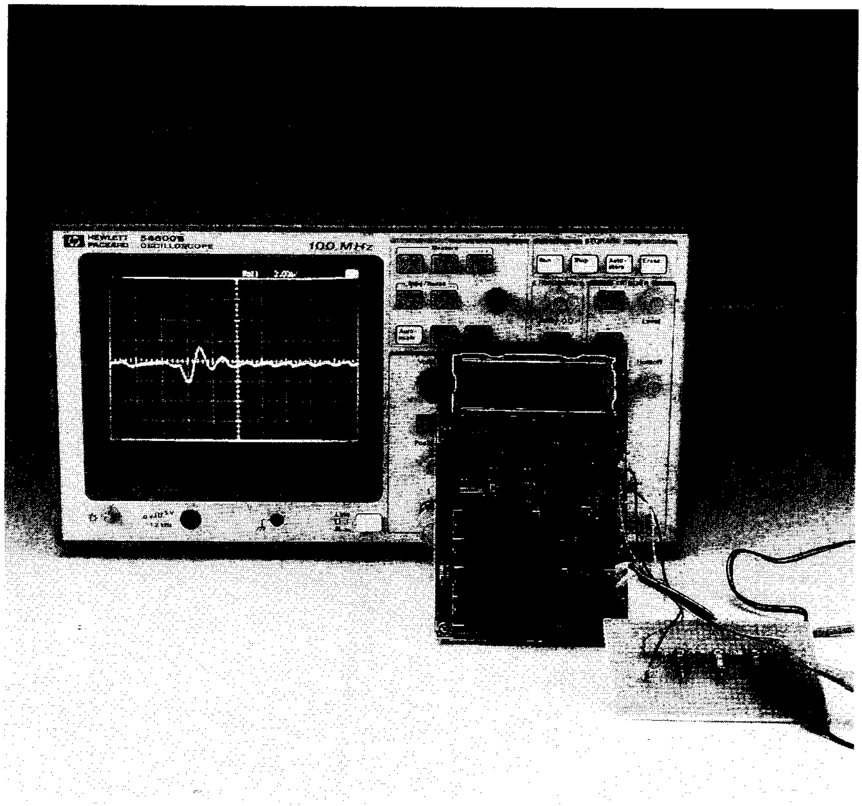
digital PID controller

closing the loop with a PIC

The term 'PID controller' normally brings to mind an analogue controller circuit with a couple of opamps. This article shows how such a controller can be implemented in digital form. In this universal controller circuit, based on a PIC and a D/A converter, all control parameters can be entered via a keyboard to adapt the controller to the task at hand.



Design By D. Kohtz

Specifications:

➤ controller type:	PID
➤ parameter entry:	hexadecimal keypad
➤ parameter display:	alphanumeric LCD
➤ adjustable parameters:	- set value - initial control quantity value - upper limit - lower limit - sampling interval (100 ms to 25 s)
➤ parameter storage:	EEPROM
➤ A/D and D/A conversion:	8 bit (measured value, control quantity)
➤ microcontroller:	PIC 16C71

The digital PID controller described in this article, based on a PIC microcontroller, is on the one hand very well suited to studying how the controller operates with various parameter values, using simple control loops. On the other hand, it can of course also be used for specific control tasks. A complete control loop requires an appropriate actuator and a suitable sensor, in addition to the controller itself.

The core of the circuit is an inexpensive PIC 16C71 microcontroller, whose built-in A/D converter digitizes the measured value (actual value) and provides the result to the controller program. The program determines the control error relative to the set value (target value) for each actual value and, based on this, calculates the control value that is output to the external D/A converter. The control parameters, which are entered by a keypad, are stored in a non-volatile memory (EEPROM).

In order to manage with the program memory of the PIC, which is only 1 kB, the parameters are entered as hexadecimal codes. They are however displayed on the LCD as text, which is easier to understand.

Basic theory

Control circuits are encountered very often in electronics. For instance, a voltage regulator is present in almost every circuit. More elaborate controllers for dealing with other physical quantities, such as rotational speed, velocity, temperature, pressure, flow and so on. Control technology deals with finding solutions for such problems. Let's look at the basic principles of this technology.

