PWM 0:1).

The power-on defaults set the PWM outputs high 100% of the time (***duty-cycle*** = 255), and run at the maximum rate of 14,456 pulses per second—a period of about 0.07 milliseconds per pulse (***rate*** = 0). **Table 5-1** in the Command Reference chapter lists all of the ***rate*** values and their corresponding frequency and period.

Jumpers J7 through J10 are provided so that the eight driver outputs may be selected in pairs to be enabled by PWM0 or PWM1. If no jumper is present the pair is always enabled. J10 controls 0/1, J9 2/3, J8 4/5, J7 6/7.

Pulse width modulation can be used to control the speed of DC motors and the torque of stepping motors. Since the topic of motor control is fairly involved, Chapter 6 is devoted solely to controlling motors.

You can turn a PWM output into an analog voltage by running it through a low-pass filter. A series resistor and a capacitor to ground will usually work just fine.

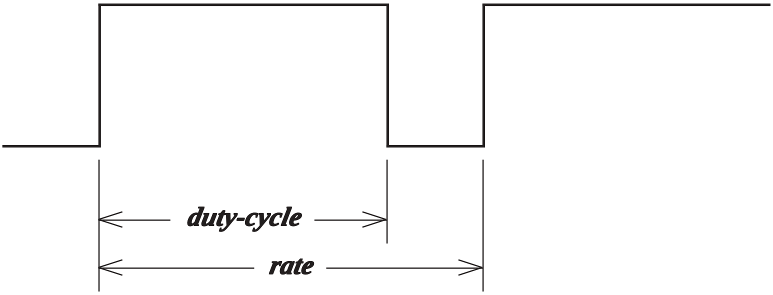


Figure 4-4: Pulse Width Modulation (IO9 PWM 0:1)
The Experimenter has two PWM outputs. They may also be used to enable the high current driver outputs in pairs. Drivers are enabled when their enable inputs are high (logic 1).

FLIP-RELAY Switches Relay On and Off

The Experimenter has one relay for controlling high-power devices. The command **`F 0 1`** turns the relay on. The first parameter is the relay number. There is only one relay on this version of the Experimenter, but the relay number was included for upward compatibility with possible future versions. The second parameter controls the state of the relay. The command **`F 0 0`** turns the relay off. The command **`F 0 2`** toggles the state of the relay. If you would like to know the current state of the relay, the command **`F 0`** reports the current state.

There are three pads adjacent to the relay. They are marked IO2 NC, IO3 NO, and IO4 COM. When the relay is off, the COM (Common) terminal connects to the NC (Normally Closed) terminal. When the relay is active, the COM terminal disconnects from the NC terminal and connects to the NO (Normally Open) terminal. An LED, D3, lights when the relay is active.

GENERAL-INFORMATION/CONTROL

This command lets you adjust the way that the Experimenter communicates over the serial port. In Chapter 3, in the section on Software Controlled Communication, we saw how to use the GENERAL-INFORMATION/CONTROL command to disable some of the extra characters the Experimenter normally sends. The functions enabling/disabling the echo of characters, appending a ***linefeed*** after each ***carriage return***, and appending a “0” after each timer measurement can each be independently controlled. These features may be useful to adapt the Experimenter’s style to your communication program.

For example, some terminal emulation programs may automatically append a ***linefeed*** to each ***carriage return*** received. Since the Experimenter normally defaults to send a ***linefeed***, this would cause the output on your screen to appear double-spaced. To correct this, the command **`G 2 0`** would turn off the appending of ***linefeeds*** but leave the Experimenter in manual controlled mode. The command **`G 2 1`** would turn them back on.

If the Experimenter is running in computer controlled mode (not echoing characters, etc.), and you wish to return it to manual control send the single-letter command **`G`**. This will restore all of the default communications features and will print the power-on message.

H-BRIDGE Control Motors

Controlling motors, especially stepping motors, is one of the most fun ways to use the Experimenter because it gives your computer a way to actually manipulate objects in the physical world. To help you with this Chapter 6 is devoted to DC and stepping motors. This section will explain how to use the H-BRIDGE command itself.

The **H**-command has five parameters. The first parameter, ***group***, tells the Experimenter which drivers, and thus which motor, you wish to control with this command. DC motors each require only two drivers. The Experimenter can support up to four DC motors, so there are four groups (numbered 0 to 3). Driver A (U8) supports the even-numbered motors (0, 2). Driver B (U7) supports the odd-numbered motors (1, 3).

Stepping motors require four drivers each, so there are only two groups (0:1). Again, driver A supports the even-numbered motor (0), driver B the odd-numbered motor (1).

The Experimenter can also support one stepping motor and two DC motors. If the stepping motor is on driver A, it is number 0, and the DC motors on driver B are numbers 1 and 3. If the stepping motor is on driver B, it is number 1, and the DC motors on driver A are numbers 0 and 2.

The second parameter, ***direction***, controls which direction the motor is to rotate. The value 1 causes the motor to go forward, 2 to go reverse, and the values 0 and 3 cause the motor to stop. If the motor is a stepping motor, stop-0 causes the motor to stop with the coils energized, and stop-3 causes the motor to stop with the coils off.

The ***duration*** parameter tells the Experimenter how long to run the motor. For a DC motor, the ***duration*** value is interpreted as milliseconds. For a stepping motor, the ***duration*** is interpreted as steps. Thus, for a DC motor, a ***duration*** of **2000** means 2 seconds, but for a stepping motor it means 2000 steps. A duration of **0** means to run the motor until a stop instruction is sent.

The ***speed*** parameter controls two different things, depending on whether the command is for a stepping motor or a DC motor. If for a DC motor, the ***speed*** parameter adjusts the ***duty-cycle*** of one PWM. Since there are only two PWM's and there can be four DC motors changing the ***speed*** of one DC motor will also change the ***speed*** of another DC motor on thatPWM. Note that a larger ***speed*** value makes a DC motor run faster, because the ***duty-cycle*** is increased.

When controlling a stepping motor, the ***speed*** parameter is the time the motor spends making each step (in milliseconds). For a stepping motor, the larger the ***speed*** value the slower the motor runs. The total time it takes to make a particular movement is the product of the ***duration*** times the ***speed*** (in mS).

The ***type*** parameter tells the Experimenter exactly what type of motor you are using, and the drive pattern to use with it. DC motors are either ***type*** 0 or 1, depending on which way the motor leads are connected. **Table 5-2** in the next chapter gives the type values for the various kinds of stepping motors. Chapter 6 will explain enough about stepping motors that this table should make sense.

The Experimenter remembers the ***speed*** and ***type*** parameters, and these parameters need not be provided again unless you want to change them.

There is also a way to determine the progress of an ongoing **`H`**-command. If sent an H-command with just the ***group*** parameter given, the Experimenter reports the remaining ***duration*** for that group.

INDIVIDUAL-OUTPUT Controls Driver Outputs Individually

Each of the Experimenter's eight driver outputs can be individually controlled with the INDIVIDUAL-OUTPUT command, including timing and number of pulses. The command has five parameters.

The first parameter, ***channel***, selects which output driver. Channels are numbered from 0 to 7, corresponding to DRIVER-A 0:3 and DRIVER-B 4:7.

The next parameter, ***state***, selects the output voltage. A value of 0 outputs a low voltage, 1 outputs a high voltage. A value of 2 causes the channel to toggle.

The ***duration*** parameter sets how long (in milliseconds) the channel will remain in the specified ***state***. When ***duration*** has expired, the output will toggle and remain in the new state as long as specified by the ***complement.duration*** parameter (also in milliseconds).

The ***cycles*** parameter limits the number of repeats before stopping. If the ***complement.duration*** is 0, the output will remain in the initial state for the entire interval of ***cycles * duration***. A value of 0 puts no limit on the number of cycles.

To determine the progress of an ongoing I-command, give the command with only the ***channel*** parameter. The Experimenter will report three numbers (separated by commas and spaces): the value of the current ***state***, the ***duration*** remaining for that ***state***, and the number of ***cycles*** remaining.

As an example, the command **`I 5 1 100 200 250`** produces a pulse on DRIVER-B 5 that is high (1) for 100 milliseconds, low for 200 milliseconds, and repeats to produce 250 pulses. While this pulse stream is running, if you issue the command **`I 5`** the Experimenter will report three numbers. For example, the Experimenter may report **`1, 25, 175`** meaning the output state is currently high, that it will remain in that state for 25 more milliseconds, and that there are 175 cycles remaining to be produced.

5

Command Reference

Experimenter firmware Version 1.0 provides eight commands, with from one to six parameters each. Each section of this chapter describes one of these commands, its parameters, responses (if any), and any special considerations for use.

If you have suggestions for additional commands, changes to these commands, or any other ideas for improving the Experimenter, please let us know. Your feedback is very important to us.

ANALOG Voltage Measurement

Reports the voltage on the specified analog input channel (IO6 ANALOG 0:7).

A ***channel***

channel analog input channel number

none	channel 0 assumed
`0` *to* **`7`**	available channel selections

Notes:

1. Measurements have 5 mV resolution over the range from 0 to 5.115 volts.
2. Values are reported as integers from **`0`** to **`5115`**, in multiples of 5.
3. Slew rates exceeding 10 volts/millisecond, or voltages above 5.12 volts or below ground on any analog input may result in erroneous measurements.

B-COMMAND

This command is reserved for a future firmware release.

COUNTER/TIMER Pulse Counting / Time Measurement

Counts pulses and measures time intervals on the specified counter/timer input channel(s) (IO8 T-IN 0:3).

C ***channel function wait***

channel counter/timer input channel

none	channel 0 assumed
0 *to* **3**	available channel selections

function counter/timer measurement function

	<u>Counter Measurements</u>:
none or **0**	report the current count (counter unchanged)
1	report count, restart counter, count on rising edge
2	report count, restart counter, count on falling edge
3	report count, restart counter, count on both edges
4	report count, reset counter, counting stopped
	<u>Single Channel Timer Measurements</u>:
5	measure and report positive-going pulse duration
6	measure and report negative-going pulse duration
7	measure and report rising-edge to rising-edge period
8	measure and report falling-edge to falling-edge period
	<u>Two Channel Timer Measurements</u>:
9	measure and report rising-edge to falling-edge time
10	measure and report falling-edge to rising-edge time
11	measure and report rising-edge to rising-edge time
12	measure and report falling-edge to falling-edge time

wait timer maximum waiting period for the start of a measurement

none	use previous (or default) value
0	unlimited wait
1 *to* **255**	waiting period, in 100 millisecond steps

<u>Notes</u>:

1. Maximum count rate is well over 1 kilohertz. Maximum count value is 65535. Further pulses beyond 65535 are ignored.
2. Timer measurements may range from about 250 microseconds to 655350 microseconds, with 10 microsecond resolution. The time is presented either in microseconds or in tens-of-microseconds (refer to Chapter 3, Software Controlled Communication for details of 0-append to timer measurements).
3. If the first edge defining a timer measurement does not occur within the specified ***wait*** time, a value of 0 will be returned, and the measurement terminated. This is to avoid “hanging” the Experimenter in the event that no signal is present.

DIGITAL I/O Input and Output Digital Ports

24 bits of digital input and output (IO13 DIGITAL-I/O A 7:0, B 7:0, C 7:0).

D **3** ***mode*** (To configure port I/O)

mode ports A, B, and C are set to inputs and outputs as follows:

mode	A7:0	C7:4	B7:0	C3:0
128	out	out	out	out
129	out	out	out	in
130	out	out	in	out
131	out	out	in	in
136	out	in	out	out
137	out	in	out	in
138	out	in	in	out
139	out	in	in	in
144	in	out	out	out
145	in	out	out	in
146	in	out	in	out
147	in	out	in	in
152	in	in	out	out
153	in	in	out	in
154	in	in	in	out
155	in	in	in	in

D ***port*** (For reading inputs or checking status of outputs)

port selected input/output port

none or **0**	port A is selected
1	port B is selected
2	port C is selected

If the selected port is an output:
Reports the current value and the remaining duration for the output, separated by a comma and a space.

If the selected port is an input:
Reads and reports the current input value and the number 0, separated by a comma and a space.

D ***port output.1 duration.1 output.2 duration.2 output.3*** (For outputs)

port selected input/output port

none or **0**	port A is selected
1	port B is selected
2	port C is selected

output.1 initial output value

none	port is read: value and duration remaining is reported
0 *to* **255**	value written to port outputs

duration.1 time duration output.1 is present on port outputs

none or **0**	duration is unlimited
1 *to* **65535**	duration in milliseconds

output.2 next output value (after duration.1 is over)

none or **0**	0 is written to port outputs
1 *to* **255**	value written to port outputs

duration.2 time duration output.2 is present on port outputs

as above for duration.1

output.3 final output value

as above for output.2

Note:

1. The **D 3** ***mode*** command is required to initialize ports for output. This also resets all outputs to 0.

ENABLE-PWM Enable Drivers with PWM

Controls the duty-cycle and period of the pulse width modulation outputs (IO9 PWM 0:1). These can enable the driver chips, controlling output power.

E ***channel duty-cycle rate***

channel counter/timer input channel

none or **0**	PWM 0 is selected
1	PWM 1 is selected

duty-cycle counter/timer input channel

none	report current pulse width setting
0	always off (low)
1 *to* **254**	variable pulse width
255	always on (high)

rate controls the pulse rate

none	use previously assigned value, default 0
0 *to* **255**	0 is fastest, 255 is slowest

Notes:

1. The PWM pulse frequency can be calculated from the equation:
   Frequency = 7372800/(510*(1+***rate***)) cycles per second.
2. The PWM pulse period can be calculated from the equation:
   Period = (510*(1+***rate***))/7372.8 milliseconds.
3. Table 5-1, on the next page, gives the frequency (in cycles per second) and period (in milliseconds) for all values of ***rate***.

