5

Electronic Design

Knowing nothing more than the rudiments of circuit theory, it is possible to use catalog components to design amplifiers, filters, discriminators, and even an elementary analog-to-digital converter.

Those rudiments must include knowledge of how to calculate values for the combination of passive components (resistors, capacitors, inductors) in series and parallel and the analysis of circuit loops by Kirchhoff's methods. Familiarity with Norton and Thevenin's theories would also help.

The catalogs are full of semiconductor devices, many costing no more than 10¢ and few costing more than \$5. They range from elementary amplifiers and logic circuits, through comparators, counters, multichannel ADCs, power transistors, magnetic and optical sensors, and many of the embeddable microcomputers.

In the same catalogs you are likely to find plug-in systems to solve your problems. These, however, are likely to cost a hundred times as much. Being designed to address the problems of a hopefully large client base, it is unlikely that they will be the best fit for your specific application.

It is well worth gaining expertise in putting your own circuits together.

5.1 THE RUDIMENTS OF CIRCUIT THEORY

Circuits can be arranged as networks, as in Fig. 5.1, nodes joined by meshes to form loops. The laws and equations can be expressed in many ways, but can be summed up as described in the following paragraphs.

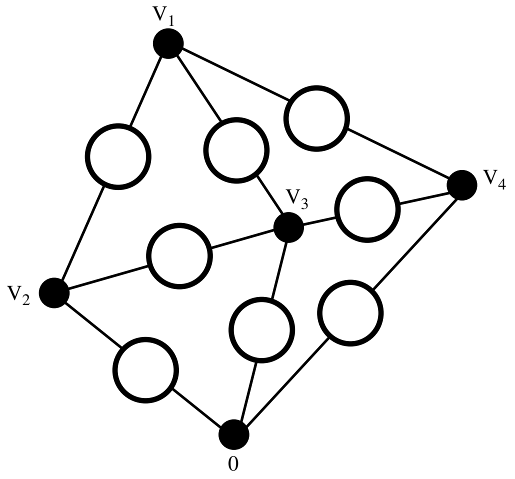


Figure 5.1 *Circuit network.*

Each node has a voltage with reference to the node 0 that is taken as reference.

Each mesh carries a current between nodes, taking the direction and sign into account. The first and only rule states that the sum of all currents at each node must be zero. In other words, all currents must "come from somewhere." (Kirchhoff's "second rule" is a simple consequence of assigning a voltage to each node.)

To analyze the circuit, we must therefore have some way of relating the current in a mesh to the difference between the voltages of the nodes at its ends. We have a number of primitive components for this analysis:

1. *A Voltage Source.* This will be a voltage that contributes to the voltage difference across the mesh regardless of the current passing through it.
2. *A Current Source.* Rather more aggressive, perhaps, the current source forces its current in the mesh regardless of the voltage across it. Clearly the resulting equation will express the voltage across the mesh in terms of this current, rather than vice versa.
3. *Resistance.* Ohm's law states that the voltage across the resistance is proportional to the current through it, but this is more a rule of thumb than a law of physics. With a wide variation in current resulting in temperature changes in the components, nonlinearities are likely to become apparent. As soon as semiconductors are involved, the nonlinearities become important.

   Resistance "contaminates" most other components, so that they come in combination with it. A voltage source is seldom "pure," but appears to have a resistance in series with it. A current source will usually appear to have a resistance across it.

4. *Capacitance.* Capacitance has a somewhat different nature. It involves time. The current through a capacitor is proportional to the rate of change of the voltage across it, so the equation for a mesh is then likely to turn out to be a state equation.
5. *Inductance.* Inductance also involves time, but this time the voltage across the inductance is proportional to the rate of change of the current through it. Once again, an inductance is seldom "pure" but usually has some resistance associated with it.

Electricians have assembled a set of rules for combining components in series and parallel. When current passes through two resistors in series, the voltage across them is, of course, the sum of the two individual voltages. The combined resistance is thus the sum of the two resistances:

$$R = R_1 + R_2$$

When they are connected in parallel, both resistors have the same voltage across them and pass the sum of the two individual currents:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

Using these two simple rules, a circuit consisting only of resistors can be "crunched" first by working out a single resistance for each mesh, then combining meshes in parallel.

Thevenin's theorem states that a circuit, however complicated, can be reduced to a single resistance in series with a single voltage source. Norton turns this on its side and says that this is equivalent to a single resistance (of the same value) in parallel with a single current source. Both theories, of course, depend on the circuit components being linear.

Flushed with their success at manipulating resistors, electricians are keen to represent capacitors and inductors in the same form. Often their calculations will involve a single supply frequency, which they multiply by 2π to get the *angular frequency*, ω. The sines and cosines of the waveform can be expressed as the real and imaginary parts of $e^{j\omega t}$, and every differentiation will give rise to a multiplying factor of $j\omega$. They call an inductance L an *impedance* of $Lj\omega$ and crunch it just as if it were a resistance.

The ratio of output to input voltages of a network is likely to involve a spattering of $(j\omega)$ symbol, and the result is called a *transfer function*. Their euphoria with transfer functions is carried over into their dealings with control systems, and their influence is to blame for the tradition of treating transfer functions as the foundation for the teaching of control theory.

