USING THE PIC16C84 SINGLE CHIP MICRO - 2

In the first of these articles, we introduced the PIC16C84 miniature microcontroller and described some of its features. A lot of that was dry — but important — groundwork. We can now get into the 'juicy' bit, building and using the PC-driven programmer. The programmer has been purposefully designed to be reliable, cheap and easy to build.

by CHARLES MANNING

Once you have developed some PIC software, you need to program or, to use the lingo, 'burn' it into the microcontroller's program memory. This is typically done from a host computer via a programming interface, or just 'programmer'.

The programmer described here attaches to the printer port of an IBM compatible PC, which means that it does not require any special interface cards. It can even be used with a notebook computer to make a fully portable development system.

Microchip added a new five-pin serial programming mode to the new 14-bit series of PIC chips. Previous PIC versions only supported a 17-pin parallel programming mode. The serial programming mode has two major benefits over the parallel mode: it allows in-circuit programming and, more importantly to us, reduces the complexity of the programming interface. This programmer only uses serial mode as there is no real benefit in supporting both serial and parallel modes.

Microchip also made a big move from their previous policies by making their programming specifications public. Before this they only gave out this information under a strict non-disclosure agreement.

The serial programming mode requires just five connections to the PIC. These are power (nominally 5V), ground, Vpp (12 to 14V), clock (RB6) and data (RB7).

The PIC enters programming mode when the Vpp pin rises sharply from zero to the programming voltage while the clock and data lines are held low. The PIC then remains in programming mode until it is reset by taking the Vpp pin to zero volts.

Serial data is transferred to and from the PIC via the data pin, RB7. The data transfer is controlled by sending clocking signals to the clock pin, RB6. When accepting data, the PIC latches the data on the falling edge (high to low transition) of the clock pin. When outputting data, the PIC increments to the next bit on the rising edge (low to high transition) of the clock pin.

The PIC 16C84 programming logic is controlled by eight commands, seven of which are useful for serial programming. The eighth command is used to enter parallel programming mode and is not used by this programmer. Each of these commands is made up of a six-bit sequence which is written to the PIC to select the desired action.

Most of these commands are followed by a data transfer operation, either into or out of the PIC. Data transfers are always 16 bits wide, though the effective data transfer is either 14 or eight bits wide, for program and EEPROM data transfers respectively. The control command set for the 16C84 is different from other PICs as it also supports writing and reading of the EEPROM data memory.

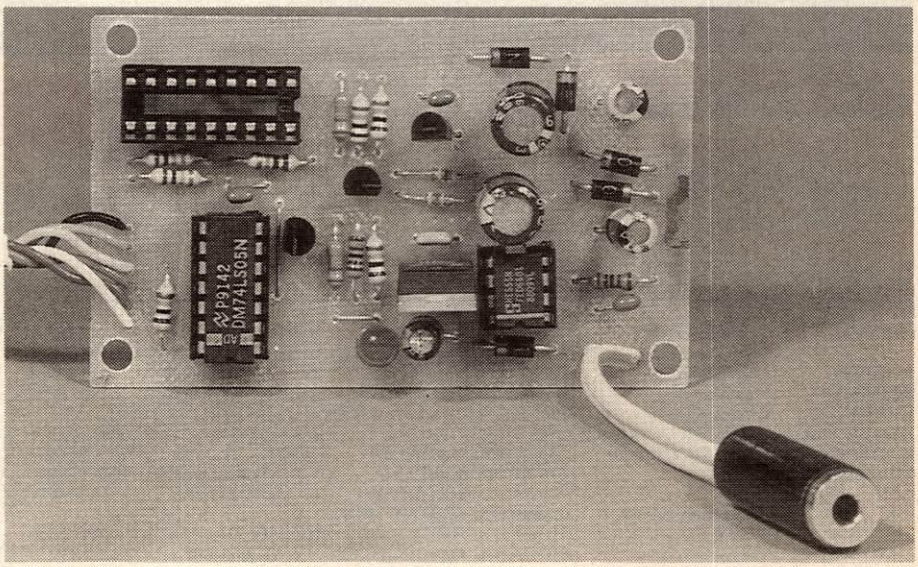
The programmer

It is possible to build a simpler programmer for the 16C84, however this could reduce its reliability. This programmer has been specially designed for the amateur and it was decided to include some extra circuitry to make it more reliable and easier to use. The circuit schematic is shown in Fig.1, while that for the matching cable to the PC printer port is shown in Fig.2.