Table 5-1: Pulse Width Modulator Rate, Frequency, and Period

*The **rate** parameter to the E-command allows you to control the pulse rate of the pulse width modulator. This table gives you the frequency (in cycles per second) and the period (in milliseconds) for each **rate** setting.*

Rate	Freq	Period	Rate	Freq	Period	Rate	Freq	Period	Rate	Freq	Period	Rate	Freq	Period
0	14,456	0.07	51	278	3.60	102	140	7.12	153	93.9	10.65	204	70.5	14.18
1	7,228	0.14	52	273	3.67	103	139	7.19	154	93.3	10.72	205	70.2	14.25
2	4,819	0.21	53	268	3.74	104	138	7.26	155	92.7	10.79	206	69.8	14.32
3	3,614	0.28	54	263	3.80	105	136	7.33	156	92.1	10.86	207	69.5	14.39
4	2,891	0.35	55	258	3.87	106	135	7.40	157	91.5	10.93	208	69.2	14.46
5	2,409	0.42	56	254	3.94	107	134	7.47	158	90.9	11.00	209	68.8	14.53
6	2,065	0.48	57	249	4.01	108	133	7.54	159	90.4	11.07	210	68.5	14.60
7	1,807	0.55	58	245	4.08	109	131	7.61	160	89.8	11.14	211	68.2	14.66
8	1,606	0.62	59	241	4.15	110	130	7.68	161	89.2	11.21	212	67.9	14.73
9	1,446	0.69	60	237	4.22	111	129	7.75	162	88.7	11.28	213	67.6	14.80
10	1,314	0.76	61	233	4.29	112	128	7.82	163	88.1	11.34	214	67.2	14.87
11	1,205	0.83	62	229	4.36	113	127	7.89	164	87.6	11.41	215	66.9	14.94
12	1,112	0.90	63	226	4.43	114	126	7.95	165	87.1	11.48	216	66.6	15.01
13	1,033	0.97	64	222	4.50	115	125	8.02	166	86.6	11.55	217	66.3	15.08
14	964	1.04	65	219	4.57	116	124	8.09	167	86.1	11.62	218	66.0	15.15
15	904	1.11	66	216	4.63	117	123	8.16	168	85.5	11.69	219	65.7	15.22
16	850	1.18	67	213	4.70	118	121	8.23	169	85.0	11.76	220	65.4	15.29
17	803	1.25	68	210	4.77	119	120	8.30	170	84.5	11.83	221	65.1	15.36
18	761	1.31	69	207	4.84	120	119	8.37	171	84.0	11.90	222	64.8	15.43
19	723	1.38	70	204	4.91	121	118	8.44	172	83.6	11.97	223	64.5	15.49
20	688	1.45	71	201	4.98	122	118	8.51	173	83.1	12.04	224	64.3	15.56
21	657	1.52	72	198	5.05	123	117	8.58	174	82.6	12.11	225	64.0	15.63
22	629	1.59	73	195	5.12	124	116	8.65	175	82.1	12.17	226	63.7	15.70
23	602	1.66	74	193	5.19	125	115	8.72	176	81.7	12.24	227	63.4	15.77
24	578	1.73	75	190	5.26	126	114	8.78	177	81.2	12.31	228	63.1	15.84
25	556	1.80	76	188	5.33	127	113	8.85	178	80.8	12.38	229	62.9	15.91
26	535	1.87	77	185	5.40	128	112	8.92	179	80.3	12.45	230	62.6	15.98
27	516	1.94	78	183	5.46	129	111	8.99	180	79.9	12.52	231	62.3	16.05
28	498	2.01	79	181	5.53	130	110	9.06	181	79.4	12.59	232	62.0	16.12
29	482	2.08	80	178	5.60	131	110	9.13	182	79.0	12.66	233	61.8	16.19
30	466	2.14	81	176	5.67	132	109	9.20	183	78.6	12.73	234	61.5	16.26
31	452	2.21	82	174	5.74	133	108	9.27	184	78.1	12.80	235	61.3	16.32
32	438	2.28	83	172	5.81	134	107	9.34	185	77.7	12.87	236	61.0	16.39
33	425	2.35	84	170	5.88	135	106	9.41	186	77.3	12.94	237	60.7	16.46
34	413	2.42	85	168	5.95	136	106	9.48	187	76.9	13.00	238	60.5	16.53
35	402	2.49	86	166	6.02	137	105	9.55	188	76.5	13.07	239	60.2	16.60
36	391	2.56	87	164	6.09	138	104	9.62	189	76.1	13.14	240	60.0	16.67
37	380	2.63	88	162	6.16	139	103	9.68	190	75.7	13.21	241	59.7	16.74
38	371	2.70	89	161	6.23	140	103	9.75	191	75.3	13.28	242	59.5	16.81
39	361	2.77	90	159	6.29	141	102	9.82	192	74.9	13.35	243	59.2	16.88
40	353	2.84	91	157	6.36	142	101	9.89	193	74.5	13.42	244	59.0	16.95
41	344	2.91	92	155	6.43	143	100	9.96	194	74.1	13.49	245	58.8	17.02
42	336	2.97	93	154	6.50	144	99.7	10.03	195	73.8	13.56	246	58.5	17.09
43	329	3.04	94	152	6.57	145	99.0	10.10	196	73.4	13.63	247	58.3	17.15
44	321	3.11	95	151	6.64	146	98.3	10.17	197	73.0	13.70	248	58.1	17.22
45	314	3.18	96	149	6.71	147	97.7	10.24	198	72.6	13.77	249	57.8	17.29
46	308	3.25	97	148	6.78	148	97.0	10.31	199	72.3	13.83	250	57.6	17.36
47	301	3.32	98	146	6.85	149	96.4	10.38	200	71.9	13.90	251	57.4	17.43
48	295	3.39	99	145	6.92	150	95.7	10.45	201	71.6	13.97	252	57.1	17.50
49	289	3.46	100	143	6.99	151	95.1	10.51	202	71.2	14.04	253	56.9	17.57
50	283	3.53	101	142	7.06	152	94.5	10.58	203	70.9	14.11	254	56.7	17.64
												255	56.5	17.71

FLIP-RELAY Switches Relay On and Off

Sets, resets, or toggles the single-pole double-throw relay, or reports its state.

F ***relay.number state***

relay.number the relay number

any	the relay number is ignored

state reads or sets the relay's state

none	report the relay's current state
`0`	relay coil off
`1`	relay coil on
`2`	toggle the relay’s state

Notes:

1. The parameter ***relay.number*** is included for compatibility with possible future versions of the Experimenter having more than one relay. Firmware Version 1.0 ignores the value of ***relay.number***.
2. When the relay coil is off, the IO2 NC (Normally Closed) contact is connected to the IO4 COM (Common) contact—when on, the IO3 NO (Normally Open) contact is connected to COM.

GENERAL-INFORMATION/CONTROL

General information about the Experimenter is reported, and special functions are controlled.

G ***select off/on***

select selects the information or control function

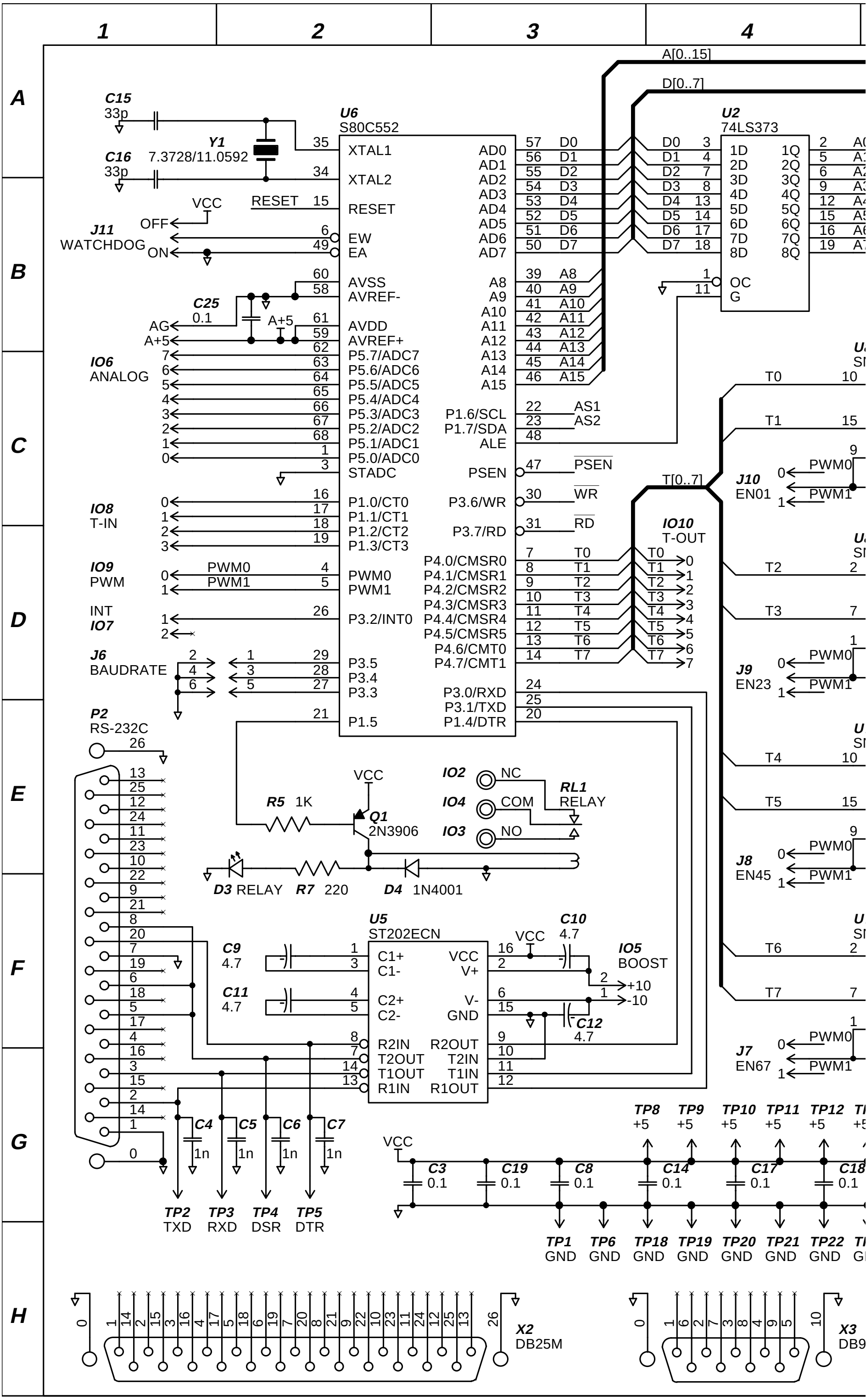
none or **`0`**	manual (default) / computer control mode (when on)
`1`	echo typing
`2`	append a ***linefeed*** after each ***carriage-return***
`3`	append a **`0`** to all non-zero timer measurements

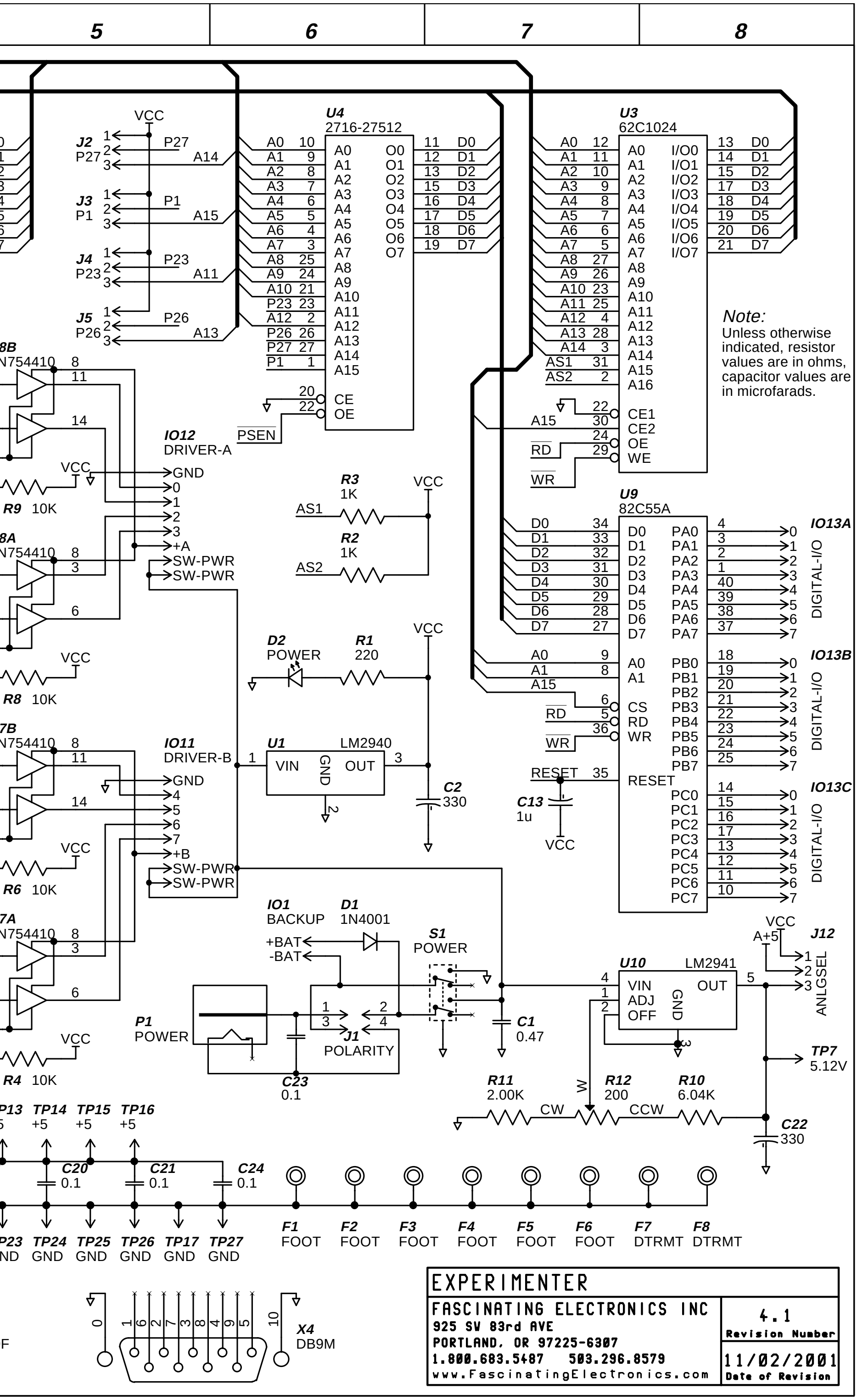
off/on disables or enables the selection

none or **`0`**	turn the selection off
`1`	turn the selection on

Notes:

1. Manual mode defaults to enable selections 1, 2 and 3; computer control mode disables them.
2. The echo typing selection also enables the prompt message (**Exp>**).


1
2
3
4
A
B
C
D
E
F
G
H
A[0..15]
D[0..7]
C15
33p
Y1
7.3728/11.0592
C16
33p
U6
S80C552
35 XTAL1
34 XTAL2
VCC
RESET 15 RESET
J11
WATCHDOG
OFF
ON
6 EW
49 EA
60 AVSS
58 AVREF-
C25
0.1
AG
A+5
61 AVDD
59 AVREF+
IO6
ANALOG
62 P5.7/ADC7
63 P5.6/ADC6
64 P5.5/ADC5
65 P5.4/ADC4
66 P5.3/ADC3
67 P5.2/ADC2
68 P5.1/ADC1
1 P5.0/ADC0
3 STADC
IO8
T-IN
16 P1.0/CT0
17 P1.1/CT1
18 P1.2/CT2
19 P1.3/CT3
IO9
PWM
PWM0 4 PWM0
PWM1 5 PWM1
INT
IO7
26 P3.2/INT0
J6
BAUDRATE
29 P3.5
28 P3.4
27 P3.3
21 P1.5
AD0 57 D0
AD1 56 D1
AD2 55 D2
AD3 54 D3
AD4 53 D4
AD5 52 D5
AD6 51 D6
AD7 50 D7
A8 39 A8
A9 40 A9
A10 41 A10
A11 42 A11
A12 43 A12
A13 44 A13
A14 45 A14
A15 46 A15
P1.6/SCL 22 AS1
P1.7/SDA 23 AS2
ALE 48
PSEN 47 PSEN
P3.6/WR 30 WR
P3.7/RD 31 RD
P4.0/CMSR0 7 T0
P4.1/CMSR1 8 T1
P4.2/CMSR2 9 T2
P4.3/CMSR3 10 T3
P4.4/CMSR4 11 T4
P4.5/CMSR5 12 T5
P4.6/CMT0 13 T6
P4.7/CMT1 14 T7
P3.0/RXD 24
P3.1/TXD 25
P1.4/DTR 20
U2
74LS373
D0 3 1D 1Q 2 A0
D1 4 2D 2Q 5 A1
D2 7 3D 3Q 6 A2
D3 8 4D 4Q 9 A3
D4 13 5D 5Q 12 A4
D5 14 6D 6Q 15 A5
D6 17 7D 7Q 16 A6
D7 18 8D 8Q 19 A7
1 OC
11 G
T[0..7]
IO10
T-OUT
T0 10
T1 15
T2 2
T3 7
T4 10
T5 15
T6 2
T7 7
J10
EN01
J9
EN23
J8
EN45
J7
EN67
PWM0
PWM1
P2
RS-232C
IO2 NC
IO4 COM
IO3 NO
RL1
RELAY
R5 1K
Q1
2N3906
D3 RELAY
R7 220
D4 1N4001
U5
ST202ECN
C9 4.7
C11 4.7
1 C1+
3 C1-
4 C2+
5 C2-
8 R2IN
7 T2OUT
14 T1OUT
13 R1IN
VCC 16
V+ 2
V- 6
GND 15
R2OUT 9
T2IN 10
T1IN 11
R1OUT 12
C10 4.7
C12 4.7
IO5
BOOST
+10
-10
C4 1n
C5 1n
C6 1n
C7 1n
TP2 TXD
TP3 RXD
TP4 DSR
TP5 DTR
TP8 +5
TP9 +5
TP10 +5
TP11 +5
TP12 +5
C3 0.1
C19 0.1
C8 0.1
C14 0.1
C17 0.1
C18 0.1
TP1 GND
TP6 GND
TP18 GND
TP19 GND
TP20 GND
TP21 GND
TP22 GND
X2
DB25M
X3
DB9



Note:
Unless otherwise indicated, resistor values are in ohms, capacitor values are in microfarads.