One particular form of network of particular interest is the *two-port* or *four-terminal* network (Fig. 5.2). The two output terminals can be tied to the

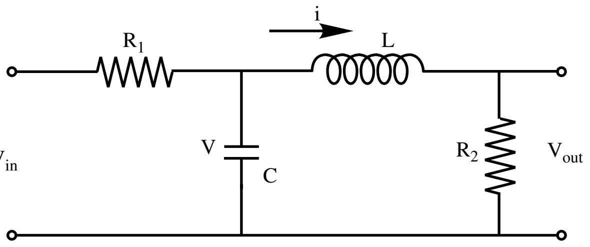


Figure 5.2 *Four-terminal network.*

two input terminals of another such network; indeed, a whole chain of networks can be linked to form a complex filter.

Four variables are involved, the input voltage and current and the output voltage and current. The equations linking them can be found by "network crunching," or more simply by building state equations. Consider the circuit of Figure 5.2.

The state variables are V, the voltage on the capacitor; and i, the current through the inductor. The voltage across R_1 is $V_{in} - V$, so the current through it is

$$\frac{V_{\text{in}} - V}{R_1}$$

The current through the inductor is, of course, just i, but is directed away from the capacitor. The currents arriving at the top node of the capacitor, including the current passing through the capacitor, must sum to zero. The current flowing into the capacitor is equal to C times its rate of change of voltage:

$$C\frac{dV}{dt} = \frac{V_{\text{in}} - V}{R_1} - i$$

Now the current through R_2 will be $i - i_{out}$, and the voltage across it is

$$R_2(i - i_{\text{out}})$$

so we can find the voltage across the inductor. The equation that defines an inductance gives us

$$L\frac{di}{dt} = V - R_2(i - i_{\text{out}})$$

and there we have our two state equations. Everything on the righthand side is either a state variable or an input—note that both V_{in} and i_{out} are inputs, as far as this system is concerned.

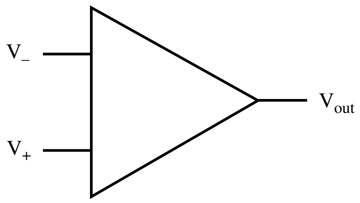


Figure 5.3 *Operational amplifier.*

5.2 THE OPERATIONAL AMPLIFIER

This type of integrated circuit seems as versatile as the transistor itself, but allows amplifier and signal conditioning circuits to be designed to a "grand scheme" with few of the niggling details of biasing that transistors present.

The output voltage V_{out} (see Fig. 5.3) is a large number of times the difference between the two input voltages—where the large number may typically be 100,000. But it is most unlikely that you will want such a high gain in your final circuit. The gain is reduced when you apply feedback around the amplifier, in exchange for certainty about the closed-loop circuit parameters.

Modern op-amps (operational amplifiers) have a very high input impedance. That means that when you apply a voltage to the input, practically no current at all is accepted. They also have a very small "input offset voltage," the actual difference in voltage between the inputs when the output is zero.

5.2.1 Virtual Earth

A large family of applications use a "virtual earth" concept, where the non-inverting input of the amplifier is connected to ground, or to a constant reference voltage.

Now let us start to analyze the circuit, even before we decide what the circuit will actually consist of. Since

$$V_{\text{out}} = A(V_+ - V_-)$$

and since we have grounded V_+, we know that

$$V_- = -V_{\text{out}}/A$$

and since A is very large, that will make V_- virtually zero, unless the amplifier is saturated. Hence the term "virtual earth."

Now, for example, if we connect a resistor of 100 kΩ between V_{out} and V_- (as shown in Fig. 5.4), both V_{out} and V_- will be zero. Let us connect a second resistor to V_-, this time of value 10 kΩ, and apply one volt to the end of it. What does this do to V_-?

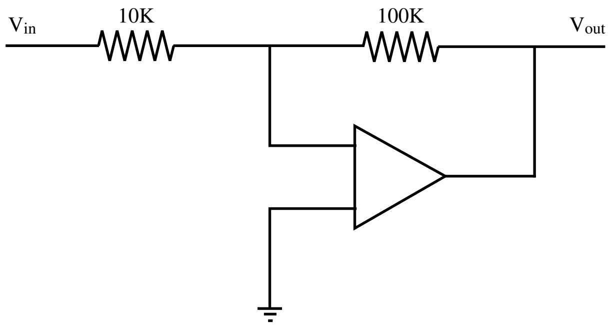


Figure 5.4 *Amplifier with virtual earth.*

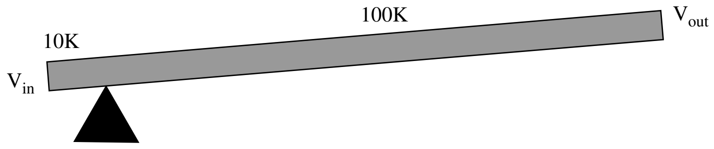


Figure 5.5 *Amplifier seesaw analogy.*

As soon as V_- is pulled away from zero volts, V_{out} changes to try to pull it back. By taking a value of −10 V, the two resistor currents cancel out and balance is restored. We have made an amplifier with a gain of −10; thus, for every volt applied at the input to the 10 kΩ resistor, the output will change by −10 V.

You can look on the schematic rather as a seesaw (Fig. 5.5), with one side of the plank 10 times as long as the other.

