10 Counters

Counters are provided as built-in elements in PLCs and allow the number of occurrences of input signals to be counted. This might be where items have to be counted as they pass along a conveyor belt, or the the number of revolutions of a shaft, or perhaps the number of people passing through a door. This chapter describes how such counters can be programmed.

10.1 Forms of counter

A counter is set to some preset number value and, when this value of input pulses has been received, it will operate its contacts. Thus normally open contacts would be closed, normally closed contacts opened.

There are two types of counter, though PLCs may not include both types. These are down-counters and up-counters. *Down-counters* count down from the preset value to zero, i.e. events are subtracted from the set value. When the counter reaches the zero value, its contacts change state. Most PLCs offer down counting. *Up-counters* count from zero up to the preset value, i.e. events are added until the number reaches the preset value. When the counter reaches the set value, its contacts change state.

Different PLC manufacturers deal with counters in slightly different ways. Some count down (CTD), or up (CTU), and reset and treat the counter as though it is a relay coil and so a rung output. In this way, counters can be considered to consist of two basic elements: one relay coil to count input pulses and one to reset the counter, the associated contacts of the counter being used in other rungs. Figure 10.1(a) illustrates this. Mitsubishi is an example of this type of manufacturer. Others treat the counter as an intermediate block in a rung from which signals emanate when the count is attained. Figure 10.1(b) illustrates this. Siemens is an example of this type of manufacturer.

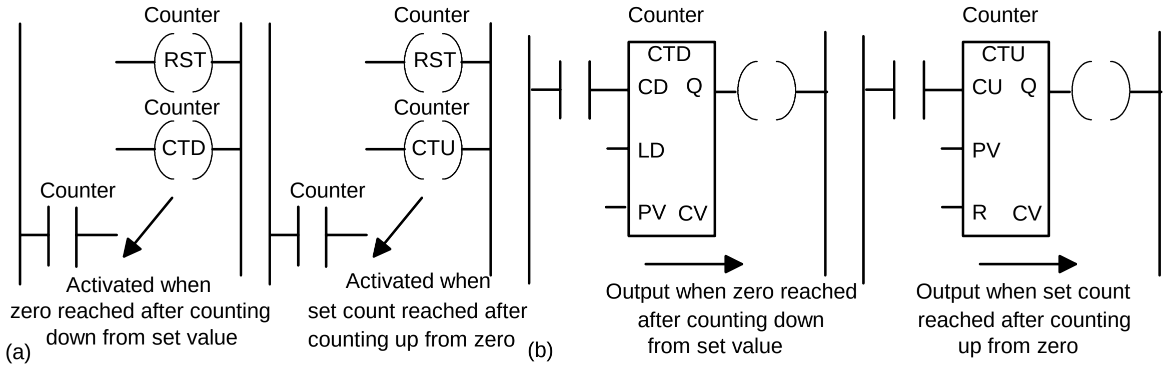

Counter
RST
Counter
CTD
Counter
Activated when zero reached after counting down from set value
(a)
Counter
RST
Counter
CTU
Counter
Activated when set count reached after counting up from zero
(b)
Counter
CTD
CD Q
LD
PV CV
Output when zero reached after counting down from set value
Counter
CTU
CU Q
PV
R CV
Output when set count reached after counting up from zero

Figure 10.1 *Forms of representation of counters. In (a) RST is reset. In (b), the IEC 1131-3 representation, CD is count down input, LD is for loading the input, PV is for the preset value, CV the current count value, CU is count up input, and R is for the reset input.*

10.2 Programming

Figure 10.2 shows a basic counting circuit. When there is a pulse input to In 1, the counter is reset. When there is an input to In 2, the counter starts counting. If the counter is set for, say, 10 pulses, then when 10 pulse inputs have been received at In 2, the counter's contacts will close and there will be an output from Out 1. If at any time during the counting there is an input to In 1, the counter will be reset and start all over again and count for 10 pulses.