EXPERIMENTER

FASCINATING ELECTRONICS INC
925 SW 83rd AVE
PORTLAND, OR 97225-6307
1.800.683.5487 503.296.8579
www.FascinatingElectronics.com

4.1
Revision Number

11/02/2001
Date of Revision

H-BRIDGE Control Motors

Configures the drivers as an H-Bridge to control DC or stepping motors.

H ***group direction duration speed type***

group selects groups of driver outputs

	For DC Motors:
0	DRIVER-A 0 and 1; controls PWM 0
1	DRIVER-B 4 and 5; controls PWM 1
2	DRIVER-A 2 and 3; controls PWM 0
3	DRIVER-B 6 and 7; controls PWM 1
	For Stepping Motors:
0	DRIVER A 0, 1, 2, 3
1	DRIVER B 4, 5, 6, 7

direction motor direction control

none	report remaining duration
0	stop (for stepper: coils on)
1	forward
2	reverse
3	stop (for stepper: coils off)

duration how long to drive the motor, then return to stop (0)

none or **0**	drive duration unlimited

1 to **65535**	for DC motors: duration in milliseconds
	for stepping motors: duration in steps

speed PWM control for DC motors, step duration for stepping motors

	For DC Motors:
none or **0**	PWM settings are not changed
1 to **255**	PWM ***duty-cycle*** set as in the **E**-command
	For Stepping Motors:
none or **0**	rate is not changed (default 255 milliseconds/step)
1 to **65535**	step duration in milliseconds/step

type selects type of motor and driver configuration

none	type of motor is not changed
	For DC Motors:
0	positive polarity (default)
1	negative polarity
	For Stepping Motors:
2 *to* **11**	see ***Table 5-2*** on the next page

Table 5-2: Stepping Motor Configuration Types

The Experimenter can control all common types of stepping motors. The ***type*** *parameter tells the Experimenter how to drive the particular motor you are using. The chapter* ***DC and Stepping Motors*** *tells you how to recognize the various types of stepping motors. Then, using this table, you can select the correct* ***type*** *parameter for the motor you are using.*

Type	Coil Configuration	Common	Drive Pattern
2	3-coil unipolar	- supply	1-phase
3	3-coil unipolar	+ supply	1-phase
4	3-coil unipolar	- supply	half-step
5	3-coil unipolar	+ supply	half-step
6	4-coil unipolar	- supply	1-phase
7	4-coil unipolar	+ supply	1-phase
8	4-coil unipolar *or*	- supply	2-phase
	2-coil bipolar	none	2-phase
9	4-coil unipolar *or*	+ supply	2-phase
	2-coil bipolar	none	2-phase
10	4-coil unipolar *or*	- supply	half-step
	2-coil bipolar	none	half-step
11	4-coil unipolar *or*	+ supply	half-step
	2-coil bipolar	none	half-step

Notes:

1. When a new stepper command is issued for a ***group*** with an ongoing stepper command, if the current step is longer than 50 milliseconds it is shortened to 50 milliseconds. Then, upon the completion of the current step, the new command begins.
2. If only the ***group*** parameter is given, the Experimenter reports the remaining ***duration*** for any ongoing stepper command on that group.
3. Since both the **`H`**-command and **`I`**-command share the driver outputs, issuing an **`H`**-command for a ***group*** that includes a ***channel*** in use by an **`I`**-command will cause that **`I`**-command to terminate.

INDIVIDUAL-OUTPUT Controls Driver Outputs Individually

Controls state and timing for eight driver outputs individually (DRIVER-A 0:3, DRIVER-B 4:7).

I ***channel state duration complement.duration cycles***

channel		selected driver
	none	channel 0 assumed
	0 *to* **7**	available channel selections
state		initial output state
	none	report state, remaining time in state, remaining cycles
	0	initial state is low
	1	initial state is high
	2	initial state is the complement of the current state
duration		time duration for the initial state
	none or **0**	duration is unlimited
	1 *to* **255**	duration of initial state, in milliseconds
comp.dur		duration for the complement state
	none or **0**	0 time between initial states
	1 *to* **255**	duration of complement state, in milliseconds
cycles		limit on the number of cycles to be repeated
	none or **0**	cycle without limit
	1 *to* **255**	perform this number of cycles, then stop

Notes:

1. To get longer durations for a single pulse, set ***complement.duration*** to **0**. The duration of the single pulse produced will be ***duration*** * ***cycles*** (milliseconds).
2. If only the ***channel*** parameter is given, the Experimenter reports the current ***state***, the ***duration*** remaining for that state, and the number of ***cycles*** remaining for that channel.
3. Since both the **H**-command and **I**-command share the driver outputs, issuing an **I**-command for a ***channel*** that is part of a ***group*** in use by an **H**-command will cause that **H**-command to terminate.

6

DC and Stepping Motors

This chapter is a tutorial on DC and stepping motors. A stepping motor is a remarkable device, much different from ordinary DC motors. Ordinary DC motors have just two wires. When they are connected to a battery the motor will run as fast as it can, depending on the power supplied and how heavily it is loaded.

In contrast, a stepping motor has two or more separate coils, and four or more wires. Each coil causes the motor to turn slightly, usually a few degrees. By sequentially energizing the coils, the motor can be made to rotate a specific amount at an accurately controlled speed. Stepping motors give us the ability to precisely control the rotational position and speed of the motor shaft.

The Experimenter is designed to directly drive up to four DC motors or two stepping motors at the same time, with supply voltages from 4.5 to 36 volts and maximum currents of 1 amp across each coil. With some additions the Experimenter can drive more motors and higher currents.

Driving DC Motors

The Experimenter is designed to drive DC motors in an H-bridge configuration. **Figure 6-1** shows an H-bridge, in its conceptual form, driving a DC motor. Of course, the Experimenter uses transistors in the driver chips, U8 and U9, instead of switches. In this configuration, not only can the power to the motor be switched on and off, but the polarity of the power can be selected too. This lets you control the direction of the motor. The ***direction*** parameter in the **H**-command selects which way the power is routed through the motor.

The **H**-command also lets you control the speed of DC motors. The ***speed*** parameter does this by varying the duty-cycle of the drivers. The drivers may be enabled and disabled by a pulse width modulator (PWM). When a driver is enabled, it provides full power to the motor. When disabled, it provides none. By rapidly switching the drivers on and off, the power level can be adjusted to intermediate values. This is less precise than controlling the speed on a stepping motor, which is done with crystal-clock precision. But it does give you some measure of control, which is adequate in many applications.

It may prove necessary to adjust the frequency of the PWM to avoid resonances with the motor's rotation. The frequency of the PWM is set using the **E**-command. The optimum frequency value for a specific motor in a particular application is best determined by experiment. Low frequencies often cause the motor to operate in a jerky manner. High frequencies may be limited by the inductance of the motor.

Because of their internal mechanical brushes, DC motors produce a tremendous amount of electrical noise. It helps to bypass the high-frequency energy through nonpolar capacitors installed across the motor's power leads. Try a nonpolar electrolytic capacitor (≥100 µF) in parallel with a disk capacitor (≥0.1 µF).

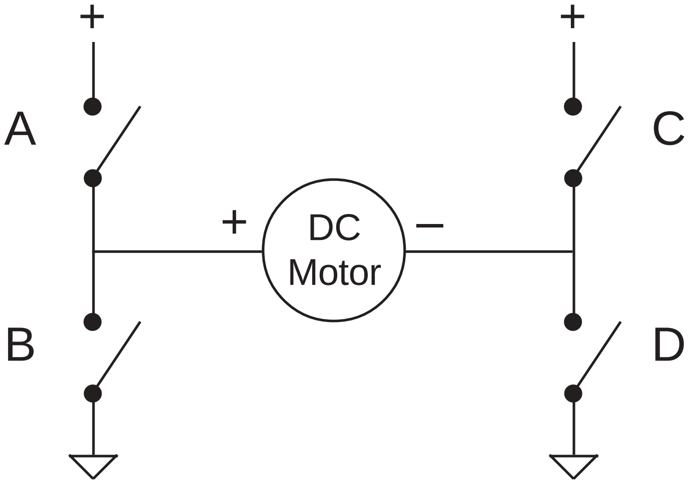


Figure 6-1: The H-Bridge Circuit

The H-bridge circuit runs the DC motor forward (by closing switches A and D) or reverse (by closing switches B and C). The driver chips (U8 and U9) use this configuration, except that transistors are used in place of switches. Since the motor may be connected in either polarity by this circuit, it is called "bipolar."

On a bipolar device, both sides of the device are switched by drivers, like the motor in this figure. On a unipolar device, one lead is connected to a supply, while the other lead is switched. Some stepping motors require bipolar drivers, others are designed to accept a unipolar supply.

Separate **+SUPPLY** and **GROUND** connections are provided for each driver chip. Be sure to connect these to an adequate power source. The voltage drops across the high-side and low-side drivers are approximately 1.2 volts at 1 amp (2.4 volts in a bipolar configuration). You may need to provide a little more supply voltage to get full speed operation out of your DC motor.

Figure 6-2 shows the connections for driving two DC motors. Note how the capacitors are installed across the motor and the battery connections to the driver. The Experimenter can directly drive DC motors with supply voltages from 4.5 to 36 volts and currents up to 1 amp.

When a DC motor is under a heavy load, its current draw increases. The current peaks when the motor is stalled. Exceeding the driver's maximum current rating, even briefly, will destroy the driver. So, if you are driving a motor that may be subject to stalling and is operating near the current rating, you may wish to use the driver circuit given in **Figure 6-8**. You will only need half of the circuit, as it is shown driving the two coils of a bipolar stepping motor, and there is only one set of coil connections to a DC motor. This circuit will safely drive several amps, with brief loads of up to five amps. It is discussed more fully in the section Driving Higher Current Motors.

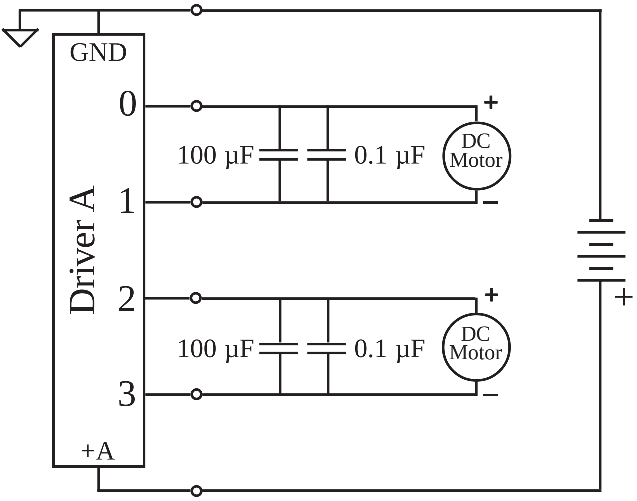


Figure 6-2: Directly Driving Two DC Motors
This shows the power supply and motor connections required to drive two DC motors. The Experimenter can directly drive motors with supply voltages from 4.5 to 36 volts and coil currents up to 1 amp. You must not exceed the voltage or current ratings, as the driver chip would be very quickly destroyed.

Stepping Motor Coil Configurations

Stepping motors differ in physical size from a fraction of an ounce to many pounds. They are constructed in various ways, and have a wide range of voltage and current ratings. A single step for some types will be less than a degree, for others fifteen degrees or more. Electronic surplus stores often have a variety of stepping motors available. Usually motors will be labelled with the number of *steps per revolution*, although this may instead be expressed as the number of *degrees per step*. Motors are also labelled with a coil voltage and current rating, although some list voltage and coil resistance. The Experimenter can drive most common stepping motors, either directly or with additional buffering for high-current motors.

There are four common coil configurations for stepping motors (**Figure 6-3**). You may find stepping motors with two coils, three coils, two coils with center taps, or four separate coils. If you get a stepping motor without a data sheet, you will need to use an ohm meter and a little experimentation to determine the coil configuration. Once you know the coil configuration you can connect the motor to the Experimenter's drivers. The numbers in **Figure 6-3** correspond to DRIVER-A outputs. You can also connect the motor to DRIVER-B in the same order. For unipolar stepping motors, the wires labelled **Supply** must be connected to the power supply. Since the driver outputs default to high when the Experimenter is powered-on, the **Supply** wires should be connected to the positive side of the motor supply. Power should not be applied to the motor when the Experimenter is off, so it is convenient to use SWP (SWitched Power).

You use the **H**-command to control stepping motors. You must tell the Experimenter the coil configuration of the stepping motor so that it can provide the proper sequencing of outputs. **Table 5-2** gives the drive types available for various coil configurations and supply connections.

Most stepping motors can be successfully driven with several different drive types. For example, a two-coil bipolar stepping motor can be driven with types 8, 9, 10, or 11. The difference between types 8 and 9 for a bipolar motor is the direction that the motor will turn. The same is true of types 10 and 11. Types 8 and 9 run the motor in full-step mode. Types 10 and 11 double the resolution of the stepping motor by running in half-step mode.

A two- or four-coil unipolar stepping motor with its common leads connected to the + supply can be driven with types 7, 9, or 11. Type 7 (one-phase drive) minimizes power consumption since only one coil is active at any time, though this reduces torque. Type 9 (two-phase drive) runs two coils simultaneously, providing somewhat more torque but doubling power consumption. Type 11 (half-step drive) alternates between driving two coils and one coil, doubling the resolution.