Let us now connect an additional 10 kΩ resistor to V_- and apply −2 V to it. Now the sum of all three currents at the V_- junction must be zero, while the voltage there remains at zero. The output must change to +10 V. We have devised a way to add input signals together, although those signals do not "see each other" since they are only joined via a point that remains at zero volts. The virtual-earth connection has become a "summing junction" (see Fig. 5.6).

We can add signals in unequal proportions by varying the values of their corresponding resistors. If we want to subtract a signal, we can first invert it using an operational amplifier with an input resistor equal to its feedback resistor.

If we replace the feedback resistor with a capacitor, we get an *integrator* (Fig. 5.7). As before, the currents at the summing junction must add to zero. Now the equation that describes the current i in a capacitor is

$$i = C\frac{dV}{dt}$$

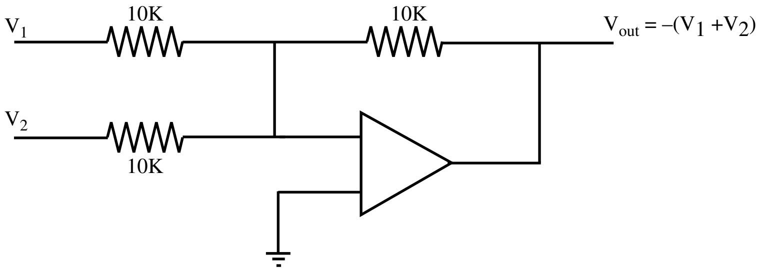


Figure 5.6 *Summing amplifier.*

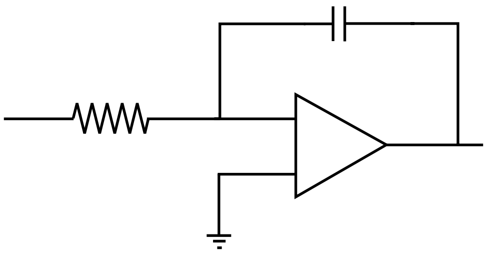


Figure 5.7 *Integrator with capacitor feedback.*

So we have

$$\frac{V_{\text{in}}}{R}+C\frac{dV_{\text{out}}}{dt}=0$$

or in other words

$$\frac{dV_{\text{out}}}{dt}=-\frac{V_{\text{in}}}{RC}$$

The derivative of V_{out} is proportional to V_{in}, so V_{out} is proportional to the integral of V_{in}.

Now we have all that we need to make an analog computer. With it, we can simulate linear systems, or add some circuit dodges to simulate nonlinear ones. This was the method of choice in the mid 1970s, but today it is so much easier to simulate a system digitally.

This does mean, however, that with some operational amplifiers and a handful of capacitors, we can produce virtually any transfer function. This could be all that we need to stabilize a difficult system.

5.2.2 Other Configurations

Instead of tying V_+ to ground, we can apply the input signal to it (see Fig. 5.8). The virtual-earth configuration has the disadvantage that the amplifier

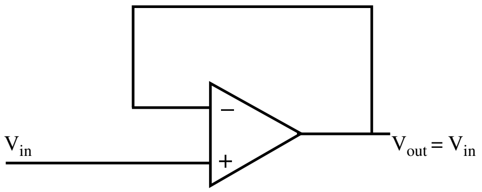


Figure 5.8 *Noninverting buffer amplifier.*

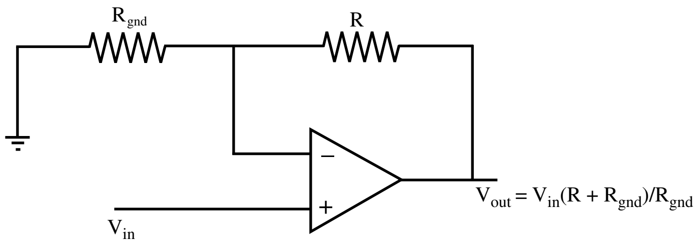


Figure 5.9 *Noninverting amplifier with gain.*

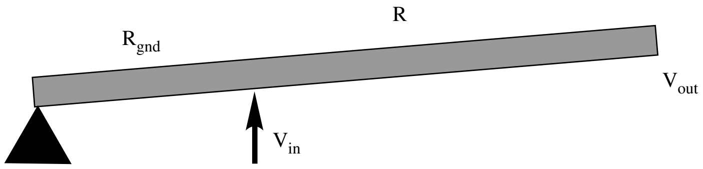


Figure 5.10 *Seesaw analogy becomes a lever.*

has an input impedance that is no greater than the input resistor. Some signal sources should not be loaded, even by this sort of resistance value.

With the signal applied to the noninverting input, however, the input impedance is extremely high. Via the feedback, the output will drive the inverting input to match V_+, so the input impedance will be many times larger than that of the op-amp itself.

To avoid any risk of oscillation, it may be preferable to apply this feedback via a resistor, rather than a direct connection. By connecting a second resistor between V_- and ground, we can have high input impedance combined with some noninverting gain (Fig. 5.9). The "seesaw" principle becomes a lever (Fig. 5.10) and the gain is seen to be $(R + R_{gnd})/R_{gnd}$.

The circuit shown in Figure 5.9 is the same one we used in the ball-and-beam experiment. When we add a capacitor between V_+ and ground, if the ball has broken contact with the track, the voltage on V_+ will decay very slowly due to the high input impedance. As soon as the ball makes contact with the track again, the effect of the capacitor is only a very short time constant.