The programmer requires an input power supply of 9 to 12V DC. Although the programmer only uses about 20mA while idle and 40mA while busy, a supply of at least 100mA is preferable.

Such power supplies are readily available and are often used to power electronic equipment. A 9V battery is also sufficient and could be used for a portable system.

A word of caution, though: many power supplies produce quite a bit more than their nominal voltage, so ensure that the voltage does not exceed 12V. With minor modifications, as detailed



One of the author's prototype boards for his PC driven programmer for the PIC16C84 microcontroller chip.

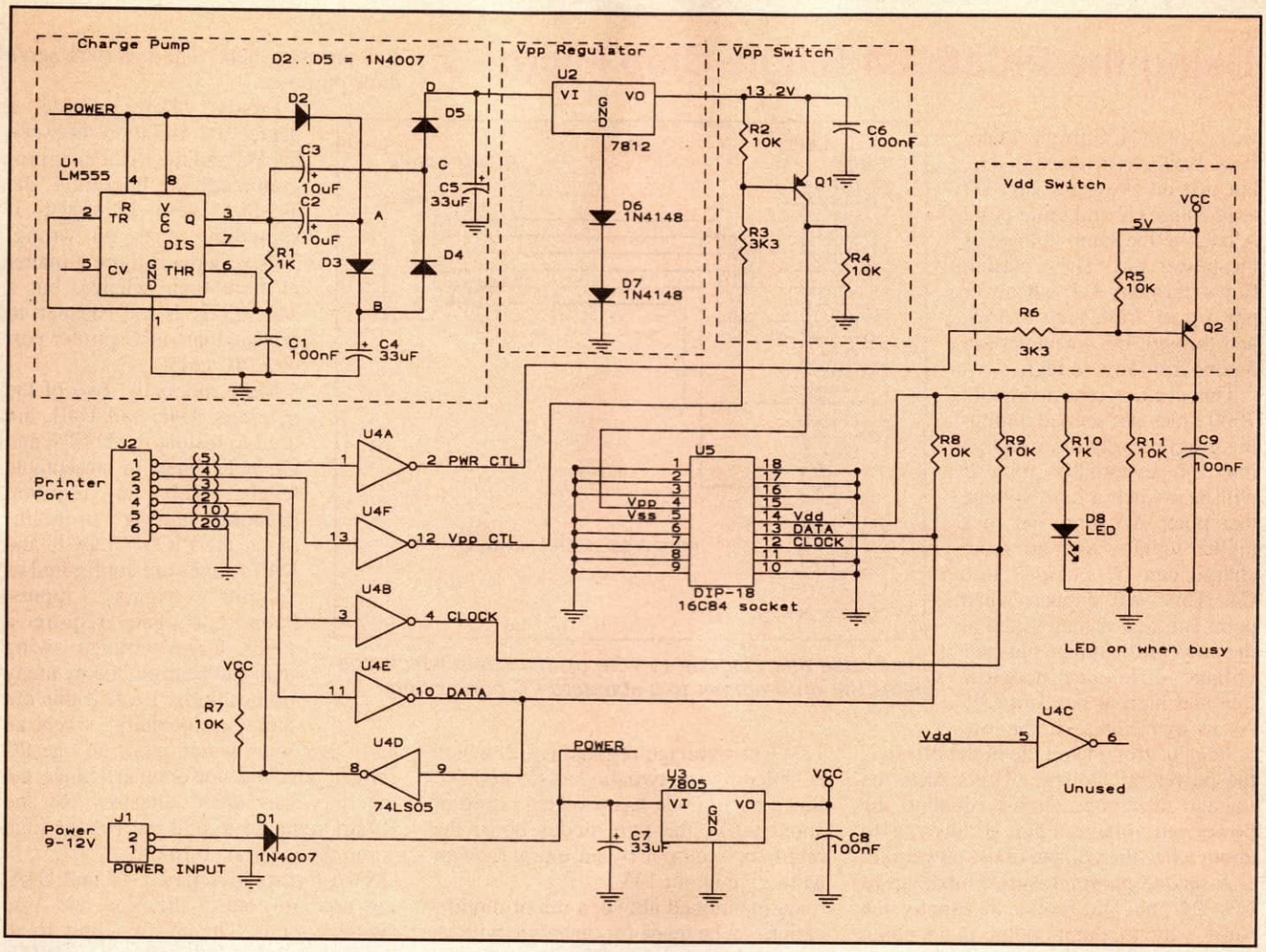


Fig.1: The schematic for the PC driven programmer. As you can see, it's very straightforward and uses a very small number of low cost components. U1 is used to generate the programming voltage.

later, the programmer will operate with an input voltage of up to 15V.