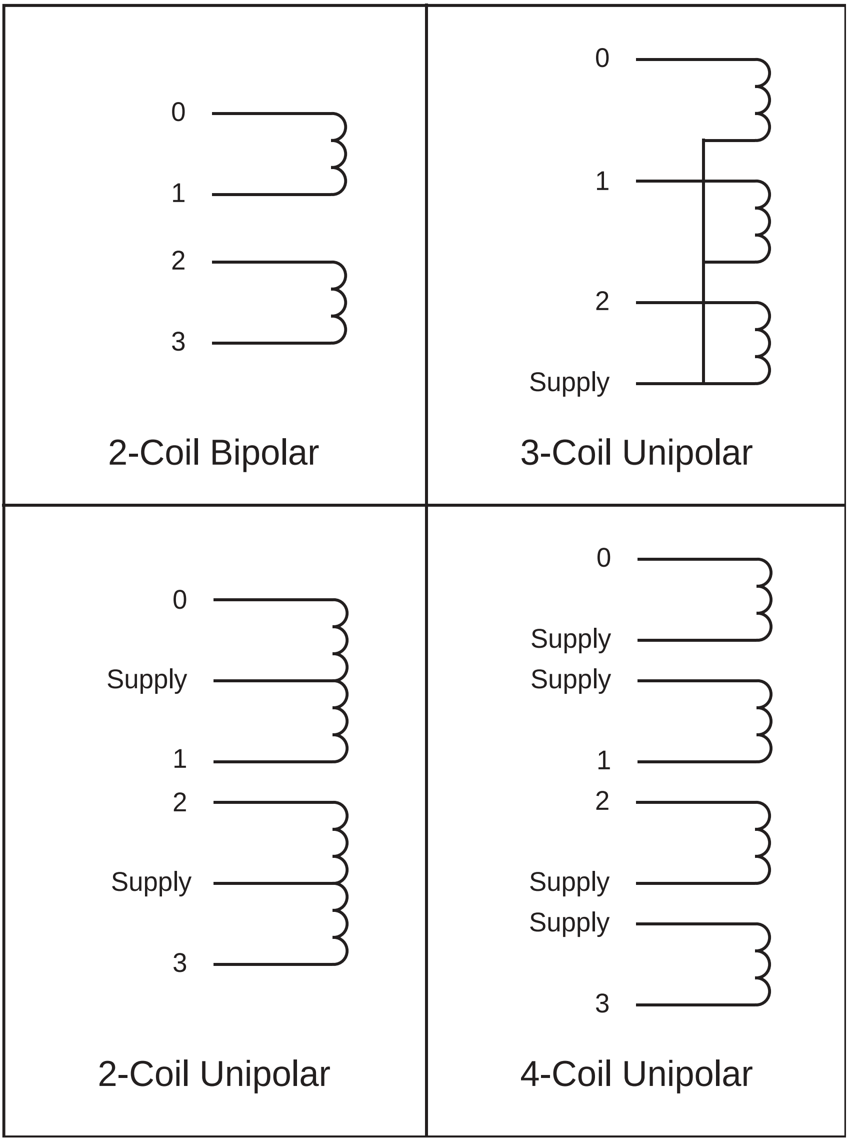


Figure 6-3: Common Stepping Motor Coil Configurations
Unlike ordinary DC motors which use internal brushes to switch current between coils, stepping motors bring out wires from each coil to be switched electronically. There are many ways of configuring coils in stepping motors. This figure shows the common ones. Using an ohm-meter you can figure out which leads connect to which coils, and learn the coil configuration for any motor you may find. The Experimenter can drive all common types of motors.

A Typical Stepping Motor

The MOT-S2002 (**Figure 6-4**) is a high quality, ball bearing stepping motor. It features a full-step of 1.8° (200 steps per revolution), and can be half-stepped by the Experimenter at 0.9° (400 steps per revolution). This motor is rated at a maximum coil current of 400 mA at 24 volts, making it especially easy for the Experimenter to run the motor at its maximum power. When lightly loaded the MOT-S2002 can be driven from a supply voltage as low as 5 volts. **Figure 6-5** shows how to connect the motor to the Experimenter.

The motor weighs 12 ounces, with an overall length of about 2.3 inches. Maximum torque is roughly 40 inch-ounces. When any stepping motor steps very rapidly, the inductance of the coils restricts the power going to the coils (**Figure 6-6**). This motor cannot run faster than 2 mS per step.

To drive the MOT-S2002 in full step mode use drive type 8, for half step mode use drive type 10. For example, the command `H 0 1 100 15 8` would drive a motor connected to DRIVER-A (0), in the forward direction (1), for 100 steps (100), at a rate of 15 milliseconds per step (15), in full step mode (8).

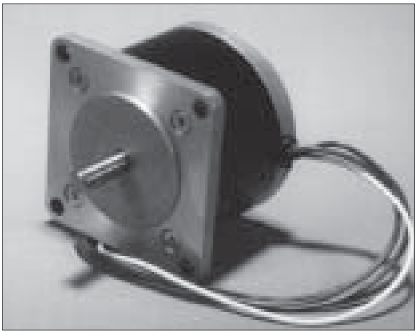


Figure 6-4: A Stepping Motor
The MOT-S2002 stepping motor is well suited to control by the Experimenter.

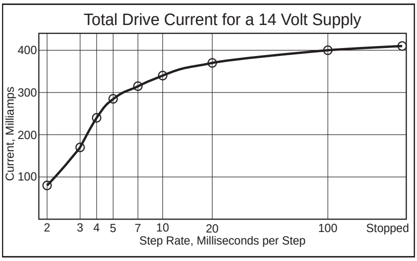


Figure 6-6: Current vs Step Rate
Shorter step times result in less current through the motor. Motor torque is similarly reduced.

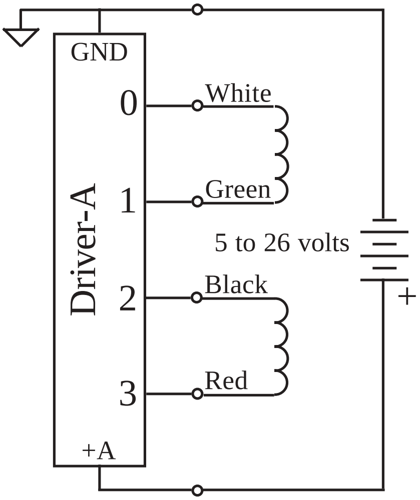


Figure 6-5: Directly Driving a Bipolar Stepping Motor
This shows the power supply and coil connections required to drive a bipolar stepping motor, our MOT-S2002. The Experimenter can directly drive stepping motors with a supply voltage of 4.5 to 36 VDC and coil currents up to 1 amp.

Limiting Current with Series Resistors

It is often useful to add resistors in series with the coils in a stepping motor. You may wish to power a six volt stepping motor from a 12 volt battery, either for convenience or to improve the high speed torque of the motor (the higher voltage helps overcome the increasing coil impedance due to its inductance). Or you may wish to reduce the motor's current draw to allow the Experimenter to drive it safely within the 1 amp maximum driver current rating (circuits for higher current drive are given in the next section).

Some of the motor supply voltage is lost in the drivers, about 1.2 volts. Since an H-bridge configuration drives both high and low sides of the coils, about 2.4 volts is lost when driving bipolar motors.

Sometimes the motor's label tells the coil resistance. Sometimes the label specifies maximum coil voltage and current instead of resistance, so you can calculate the coil resistance from Ohm's Law (**Equation 6-1**). Or you can measure the resistance with an ohmmeter.

Eq. 6-1: $R_{coil} = E_{coil\text{-}max} / I_{coil\text{-}max}$

R_{coil} the internal resistance of the coil
$E_{coil\text{-}max}$ the maximum rated coil voltage
$I_{coil\text{-}max}$ the maximum rated coil current

Now, using another form of Ohm's law, we can calculate the additional series resistance required:

Eq. 6-2: $R_{series} = (E_{supply} - E_{drop}) / I_{chosen} - R_{coil}$

R_{series} the additional resistance required
E_{supply} the power supply to the driver.
E_{drop} the voltage drop across the driver, about 1.2 volts unipolar, 2.4 volts bipolar.
I_{chosen} the new coil drive current.
R_{coil} the internal resistance of the coil.

Now we need to calculate the power that will be dissipated in the resistors:

Eq. 6-3: $P_{resistor} = (I_{chosen})^2 * R_{series}$

$P_{resistor}$ is the power dissipated in the resistor.
I_{chosen} the new coil drive current.
R_{series} the additional resistance required

Choose the standard resistor closest to the value calculated in **Equation 6-2**. A good rule of thumb is to get resistors rated for twice the power calculated in **Equation 6-3**. Even so, they will get hot! You may also use series or parallel combinations of resistors to get the desired values. **Figure 6-7** shows a unipolar stepping motor with series resistors directly connected to the Experimenter.

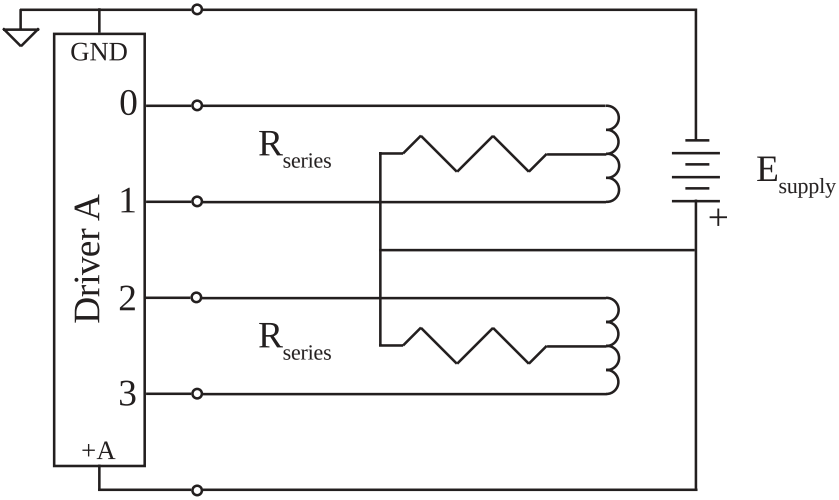


Figure 6-7: Driving a Unipolar Stepping Motor through Series Resistors
This shows how series resistors may be installed to limit coil current when directly driving a unipolar stepping motor.

Driving Higher Current Motors

Sometimes you need more power than the 1 amp drivers can give. If so, you can build a high-current driver. With relatively cheap power transistors, you can greatly boost the power drive capability of the Experimenter. **Figure 6-8** shows a driver capable of several amps connected to a bipolar stepping motor. The transistors are connected in an emitter-follower configuration. This configuration provides current gain, with the output voltage following the input voltage. The diodes dissipate the inductive surge produced when a coil is switched off. Typically power transistors have current gains of 20 or better. So by using more powerful transistors than the TIP41 and TIP42 (and bigger diodes), you could drive motors rated up to 20 amps. Calculate the series resistor as shown in the previous section, but allow for 3 volts of drop across the power transistors (E_{drop} = 3).

When driving large currents, lots of power gets dissipated as heat in the transistors. It is essential that adequate (big) heatsinks be provided for them. If the transistors get too hot, they will fail. A little heatsink grease between the transistor and heatsink improves the heat conduction out of the transistor, helping to keep the semiconductor junctions a little cooler. And for really big loads, a fan greatly increases the efficiency of a heatsink.

The tab on the TIP41 and TIP42 transistors is connected to the collector. The collector on the TIP41 is connected to the positive supply. The collector on the

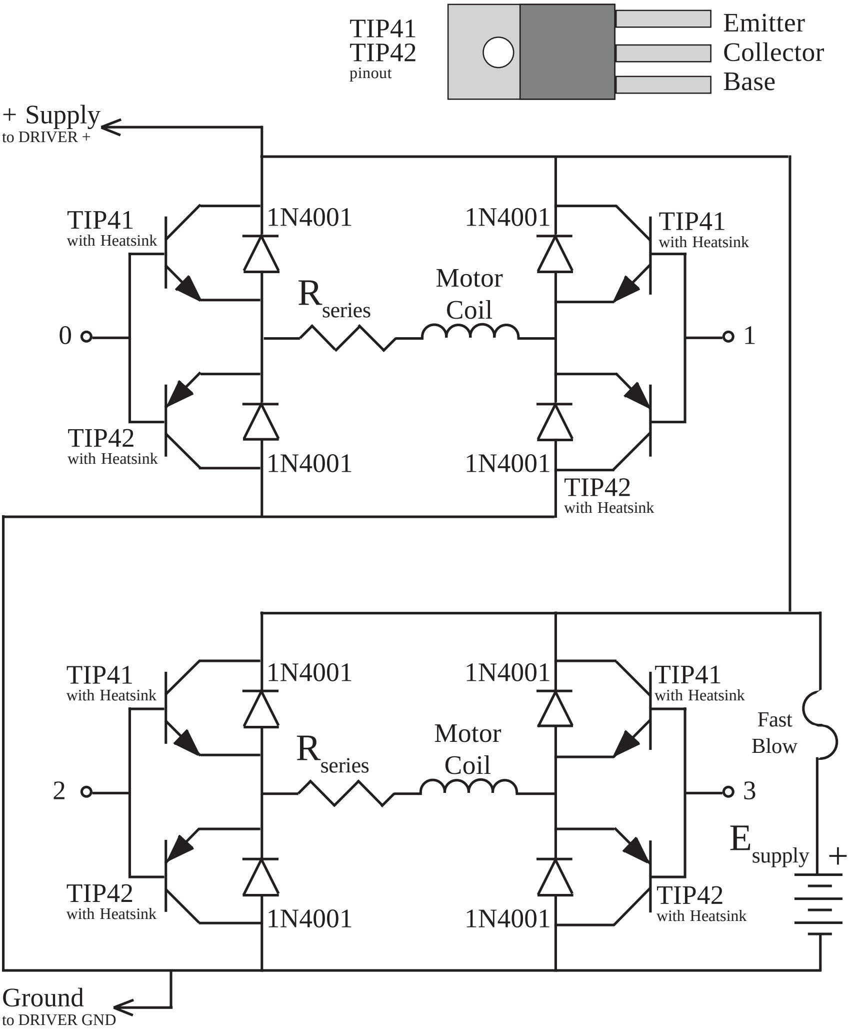


Figure 6-8: A High-Current Bipolar Driver

This simple circuit may be added to the driver outputs on the Experimenter. It will drive loads of several amps with up to 36 volts on the supply. Each active transistor dissipates about 1.5 watts in heat for each amp of current drawn, so good heatsinks are required. Add heatsinks to all eight of these transistors.

Warnings: The collector shorts to the tab on these transistors! Do not let the heatsinks on the NPN transistors touch those on the PNP transistors or there will be a short circuit from the supply to ground. Also, be sure to send the +Supply and Ground to the driver on the Experimenter as shown.