As with the integrator, the time constant is given by the value of *RC*. If our capacitor is chosen to be one microfarad and if the track resistance is 10 Ω, the result will be 10 μs, which is much too small to have any effect on the system's performance.

5.2.3 Differential Amplifier

Some signals, such as strain-gauge outputs, appear as small differences between two voltages that are not close to ground. Other signals might have mains noise superimposed on the pair of leads that we are trying to measure. In these and many other cases we would like an amplifier that has good *common-mode rejection*, which will amplify the difference voltage and not respond to signals that vary both lines together.

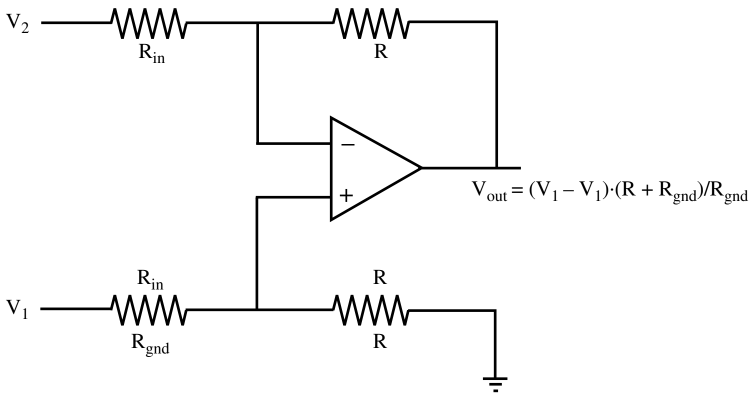


Figure 5.11 Differential op-amp circuit.

The differential circuit shown is Figure 5.11 is the answer. If we ground input *A* and vary input *B*, we see that the "seesaw" gain of R_2/R_1 applies to the difference of the inputs. But if we tie inputs *A* and *B* together and vary them, we see that the voltages on V_+ and V_- remain equal if V_{out} remains at zero. The common-mode gain is thus zero.

5.3 FILTERS FOR SENSORS

In addition to amplifying sensor signals, we may need to process them in other ways. One requirement might be to remove high-frequency noise.

5.3.1 Antialiasing

It is, of course, possible to take averages of digitized readings to smooth out some types of noise, but the digitizing process itself suffers from *aliasing* (Fig.

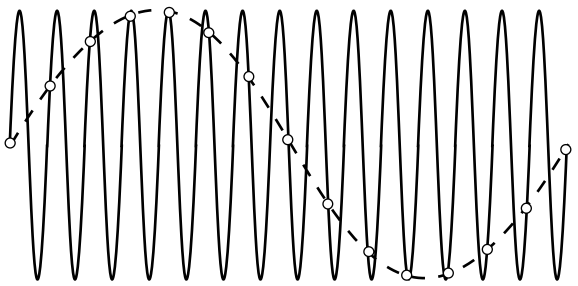


Figure 5.12 *Example of aliasing.*

5.12). As frequencies rise above half the sampling frequency, the sampled signal can appear to drop in frequency as the actual frequency goes up. The phenomenon is similar to the effect in old Western movies when the wagon wheels appear to turn backward.

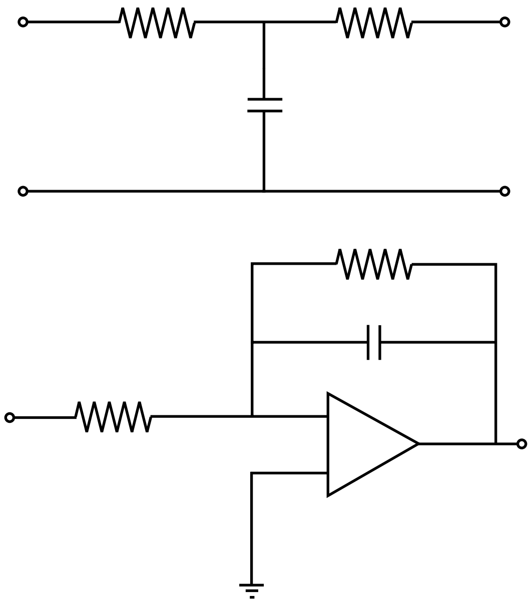


Figure 5.13 *Two simple lowpass filters.*

The only way to eliminate the high-frequency noise is to attack it before digitizing, using an *antialiasing filter*. This can simply be a lowpass *RC* circuit (e.g., see Fig. 5.13), an op-amp with feedback consisting of a resistor in parallel with a capacitor, or a higher-order filter with several capacitors.

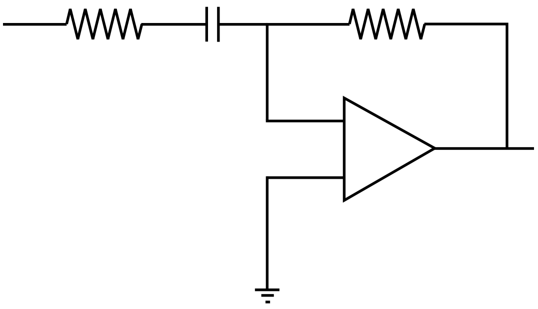


Figure 5.14 Differentiator circuit.

