Copyright © 2008, Wimborne Publishing Ltd
(Sequoia House, 398a Ringwood Road, Ferndown, Dorset BH22 9AU, UK)
and TechBites Interactive Inc.,
(PO Box 857, Madison, Alabama 35758, USA)

All rights reserved.

The materials and works contained within EPE Online — which are made available by Wimborne Publishing Ltd and TechBites Interactive Inc — are copyrighted.

TechBites Interactive Inc and Wimborne Publishing Ltd have used their best efforts in preparing these materials and works. However, TechBites Interactive Inc and Wimborne Publishing Ltd make no warranties of any kind, expressed or implied, with regard to the documentation or data contained herein, and specifically disclaim, without limitation, any implied warranties of merchantability and fitness for a particular purpose.

Because of possible variances in the quality and condition of materials and workmanship used by readers, EPE Online, its publishers and agents disclaim any responsibility for the safe and proper functioning of reader-constructed projects based on or from information published in these materials and works.

In no event shall TechBites Interactive Inc or Wimborne Publishing Ltd be responsible or liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or any other damages in connection with or arising out of furnishing, performance, or use of these materials and works.

READERS' TECHNICAL ENQUIRIES

We are unable to offer any advice on the use, purchase, repair or modification of commercial equipment or the incorporation or modification of designs published in the magazine. We regret that we cannot provide data or answer queries on articles or projects that are more than five years' old. We are not able to answer technical queries on the phone.

PROJECTS AND CIRCUITS

All reasonable precautions are taken to ensure that the advice and data given to readers is reliable. We cannot, however, guarantee it and we cannot accept legal responsibility for it. A number of projects and circuits published in EPE employ voltages that can be lethal. You should not build, test, modify or renovate any item of mains-powered equipment unless you fully understand the safety aspects involved and you use an RCD adaptor.

COMPONENT SUPPLIES

We do not supply electronic components or kits for building the projects featured; these can be supplied by advertisers in our publication Practical Everyday Electronics. Our web site is located at www.epemag.com

We advise readers to check that all parts are still available before commencing any project.

www.epemag.com

To order you copy for only $18.95 for 12 issues go to www.epemag.com

Constructional Project

EPE HYBRID COMPUTER

PETROS KRONIS

Part 2

Real-time computation of complex system behaviour is greatly simplified by combining analogue and digital processing techniques.

LAST month the circuit technicalities for this design were discussed at length, and the initial constructional aspects were described. In this final part the remaining constructional details are presented (see Fig.18), plus guidance on actually using the design to simulate real-world engineering problems.

TESTING

Before any serious programming is attempted it is good practice to carry out the following simple testing procedures which will help to identify any errors in the construction, or problems in the operation of the computer.

The analogue computer is programmed by connecting the various processing units together via the patch panel. It is necessary that you familiarise yourself with the arrangement of the patch panel.

The panel layout for one amplifier is shown in Fig.19. The layout for the other nine amplifiers is the same except for the reference voltage socket 7, and the four rightmost sockets of the first row of sockets, as indicated in the righthand column.

Functional notations for the sockets are given in Fig.20.

TESTING THE ADDING AMPLIFIERS

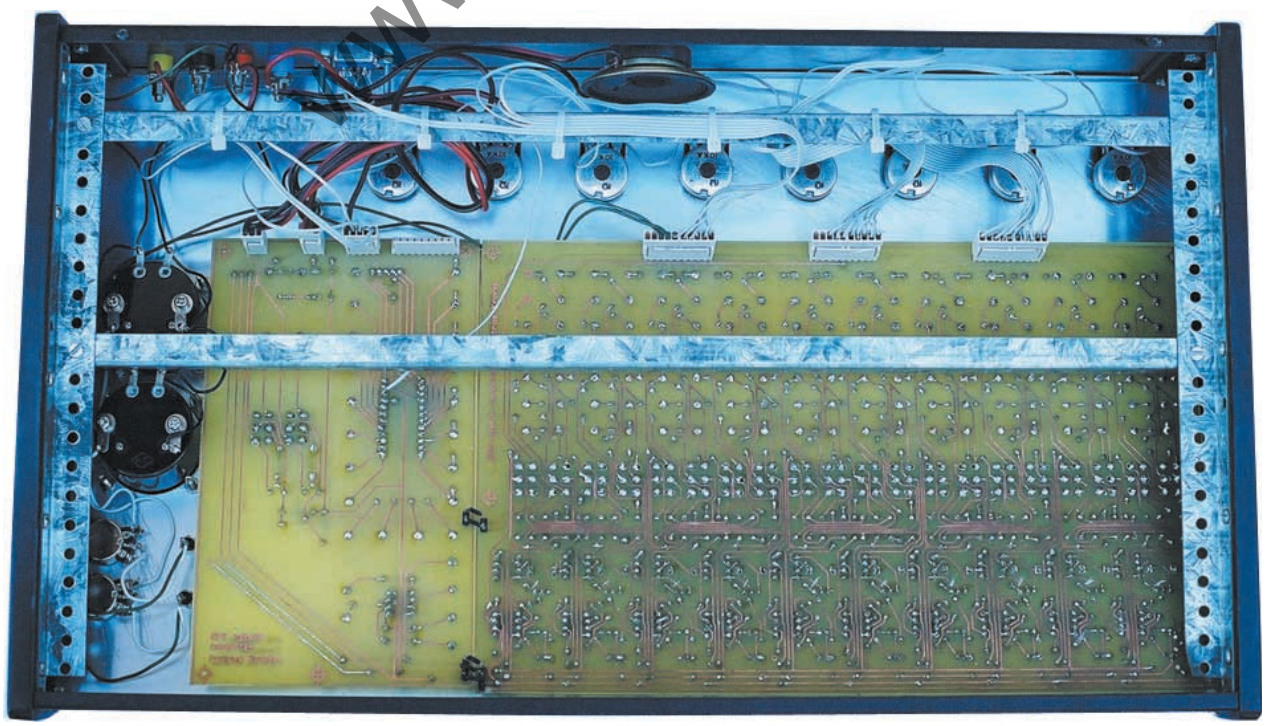
To test the Adding Amplifiers, do it in the following order:

1. Switch all ten amplifiers to **Add** using the amplifier toggle switches, S1 and S2.
2. Switch the power on.
3. Set up a reference voltage of 1V. This can be done by connecting a +15V reference voltage to the input of any of the Coefficient Multiplier potentiometers (VR15) and adjusting the dial until the potentiometer output reads 1V (measured using a multimeter).
4. Apply the 1V reference voltage to the ×1 inputs of the amplifiers. Measure the amplifier output voltage. It should read 1V. Test all 10 amplifiers in turn.
5. Repeat the test, applying the 1V reference voltage to the ×10 amplifier inputs. The amplifier outputs should read 10V.