The power is fed into the programmer via a diode, D1, which prevents reversed polarity power from being fed into the circuit and perhaps causing damage to the programmer and the PC. Simple goof-proofing like this has saved many circuits. There is a voltage drop of about 0.6V across the diode.

The PIC requires two different voltage levels during programming, 5V for normal power on the Vdd pin and a programming voltage of 12 to 14V on the Vpp pin. The 5V level is easily produced from the input power by using a common 7805 type 5V fixed voltage regulator, U3.

The 13V level is more difficult to produce, since it is at a higher voltage than the input voltage. This is overcome by using a 'charge pump' voltage converter. The programmer could have been kept a bit simpler by using a higher input voltage of 18V or so. The disadvantage of this approach is that 18V power supplies are not commonly available. Besides, this way provides an excellent opportunity to learn about a simple way to boost power supply voltages.

Charge pump (or flying capacitor) voltage converters are a lot simpler for the amateur to construct than their main rivals, the switch-mode power supplies, because they do not use inductors. They do however have some disadvantages; they are typically limited to low power throughput and are less efficient than switch-mode power supplies. Since the programmer only requires about 5mA at 13V, the complexity of tracking down suitable cores and winding inductors is just not worth the effort here.

The operation of a charge pump voltage converter may seem a bit strange at first, especially since all the energy is transferred through capacitors. However, once the principles are understood it all makes a lot more sense.

Our trusty friend, the common old 555 timer IC, U1, is configured as a square wave oscillator. The frequency of the oscillator is set by R1 and C1 to give a frequency of f(Hz)=0.7/(R*C). A square wave of about 7kHz is thus produced at pin 3 of U1.

Consider now diodes D2 and D3, ignoring for now the voltage drop across them for the sake of simplicity. When the circuit is switched on, current will flow from the power rail through the diodes until the voltages at points A and B are the same voltage as the power rail.

Now if pin 3 of U1 was high and goes low, the voltage at point A will decrease. Current will flow through D2 to recharge C2, so that point A is restored to the power rail voltage. When pin 3 goes high the voltage at point A is pushed higher than the voltage of the power rail and point B. Current cannot flow backwards through diode D2, but it can flow through D3 — discharging C2 and charging C4 until points A and B are at the same voltage.

When pin 3 goes low again, point A will be at a lower voltage than the power rail because some of the charge in C2

Using the PIC16C84 Single Chip Micro - 2

was lost to C4. Current cannot flow backwards through D3, but instead flows through D2 to recharge C2 and bring point A back to the same voltage as the power rail. Thus, current flows through D2 whenever pin 3 goes from high to low, and through D3 whenever pin 3 goes from low to high.

This cycle is repeated about 7000 times per second, pumping progressively more charge into C4. Eventually point B will be at such a high voltage that point A cannot get to a higher voltage and no more charge can be pumped into C4. This will happen when point B is at a voltage equal to the power rail voltage plus the voltage difference between low and high at pin 3 of U1. As an approximation, the low voltage is ground and the high voltage is the power rail voltage. This makes the voltage difference almost equal to the power rail voltage. Point B thus gets to about twice the voltage of the power rail.

A second pumping stage, made up of C3, D4 and D5, works in exactly the same way to pump point D to about three times the power rail voltage.

If you are not entirely convinced at this point, consider the following mechanical analogy. Consider a foot pump; it has an input valve (D2), an output valve (D3) and a piston (C3). When the piston is pulled back by a force (U1 output goes low), air flows through the input valve. When the piston is forced forward (U1 output goes high) air is forced out of the output valve. The name 'charge pump' does make sense!