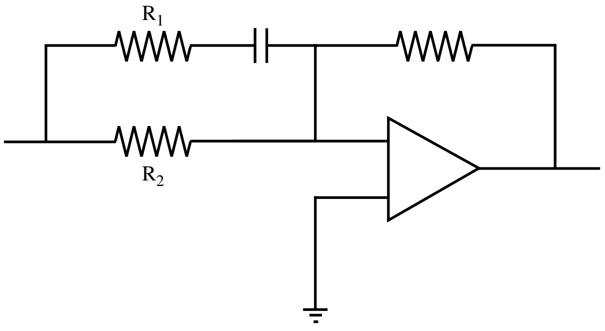


Figure 5.15 Phase advance circuit.

5.3.2 Differentiating and Phase Advance

In principle we could swap the resistor and capacitor of an integrator to form a differentiator. In practice, the gain increases indefinitely at high frequencies, so the output would be swamped with noise. We have to add a resistor in series with the capacitor (see circuit in Fig. 5.14). The output is a lowpass version of the derivative, with a limit on the high-frequency gain.

When we add this estimated derivative to the original signal, as when adding an estimated velocity to a position signal, the result is a *phase advance* (see circuit in Fig. 5.15).

At high frequencies, the capacitor can be regarded as a short circuit, while at DC it acts as an open circuit. The ratio between the gains at high and at low frequencies is thus $(R_1 + R_2)/R_1$.

5.3.3 Switched Filters

Possibly the signal that we seek lies is modulated by an alternating voltage. Many sensors use AC signals, such as the *E and I pickoff* (Fig. 5.16) used in the past for aircraft instrumentation and the *linear variable differential transformer* (LVDT) (Fig. 5.17). The output is an AC signal that is zero at a central

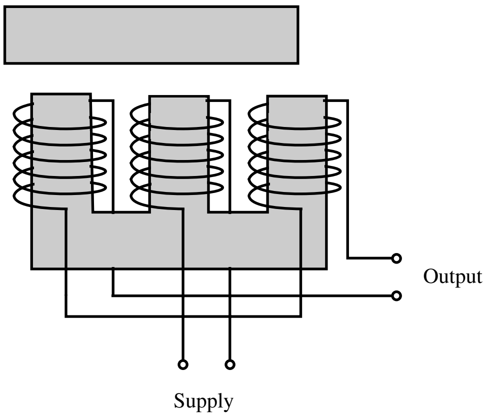


Figure 5.16 *E and I pickoff.*

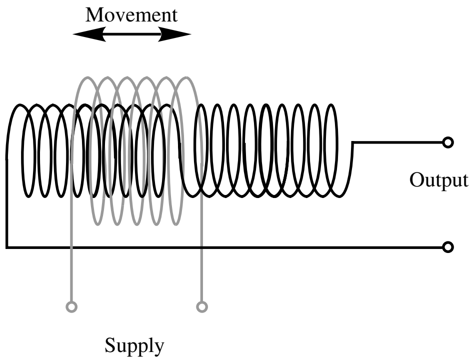


Figure 5.17 *Linear variable differential transformer.*

position, increasing with opposing phases as the sensor is displaced on either side of the center.

To convert such a signal to a DC one that can vary positive and negative, we require a *phase-sensitive discriminator.* For this, we can use a semiconductor switch.

A circuit such as the CMOS 4066 depends on field effect transistors to close or open connections between pairs of contacts. The control signal is designed to be operated by the output of a PC logic line or a microprocessor, but here we use the oscillator supply for the switching signal (see Fig. 5.18).

5.3.4 A Single-Chip ADC

In Chapter 3, many of the experiments depended on the availability of an ADC. If you have no suitable card, a four-channel or eight-channel ADC can

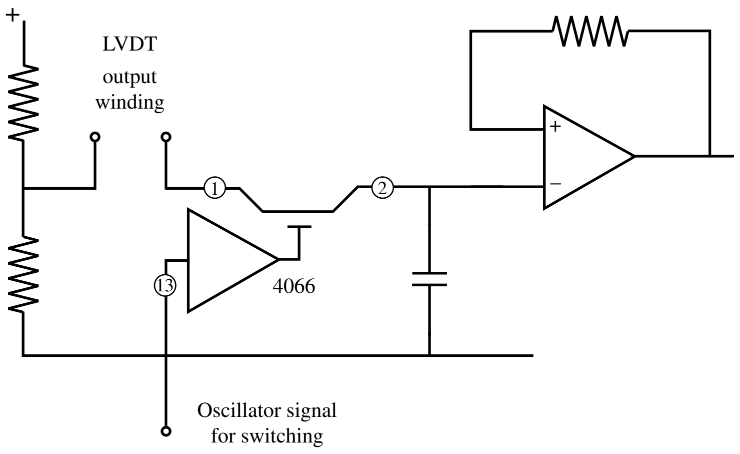


Figure 5.18 *Half-wave switched discriminator.*

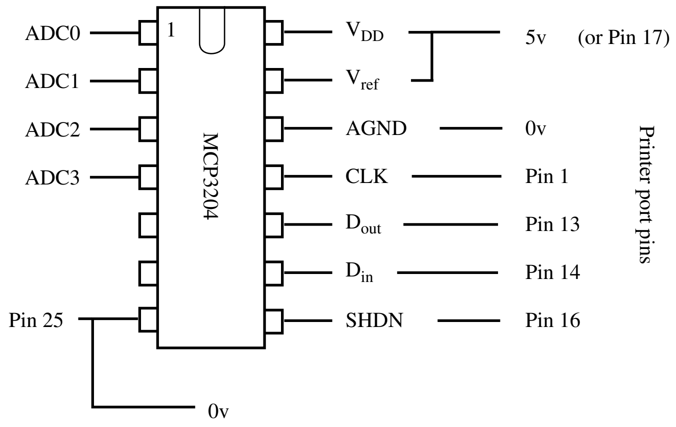


Figure 5.19 *Circuit and printer port connections.*

be built from a single chip, connected to the parallel port of the PC with no additional electronic components.