INTEGRATING AMPLIFIER TESTING

Test the Integrating Amplifiers in the following order:

1. Switch all ten amplifiers to **Integrate** using the amplifier toggle switches, S1 and S2. Switch the mode switches to **Compute/Auto Reset** (S3) and **Compute/Auto Hold** (S4).
2. Make the connection for integration with a nose gain of 1, i.e. connect socket 6 to socket 13 (see Fig.19).
3. Switch the power on.



www.epemag.com

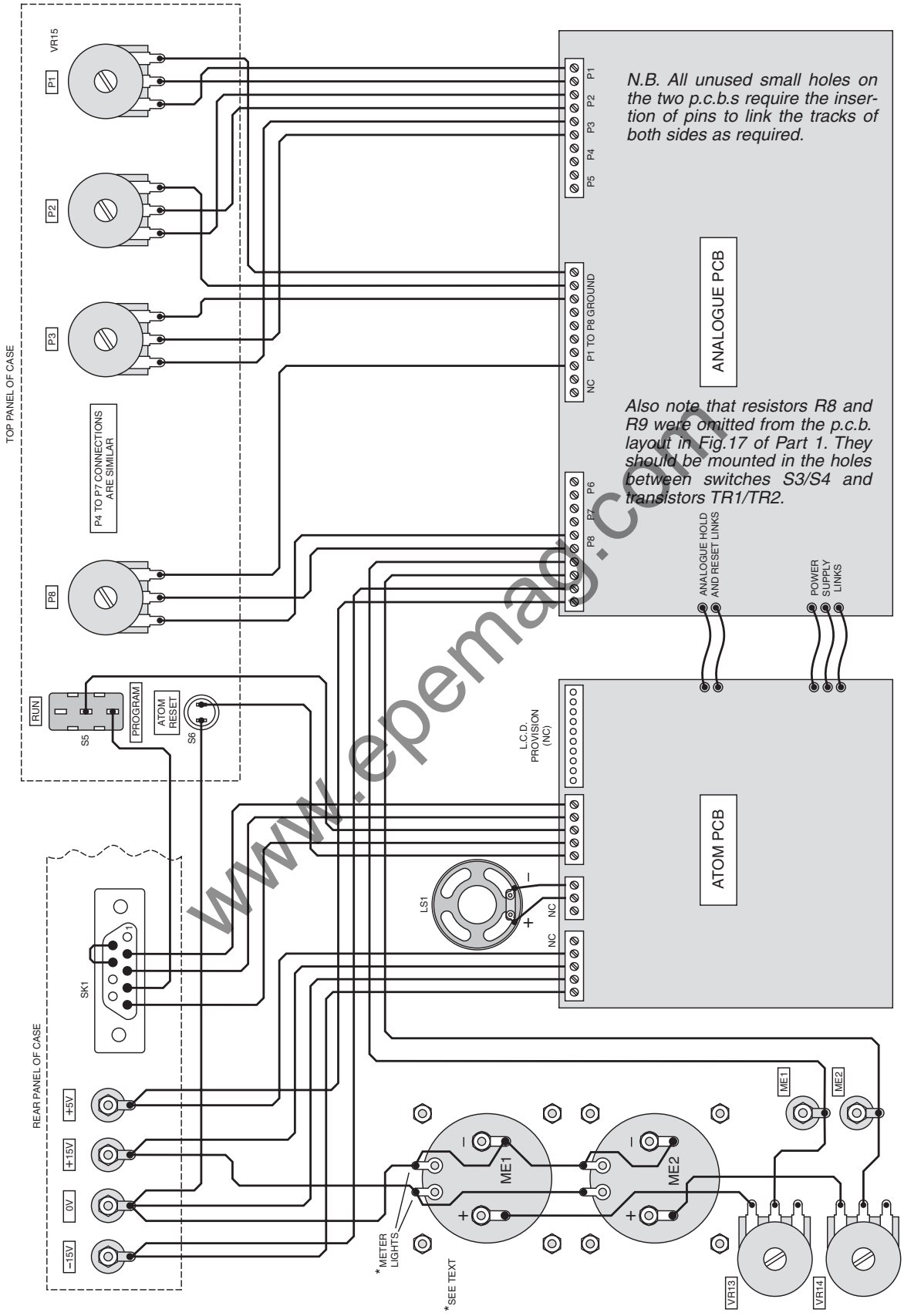


Fig.18. Interconnection wiring diagram.

www.epemag.com

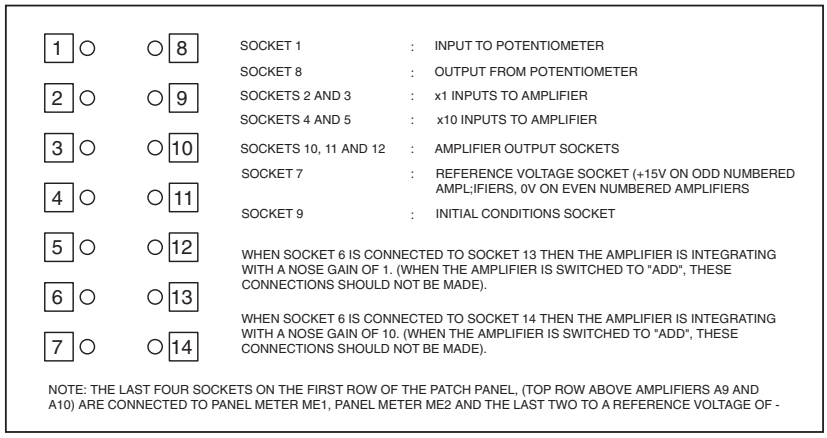


Fig.19. Patch panel socket layout and functions for one amplifier module.