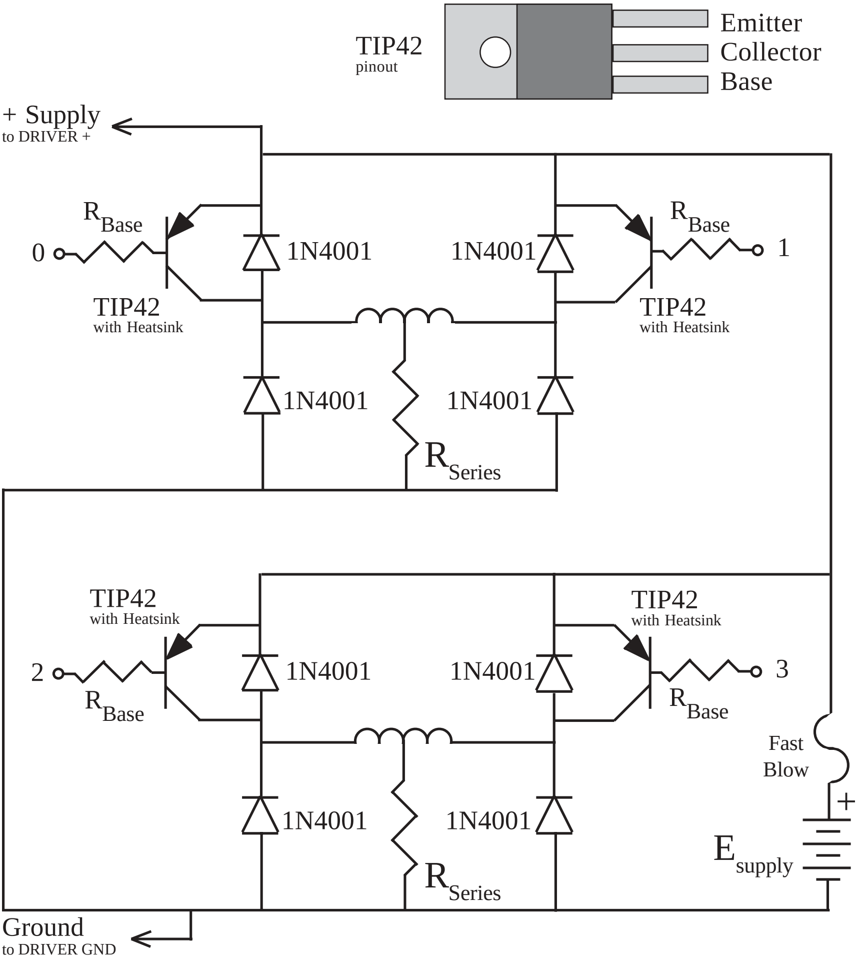


Figure 6-9: A High-Current Inverting Unipolar Driver

This circuit differs from the bipolar driver of ***Figure 6-8*** *in that the transistors are driven to saturation. This results in a lower voltage drop across the transistors than in the emitter-follower configuration. You will need to calculate values for the base resistors on each transistor. A base current of about 5% of the collector current is a good starting point.*

Good heatsinks are required on all four of these transistors. Since the collectors on these transistors short to their mounting tabs, use separate heat sinks for each transistor.

You must also connect the positive supply and ground to the inputs on the driver IC. Do not apply power to this circuit without providing power to the Experimenter.

TIP42 is connected to the negative supply. You may use individual heatsinks on each transistor. Or you may use the same heatsink for the four TIP41 transistors and another heatsink for the four TIP42 transistors. But do not use the same heatsink for all eight transistors as this would short the positive supply to the negative supply.

Unipolar motors do not require quite as complex a driver as bipolar motors. **Figure 6-9** shows a high-current inverting unipolar driver. In this circuit, the transistors are driven to saturation. This results in lower voltage drop and lower power loss. Since this is an inverting driver, you should choose a configuration ***type*** for a +supply common unipolar motor, even though the common lead is connected to ground (see **Table 5-2**).

You must calculate values for the base and series resistors. Choose base resistors that will limit the base current to about 10% of the desired collector current. Be sure to use appropriately power rated resistors and adequate heatsinks on the transistors. Use separate heatsinks for each transistor.

Applications of Computer Controlled Motors

We have looked at controlling DC motors and stepping motors with the Experimenter. What can you build with them? Well, many high school and college students have built robots controlled by the Experimenter. Gear reduction DC motors make good drive motors for wheels, and stepping motors work well for positioning sensors.

Some of our customers have automated instruments and machinery using stepping motors, like telescopes and milling machines. Or adjust the position of a prism in a spectrascope using a gear reduction stepping motor.

Remember that the Experimenter provides measurement capabilities that can complement motor control. For example, you can use a potentiometer read by an analog input to provide feedback on the motor position. Or use a magnetic switch or hall effect sensor sending a signal to a counter/timer input to count shaft rotations.

7

Using Analog Inputs

Many sensors output analog voltages. With its eight analog voltage measurement inputs, the Experimenter provides an easy way to interface these sensors to your computer. This chapter will give you some hints on using these inputs.

Scaling and Filtering

The two basic operations we must perform on an analog signal are scaling and filtering. Scaling is adjusting the range of the signal to match the Experimenter's input range of 0 to 5.12 volts. Filtering removes noise from the signal.

Scaling is easily accomplished by using an operational amplifier (opamp) and resistors to set the gain. The Texas Instruments TLC2274C quad opamp is a very good choice (**Figure 7-1**). This amplifier has the remarkable property that its output will swing virtually from one supply rail to the other. By supplying the amplifier from the 5.12 volt analog supply, the Experimenter's entire analog input range can be used. The TLC2274C has very low input bias current of 1 pA (typical) and low input offset of 300 µA (typical). Its input voltage range is from -0.3 to 4.2 volts (typical). We use this part in the Observer Meteorological Station, and it is also available separately from Fascinating Electronics, Inc.

There are many possible opamp circuits. A simple positive gain amplifier circuit is shown in **Figure 7-2**. The circuit gain is given by **Equation 7-1**, and the formula for calculating R2 is given by **Equation 7-2**. For example, if you wanted an amplifier with a gain of 100 and R1 value of 1KΩ, the value needed for R2 would be 99 KΩ. You may choose the nearest standard value (100 KΩ). For precision applications, use 1% tolerance resistors.

This circuit may be used to scale the output of a temperature sensor or to amplify the signal from a photodetector (photo diode or transistor). Other opamp circuits subtract offset voltages from signals or amplify differential signals (for devices like strain gauges).

Eq. 7-1: Gain = 1 + R2 / R1

Eq. 7-2: R2 = R1 * (Gain -1)

The Experimenter's analog inputs may produce false readings if the voltage varies at a slew rate of 10 volts/millisecond or greater. This can often be traced to high frequency electrical noise. Either one or both capacitors, C1 and C2, may be used in the circuit (**Figure 7-2**) to filter high frequency electrical noise from the signal. Try values of about 0.1 µF for these capacitors.

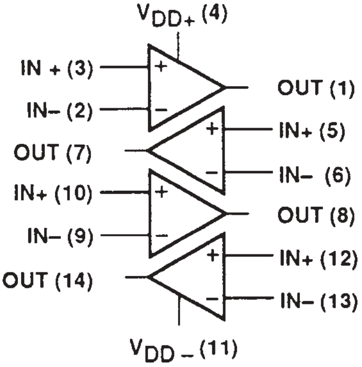


Figure 7-1: TLC2274C Quad Rail-To-Rail Opamp Pinout
The TLC2274C is very useful for scaling analog signals. It features rail-to-rail output swing, very low input current, and low input offset voltage.

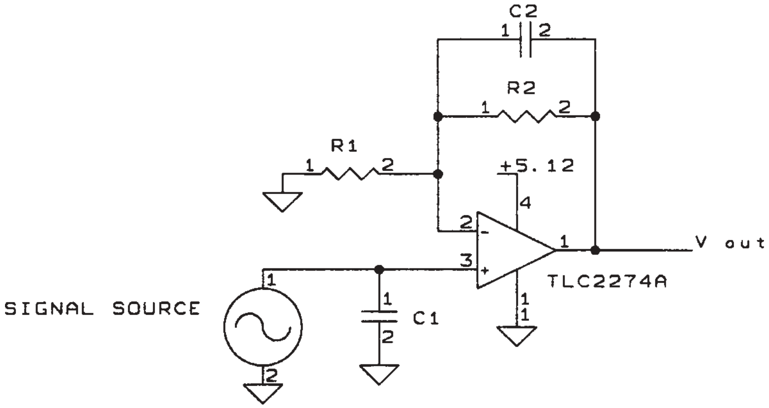


Figure 7-2: A Postive Gain Amplifier
By using this simple circuit, you can amplify a signal by even very large gain factors. R1 and R2 set the gain, C1 and C2 may be used to filter electrical noise.

Dual Wiper Potentiometer

In the ANALOG command section (Chapter 4: Command Tutorial), a potentiometer (pot) was used to create an analog voltage. As the shaft of the pot rotates, a varying voltage on the wiper may be measured.

As an angle sensor, most pots have two characteristics that limit their usefulness. They have mechanical stops that limit rotation, usually to about 300 degrees. Pots that do not have mechanical stops are able to rotate freely, but they have a "dead zone" where the wiper is not touching the voltage divider.

A pot that overcomes both of these problems is available for your experimentation from Fascinating Electronics, Inc. Used primarily in the wind vane (part of the Observer Meteorological Station), a special pot with two wipers is available. The wipers are offset 180° from each other, so that one wiper is always making contact with the voltage divider. A circuit for using the pot is shown (**Figure 7-3**). The pull-down resistors, R1 and R2, draw the output voltage to ground when a wiper leaves the voltage divider. C1 and C2 remove any electrical noise that may be produced as the wipers slide along the voltage divider.

The program (**Listing 7-1**) converts the Experimenter's voltage measurements into angle values and displays them on the screen. This code must be inserted into the program TEMPLATE.BAS (**Listing 3-2**) in order to run.

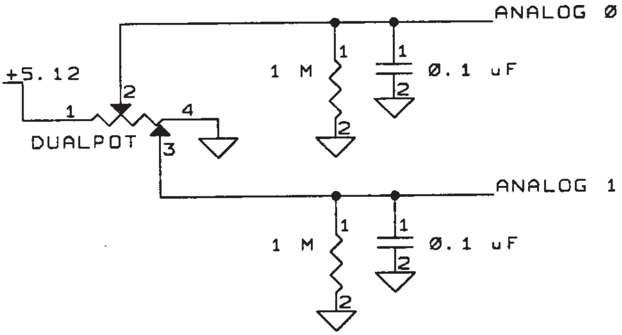


Figure 7-3: Dual Wiper Pot Circuit
The dual wiper pot provides continuous rotation and continuous resolution for angle measurements. Purchased for the wind direction sensor (part of the Weather Monitoring Station), the dual wiper pot has many other applications—robotics, machinery, measurement instruments, and general experimentation.

The resistors pull the wiper voltage to ground when the wiper moves off of the voltage divider. The capacitors filter electrical noise from the signals.

```
CLS
delta = 700                         'delta is the overlap between wipers
DO WHILE UCASE$(INKEY$) <> “Q”
  PRINT #1, “A 0″
  INPUT #1, w1                       'voltage on wiper 1
  PRINT #1, “A 1″
  INPUT #1, w2                       'voltage on wiper 2
  LOCATE 1, 1
  PRINT USING "####  ####"; w1; w2   'print raw wiper voltages

  ' Decide if we should update the delta value.
  IF w1 > 5120 - 2 * delta AND w1 < 4920 AND w2 < 2 * delta AND w2 > 200 THEN
    delta = (9 * delta + (w2 + 5120 - w1) / 2) / 10
    PRINT "w1 > 5120 - 2 * delta AND w1 < 4920 AND w2 < 2 * delta AND w2 > 200"
  ELSEIF w2 > 5120 - 2 * delta AND w2 < 4920 AND w1 < 2 * delta AND w1 > 200 THEN
    delta = (9 * delta + (w1 + 5120 - w2) / 2) / 10
    PRINT "w2 > 5120 - 2 * delta AND w2 < 4920 AND w1 < 2 * delta AND w1 > 200"
  ELSE
    PRINT "No change in delta value.                                          "
  END IF

  ' Decide if we have a valid bearing off of one of the wipers.
  IF (w1 > delta AND w1 < 5120 - delta) AND (w2 > 5120 - delta OR w2 < delta) THEN
    PRINT "w1 yields valid bearing."
    bearing = 180 * (w1 - delta) / (5120 - 2 * delta)
  ELSEIF (w2 > delta AND w2 < 5120 - delta) AND (w1 > 5120 - delta OR w1 < delta)
THEN
    PRINT "w2 yields valid bearing."
    bearing = 180 + 180 * (w2 - delta) / (5120 - 2 * delta)
  ELSE
    PRINT "Do Not Update Bearing.  "
  END IF

  ' Print current results for bearing and delta.
  PRINT USING “Bearing = ###.#   Delta = ####.#″; bearing; delta
LOOP
```

Listing 7-1: POT-CODE.BAS, Converts Dual Wiper Potentiometer Voltage Measurements to Angles

*This program reads the wiper voltages on a dual wiper pot connected to analog inputs 0 and 1, and converts the measured voltages to a bearing in degrees. This code must be inserted in TEMPLATE.BAS program (**Listing 3-2**) in order to run.*

*The program constantly measures and crosschecks the voltages on the wipers. If one voltage is midrange, the other should be about 0. As one voltage approaches an extreme, the other should be near the other extreme. The variable **delta** is a measure of the amount of overlap where both wipers are on the resistive band while they are near the extremes. When both wipers are solidly on the resistive material the program updates the value of **delta**, giving a more accurate reading for your particular potentiometer.*

!

Prologue to the Applications

The following chapters show a few examples of projects that have been built using the Experimenter. Fascinating Electronics Inc. stocks cases and other accessories, motors, ultrasonic ranging components, temperature, pressure, and humidity sensors, and many complete kits related to the Experimenter. Some projects require much larger programs than can be included in this manual, so sample programs are available on our application disks at very reasonable prices. With appropriate sensors and motors the Experimenter can link your computer to the physical world in many ways.

If you have built something with the Experimenter that others may find interesting or useful, please let us know. We would like to include your application in our manual or in other publications. If you have developed some neat software you wouldn’t mind sharing with others, we would like to add it to the application disks.

We hope the Experimenter will more than live up to your expectations. A world of new computer applications awaits your creativity. Building devices that link to the physical world is challenging, fascinating, and educational. Take the following chapters as a starting point. But remember that applications for the Experimenter are truly only limited by your imagination.

A

An Ultrasonic RADAR

This is a fun project that puts a live-action RADAR display on the screen of your computer. We've shown this project to a wide range of people—from kids to seniors, from the computer illiterate to operating system software consulting engineers. And so far, everyone has been impressed.

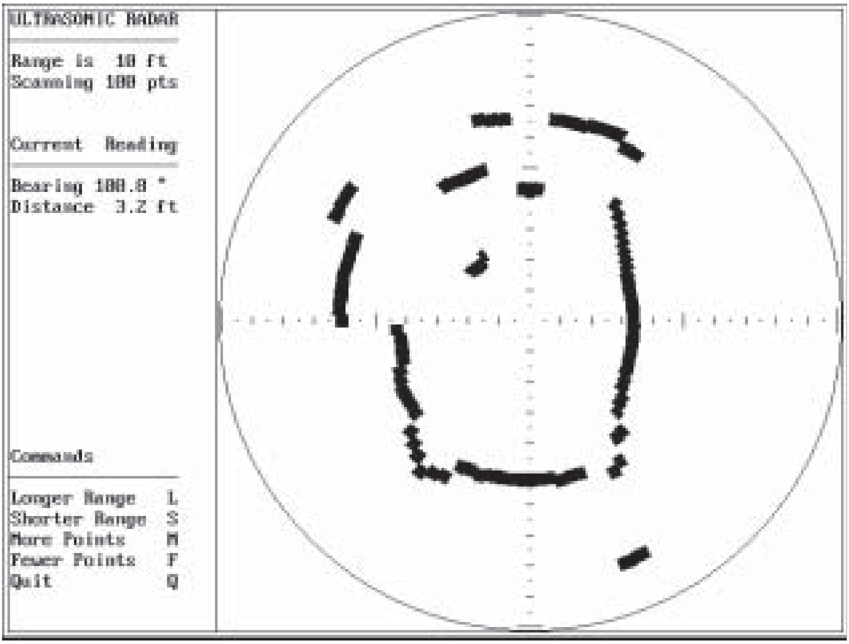
The RADAR uses an ultrasonic rangefinder to measure the distance to surrounding objects. A stepping motor rotates the rangefinder, giving 360° coverage. Your computer paints a graphic display of the surroundings. VGA and EGA systems use color for a better-looking display. CGA and Macintosh systems and are supported too. **Figure A-1** shows the RADAR on a VGA display.