Of course we live on planet earth, so a bit of reality creeps in to mess up this perfect picture. The 0.6V drop across each of the diodes does reduce the efficiency of the charge pump, and point D thus never actually gets all the way to three times the power rail voltage. Power is drawn from the circuit constantly, thus reducing the voltage even more. However the output is still high enough to provide a reliable supply to the Vpp voltage regulator.

The output from the charge pump is reduced to the 12 to 14V range by using a 7812 type 12V fixed voltage regulator, U2, which is 'standing' on two diodes, D6 and D7. Since each diode has a voltage drop of about 0.6V, the two diodes boost the regulator's output by an extra 1.2V or so to give about 13V.

As mentioned above, a minor modification can be made for operation with an input power supply of between 12 and 15 volts. All that is required is to remove C3. This effectively disables the second pumping stage and prevents the charge pump output voltage from rising too high and destroying U2.

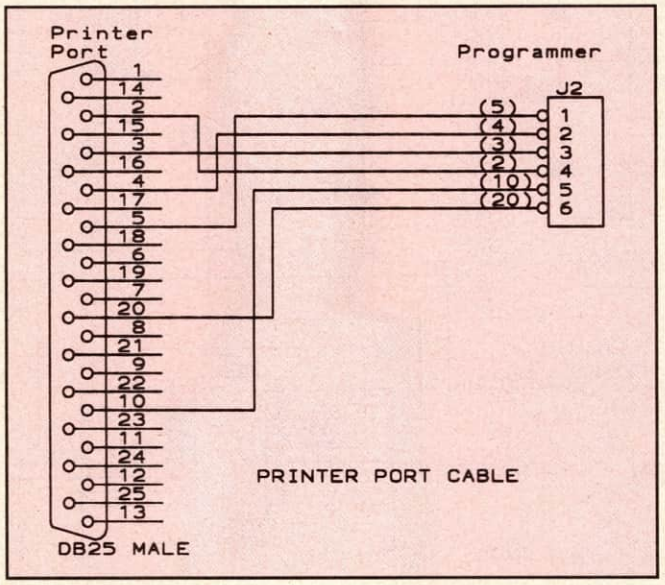


Fig.2: Use this diagram to wire up the cable which connects the programmer to a standard PC printer port.

Signal buffer

The signal lines between the PC and the programmer are passed through a 74LS05 open collector inverter, U4. This device has six inverters, of which five are used here. The inverters serve three purposes.

Firstly, they provide a degree of isolation between the PC and the PIC being programmed, so hopefully the 74LS05 gets damaged if something does go wrong. This is perhaps more a matter of faith than science, but a 74LS05 is a lot cheaper to replace than a PC printer port or a PIC chip!

More rationally, two of the inverters, U4E and U4B, are used to restore the DATA and CLOCK lines to acceptable levels. This is required because, during programming, the PIC's CLOCK and DATA lines are configured as Schmitt trigger inputs. Schmitt triggers require a much larger voltage swing than can be provided by many printer ports. I found this out quite accidentally, when an early prototype would work on one PC printer port and not another! Since the inverters have open collectors, R8 and R9 are required to pull the voltage high when the inverters turn off.

Two further inverters, U4F and U4A, are used to control the Vpp and Vdd switch circuits. The power supply to all the PIC's pins must be off when a PIC is being removed or plugged into the socket. The Vdd switch circuit is only turned on while the PIC is being accessed.

Since the inverters have open collector outputs, they do not sink or source current while the output is high. When the output of U4F goes low it sinks current, causing PNP transistor Q2 to turn on and supply power to the PIC's Vdd pin. This power is also used as a pull-up voltage for R8 and R9. Current flowing via R10 lights the LED, D8, to indicate that the programmer is busy. When the Vdd switch is turned off, the charge in C9 is quickly drained via the LED and via R11 so that it is safe to remove the PIC from the circuit.

A similar circuit is used for switching on the programming voltage Vpp, using transistor Q1.