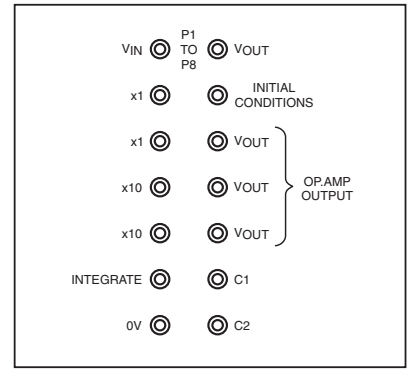


Fig.20. Functional notations for the amplifier module sockets.

4. Using potentiometer VR13 adjust the sensitivity of panel meter ME1 to give maximum deflection at +15V and connect the amplifier output to the panel meter.

5. Apply a 1V reference voltage (as set in the previous tests using VR15) to the ×1 inputs of the amplifier. The output should increase linearly at a rate of 1V per second until the amplifier saturates. The appropriate over-voltage warning l.e.d. (D4) should come on as the amplifier saturates. Bear in mind that the amplifiers invert the input voltage, both when adding and when integrating,

6. If the reference voltage is applied to the ×10 inputs, the output increases at a rate of 10V/s, which is too fast to see unless shown on an oscilloscope. If the integrator is connected with a nose gain of 10, i.e. connection of socket 6 to socket 14, integration is even faster, by a factor of 10.

HOLD MODE TESTING

To test the Hold mode, follow the same procedure described in the first four numbered paragraphs in the previous test, and then:

1. Before the amplifier saturates, switch the Compute/Auto Hold toggle switch (S4) to **Manual Hold**. The processing should freeze and the output should remain constant. (Some drift may be present.)

2. Now switch the Compute/Auto Reset toggle switch (S3) to **Manual Reset**. The output of the integrators should become zero.

PROGRAMMING CONVENTIONS

In the preceding simple examples, the process of connecting the various analogue computer units is described in English. However, this is not very efficient, or universally understandable, and programmers have developed a generally accepted code for the description of an analogue computer program.

The program is represented by a flow diagram, which is a collection of symbols representing the various units of the analogue computer, connected together. The symbols used are those given in Fig.5 (Part 1).

Using this symbolic convention, the analogue computer flow diagram in Fig.21 shows two successive integrations by amplifiers A1 and A2, of a step function of 1V set up by coefficient multiplier P1.

Panel meters ME1 and ME2 are used to monitor the outputs of amplifiers A1 and A2 respectively. If we could plot these outputs on an X-Y plotter we would find that when a step function is integrated the result is a ramp function, and when a ramp function is integrated a square function (x^2) is the result.

These results are shown in the circles of Fig.21. Note that the amplifiers invert the input signals, i.e. a positive step function produces a negative ramp function and a negative ramp function results in a positive square function.

To set up this program on the analogue computer, the patch panel is wired as shown in Fig.22. This diagram is not normally drawn because a programmer with some experience can wire the patch panel by simply looking at the flow diagram. It is given here to help beginners.

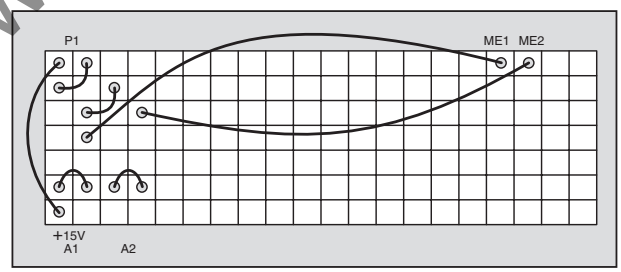


Fig.22. Patch panel connections for the flow diagram of Fig.21.

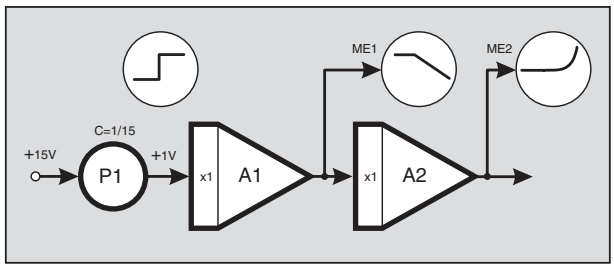


Fig.21. Analogue computer flow diagram example.

To run the program follow these steps:

1. Wire the patch panel.

2. Switch the power on.

3. Calibrate panel meters ME1 and ME2 by applying a reference voltage of +15V and adjusting the sensitivity of the meters to read that value at around maximum deflection, using their potentiometers, VR13 and VR14.

4. Switch amplifiers A1 and A2 to **Integrate**.

5. Set mode switches to **Compute/Auto Hold** (S4), and **Manual Reset** (S3).

6. Switch **Manual Reset** (S3) to **Compute/Auto Reset** to begin the computation function.

7. Observe the panel meters.

OFFSET NULL TRIM

When the input of an ideal op.amp powered by a split power supply (e.g. +15/0V/–15V) is grounded (0V), the output should be zero. However, this is not the case in a real op.amp and a corrective voltage is necessary to eliminate the offset error. The offset voltage of the OPA177 used in the *EPE* Hybrid Computer is very low, 25μV maximum. But even so, the offset null procedure is sometimes needed to reduce errors and unwanted output drift.

Adjust the offset voltage as follows:

1. Set all amplifiers to **Add**.

2. Connect the output of amplifier A1 to meter ME1.

3. Ground one of the ×10 inputs of the amplifier.

4. Turn the sensitivity of the meter as low as possible (VR13).

5. Switch the power on.

6. Increase the sensitivity of the meter gradually to maximum.

7. Adjust the offset null potentiometer (VR1 to VR10, as required) to make the meter read zero.

8. Repeat the procedure for the other amplifiers.

9. Make sure you do not disturb the offset null potentiometers during the programming and execution of the program.

INITIAL CONDITIONS EXAMPLE

In the previous examples we assumed that at the start of the computation, i.e. at time = zero, all variables had zero value. This may not always be the case.

Suppose we wanted to give an initial value to the output of an integrator before the computation begins. We might like to investigate the flight of a rocket, for

www.epemag.com

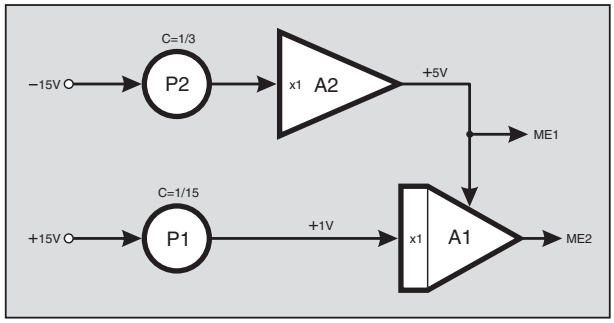


Fig.23. Initial conditions setup.

example, not from the point of launch but from some height above the launching pad, at which the rocket will have some velocity and acceleration. This can be done by connecting wires carrying reference voltages, to the initial conditions sockets of the integrators.