The MCP3204 and MCP3208 chips from Microchip Technology Inc. are designed to communicate using a serial technique. Data bits are clocked in and out in response to a clock signal generated by the computer to which the chip is connected.

When the "shutdown" line is pulled to ground, the chip is selected. Data on the D_{in} line is clocked into the chip to set the channel number to be converted. After an extra clock cycle, the chip starts to output the 12-bit result on the D_{out} line, most-significant bit first. When the 12th bit has been received, the computer sets the shutdown line high to reset the chip before the next conversion. (See configuration in Fig. 5.19.)

Just one input line is needed, so one of the control bits of the printer port can be used. Three output lines from the computer are needed for chip select, clock, and D_{in}. In fact, a fourth line could be used here, since the chip takes so little current that it could be powered by a further data bit. However the voltage on that data bit might be too rough for our required accuracy.

For the code that follows, the connections have been made in the same sequence as tabulated here:

MCP3204		Printer Port Pin, Function	
14	V_{DD}	17	-select
13	V_{ref}	17	-select
12	AGND	25	ground
11	CLK	1	-strobe
10	D_{out}	13	printer present
9	D_{in}	14	-auto LF
8	SHDN	16	initialize
7	DGND	25	ground

The four analog inputs are connected to pins 1, 2, 3 and 4.

The code is not very elegant, but I hope that it is clear to follow.

```
CONST pdata = &H378      'Address of the printer port
CONST pinp = &H379       'associated input register
CONST pout = &H37A       'and control register

'MCP function            Printer function
' On the pinp port:
CONST DOUT = &H10        'printer present

'On the pout port:
CONST CLKbar = 1         '-Strobe
CONST DINbar = 2         '-auto linefeed
CONST SHDN   = 4         'initialize
CONST VDDbar = 8         '-select

CONST CLKlo = CLKbar + DINbar 'VDD high, shutdown low
CONST CLKhi = DINbar          'same with CLK high

'Demonstration program:
CLS
SCREEN 9
WINDOW (0, -1)-(1000, 1)
DO
   FOR i = 0 TO 1000
      PSET (i, ADC(0))
   NEXT
CLS
LOOP UNTIL INKEY$ <> ""

FUNCTION ADC (chan%)
DIM i AS INTEGER, bits AS INTEGER

bits = 0

OUT pout, CLKlo           'VDDbar=0, shutdown=0

OUT pout, CLKlo - DINbar  'Start bit must be 1
OUT pout, CLKhi - DINbar  'clock it
OUT pout, CLKlo - DINbar  '(subtract DINbar to set DIN
                          high)

OUT pout, CLKlo - DINbar  'First bit 1 for single ended
OUT pout, CLKhi - DINbar  'clock it
OUT pout, CLKlo - DINbar

OUT pout, CLKlo   'For 3204, don't care - so output zero
OUT pout, CLKhi   '(for 3208, same as MSB of address)
OUT pout, CLKlo
IF (chan% AND 2) THEN     'High bit of channel number
   OUT pout, CLKlo - DINbar
   OUT pout, CLKhi - DINbar
   OUT pout, CLKlo - DINbar
ELSE
    OUT pout, CLKlo
    OUT pout, CLKhi
    OUT pout, CLKlo
END IF

IF (chan% AND 1) THEN     'Low bit of channel number
    OUT pout, CLKlo - DINbar
    OUT pout, CLKhi - DINbar
    OUT pout, CLKlo - DINbar
ELSE
    OUT pout, CLKlo
    OUT pout, CLKhi
    OUT pout, CLKlo
END IF

    OUT pout, CLKlo
    OUT pout, CLKhi    'The chip samples input now
    OUT pout, CLKlo
    OUT pout, CLKhi   'extra clock was needed

    OUT pout, CLKlo   'null bit starts now
    OUT pout, CLKhi

FOR i = 0 TO 11
    OUT pout, CLKlo   'next bit ready now, MSB first
    bits = 2 * bits + (INP(pinp) AND &H10) \ &H10
    OUT pout, CLKhi
NEXT

ADC = (bits XOR 2048) / 2048! - 1
OUT pout, SHDN + CLKbar + DINbar   'shutdown high

END FUNCTION
```

5.3.5 A More Rudimentary ADC

For serious applications, it would probable be advisable to use a commercial ADC chip or to use the ADC capabilities of an embeddable microcomputer. To support the experiments of Chapter 3, however, it is possible to construct a converter from very simple components and use the power of software. That, after all, is part of the essential art of mechatronics.