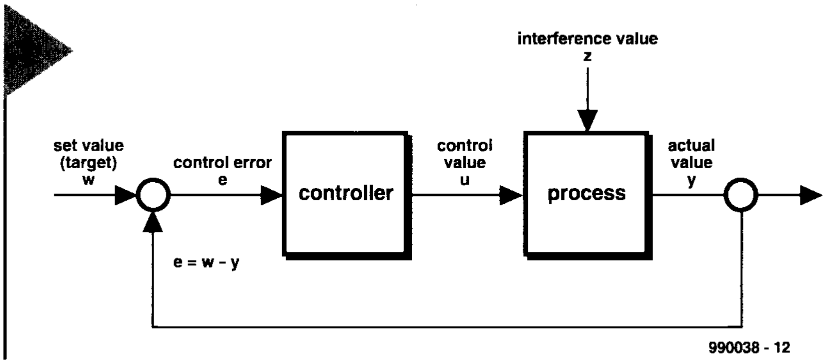
In control theory, the controlled activity is referred to as the **process**. The job of the controller is to maintain the value of the **controlled quantity** at a set-point or **target value**, in spite of the effects of external `interference`. In order to do this, the controller must measure an **actual value y**, calculate the **control error e** by comparing the actual value to the **target value w**, and output a `control value` **u** that depends on the value of the control error and which achieves the desired effect. (Note: the official term for the control value is 'manipulated variable'.)

A characteristic feature of every control system is **closed loop operation**, as illustrated in **Figure 1**. It is necessary that a given input signal produces a specific output signal, for both the controller and the process. In general, the relationship between the input an output signals is `time-dependent`.

The step response is an essential characteristic of both processes and controllers. It reflects how the output signal responds when the input signal **u** changes instantaneously from one level to another level. The waveform shown in **Figure 2** illustrates a step response that is typical of many processes. For example, if a given DC voltage is applied to a resistive heating element, the graph of its temperature as a function of time will be similar to this curve.

A controller is also a transfer element that produces a control value (output signal) in response to a control error (input signal). Depending on the way in which the controller responds, it can be classified as a proportional, integral or derivative controller. In a proportional controller (*P* controller), the transfer element is in principle nothing more than an amplifier. Its output signal changes in proportion to its input signal, with the gain being an adjustable parameter. As the names suggest, the transfer element of an integral controller (*I* controller) is an integrator, and the transfer element of a derivative controller (*D* controller) is a differentiator. With the latter types of controllers, both the time constant and the gain are adjustable parameters.

A controller that combines all three types of response, called a PID controller, has proven its worth as a sort of 'universal' controller. As shown in **Figure 3**, the control value consists of the sum of the outputs of the three types of controllers, which can be assigned individual weighting factors according to the desired controller response.



The mathematical relationship between the control error and the control value is called the **control algorithm**. For a PID controller, this is a differential equation.

The primary difference between a digital controller and an analogue controller is that with a digital controller the actual value is not measured continuously, but is instead periodically sampled at some fixed interval T_0. In the time between two successive samples, the control error must be determined and the control value calculated using the control algorithm.

The following equation can be used as the control algorithm of a PID controller (see the sidebar 'Digital PID controller design' for the derivation of this equation):

$$u_k = u_{(k-1)} + q_0e_k + q_1e_{(k-1)} + q_2e_{(k-2)}$$

In this equation, the index k represents the running count of the individual measurements. This means that $e_{(k-1)}$ is the control error for the previous measurement. The control parameters q_0 through q_2 can be approximately determined from the characteristic parameters of the process step response using the following formulas:

$$q_0 = [1.5T_G/(K_pT_u)](1 + T_u/2T_0)$$

$$q_1 = [1.5T_G/(K_pT_u)](T_0/2T_u - T_u/T_0 - 1)$$

$$q_2 = 3T_G/4K_pT_0$$

The step response of the process must be measured experimentally to obtain these values. The parameters must also satisfy the following conditions:

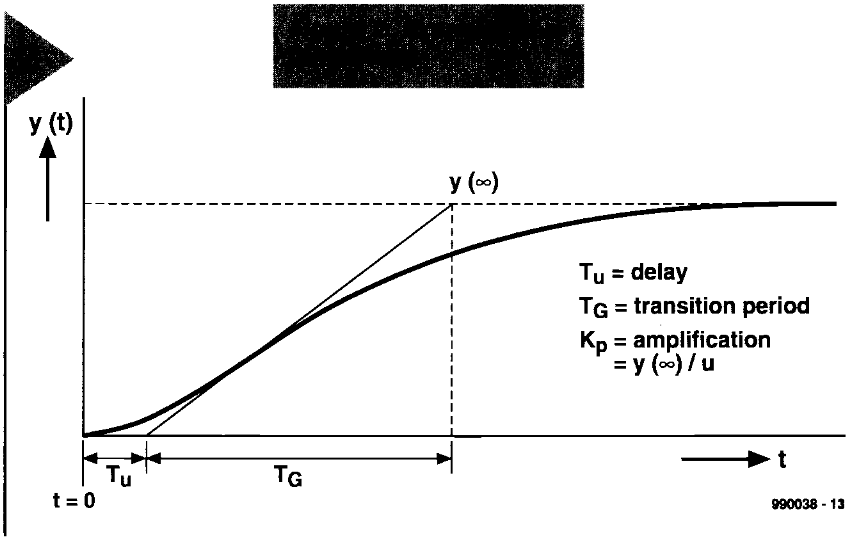
$$q_0 > 0$$
$$q_1 < 0, |q_1| > q_0$$
$$|q_0 + q_1| < q_2 < q_0$$

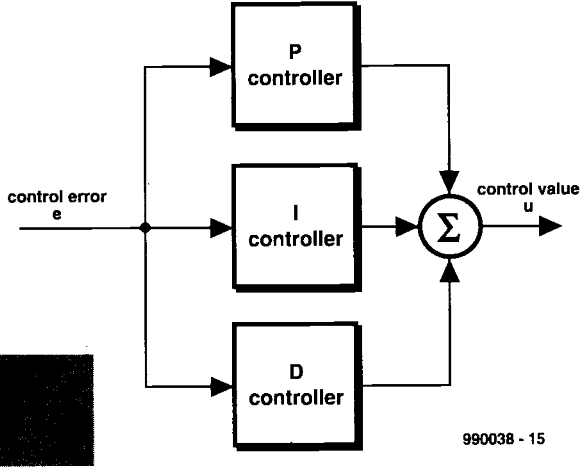
As a guideline for the value of the sampling interval T_0, it should chosen to be around one tenth of the time required for the process step response to reach 95% of its final value.

HARDWARE AND SOFTWARE

Figure 4 shows the circuit diagram of the controller. Its input is designed for voltages between 0 and 5 V, which can come from any desired type of sensor. The input voltage is passed to the internal A/D converter of the PIC 16C71 via opamp IC5a, which acts as a buffer. The low-pass filter formed by R8 and C8 at the output of the opamp keeps high-frequency interference from reaching the A/D input (RA0 of the PIC).

The controller output provides a





[redacted]

voltage that also ranges from 0 to 5 V. The output current of opamp IC5b is certainly adequate for use with the previously-mentioned circuit that simulates the process to be controlled. For controlling a real process, you will naturally have to provide a suitable power stage that can be driven by the maximum load current (5 mA) of the controller output.

IC5b matches the voltage range of the D/A converter to that of the A/D converter of the PIC. Its gain is set to 1.935 by R10 and R11.

[redacted]

An 8-bit DAC IC (Analog Devices AD577) is used for the D/A converter. It can be operated from a single 5 V supply and does not need an external reference. Its external circuitry is limited to C7, which is necessary to decouple the supply voltage. In addition, pull-up resistors R2 and R3 are connected to the clock and data pins of the EEPROM IC (24C01).