Let us assume that we want to integrate a step function of +1V, but this time we want the integrator to begin integrating from an initial value of +5V.

Set up the program illustrated in Fig.23 on the patch panel (a first test of your understanding of how to implement such programs, as discussed in relation to Fig.21). P1 and P2 are two VR15 potentiometers.

To run this program follow these steps:

1. Switch amplifier A1 to **Integrate** (S3) and A2 to **Add** (S1 and S2).

2. Switch the power on.

3. Adjust P1 to give +1V at the input to A1 and P2 to give +5V at the output of A2.

4. Switch **Compute/Auto Hold** to **Manual Hold** (S4).

5. Switch **Compute/Auto Reset** to **Manual Reset** (S3). Wait a few seconds for the output of A1 to settle to +5V.

6. Switch **Manual Hold** to Compute/Auto Hold (S4).

7. Switch **Manual Reset** to **Compute/Auto Reset** (S3) and observe the results.

You should find that the integration begins from a value of +5V and the output of A1 reduces at a rate of –1V/s until it saturates at –15V.

The initial condition value is invariably formed using an adder and a potentiometer, unless of course the exact value is available as a reference voltage. It is common practice, therefore, not to show these two computing elements in the program diagram but simply to show the value being applied to the initial condition socket as shown in Fig.24.

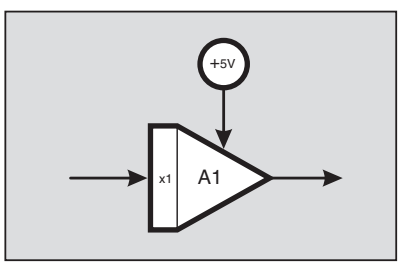


Fig.24. Representation of initial conditions.

SPRING-MASS EXAMPLE

Let's take the spring-mass system and the capacitor-inductor circuit as examples of simulating engineering problems on the analogue computer.

The diagram in Fig.25a shows a mass *m*, suspended by a spring of stiffness *K*. Its electrical equivalent system is shown as a capacitance-inductance series circuit in Fig.25b. Both systems when disturbed will oscillate. We now illustrate how to simulate the systems on the analogue computer and observe their behaviour.

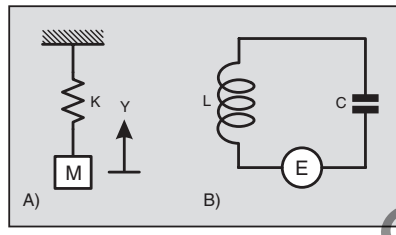


Fig.25. Mechanical and electrical equivalent circuits.

To do this we must first write down the equation of motion, which in the mechanical system involves no more than Newton's Law of Motion:

Force = mass × acceleration, which we will express as $F = m \times a$.

Imagine that the mass (*m*) is pulled down a small distance (*d*) from the equilibrium position and then released. At the moment of release the spring pulls the mass up with a force (*F*) of *Kd*. Applying the equation of motion, we get:

$$-Kd = ma$$

Dividing both sides of the equation by mass *m* and rearranging so that the highest derivative (*a*) is on the left hand side of the equation and all other terms on the right hand side, we get:

$$a = -Kd/m$$

From this equation we draw the flow diagram shown in Fig.26.

Note that there are special textual symbols that can be used to express the graphical symbols, but their discussion and use is beyond the scope of this article.

As the flow diagram shows, by integrating the acceleration (*a*) using amplifier A1 we obtain the velocity (*v*) which is shown on panel meter ME1. Amplifier A2 integrates the velocity to give the displacement (*d*) shown on panel meter ME2.

We use amplifier A3 as an inverter to obtain a displacement of –*d*, and potentiometer P1 to multiply by a constant factor of *K/m*. The output of the potentiometer is now $-\frac{K}{m}d$ which, looking at our equation, is equal to the acceleration, *a*. So we connect the output of P1 back to the input of A1 to complete the loop.

The system as it stands will not oscillate unless it is disturbed. Theoretically, once it is disturbed it will continue to vibrate indefinitely, because our equation does not take into account any air resistance (or electrical resistance in the case of the electrical equivalent system) or any energy losses in the system. In practice, however, you will find that the oscillations will reduce and die out due to these effects. The analogue computer should produce results very near to the theoretical predictions.

To run this program follow these steps:

1. Connect the patch panel.

2. Ensure A1 and A2 are Integrating and A3 is Adding.

3. Set mode switches to **Compute/Auto Reset** (S3) and **Compute/Auto Hold** (S4).

4. Switch the power on.

5. Play around with the dial setting of P1. By reducing its value you will see the spring extending (displacement *d* on ME2), until the amplifiers saturate. Now increase the value of P1 to make the system oscillate at different frequencies, as you vary the value of *K/m*. Observe the meters ME1 and ME2 to see how the system behaves.

You should find that a high value of *K/m* (i.e. a high spring stiffness or small mass, or both), gives a high frequency of oscillation and vice versa. The other point to note is that the oscillation is sinusoidal. The output of A1 (the velocity, *v*), is a cosine function, whereas that of A2 (the displacement, *d*) is a sine function.

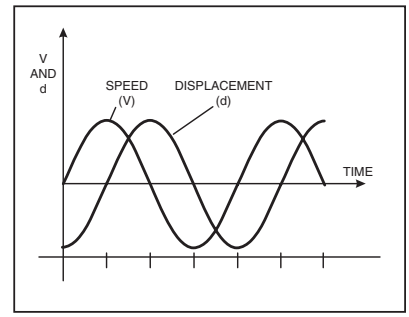


Fig.27. Solution to the problem of the spring-mass system.