Although using a totem-pole (or active pull-up) driver could have done away with the pull-up resistors, an open collector driver was selected for this job for a few reasons. Firstly, the PIC DATA pin is used for both

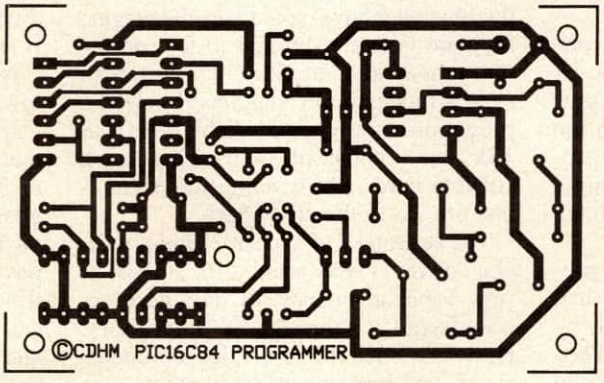


Fig.3: The PCB pattern for the programmer board, reproduced here actual size as usual, for those who want to 'roll their own'.

input and output; if a totem-pole type driver was used, there would be no way to turn off the driver and it would 'fight' with the PIC's DATA pin when it is in output mode. An open collector driver does not have this problem; when its output is high, the driver does not source current and thus does not affect the output of data from the PIC.

Secondly, open collector drivers can be used much like a transistor in the grounded emitter configuration; this property is used to sink base current for the two switching transistors, Q1 and Q2. A totem-pole driver would be able to cope with controlling the Vdd switch, but not the Vpp switch because the Vpp base voltage is much higher than a totem-pole device could handle. As a result Q1 would never turn off...

An inverting driver was selected for U4 because the PC printer port outputs are low when the PC is started up. If a non-inverting buffer was used then the Vpp and Vdd switches would be on by default. This was considered undesirable. The inverting driver turns Vpp and Vdd off by default.

The output of the data output buffer, U4D, is pulled up by R7 to provide robust signal levels to the printer port.

Construction

This programmer can easily be built on either Veroboard or a PCB made from the etching pattern provided. The pattern for a suitable small PCB is shown in Fig.3, actual size, while the matching wiring overlay diagram is shown in Fig.4.

Construction is quite straightforward. Care should be taken to ensure that the components are soldered in correctly. Many of the components are polarity sensitive and must be inserted with the correct orientation.

If you are using the PCB, ensure that you solder in the two wire jumpers near D8 and U4. It is wise to use sockets for U1 and U4 so that they can be replaced in the unlikely event of a failure. This also makes testing easier.

Note that although 7805 and 7812 (i.e., TO-220) voltage regulators have been specified, the circuit will function just as well with the smaller 78L05 and 78L12 regulators.

Virtually any six-conductor cable is suitable for the cable between the PC and the programmer. The cable should be long enough to get from the back of the PC onto the workbench, though no longer than is necessary.

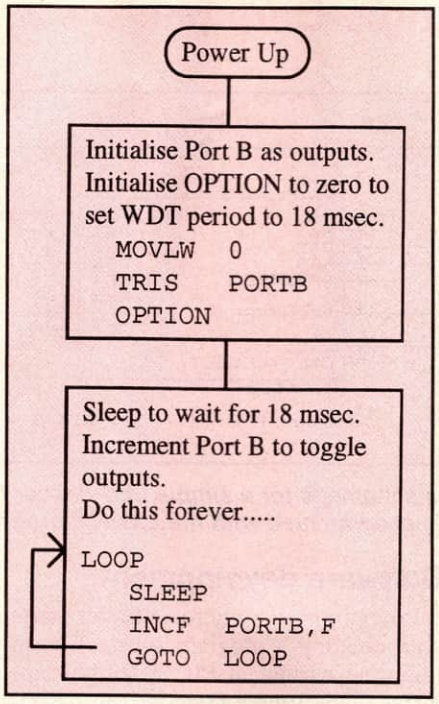


Fig.5: A flow chart for the author's LIGHTS program, used as a simple example of PIC chip programming.

The cable on the prototype is about 1.5m long. The PC end is terminated with a 25-way male D type connector. This should be a solder lug type and the conductors are soldered directly into the relevant solder lugs. The back of the connector should be protected with a purpose made cover or 'back sheel'. This protects the back of the plug against short circuits and the strain relief prevents conductors from damage.

If you have recently won the lottery, you might want to splash out and buy a zero insertion force (ZIF) socket for U5. These are normally used in professional programmers, because they are a bit easier to use. They are however quite expensive, and often difficult to find. A common old IC socket will be fine, so long as you are reasonably careful when inserting and removing the PIC during programming.