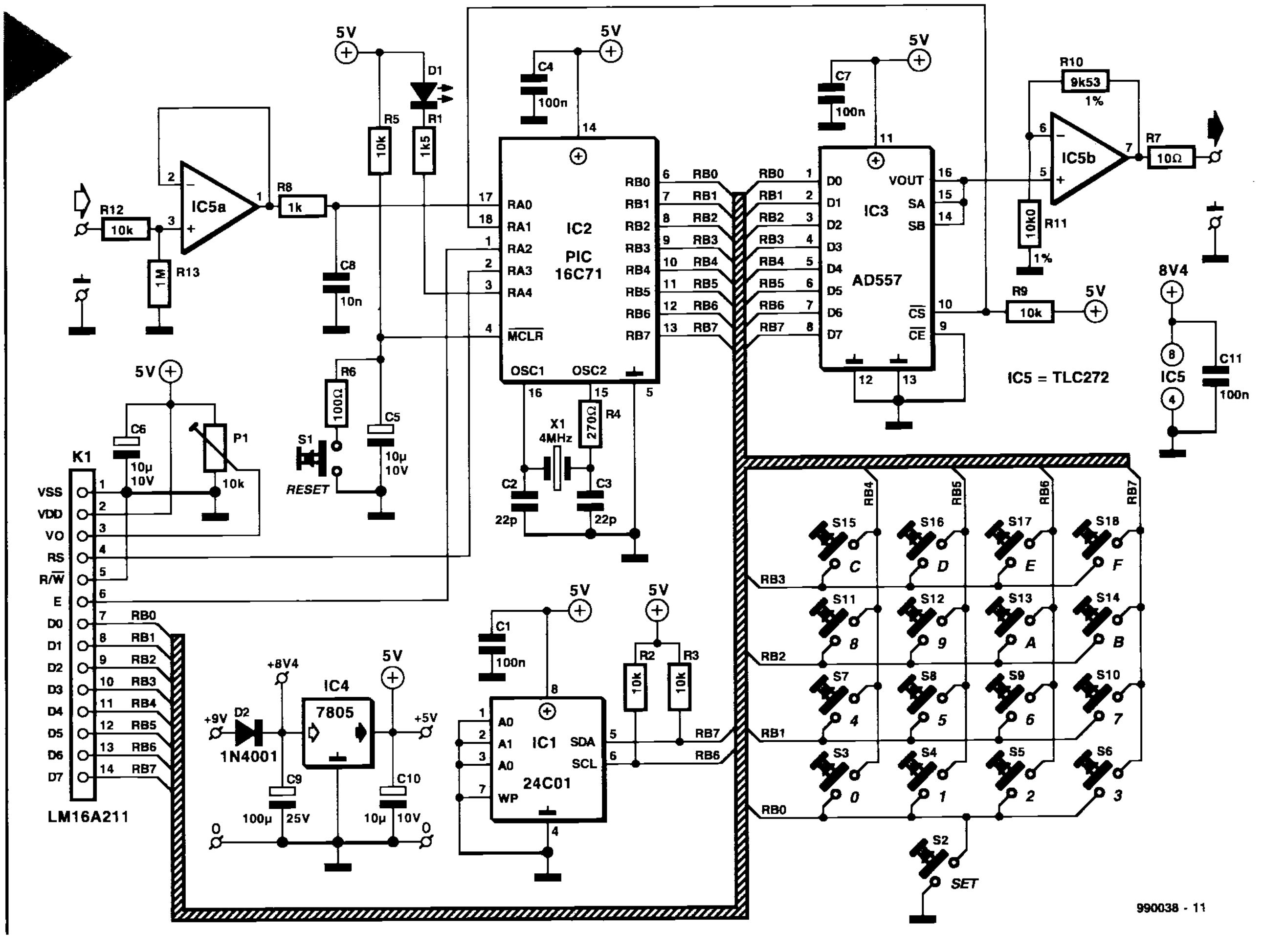
Furthermore, only a small amount of extra hardware is needed around the PIC itself. In addition to the clock circuitry with a 4 MHz crystal, there is a reset circuit which requires only a few passive components. The power-on reset is provided by R5 and C5, while a manual reset is provided by R6 and S1.