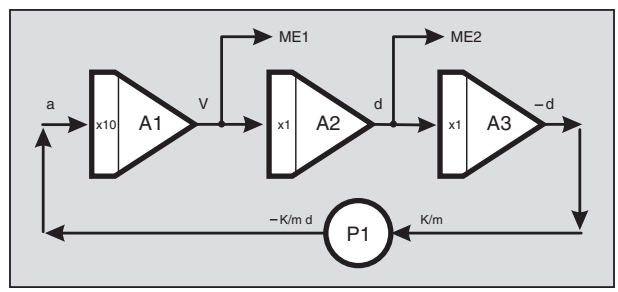


Fig.26. Flow diagram for spring-mass system.

There is a 90 degree phase difference between the two. That is, the mass comes to a momentary stop when the displacement is at its maximum value. This is shown by the diagram in Fig.27.

Notice that we have managed to program the analogue

www.ETEmag.com

computer to solve the differential equation and produce the solution without having any mathematical knowledge on the solution of differential equations. The only knowledge needed is to be able to apply Newton's Law of Motion. This is one of the great advantages of an analogue computer.

DAMPING EFFECTS

Damping is caused by air resistance or electrical resistance. This form of damping is called *viscous damping* in mechanical systems and is proportional to the velocity.

Viscous resistance is sometimes unwanted. For example, it forces us to burn fuel continuously to drive cars or to propel aeroplanes through the air. Spacecraft travelling in empty space do not have to do this, although they have to burn fuel to decelerate.

In other cases, we find damping very useful. Mechanical vibration is one of these cases, where damping helps to reduce unwanted and dangerous vibrations. Without it, motor cars would provide a very rough ride. Representation of mechanical and electrical damped systems is shown in Fig.28.

In the mechanical system, damping is provided by a dash pot which is full of oil. This system resembles very closely the suspension system of a motor car.

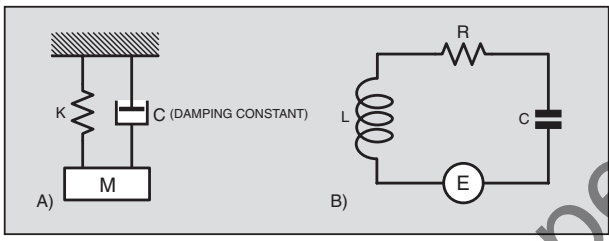


Fig.28. Mechanical and electrical systems with damping.

The following equation applies to the mechanical system:

$$F = ma + Cv + Kd$$

where C is the viscous damping factor.

The equivalent flow diagram is shown in Fig.29.

Connect the program on the patch panel and run it. You can try various forcing functions (F), such as impulse, step, sinusoidal, etc. The impulse function can be achieved by momentarily touching the wire connected to the input of A1 on a reference voltage socket carrying +15V. The diagrams in Fig.30 and Fig.31 show what results to expect when impulse and step forcing functions are applied.

Turn the dial of potentiometer P1 to select various damping values and observe the results. At a particular value of damping the output reaches the final value with no oscillation and at the shortest possible time. This value is called *Critical Damping*.

One application of the above is in the construction of analogue measuring instruments, like moving coil meters, etc. The input of such instruments is usually a step function and the needle is damped to a value very near the critical damping value so that the meter can be read with the minimum of delay.

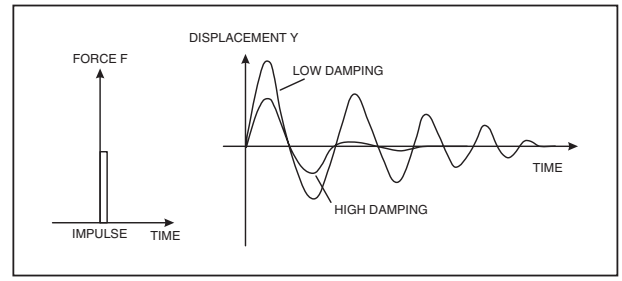


Fig.30. The effect of the impulse function.

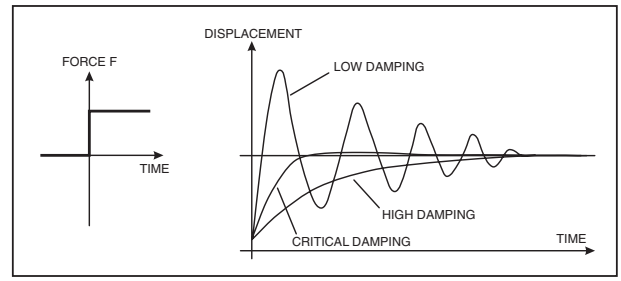


Fig.31. The effect of the step function.

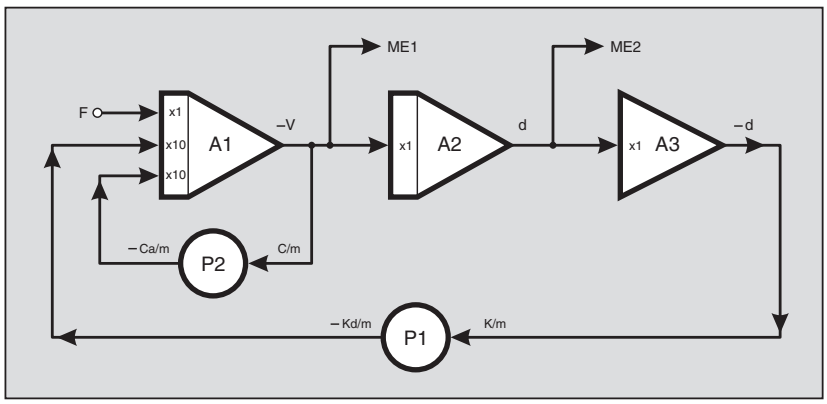


Fig.29. Flow diagram for the damped system.

CRACKING THE ATOM

The ATOM is a powerful microcontroller (MCU) which can be programmed in BASIC. It is a customised variant of Microchip's PIC16F876 and offers powerful commands enabling the programmer to carry out operations which would be difficult to achieve in PIC Assembly language, such as floating point mathematical operations. With ATOM BASIC one can perform 32-bit multiplication and division with real numbers up to a maximum value of 4,294,967,295, and this can be done in one line of code.

ATOM BASIC can handle bits, nibbles (4-bit), bytes, words (16-bit), and long values (32-bit), binary, hexadecimal, decimal, integer and floating point mathematical operations. The programmer is offered the usual BASIC facilities such as arithmetic and logical operations, subroutines, arrays, etc. as well as specialised commands which can control servo and stepper motors, produce sound, etc.