The Ultrasonic Rangefinder

We use the same ultrasonic rangefinder that is used on Polaroid cameras. The rangefinder emits a brief pulse of high frequency sound. Any object hit by the sound produces an echo. The distance to the object is measured by very accurately timing from the pulse to the echo.

The Polaroid rangefinder is made up of two parts, a transducer and a driver board. The transducer (**Figure A-2**) acts as both a speaker and a microphone. It emits the sound pulse and listens for the echo. The driver board (**Figures A-3** and **A-4**) provides the high voltages required to run the transducer, sensitive amplifiers for echo detection, and control logic.

The Experimenter controls the driver board, measures the time to the echo, controls the stepping motor that rotates the transducer, and communicates with your computer. **Figure A-5** shows the connections between the transducer, driver board, and the Experimenter.



```
ULTRASONIC RADAR

Range is  10 ft
Scanning 100 pts


Current  Reading

Bearing 100.8 °
Distance  3.2 ft




Commands

Longer Range   L
Shorter Range  S
More Points    M
Fewer Points   F
Quit           Q
```

Figure A-1: The RADAR Display
This is how the radar display looks on a VGA display, minus color enhancements. The software will also work on EGA and CGA displays, though at lower resolution. Both the range of the display and the number of points in the scan can be varied on-the-fly. Available on Application Disk #1.

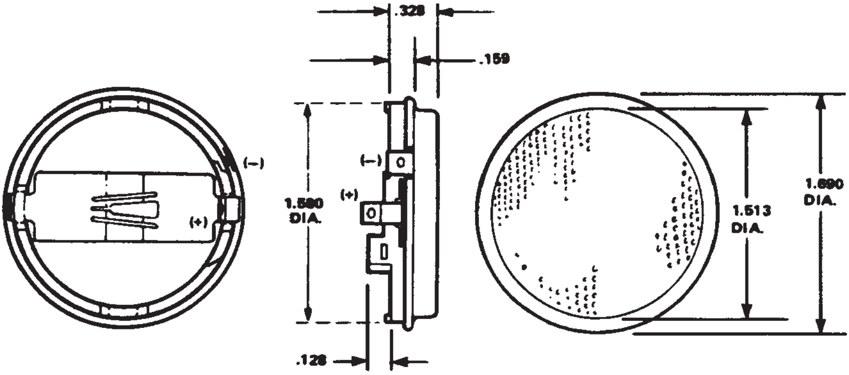
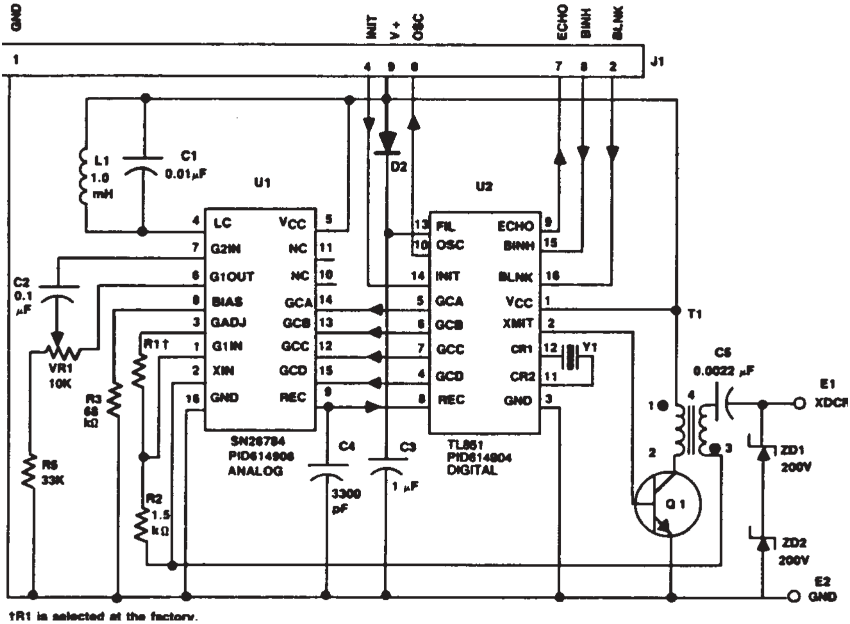


Figure A-2: The Ultrasonic Transducer
The ultrasonic transducer acts as both a speaker and microphone. The (+) terminal connects to the E1 output of the driver board, the (–) terminal to E2. Dimensions are given in inches.



†R1 is selected at the factory.

Figure A-3: Driver Board Schematic
The driver board provides the high voltages required to run the transducer, sensitive amplifiers for echo detection, and control logic.

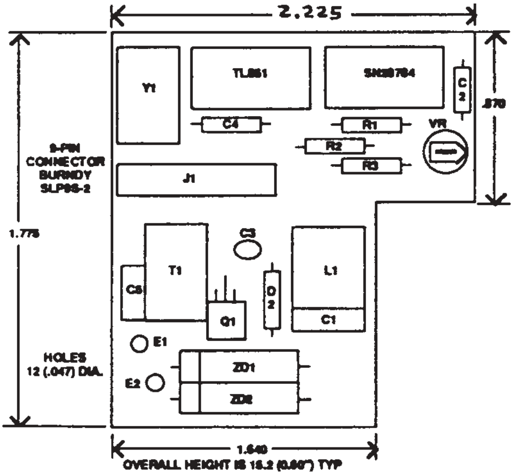


Figure A-4: Driver Board Component Location Diagram
This shows the location of components on the driver board. Dimensions are in inches.

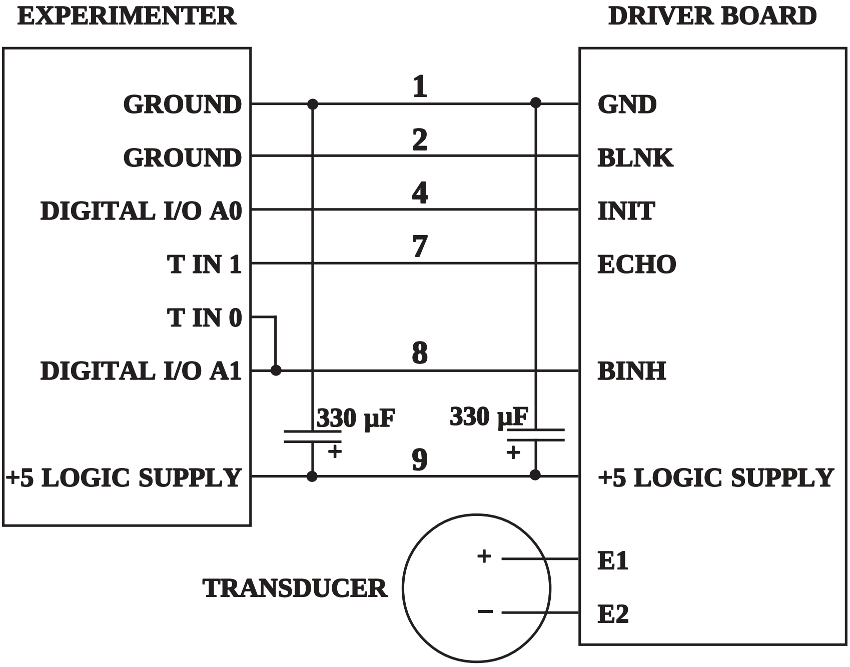


Figure A-5: RADAR Schematic
This shows the connections between the Experimenter, driver board, and ultrasonic transducter. The Experimenter and driver board link through a supplied nine pin flex cable.

The driver board interfaces through a nine conductor flex cable. The connector for this cable can be installed in the X1 connector mounting area on the Experimenter. The cable has a black stripe on it to indicate pin 1. Do not substitute for this cable! We have found that crosstalk and voltage drop in substitute cables is the major cause of these projects not working.

After you wire the connections and install the flex cable, but before applying power, verify your work with an ohmmeter. Verify that each signal on the Experimenter shown in **Figure A-5** goes to the appropriate pin on U2 in **Figure A-3**.

The driver board's power requirements are normally under 100 milliamps, but peak at about 2 amps during the transmit period. To handle this peak demand, two capacitors of 330 µF or greater must be added: one across the power inputs on the driver board, and one by the flex cable connector on the Experimenter.

Timing for the rangefinder is shown in **Figure A-6**. When the Experimenter sets INIT high, the driver board sends a burst of sixteen high-voltage drive pulses

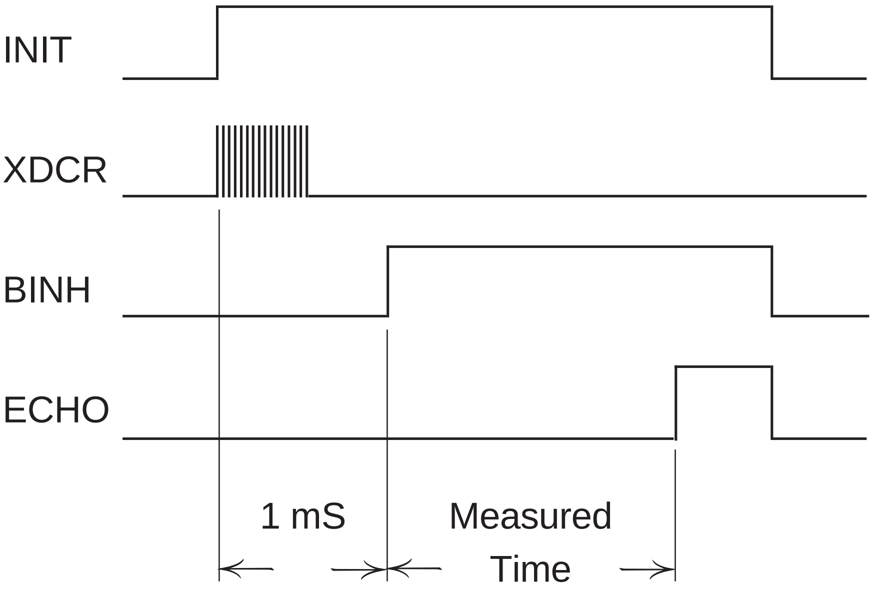


Figure A-6: Timing Diagram
The Experimenter sets INIT high, causing the driver board to send pulses to the transducer. The Experimenter waits one millisecond for the pulse transmit and for the transducer to settle down. The Experimenter sets BINH high to start the driver listening for an echo. If an echo is heard, the driver board sets ECHO high. The Experimenter measures the time from BINH going high to ECHO going high.

to the transducer. You may be able to hear this burst as a click. It take about 360 microseconds to transmit the pulses. The Experimenter waits for one millisecond to allow time for pulse transmit and for the transducer to settle down. The Experimenter then sets BINH high to start the driver listening for an echo. If an echo is detected the driver board sets ECHO high. The Experimenter measures the time from BINH going high to ECHO going high. Then the Experimenter resets INIT and BINH low. If no echo is detected in a reasonable length of time, the Experimenter terminates the measurement.

The measured time is sent to your computer, which calculates the distance based on the speed of sound. The speed of sound is approximately 1100 feet per second, but varies with temperature, humidity, and barometric pressure. You can calibrate your rangefinder by placing an object a carefully measured distance in front of the transducer and adjusting the speed of sound parameter in the software. This is discussed later in the section on software. Since the Experimenter measures time with 10 microsecond resolution, the rangefinder can measure distance with about 0.07 inches resolution!

Stepping Motor

This project uses a stepping motor to rotate the ultrasonic transducer. A slip ring maintains electrical connection with the transducer. This lets the RADAR scan in precise increments, with full 360° coverage. You can use a wide variety of stepping motors. The MOT-S2002 bipolar stepping motor, available from Fascinating Electronics Inc., works very well. If you provide your own motor be aware that some of the smallest stepping motors may have trouble reliably pointing the transducer, due to their limited ability to overcome the inertia of the rotating mass. But a stepping motor like the MOT-S2002 will have more than enough torque and can be run at a fraction of its rated power.

The beam from the rangefinder is roughly 10° wide (at its -10 dB points), so stepping less than about 5° does not necessarily give you much additional resolution. The software lets you select the number of motor steps between readings, so you can easily adjust for the optimum resolution. If you use a different motor, the RADAR program must be told the drive type and the number of steps per revolution the motor will make with that drive type. This is covered later, in the RADAR Software section.

Mechanical Assembly

The ultrasonic transducer must be mounted on the stepping motor shaft, and electrical connections between the transducer and the driver board must be made. **Figure A-7** is a photograph of one way of doing this.

This method requires a brass tube of ¼" inside diameter, a short section of brass tube that will fit around the first tube with electrical insulation between them, a plastic cap to surround the transducer, and an alternator brush assembly to complete the slip ring. You can find these parts at hobby, hardware, and automotive parts stores. There are many types of alternator brush assemblies available. Look for one that would be easy to mount on the stepping motor, with few features that will get in the away.

Drill a hole in the side of a plastic cap and glue the ¼" ID brass tube in it. The driver board has two wires with clips. Remove these wires from the driver board and clip them on the transducer. Solder the wire from the negative (-) terminal of the transducer (see **Figure A-2**) to the brass tube. Run the positive wire from the transducer, through the tube, and out through a hole drilled in the side of the tube. Mount the transducer in the lid by surrounding it with foam weather stripping.

Secure a short ring of the slightly larger diameter brass tubing around the long brass tube, but insulated from it with electrical tape. Solder the positive wire to this ring. Slip the tube over the motor shaft and secure it with epoxy.