The cheap dual-wipe type sockets seem better for this job than the fancy machined pin type, because they seem stand up better to repeated insertions. It is an easy matter to carefully push the PIC into the socket and extract it with a lever such as a flat blade screwdriver.

A word of caution, though: it is easy to get careless and bend an IC pin. Do this a few times and you will break off a pin — which renders an otherwise perfect IC completely useless, causing wailing and grinding of teeth!

A good way to protect against this to keep the IC permanently plugged into an IC socket piggy-back style. The PIC and its protective socket are treated as a unit and handled together.

If a pin gets damaged, then only the IC socket needs to be replaced — a much cheaper exercise. Machined pin IC sockets are the best for this job; the pins are a lot stronger than those of dual wipe IC sockets and stand up well to general abuse.

Programmer testing

The programmer is very easy to test and no calibration is required. Firstly inspect the finished PCB for any bad solder joints and short circuits. You might also consider a last check for missing or wrongly inserted components.

Next, remove U1 and U4 from their sockets and unplug the programmer from the printer port. Then turn on the power, and check that the power entering the programmer is the correct polarity and in the range 9 - 12V. The output from U3, measured across C8, should be 5V. The voltage measured between pins 7 and 14 of U4 should also be 5V. If this is correct, then U3 and its associated circuitry are working correctly.

The LED should not be lit. The voltage between pins 5 and 14 of the PIC socket should be zero (well at most, a few millivolts). Now place a jumper wire between pins 7 and 2 of U4's socket.

This should cause the LED to light up, and the voltage between pins 5 and 14 of the PIC socket

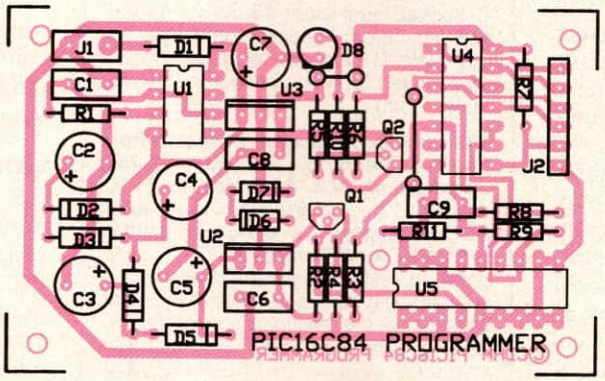


Fig.4: The overlay diagram for the programmer board, showing where everything goes and their correct orientation. Note the two small links, one near D8 and the other between Q2 and U4.

Using the PIC16C84 Single Chip Micro - 2

should now be 5V. Removing the jumper should extinguish the LED and cause the voltage to return to zero. This has tested the Vdd switching, LED and associated components.

Now remove the power and plug U1 into its socket. Turn on the power and check the voltages in the charge pump. Point B, measured across C4, should be at almost twice the power rail voltage (as measured between pins 1 and 8 on U1). Point D, measured across C5, should be at almost three times the power rail voltage and at least 18 volts. If any of these voltages is incorrect, then recheck the charge pump circuit, the polarity of diodes D2 - 5 and capacitors C2 - 5 in particular.

The output of U2, measured across C6, should be about 13.2V. The Vpp switch should be off, so the Vpp voltage, measured between pins 5 and 4 on the PIC socket, should be zero.

Now place a jumper wire between pins 7 and 12 of U4's socket. This should turn the Vpp switch on. The Vpp voltage, between pins 5 and 4 on the PIC socket, should now be about 13.2V. The current drain should cause the voltage of the charge pump output, measured across C5, to drop slightly, but it should remain above 16V.

Removing the jumper should cause the Vpp voltage on the PIC socket to drop back to zero. This has tested the Vpp regulator and Vpp switch circuits. Remove power. U4 should now be plugged into its socket.

Once the software has been installed on the PC, further tests can be run. Turn off the PC and plug the programmer into the printer port. Turn on the PC and the programmer's power source. Start the programmer software by typing PP84.

Before proceeding, you will need to select which printer port you are using. This is done by entering S and selecting the desired printer port. The programmer itself, and the connection between the PC and the programmer, can be tested by entering T.

Once in the test mode, the various voltage levels can be set and tested on the various pins.