The ATOM retains the 384 bytes of RAM and 8K of FLASH program space of the fundamental PIC16F876. It has a built in 5V regulator, a serial port for in-circuit programming and data communication, and an analogue-to-digital converter (ADC).

To program the ATOM, first install the ATOM software which is supplied on a CD-ROM when you buy the chip, or which can be downloaded from the BASIC MICRO manufacturer's internet site at **www.basicmicro.com**. The software enables the programmer to write, compile, load and run programs under an Integrated Development Environment (IDE).

If the software is downloaded, double-click the **.EXE** application and it will automatically unzip. Then double-click the **setup.exe** file, and follow the on-screen prompts.

If the software is installed from the CD-ROM, insert it into your computer. If autorun is enabled the installer menu will appear, select your software and the installation process will begin automatically. Restart your computer after the installation is complete.

Start the software and spend a while reading the *getting started* section of the manual and familiarise yourself with the IDE. Understanding all the features of the IDE will make it easier to use the ATOM more efficiently.

Before you program the microcontroller, connect the Hybrid Computer's RS232 output socket to a vacant serial COM port on your PC, using an appropriate cable. Usually COM2 will be available.

FIRST ATOM PROGRAM

To enter and run your first ATOM program, follow these steps:

1. Run the ATOM software.
2. Click the **Build** button at the bottom of the screen (see Fig.32).

www.epemag.com

3. Select File, New, Mbasic file, enter file name, and Save.

4. Enter the following program:

```
main
        debug ["Hello"]
        pause 100
        goto main
end
```

5. Save this program with any name of your choosing.

6. Configure the COM port by selecting Tools, System setup, COM2.

7. Switch the Hybrid Computer on and place toggle switch S5 to **Program**.

8. Click on the **Debug** button at the top left corner of screen.

9. Wait until the program is automatically compiled and loaded into the ATOM.

10. Click the **Connect** button, then **Animate** or **Run** to run the program.

You should get the word "Hello" repeatedly displayed in the output window. The command **debug** instructs the ATOM to display the "Hello", and the **pause 100** instruction to wait 100ms before executing the next instruction, to again repeat "Hello".

If this program works successfully then it can be assumed that the ATOM is operating correctly and you can proceed to more useful programming.

RS232 SERIAL TESTING

The following program is a test for the Hybrid Computer's ability to communicate with your PC via the RS232 serial link.

```
Main
        serout s_out,i9600,["hello"]
        pause 100
        goto main
end
```

This is almost the same program we used before, but this time the **serout** command outputs the data from the Hybrid Computer's serial port at a baud rate of 9600 bits per second.

This time do not click on the **Debug** button as the debug mode does not work with the **serout** command. Click on **Program** instead, to enter the routine that compiles the program and loads it into the program area of the ATOM.

When the program has been loaded into the ATOM, switch S5 to **Run**, click on **Terminal1** at the bottom of the screen, configure the port to COM2, 9600 bits per second, no parity, no flow control, echo, and click **Connect**. You should see the word "hello" repeated every 100ms.

To make sure that the data arrives correctly and can be read by an independent program, minimise the PC's Basic Micro software window, and click the Windows Start button, then select programs, accessories, communications, hyperterminal.

This is a program which is available with Windows and can be used to read serial data. Configure the hyperterminal to **Connect using: direct from COM2**. Click Configure and set the port configuration to 9600 bits/s, 8 data bits, no parity, 1 stop bit, and no flow control. Then click on **Advanced** and disable the **Use FIFO buffers** and click **OK**. Then click **Connect** and see if the word "hello" comes through.

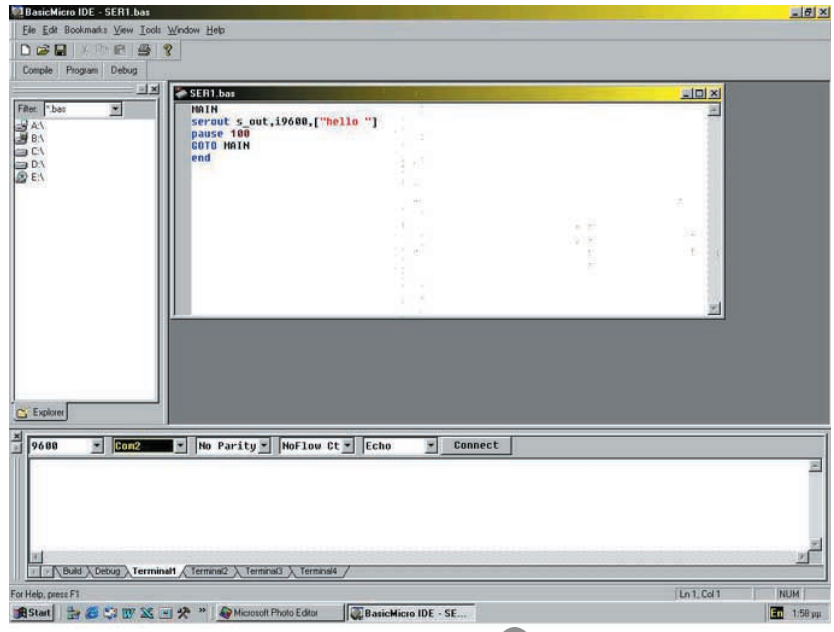


Fig.32. The ATOM software Integrated Development Environment screen.

When finished click disconnect and go into the properties again to return the properties to default. If you fail to do this, you will not be able to reprogram the ATOM because the programming software uses the same COM2 port to send and receive program information.

ADC TESTING

The ATOM has the capability to convert analogue signals to digital values. This is a very useful system to have in a hybrid computer as the two parts of the computer cannot communicate with each other unless the information is converted from one system to the other.

One point to bear in mind here is that the ATOM microcontroller operates on a supply voltage of +5V whereas the analogue computer works with ±15V. The analogue voltage to be converted to digital ***must*** be restricted to values between 0V and +5V. An application of a voltage outside this range on an ADC pin of the ATOM may cause damage to the system.

This may seem very restrictive but there are ways to get round the problem by scaling the input signals, as we shall explain later. For the moment set up the program in Fig.33 on the analogue computer and connect to the ATOM's ADC pin AX0.