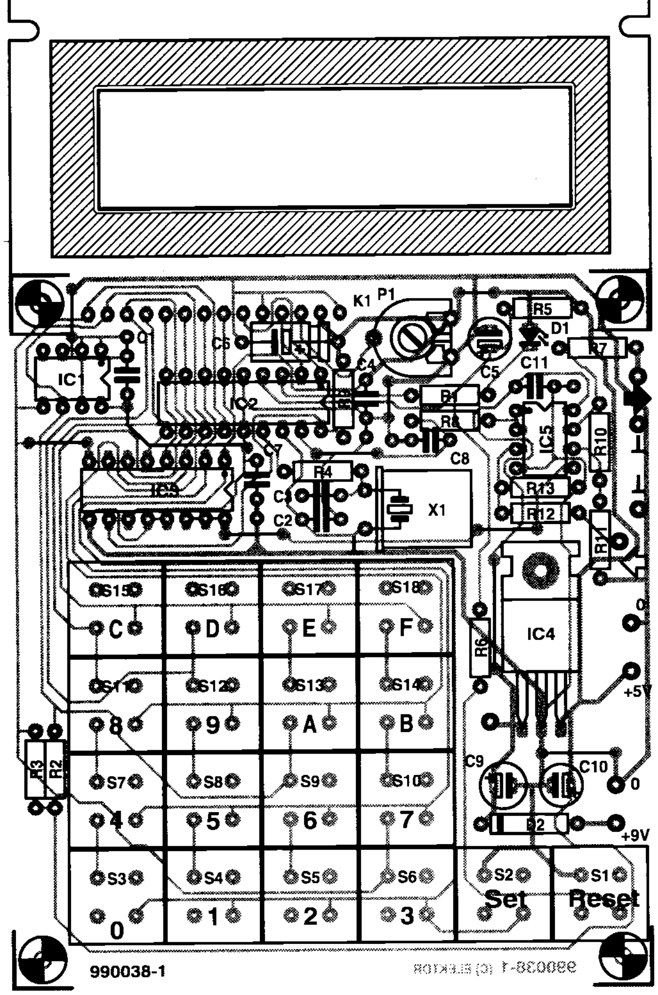
The alarm LED D1 is driven directly from the RA4 port output, and the D/A converter chip and the LCD module are also connected directly to the output port pins of the PIC. In the input mode, the same port pins are used for scanning the contacts of the parameter-input switches. Except for S2, these are arranged as a 4×4 matrix, with four port connections on each side. S2 is used as the SET pushbutton for confirming input values; it connects RB0 to earth when actuated.

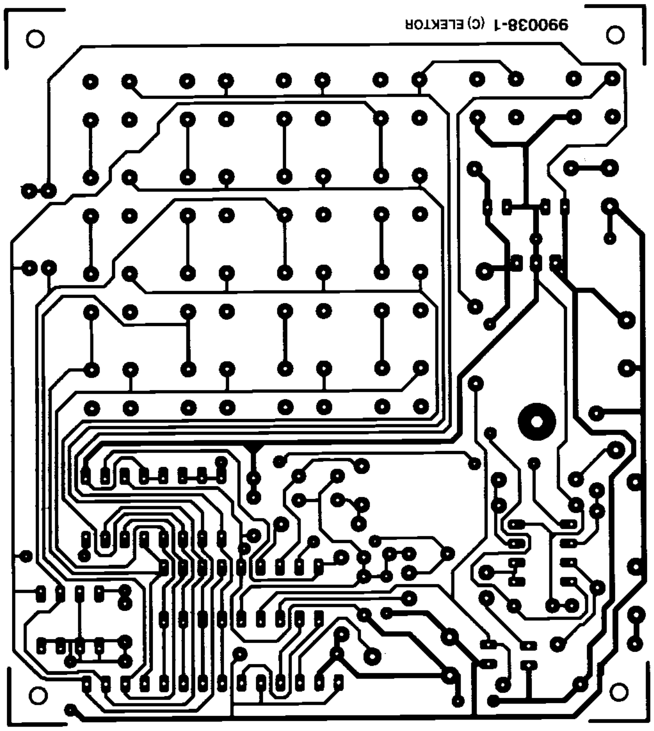
The power supply is just as simple as the rest of the circuit. It consists of a 5 V regulator, two electrolytic capacitors and a reverse-polarity protection diode at the input connector for an external 9 V mains adapter.