The heart of the device is a capacitor that charges at a steady rate, to give a ramp of voltage. An output bit from the computer first drives a transistor switch that discharges the capacitor and holds it at zero voltage. An output command frees it to ramp upward to approach the supply voltage. Four comparators on a single chip compare the ramp against three inputs and a reference voltage. These outputs are connected to the bits that serve as input bits on the printer port, where they are monitored in a counting loop. The loop ends when the ramp passes the reference voltage. The circuit is shown in Figure 5.20.

Take care when ordering components. The traditional package for small integrated circuits was *dual inline* (DIL). Here the two rows of pins are spaced at 0.1 in. intervals and the component can easily be mounted and soldered in a *project board*. For hand assembly, you must be sure to obtain DIL components.

Manufacturing has now turned almost completely to *surface-mount* technology, where, instead of pins, the component has a row of small metal pads that bond to the surface of the printed-circuit board when dabs of solder paste are heated and melted.

If you have faith in your circuit skills—and if your computer is not a cherished favorite—you can obtain 5 and 12 volt supplies from inside it. Take them from one of the spare power connectors intended for plugging into disk units.

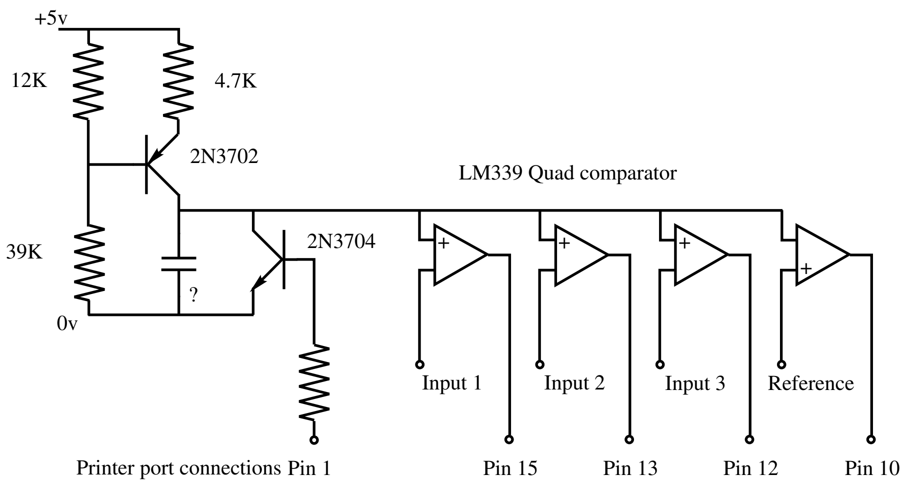


Figure 5.20 *Circuit of ramp ADC.*

As always, there are drawbacks. The PC, even in DOS mode, is interrupted frequently to service "housekeeping" interrupts, including updating the clock that controls the TIMER function. If one of these interrupts happens during the ramp, the wrong answer will be given. However, the count given for the reference voltage will also be less that it should, so a second attempt can be made or the previous value retained.

The software is as follows. First we have the three addresses for the printer port and the bit definitions, corresponding to the connector pins:

```
CONST pdata = &H378
CONST pinp = &H379
CONST pout = &H37A

' On port pinp:
CONST pin15 = 8          'True if high, used for error
CONST pin13 = &H10       'True if high, printer present
CONST pin12 = &H20       'True if high, out of paper
CONST pin10 = &H40       'True if high, ack
CONST pin11 = &H80       'True if low,  -busy

'On port pout:
CONST pin1 = 1           'True gives low , Strobe
CONST pin14 = 2          'True gives low, Auto linefeed
CONST pin16 = 4          'True gives high, initialise
CONST pin17 = 8          'True gives low, select
```

We define some shared variables and set up some other values:

```
DIM SHARED count AS INTEGER, mincount AS INTEGER
DIM SHARED Table(15)AS INTEGER
DIM SHARED Adc(3), bins(3,3)
FOR i% = 0 to 2
   For j% = 0 to 3
      READ bins(i%, j%)
   NEXT
NEXT
DATA 0, 2, 4, 6
DATA 0, 1, 4, 5
Data 0, 1, 2, 3
```

After calling `DoADC`, we find the values in an array `Adc()`—the syntax looks the same as when `Adc()` was a function:

```
SUB DoADC
FOR i% = 0 to 15
   Table(i%) = 0
Next
count = 0
OUT pout, pin1                          'release ramp
DO
   x% = INP(pinp)\8                     'Whole number
divide
   Table(x% AND 15) = count
   count = count + 1
LOOP UNTIL x% AND pin10                 'ramp crosses
                                        reference

OUT pout, 0                             'reset ramp
IF count < mincount THEN EXIT SUB       'leave old results
FOR i% = 0 TO 2
   k% = Table(0)
   FOR j% = 1 TO 3
      IF Table(bins(i%, j%)) > k% THEN
         k% = Table(bins(i%, j%))
      ENDIF
   NEXT
   Adc(i%) = 2 * k% / count - 1
NEXT
END SUB
```

So, how does it work? Before the ramp crosses the reference voltage, each time we input from `pinp` and divide by 8, we will get an answer between 0 and 7. We store the value of count in the corresponding bin. When we have finished "filling the bins," we look at what we have caught.