Adjust the dial of the potentiometer P1 to give an output of –5V maximum. Once adjusted do not touch the dial of P1 while the program is running. The –5V signal is inverted by A1 and P8 is used to vary the signal value from 0V to 5V. This way there is no danger of an inadvertent movement of the dial of the potentiometer to present the ADC pin with a voltage outside the range of 0V to +5V.

Enter the following ATOM program.

```
;program to convert analogue value to digital and transmit it to PC
;variable definitions
        value VAR byte
;A/D sampling definition
        clk con 2
;main program
        main
;A/D pin AX0, right justified
        adin AX0,CLK,AD_RON,value
;send value to PC
        serout s_out,i9600,[value]
        pause 100
        goto main
end
```

Do not use **DEBUG** to run the program as this mode does not work with the **adin** command. Click on **Program** to compile and load the program in the ATOM, ensuring than when you do this the toggle switch on the hybrid computer is switched to **Program**. When you run the program switch back to RUN. You can either set up **Terminal 1** to observe the results or use the Hyperterminal program as done before.

While the program is running turn the dial of the potentiometer to vary the analogue value. You will find that as the variable **Value** was declared as a byte the analogue value of 0V to 5V is converted to a digital value between 0 and 255 decimal and displayed as the corresponding character of the ASCII code.

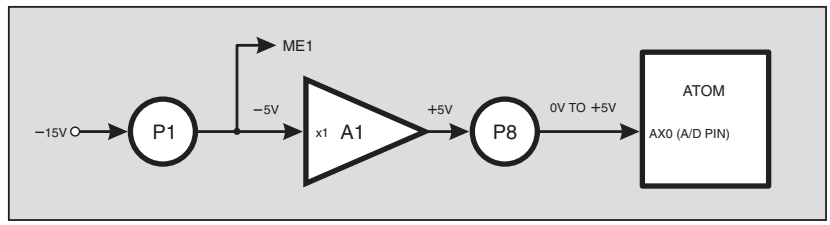


Fig.33. ADC setup program.

www.obemag.com

ATOM CONTROL

Set up the initial conditions program for the analogue computer, given earlier in Fig.23. There we used **Manual Hold** and **Manual Reset** to freeze the calculation and reset the analogue computer. We are now going to use the ATOM microcontroller to control the operations of **Hold** and **Reset** under program control.

Enter the following ATOM program and use **Program** to compile and load it into the chip.

```
;program to test auto Hold/Reset
        hold var byte
        reset var byte
;initialise variables, 1 ATOM pin P1, 0 is pin P0
        hold=1
        reset=0
        begin:
;place computer in reset and wait 3 seconds
        high hold
        high reset
        pause 3000
;place the computer in compute mode, and execute for 5 seconds
        low reset
        low hold
        pause 5000
;repeat operation
        goto begin
end
```

Before you run the program connect ATOM socket P0 to the **MODE Reset** socket, and socket P1 to the **MODE Hold** socket. When the program is run you should hear the relays click when the computer is placed in the **Reset** and **Hold** modes. When the computer is in the **Hold/Reset** mode the amplifier output will take its initial condition value.

AUDIO CIRCUIT TESTING

ATOM BASIC has commands to produce sounds, and even music. The following is a very simple program to produce some sounds to test the audio circuit.

```
begin:
;play 2 notes out of pins p1 & p2 of duration 1s and 2s
        sound2 p1\p2,[1000\2500\3500,
        2000\4500\6500]
        goto begin
end
```

Before you run this program connect ATOM sockets P1 and P2 to the two audio sockets.

FLIGHT SIMULATION

The flight of an object is described mathematically by differential equations, and these equations can be solved easily and efficiently by the analogue computer. To give an example of how it can be used as a flight simulator, we get the analogue computer to solve the equations, transmit the velocity of the aircraft to the PC and program the PC to produce the graphics and animation.

Note that we select to transmit the speed and not the height of the aircraft as the voltage range of the analogue computer would not be sufficient to represent the range of height of flight envisaged. A Harrier Jump Jet in vertical flight has two forces acting on it, its weight (*mg*) as a downwards force, and the engine thrust (*T*) as an upwards force.

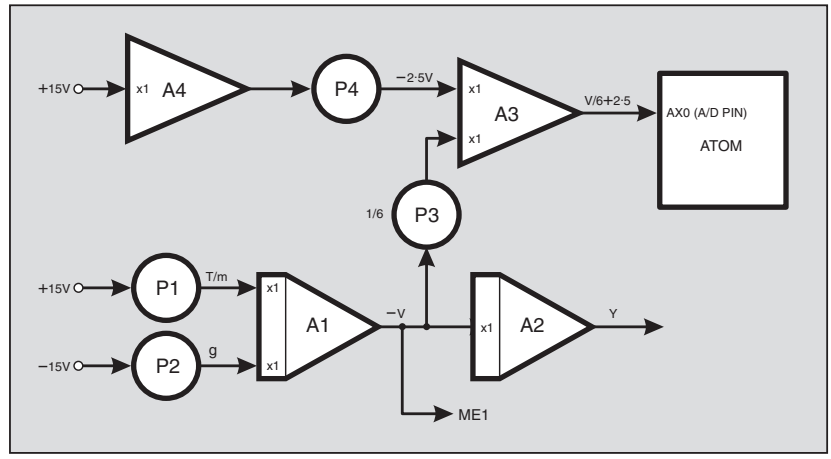


Fig.34. Flow diagram for the Harrier Jump Jet simulator.

The equation of motion according to Newton is:

$T - mg = m \times a$
(force = mass × acceleration)

where:
T = engine thrust
mg = aircraft weight
m = mass
a = acceleration

The analogue computer program in Fig.34 can solve this equation.

Amplifier A1 integrates $T/m - g$, which is the acceleration of the aircraft and outputs $-v$, which is the velocity inverted. Amplifier A2 integrates the inverted velocity to produce the altitude of the aircraft. The other two amplifiers and potentiometers are necessary to scale the velocity and present it to ADC pin AX0 of the ATOM, in the range 0V to +5V.

The output of amplifier A1 can swing between –15V and +15V. Potentiometer P3 divides this output by six (value is now between ±2·5V), and applies this signal to the input of amplifier A3. Amplifier A4 and potentiometer P4 are used to produce a reference voltage of –2·5V which is added to the signal at the other input of A3.

You can see now that as A3 also inverts the signals the output of A3 is in the range that we require, i.e. 0V to 5V. The ATOM now converts the analogue signal to a digital value and transmits it through the serial port to the PC for processing. The PC program can now take this value and scale it back to the original value, by subtracting 2·5 and multiplying by 6.