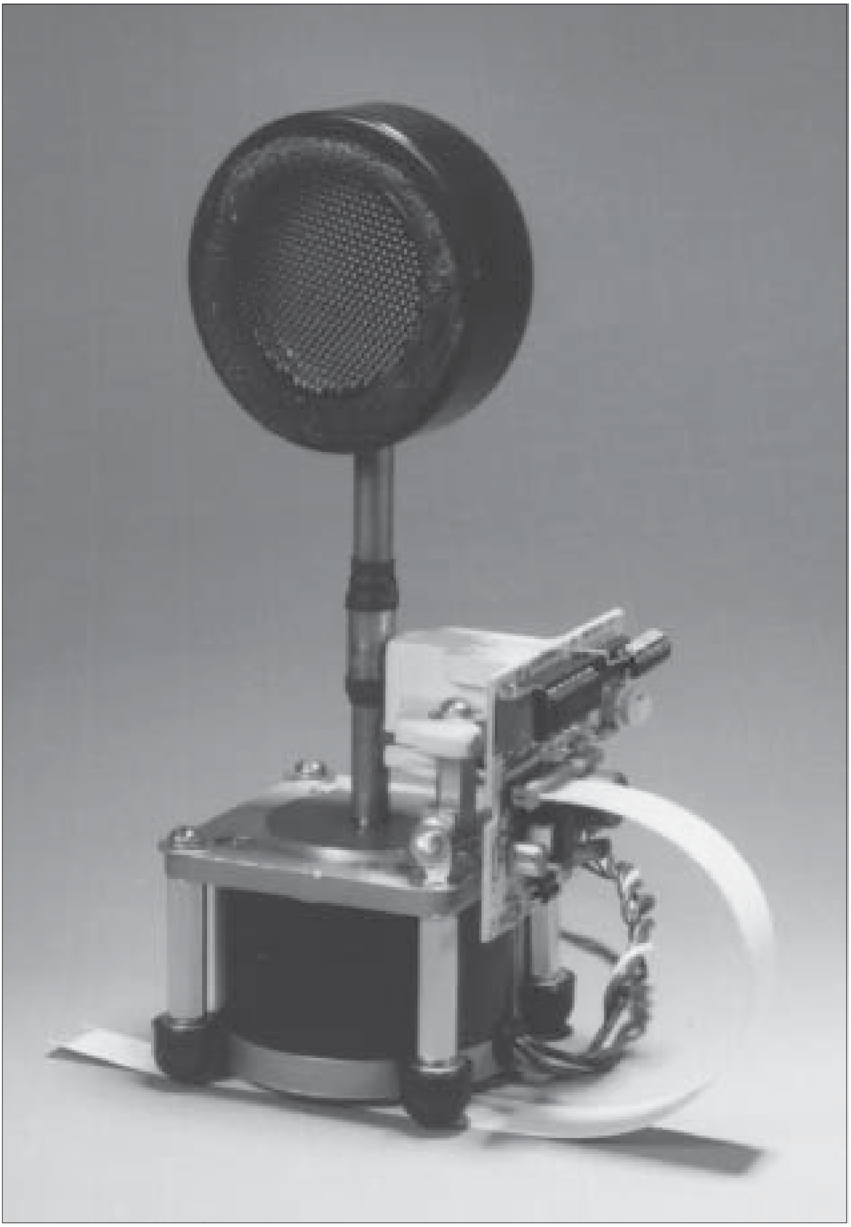


Figure A-7: RADAR Mechanical Assembly
A stepping motor rotates the ultrasonic transducer to scan the surroundings for the RADAR display. Alternater brushes are used for a slip-ring, allowing the transducer to rotate continuously, but still maintain electrical contact with the driver board.

Mount the alternator brush assembly on the stepping motor so that one brush contacts the ring, and the other contacts the tube. You may have to add spacers, and cut, file or sand off features from the brush assembly to get it to fit on the motor. Then connect terminals E1 and E2 from the driver board to the brushes. E1 must connect to the ring, E2 to the tube. Verify that there are no opens or shorts by using an ohmmeter.

RADAR Software

This section includes a short program that measures distances with the rangefinder. But because the RADAR display program is quite long, it is not included in this book. You wouldn't want to type it in anyway! It is on Application Disk #1, along with the programs in this book. Windows and Macintosh versions are also available. If you are using a different computer (Atari, Commodore, Cray, etc.), you will need to translate the program to the dialect of BASIC used by your machine.

Listing A-1 is a simple distance measurement program. It pulses the rangefinder several times per second and reports the distance measured with 0.07 inch resolution. You can calibrate the unit to the speed of sound for your local conditions by placing a flat object, like a book, a carefully measured distance from the transducer. When you run the program:

If the reported distance is **more** than the actual distance, decrease the speed of sound parameter. Pressing the 1 key decreases the parameter 10 feet-per-second (fps), the 2 key decreases 1 fps.

If the reported distance is **less** than the actual distance, increase the speed of sound parameter. Pressing the 4 key increases 10 fps, the 3 key increases 1 fps.

You may give the value you determined for the speed of sound to the RADAR program to make it more accurate. You also need to tell the RADAR program the drive type for the motor you are using and the number of steps per revolution it will make at that drive type, which COM port and baud rate to use, and whether to display in black-and-white or color. All of this information is set in a file called RADAR.DAT, included on Application Disk #1. You may edit the file with any text editor, then save it in ASCII format. The RADAR program will look for this file in the local directory, and set itself up accordingly.

The RADAR display program is easy to run. Pressing the L-key makes the displayed range longer (up to 35 feet). Pressing the S-key makes the range shorter (down to 5 feet). Pressing the M-key puts more points in the scan (up to the resolution of the motor at that drive type). Pressing the F-key puts fewer points in the scan (down to a minimum of at least 12, depending on motor resolution). The Windows and Macintosh versions use on-screen buttons for these functions.

```
' This program instructs the Experimenter to initiate an ultrasonic pulse,
' measure the return time, and convert that time nto a distance.

feetPerSec! = 1100            'Speed of sound, in ft/sec.
blanking! = .001              'Blanking time, in seconds.

' Print the header.
CLS
PRINT "Type Q to Quit."
LOCATE 8, 30, 0
PRINT " Ultrasonic  Ranging "
LOCATE 9, 30, 0
PRINT "---------------------"

PRINT #1, "D 3 139"           'DIGITAL I/O port A is output.
DO
   PRINT #1, "D 0 0 100 1 1 3"'Wait 100 mS, then make a pulse.
   PRINT #1, "C 0 11"         'Measure the echo delay.
   INPUT #1, time!            'Measurement units are 10's of uS.
   time! = time! / 100000 + blanking!
   PRINT #1, "D 0 0"          'Turn off ultrasonic rangefinder.

   ' Calculate and Print the results.
   distance! = time! * feetPerSec! / 2
   LOCATE 10, 30, 0
   PRINT USING "Time =  0.###### sec"; time!
   LOCATE 11, 30, 0
   PRINT USING "Distance = ##.## ft"; distance!
   LOCATE 12, 30, 0
   PRINT USING "Speed  =   #,### ft/s"; feetPerSec!

   ' Check for change of value for speed of sound.
   key$ = UCASE$(INKEY$)
   SELECT CASE key$
   CASE "1"
      feetPerSec! = feetPerSec! - 10
   CASE "2"
      feetPerSec! = feetPerSec! - 1
   CASE "3"
      feetPerSec! = feetPerSec! + 1
   CASE "4"
      feetPerSec! = feetPerSec! + 10
   CASE "Q"
      END
   END SELECT
LOOP
```

Listing A-1: DISTANCE.BAS, Distance Measurement Code

This simple program measures and displays the distance from the ultrasonic transducer to an object. Minimum distance is about half a foot. Maximum distance is about 35 feet. Resolution is 0.07 inches. You can use this program to determine the speed of sound for your current local conditions. In order to run, this code must be surrounded with the TEMPLATE.BAS program, ***Listing 3-2.***

B

Observer Meteorological Station

The weather affects everyone. But instead of simply being a source of rained-out picnics, the weather can be a source of endless fascination. Weather is a constantly changing panoply of winds, heat, pressure, and moisture. This project is a professional caliber meteorological station that can help you unlock the secrets of weather. These stations are in use in many countries for such diverse applications as greenhouse automation, Earth science projects, agricultural surveys, as well as basic meteorological data collection.

Rugged Meteorological Instruments

At Fascinating Electronics Inc., we designed a meteorological station that is both fun to build and to use. We developed kits for all of the standard meteorological measurements. They are available separately, or together in a package we call the Observer.

The Observer system (**Figure B-1**) includes an anemometer (**Figure B-2**), wind vane (**Figure B-3**), rain gauge (**Figure B-4**), thermometers (**Figure B-5**), hygrometers (**Figure B-6**), and a barometer (**Figure B-7**). The Experimenter is housed in a sturdy metal case, along with the barometer and signal conditioning electronics.

All instrument calibration is performed on your computer. This makes the instruments much simpler, easier to build, and reduces their cost while providing great accuracy.

Instruments are rugged, made with heavy gauge PVC, and with extensive use of stainless steel hardware. The anemometer and wind vane feature shielded steel ball bearings for measurement sensitivity and durability. The temperature

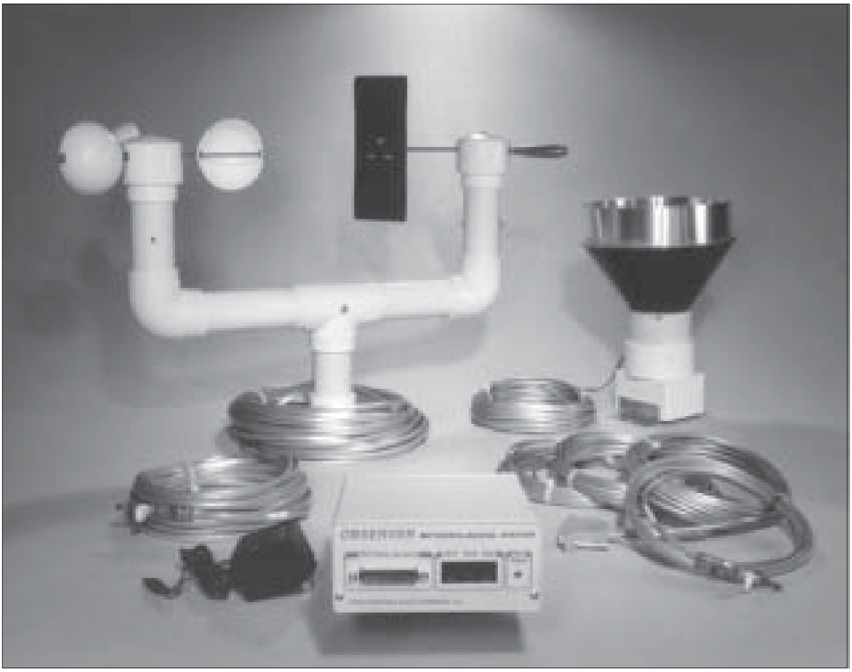


Figure B-1: The Observer Meteorological Station.
Full-size instruments featuring extensive use of stainless steel hardware and heavy gauge PVC are tough, made to last. The Experimenter provides the interface between the instruments and the computer.

sensors use current output (rather than voltage output) integrated circuit sensors, for accuracy that does not degrade with long cable lengths. The barometer and humidity sensors also use reliable solid-state sensors.

This is not a disposable weather station! If any instrument should break it can be repaired. All instruments can be disassembled and any broken components replaced. If you can build it, you can fix it.

This is a project that will give you many years of satisfaction. It will provide you with a record of exciting storms, details on your own microclimate, and a deeper understanding of the world around you.

These few pages will give you some idea of the construction and capabilities of the Observer. Unfortunately, we do not have space to go over the construction in detail as we did with the Ultrasonic RADAR. If you would like more information, please take a look at our website where you can find photos and download assembly instructions.

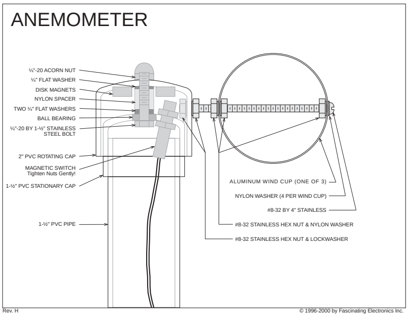
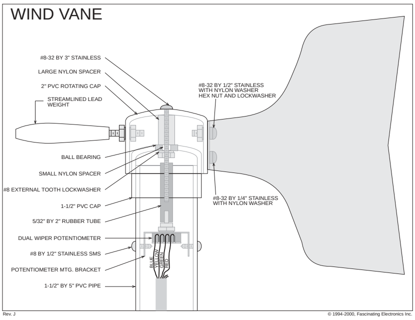

ANEMOMETER
¼"-20 ACORN NUT
¼" FLAT WASHER
DISK MAGNETS
NYLON SPACER
TWO ¼" FLAT WASHERS
BALL BEARING
¼"-20 BY 1-½" STAINLESS STEEL BOLT
2" PVC ROTATING CAP
MAGNETIC SWITCH Tighten Nuts Gently!
1-½" PVC STATIONARY CAP
1-½" PVC PIPE
ALUMINUM WIND CUP (ONE OF 3)
NYLON WASHER (4 PER WIND CUP)
#8-32 BY 4" STAINLESS
#8-32 STAINLESS HEX NUT & NYLON WASHER
#8-32 STAINLESS HEX NUT & LOCKWASHER
Rev. H
© 1996-2000 by Fascinating Electronics Inc.

Figure B-2: Anemometer.
From gentle zephyr to full gale, this rugged three-cup anemometer accurately measures wind speed. Tested in hurricane force winds. Built of robust precision drilled schedule 40 PVC, three full-size (3") aluminum wind cups, with shielded steel ball bearings and stainless steel hardware. Output is a magnetic switch closure.

Figure B-3: Wind Vane.
Even a gentle breeze turns this precise and sensitive wind vane. With a beautiful laser-cut gold anodized aluminum tail, shielded steel ball bearings, stainless steel hardware and ball bearing, custom molded powdercoated lead counterweight and rugged schedule 40 PVC body this instrument is built to withstand the elements. A special dual-wiper potentiometer translates the wind direction into two voltages, with 1° resolution and no "dead band."



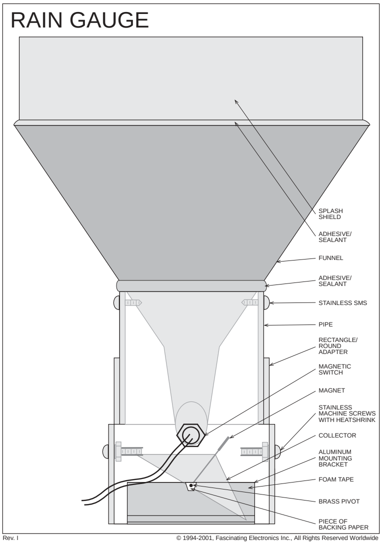


Figure B-4: Rain Gauge.
From drought to flood, this self-emptying rain gauge keeps the rainfall tally. With its large diameter funnel, it is sensitive to less than 0.01" of rain. Metal splash shield, mounting brackets and rain collector for durability. Sturdy, heavy gauge plastic base and funnel. Tested to rainfall rates exceeding 4 inches per hour. When a precise amount of rainwater collects, the collector tips, triggering a sealed magnetic switch. The calibration value is entered in software for best accuracy.

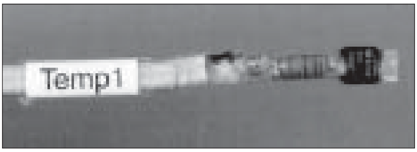


Figure B-5: Temperature Sensors.
A tiny integrated circuit temperature sensor accurately converts the ambient temperature into a current. Output is a current, rather than a voltage, allowing almost any length of wire to connect the sensor to the Observer meteorological station. The sensor is sealed in a moisture-tight adhesive lined heat shrink tube for long life.