Any time that the ramp has not crossed input channel 0, the first bit of the bin number will not be set. The bin number will therefore be 0, 2, 4, or 6. So, if we look for the largest value in these four bins, we will find the value of count just before the status of pin 15 changed.

Similarly, the value for channel 1 will be given by the largest value in bins 0, 1, 4, and 5, where the bit of value 2 in the bin number is not set.

Now all that remains is to scale the answer to the range –1 to 1 and store the result in the `Adc` array.

Before starting any interrupts, we should call `DoAdc` several times and note the largest value of count that it gives. Reduce this by, say, 5%, and save the answer in `mincount`. Since we will always `PLAY "n0"` between calls, the ramp should have had time to return safely to zero.

You have a further task to complete. You must choose an appropriate value for the ramp capacitor. Since we are using a software loop for timing, this will depend on the computer speed and on whether you are using QBasic, Quick Basic, or a compiled .EXE file. Ideally you want `count` to be about 1000, to give a resolution of 0.1%. Too low, and you lose accuracy; too high, and you risk an overflow error if `count` ever exceeds 32,767. Start small, run the program with a `PRINT count` line, and dab on extra capacitors in turn until you find something close to what you want.

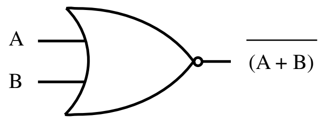
5.4 LOGIC AND LATCHES

There have been several generations of changing technology—including RTL, DTL, TTL, and CMOS—of the extensive family of logic circuits. Now the 7400 and 4000 series CMOS seem to be winning the day, but the variations are legion. Do you need “high speed,” “very high speed,” “low voltage,” “advanced,” or “Schottky”? A 2004 catalog contained 15 pages with well over 100 devices listed per page.

It can be shown mathematically that every possible logic circuit can be constructed from a combination of NOR gates. What is a NOR gate? If either of its two logic inputs is at a logic 1, near 5V, the output will be at logic 0 near ground. Only if neither one input NOR the other is a 1 will the output be high. It can be described by a “truth table” as shown on the right in Figure 5.21.

In the days of *resistor–transistor logic* (RTL), such a gate was constructed from just one transistor and three resistors (see Fig. 5.22).

Two NOR gates can be connected together to make a 1-bit memory (Fig. 5.23). If the two free inputs are low, then either output *A* is high, forcing output *B* to be low, hence allowing *A* to be high—and so on, round in a circle, or equally output *B* can be high forcing *A* to be low, and so on. If input *a* becomes high, *A* will be forced low and *B* will become high. When *a* falls again, the outputs remain in the state they were left. If output *b* becomes high for a moment, the output will “flip” into the other state—the circuit is sometimes known as a “flip-flop”.



Truth table of $\overline{(A + B)}$

		A = 0	A = 1
B	0	1	0
B	1	0	0

Figure 5.21 NOR gate and truth table.

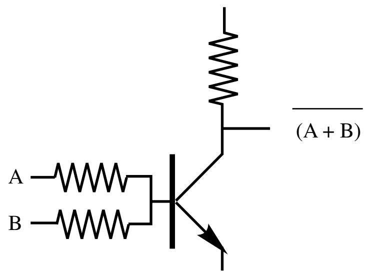


Figure 5.22 Transistor NOR gate.

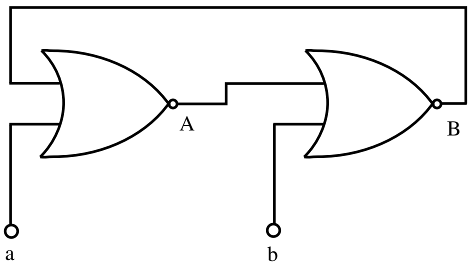


Figure 5.23 Two NOR gates as a flip-flop.

Today, serious circuits are more likely to use one of the mass of "latches" or *JK* flip-flops that fill the catalog, four or more to a chip. Even these are not likely to be found beyond a "breadboard" experimental circuit, since entire systems can be implemented by a *field-programmable gate array* (FPGA) or burned onto an *application-specific integrated circuit* (ASIC).

Logic can be analyzed by *Boolean algebra*, by which expressions such as

$$(A \text{ and not}(B \text{ or } C)) \text{ or } ((\text{not } A) \text{ and } (\text{not } B) \text{ and } (\text{not } C))$$

can be simplified for easier circuit design. Some essential rules can be deduced from common sense:

$$\begin{aligned} A \text{ and } (B \text{ or } C) &= (A \text{ and } B) \text{ or } (A \text{ and } C) \\ A \text{ or } (B \text{ and } C) &= (A \text{ or } B) \text{ and } (A \text{ or } C) \\ \text{not } (A \text{ or } B) &= (\text{not } A) \text{ and } (\text{not } B) \\ \text{not } (A \text{ and } B) &= (\text{not } A) \text{ or } (\text{not } B) \end{aligned}$$

while

$$B \text{ or } (\text{not } B) = \text{true}$$

In some computer languages, logic operators are expressed as words such as these, while others use symbols such as &, |, and !. Mathematicians grace the subject with the term *propositional calculus* and use a host of other symbols.

As an exercise, try simplifying the first expression above.

For mechatronics, logic design is likely to be a means to the end of including logic states in the controller. If there is a computer involved, it is usually easier to sidestep the problem of logic design and use software.