As far as the software is concerned, the tables for the predefined display text and the instructions needed to control the serial EEPROM both take up quite a bit of storage space. The mathematical routines for calculating the control quantities also take up a certain amount of space. The actual controller program is modest by comparison.

The program is protected by a watchdog timer, which in case of a failure triggers a reset and an alarm (blinking LED D1) and also sets the control quantity to zero. If this happens, the controller must be restarted, but it is not necessary to re-enter the control







parameters, since the values that were previously stored in the EEPROM can be used again.

Since the program needs a very large number of registers, some registers must be used for more than one purpose. In principle, this is not a problem, since (for example) the registers used for performing calculations are never used during parameter input. The only disadvantage is that the registers that are used more than once do not have meaningful names.

If you want to carefully study or modify the program, you will find the source code on a diskette available through our Readers Services.

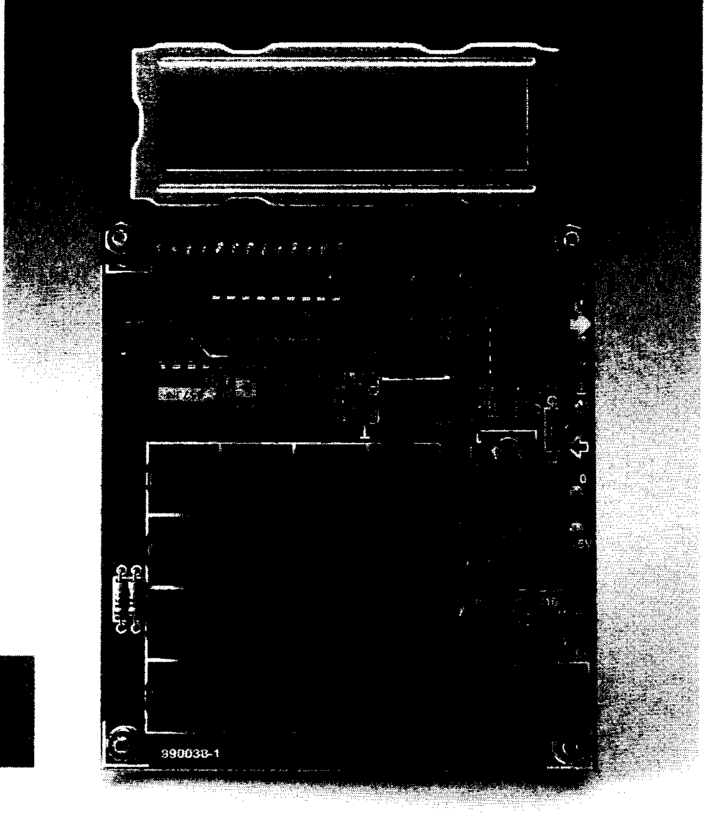
Construction and operation

Constructing the circuit on the printed circuit board does not require any unusual skills. Since the circuit board is single-sided, a few jumpers are unavoidable. Do not overlook them when stuffing the board.

The keypad module specified in the list of components must be used for the parameter input pushbuttons, due to the internal connections. An unregulated 9 V mains adapter can be used for the power supply, as long as it can supply at least 200 mA.

When setting up the board, first adjust P1 to set the contrast of the display for optimum readability.

A simple circuit, shown in **Figure 7**, has been designed for testing the controller and experimenting with it. This circuit simulates the process to be controlled. The step response of the process is determined by the time constants of three RC low-pass circuits



[redacted]

connected in series. The time constants of this 'electronic process' can thus be easily changed, and interference can be easily simulated by loading the output with the switchable resistor. To use the simulation circuit, connect its input to the output of the controller and its output to the input of the controller.