Now the programmer should be fully functional, and it's time to try out a simple development cycle.

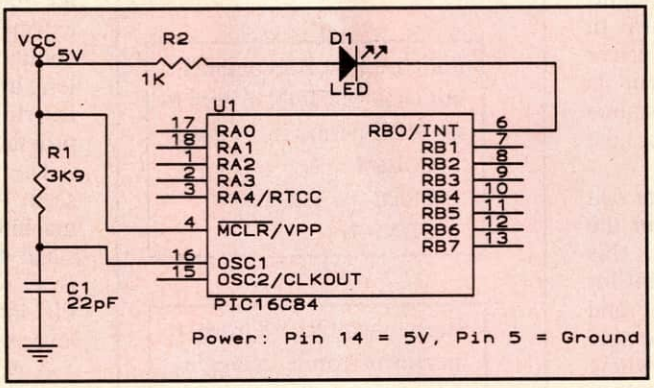


Fig.6: The schematic for a simple LED flasher based on a PIC chip programmed with the LIGHTS program.

Software development

Programming a microcontroller starts with creating a program source file, in this case written in PIC assembly language. This source code is then passed through a translator program known as an *assembler*. The assembler converts the human readable form of the program into the actual binary code which can be downloaded into the PIC.

If the assembler cannot understand part of the assembly language source file, then it displays the errors. The source file must then be edited to correct these errors and the program should be passed through the assembler again.

Once the code has been successfully assembled, it can be programmed into the PIC. This is done by the programmer control program, PP84. The PIC is then plugged into the circuit and tried out. If something goes wrong, the error has to be tracked down in the source code and the whole cycle is repeated. This is often referred to as 'crash and burn' development.

Traditionally, the first program that any 'C' language programmer tries is called the 'hello world' program. This is almost a peace offering to the gods of 'C', and to skip this important step is to invite a cursed programming career. Similarly, microcontroller developers religiously type in LIGHTS as a first program for any new development.

LIGHTS is just a very simple program which flashes an LED. This isn't really as crazy as it seems; getting LIGHTS to work proves that a lot of the circuit works. The code for LIGHTS is included on the software distribution disk.

Fig.5 shows a flowchart for the LIGHTS program, while Fig.6 shows the schematic for the simple LED flasher based on a PIC 16C84 chip programmed with this program. As you can see it's about as simple as it gets and can easily be built on a breadboard or a scrap of veroboard. This is a useful circuit for experimenting with various features, such as watchdog timers and the various oscillator modes.

Conclusion

So far in this series we have described the PIC microcontroller series, with particular emphasis on the PIC 16C84, and built a simple programmer for it. This is everything that you need to start developing your own microcontroller based projects.

In the third and final article in this series, we'll present a sample project which uses many of the 16C84's special features, to implement an electronic lock. ❖

PARTS LIST

Semiconductors

U1	LM555 timer IC
U2	7812 or 78L12 voltage regulator
U3	7805 or 78L05 voltage regulator
U4	74LS05 hex inverter (o/c outputs)
Q1,Q2	PNP transistor such as BC327
D1-5	1N4007
D6-7	1N4148
D8	LED 3mm or 5mm

Resistors

All 0.25 watt:

R1,10	1k
R2,4,5,7,8,9,11	10k
R3,6	3.3k

Capacitors

C1,6,8,9	0.1uf monolithic ceramic
C2,3	10uF 50VW radial electrolytic
C4,5,7	33uF 50VW radial electrolytic

Miscellaneous

PCB, 78 x 48mm, or Veroboard; 18-pin IC socket, dual-wipe; 14-pin IC socket, dual wipe; 8-pin IC socket, dual wipe; 25-way male D type connector with backshell cover; power connector; six-way cable (1.5m).

Kits for this project are available from Charles Manning, PO Box 33-256, Christchurch, N.Z. All prices are in NZ dollars. Visa, MasterCard and bank drafts accepted for overseas orders. (Add GST for NZ orders.)

Software - includes a manual, assembler, programming software, other utilities and software for the electronic lock project.

(Specify 5.25 or 3.5 inch) $20
PCB only $10
PCB plus all components and software $50
PIC16C84 (4MHz) each $15
PIC16C84 (10MHz each $20
P&P (airmail) per order $8