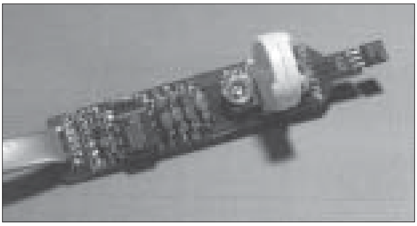


Figure B-6: Temperature plus Humidity Sensors.
This approximately ½" by 2-½" circuit board supports circuitry for both one temperature sensor and one humidity sensor. A capacitive element changes in value in response to the relative humidity of the air. This capacitance is translated into a digital frequency by an integrated circuit timer. The timer and temperature sensor are sealed in hot-melt adhesive and heat shrink.

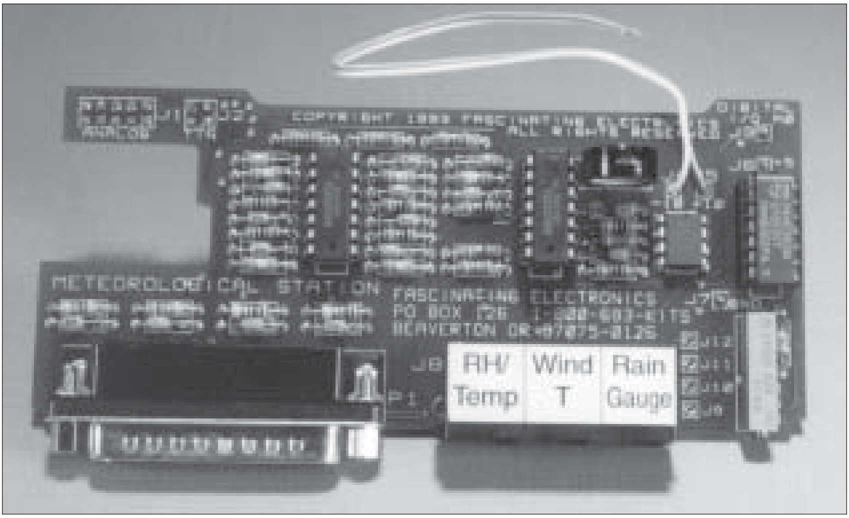
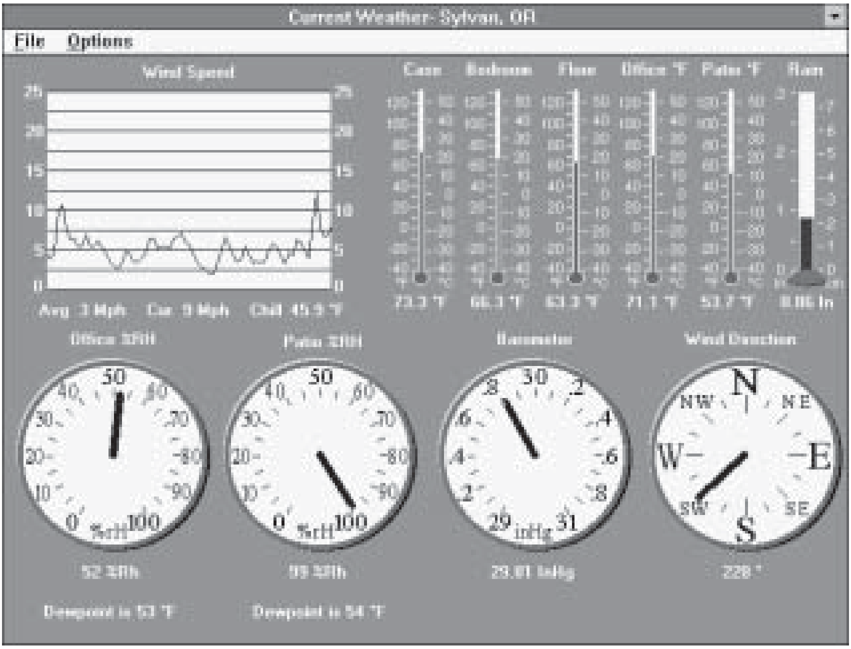


Figure B-7: Signal Conditioning and Barometer Board.
Filters signals from the weather instruments for measurement by the Experimenter. This small circuit board mounts on the Experimenter, over the wiring grid. The wind instruments, rain gauge, and single temperature/humidity sensor connect through modular jacks. The additional temperature/humidity sensors plug in through a DB25 connector. A solid-state barometer on the board measures local air pressure, can be temperature compensated for high precision over wide temperature changes, and can be adjusted over a very wide elevation range.



Avg 3 Mph Cur 9 Mph Chill 45.9 °F

73.3 °F 66.3 °F 63.3 °F 71.1 °F 53.7 °F 0.86 In

52 %Rh 93 %Rh 29.81 InHg 228 °

Dewpoint is 53 °F Dewpoint is 54 °F

Figure B-8: Current Weather Display.
The current weather display shows: (top row, left to right) wind speed moving graph, five thermometers, daily rainfall, (lower row) two hygrometers, barometric pressure, and wind direction. Digital values are displayed below each instrument graphic. Wind chill and dew point are also presented.

Meteorological Station Software

A very easy to use menu driven program runs the Observer. Graphical instruments display the current conditions (**Figure B-8**). The software automatically stores weather data on your computer's hard drive. You can graph the results (Figure **B-9**). The past day's minimum and maximum records are displayed with 1 minute resolution (**Figure B-10**).

Data is stored each minute for the past 24 hours, giving a detailed recap of the past day's weather and providing a basis for estimating weather trends. Hourly data and daily minimum and maximum records are permanently stored for historical analysis, and can be graphed or exported to other programs for further analysis.

The Experimenter's high current drivers and relay can be used as alarms, easily configured for a variety of weather conditions.

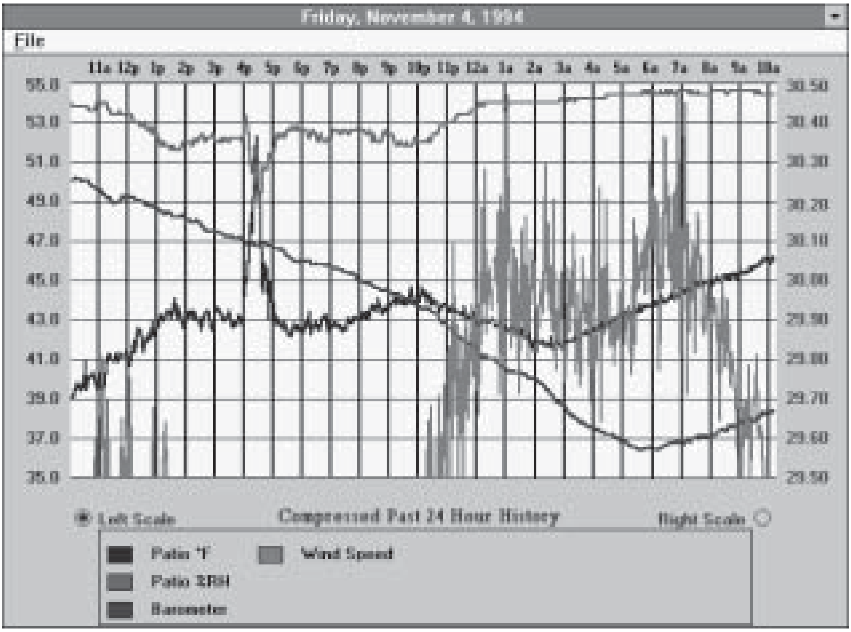
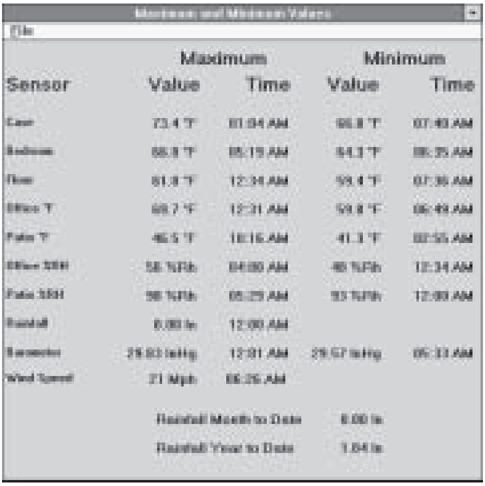


Figure B-9: Past 24 Hour History.

This is an actual data graph from a recent Pacific storm. Relative humidity (top line) hovered just below 90% most of the afternoon, except when the sky cleared between 4 PM and 5 PM, allowing the temperature (third from top) to rise from 43°F to just over 52°F, before falling back to 43°F as the sun set. The barometer (second from top) fell steadily, warning of an approaching front. The first 5 MPH wind gusts from the front appeared about 10:15 PM, and grew stronger throughout the night. Winds peaked at just under 20 MPH at 1 AM and again at around 7 AM.



Maximum and Minimum Values

File

	Maximum		Minimum	
Sensor	Value	Time	Value	Time
Cave	73.4 °F	01:04 AM	66.8 °F	07:40 AM
Bedroom	66.8 °F	05:19 AM	64.3 °F	06:35 AM
Den	61.8 °F	12:34 AM	59.4 °F	07:36 AM
Office °F	60.7 °F	12:31 AM	59.8 °F	06:49 AM
Patio °F	46.5 °F	10:16 AM	41.3 °F	02:55 AM
Office %RH	58 %Rh	04:00 AM	48 %Rh	12:34 AM
Patio %RH	98 %Rh	05:29 AM	93 %Rh	12:00 AM
Rainfall	0.00 In	12:00 AM		
Barometer	29.83 inHg	12:01 AM	29.57 inHg	06:33 AM
Wind Speed	21 Mph	06:26 AM		

Rainfall Month to Date 0.00 In

Rainfall Year to Date 1.04 In

Figure B-10:
Daily Minimums & Maximums Display.

The time and value of the daily minimum and maximum for each instrument are displayed with one minute resolution. Rainfall month-to-date and year-to-date are also presented.

C

An Autonomous Robot

This fascinating project was built as a metal shop project by high school students in Newberg, Oregon. The robot is operated by an internal computer, and includes a speech synthesizer, keyboard, and video display.

Hardware Features

The robot has two arms which rotate at the shoulder and bend at the elbow. A simple gripper is mounted on the end of each arm. All three functions (shoulder, gripper and elbow) are controlled by stepping motors. High current drivers were built for some of these motors.

In order to control six stepping motors from one Experimenter the students built a latching driver circuit. The circuit takes the T-OUT signals in two groups (0:3, 4:7), and either latches or passes the signals to driver ICs. In this way both arms can make the same motion simultaneously, or may move independently one at a time.

The waist rotates, driven by a small DC motor. DC motors were also used to drive the main wheels, which were originally built for a child's electric car.

The robot's "brain" is a PC motherboard. A text-to-speech synthesizer is driven through a printer port. A small monochrome monitor and 3.5" floppy drive mount in the robot's head. Speakers for the speech synthesizer mount on the robot's shoulders.

A 12 volt automobile battery provides power the motors and electronics. A switching supply generates 5 volt power for the computer. +12 volts required by the monitor is supplied through a 1 amp low dropout voltage regulator.

Possible enhancements include adding microswitches to detect collisions with objects by the body or the arms, and adding an ultrasonic RADAR for obstacle avoidance. The software can be endlessly enhanced, adding new and interesting behaviors.

Mechanical Hints

The Newberg High School students work with very good shop equipment. Few of us are so fortunate. If you want to build a robot but lack metal working machinery, why not try working out a deal with your local high school or community college industrial arts teacher? He may be receptive to doing a project like this.

Based on the student's experience, it is difficult to build to tolerances adequate for gear drive. The students had much more success with belt drive. Where subject to great tension, such as at the shoulder joint, a belt with large teeth is needed to prevent slippage. A modest reduction ratio can provide a great deal of torque from even small stepping motors. Aluminum is much lighter than steel. Even so, it is amazing how much a five foot tall robot can weigh.

Almost all of the mechanical parts were acquired surplus. Some were scavenged from copy machines. The robot was basically built from scrap that had been donated to the high school. Despite its humble origins, the robot was a great success. The students learned not only machining, but also electronic wiring and BASIC programming.

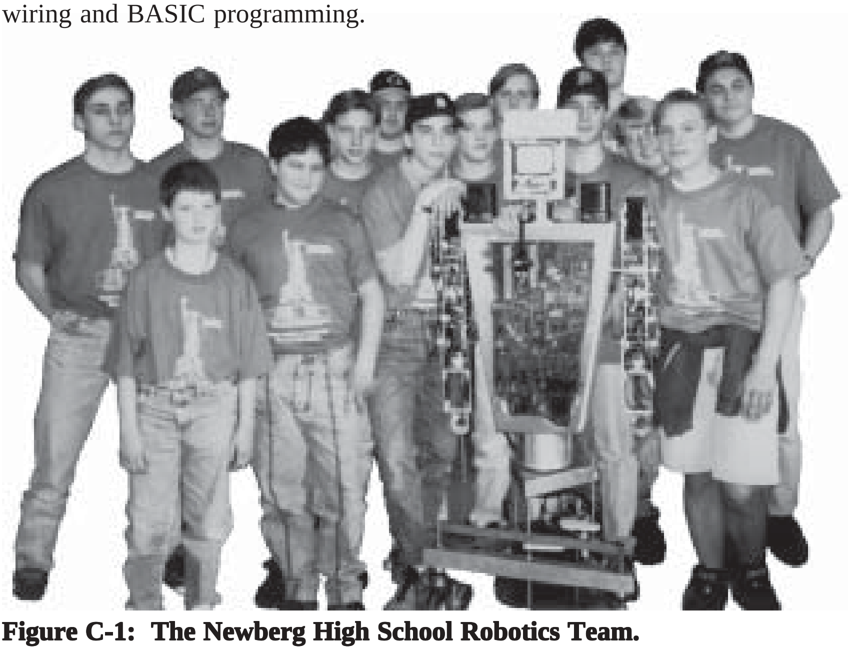


Figure C-1: The Newberg High School Robotics Team.

Experimenter Specifications 11/02/2001	
Serial Interface	Standard RS-232C, DB-25 female connector. Supports baud rates from 300 baud to 38.4 Kbaud with hardware and Xon/Xoff handshake.
Drivers	Eight drivers, source and sink up to a maximum of 1 amp each. Uses an external power source of +4.5 to +36 volts DC. Thermal overload and output clamp diode protected. Not short circuit protected.
Relay	One SPDT relay with LED status indicator, for loads up to 10 amps.
Analog Inputs	Eight analog channels with 5 millivolt resolution from 0 to 5.115 volts.
Counter/Timers	Four counter/timers. Resolution is 10 µS, durations from 250 to 655,350 µS. Measures period, pulse width, channel-to-channel delay. Counts to 65,535 from DC to >1 kilohertz.
Digital I/O	Twenty-four digital input/output lines with CMOS voltage and drive levels.
Logic Supply	A low dropout voltage regulator provides +5 volts for Experimenter logic circuitry from an external supply of +5.5 to +15 volts. Has LED power indicator and convenient power switch.
Analog Supply	An adjustable low dropout voltage regulator provides a quiet, precise, 5.120 volt reference and power supply for analog measurements and circuits.
Circuit Board	Durable epoxy-glass, double-sided with plated holes. Circuit numbers, graphics and solder mask on both sides. Measures approximately 5.25 by 6.2 inches. Rests on six rubber feet or may be installed in an optional metal case.
Wiring Grid	Large wiring grid (360 pads) with prewired +5 and GND pads for adding your own circuits. Also has mounting pads for adding one DB-25, two DB-9, one high density and one 5 mm pitch connectors.