Control parameter values can be entered only immediately after the circuit is switched on or directly following a reset. They are entered in hexadecimal form by means of the keypad. Two characters will be displayed after two keys have been pressed in turn. The SET button must be pressed for the entered value to be accepted and stored in EEPROM. The target value and the initial value of the control quantity can be set between 00_H and FF_H, which corresponds to a voltage range of 0 to 5 V. The upper and lower limit values can also be set between 00_H and FF_H. The length of the sampling interval (display indication 'INTERVAL') is determined by multiplying a basic increment of 0.1 s (fixed in software) by the entered hexadecimal value.

[redacted]

The parameters q_0, q_1 and q_2, which are important for the control algorithm, are entered as fixed-point numbers, with one byte (two hex characters) before the decimal point and one byte after the decimal point. The bits after the decimal point are weighted according to $1/2^n$, where n represents the bit position (MSB = 1, LSB = 8). A hex value of 80 after the decimal point thus corresponds to 1/2 (0.5 decimal). Since q_1 is always negative, a short explanation of the representation of negative binary numbers is in order. A negative binary number is formed by bitwise inverting a positive number and adding one to the result. An example is:

13_D =	$0000\ 1101_B = 0D_H$
inverted:	$1111\ 0010_B = F2_H$
+1:	$1111\ 0011_B = F3_H$

For 8-bit signed numbers, this results in a range of values from $+127_D$ ($7F_H$) to -128_D (80_H). Negative 8-bit numbers can be recognized by the fact that the MSB is a 1.

The integer and fractional bytes of q_0 through q_2 must be entered separately (for example, Q0HI and Q0LO are entered by pressing two keys for each byte, with SET pressed between the first and second pair of value-entry keys).

If you use the simulation circuit shown in **Figure 7**, you can achieve satisfactory results with the following parameter values:

q_0 =	$13.5_D = 0D80_H$
q_1 =	$-18.5_D = ED80_H$
q_2 =	$6.5_D = 0680_H$

sampling interval = 0.4 s (enter '04')
target value = 80_H (approx. 2.5 V)
initial control value = $A0_H$ (approx. 3.1 V)

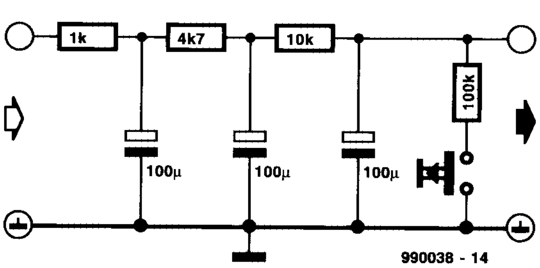
Since the electrolytic capacitors in the simulation circuit have quite large tolerances, it is possible that the controller will not operate optimally with the above values. In this case the values of q_0 through q_2 should be empirically adjusted.

The output signal of the controller is set to 0 V during parameter entry. After parameter entry has been completed by pressing SET, the control value (CONTROL VAL) is output at a level equal to its entered initial value until the entered target value (SETVALUE) is reached. After this, the actual control process starts up and, depending on the selected control parameter values, more or less exactly maintains the controlled variable at the set value, even in the presence of interference. During the control process, the current levels of the control value (CONTROL VAL) and the actual value (ACT.VAL) are continuously displayed.

If either the upper or lower limit value (UPPER LIMIT or LOWER LIMIT) is exceeded, the alarm lamp (LED D1) blinks for the duration of out-of-limits condition. No new values are displayed during this interval. The controller output goes to zero if the upper limit is exceeded, and to FF_H (5 V) if the lower limit is exceeded.

If the controller is started anew by pressing Reset, the parameter entry process must be run through again by pressing SET after each parameter value is displayed. SETVALUE appears first on the display, with the last saved value. The values of CONTROL VAL, UPPER LIMIT, LOWER LIMIT and INTERVAL follow in turn, each appearing after SET has been pressed. The stored values are displayed for each parameter in turn, and can be modified if necessary.

(990038-1)


1k
4k7
10k
100k
100µ
100µ
100µ
990038 - 14