To run this program, complete the wiring on the patch panel and write the following ATOM program.

```
;program to convert analogue speed to digital and transmit it to PC
;variable definitions
        speed VAR byte
;A/D sampling definition
        clk con 2
;main program
        main
;A/D pin AX0, right justified
        adin AX0,CLK,AD_RON,speed
;send value to PC
        serout s_out,i9600,[speed]
        goto main
end
```

For the graphics and animation now use a Visual Basic program that has been specially written to work as follows:

Fig.35. Harrier Jump Jet setup program.

www.eGmag.com

The initial PC screen (Fig.35) shows a landscape with mountains, a cloudy sky and the Harrier on the ground near the control tower. As we apply power to the engines the aircraft takes off vertically and moves up the screen until it reaches a point three quarters of the way up.

At this point and if the aircraft is still gaining altitude, the picture of the aircraft remains stationary and the background picture begins to move down giving the impression of movement of the aircraft. If the thrust of the engines is reduced, the reverse movements occur, with the aircraft moving down until it lands.

The representation of the initial screen picture in Fig.35 also includes an animated instrument panel showing speed, height and other information.

This program is supplied with this project's software. With very little code, a realistic flight simulator has been produced, which communicates with the Hybrid Computer for input information, and which produces graphics with animation.

The program is "standalone" and does not need VB itself to be loaded on the PC. The program's source code is included for those who have a suitable version of VB which can handle serial communications and wish to experiment.

Before running the program, connect the patch panel, connect the serial cable to the PC's COM port, switch the Hybrid Computer on, adjust potentiometer P1 to give a high enough thrust for take off, and click **START** and **ENGINES ON**. After take off, adjust P1 to gain height, hover, and finally to land.

Try to make a smooth landing as you will receive a landing report. At all times the instrument panel will display information about the altitude and speed of the aircraft.

Just for comparison, included with the software is a purely digital version of the same program which does not need the Hybrid Computer to work. All the calculations are done by the PC program. You can download and run this program on your PC before you build your Hybrid Computer, to get a taste of what to expect.

To operate this program after double clicking on the file, click **START** and **ENGINES ON**, and apply thrust by pressing the numeric keys (0 to 9). Keying 9 gives maximum thrust whereas 0 gives no thrust. A value around 4 to 5 achieves hover.

ADDITIONAL CONSIDERATIONS

It was mentioned previously that analogue computers have the disadvantage of a limited voltage range of operation, in our case ±15V. Had we decided, in the Harrier example, to transmit the height to the PC for processing, then we would be limited to a flight from ground to an altitude of 30m if we assume 1V to represent one metre in height.

This problem was avoided by transmitting the speed and letting the PC work out the height. This way +15V represents a speed of 54,000km/hour (more than enough range!), if we assume 1V to be equal to 1m/s.

There is another solution to the problem of limited voltage range which is called Amplitude Scaling. This involves working out additional multiplication factors which are applied to the inputs of amplifiers.

For example, assume for convenience that our voltage range is ±10V, and that we wanted to investigate the flight of the Harrier up to a height of 1000m. We need, therefore, to apply a scale factor of 10:1000 or 1:100, i.e. 0·01. Similar scale factors are worked out for the velocity and acceleration, say 0·2. The equation of motion now becomes:

$$0{\cdot}2 \times a = 0{\cdot}2 \times \left(\frac{T}{m} - g\right)$$

Another consideration which crops up with analogue computers is the need for time scaling, because the solution of differential equations may extend over periods of time ranging from microseconds to many hours. Apart from the fact that it is not convenient to record results that occur within a split second, or results that take many hours to produce, errors occur at high speeds (or frequencies), like phase shifts in computing elements and measuring instruments. Also, at very low speeds, error voltages tend to build up when integrated. Time scaling involves reducing or increasing the gain of Integrators. Time is not involved in the operation of Adders.

SPECIAL TECHNIQUES

Many special circuits and techniques are also used with analogue computers. Some of these involve the use of diode shaping circuits to produce non-linear functions. In physical systems we often find such effects as backlash, Coulomb or dry friction, dead space, etc. Expanding on these topics, though, is beyond the scope of this article.

Another useful unit used with analogue computers is the analogue multiplier which is capable of multiplying two variables. Four quadrant multiplier chips are readily available, but this too is beyond the scope of this article.

The *EPE* Hybrid Computer is a very powerful and versatile tool. As the flight simulator program shows, the programmer can get both aspects of the system, the analogue and the digital, to work together for best results. Moreover, the machine is a great development tool, thanks to the powerful on-board ATOM microcontroller. The ease with which one can write and load programs is a great bonus. We are sure electronics enthusiasts will find many uses for this design.

Petros Kronis is the Engineering Science Teacher at the A' Technical School, Limassol, Cyprus.

RESOURCES

The VB6 software for this project is available for free download from the *EPE* ftp site, or on CD-ROM (for which a charge applies) from the *EPE* Editorial office, see the *EPE PCB Service* page for details. Software for the ATOM can be supplied on CD-ROM when you buy this microcontroller (see this month's *Shoptalk* page for details) or can be downloaded from **www.basicmicro.com**.

SUGGESTED BOOKS

Introduction to Analogue Computers, Technical Education & Management Inc., Foulsham. ISBN 05720027895.

Introduction to Electronic Analogue Computers, C. A. A. Wass, Kenneth Charles Garner, Pergamon Press. ISBN 0080110711.

Design and Use of Electronic Analogue Computers, C. P. Gilbert, Chapman & Hall. ISBN 0412074605.

Analogue and Hybrid Computers, I. V. Borsky, J. Matyas, Iliffe. ISBN 0592017079.

Analogue Computers, Michael Brand, Timothy Eduard Brand, E. Arnold. ISBN 0173122552.

INTERNET LINKS

From the internet and using a search engine such as **www.google.com**, type the words "analogue computer" and search to find a lot of information.

www.science.uva.ul/faculteit/museum/analog computers/ gives details of an analogue computer aircraft simulator at the Royal Aircraft Establishment.

www.dcoward.best.vwh.nct is an analogue computer museum with pictures of various machines.

Please Note: Part 1 Comp. List, the OPA177 is a *single* precision op.amp i.c. □

www.epemag.com