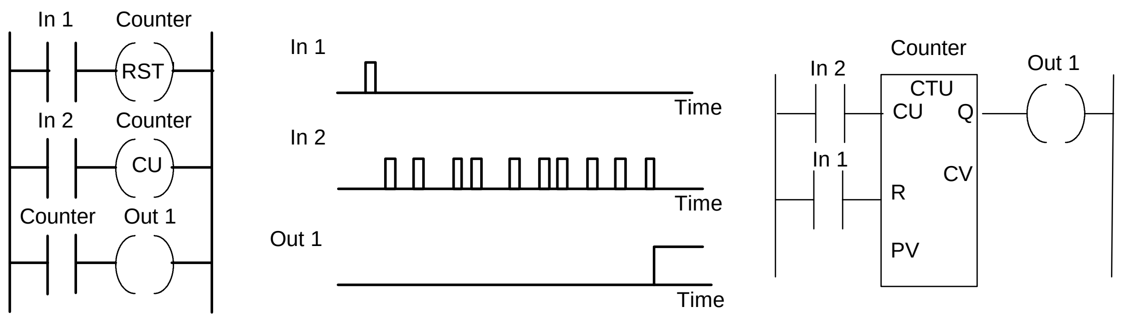
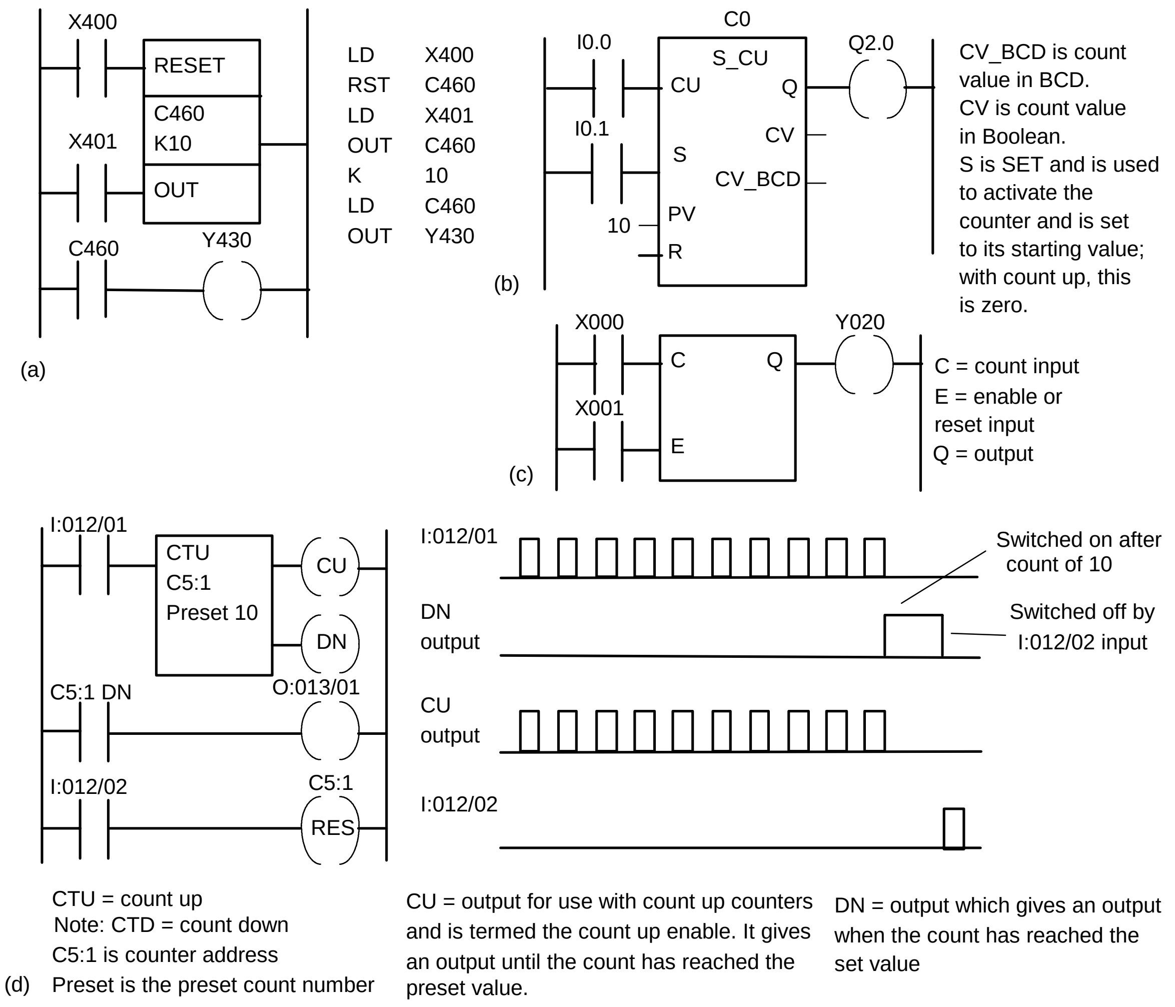


Figure 10.2 *Basic counter program*

Figure 10.3(a) shows how the above program, and its program instruction list, would appear with a Mitsubishi PLC. The reset and counting elements are combined in a single box spanning the two rungs. You can consider the rectangle to be enclosing the two counter ( ) outputs in Figure 10.2. The count value is set by a K program instruction. Figure 10.3(b) shows the same program with a Siemens PLC. With this ladder program, the counter is considered to be a delay element in the output line (as in Figure 10.1(b)). The counter is reset by an input to I0.1 and counts the pulses into input I0.0. The CU indicates that it is a count-up counter, a CD would indicate a count-down counter. The counter set value is indicated by the LKC number. Figure 10.3(c) is the program for Toshiba and Figure 10.3(d) for Allen-Bradley.

10.2.1 Counter application

As an illustration of the use that can be made of a counter, consider the problem of items passing along a conveyor belt, the passage of an item past a particular point being registered by a light beam to a photoelectric cell being interrupted, and after a set number there is to be a signal sent informing that the set count has been reached and the conveyor stopped. Figure 10.4(a) shows the basic elements of a Siemens program that could be used. A reset signal causes the counter to reset and start counting again. The set signal is used to make the counter active. Figure 10.4(b) shows the basic elements of the comparable Allen-Bradley program. When the count reaches the preset value, the done bit is set to 1 and so O:013/01 occurs and the corresponding contacts are opened and the conveyor stopped.



```
LD    X400
RST   C460
LD    X401
OUT   C460
K     10
LD    C460
OUT   Y430
```

CV_BCD is count value in BCD.
CV is count value in Boolean.
S is SET and is used to activate the counter and is set to its starting value; with count up, this is zero.

C = count input
E = enable or reset input
Q = output

CTU = count up
Note: CTD = count down
C5:1 is counter address
Preset is the preset count number

CU = output for use with count up counters and is termed the count up enable. It gives an output until the count has reached the preset value.

DN = output which gives an output when the count has reached the set value

Figure 10.3 (a) *Mitsubishi program, (b) Siemens program, (c) Toshiba program, (b) Allen-Bradley programs*

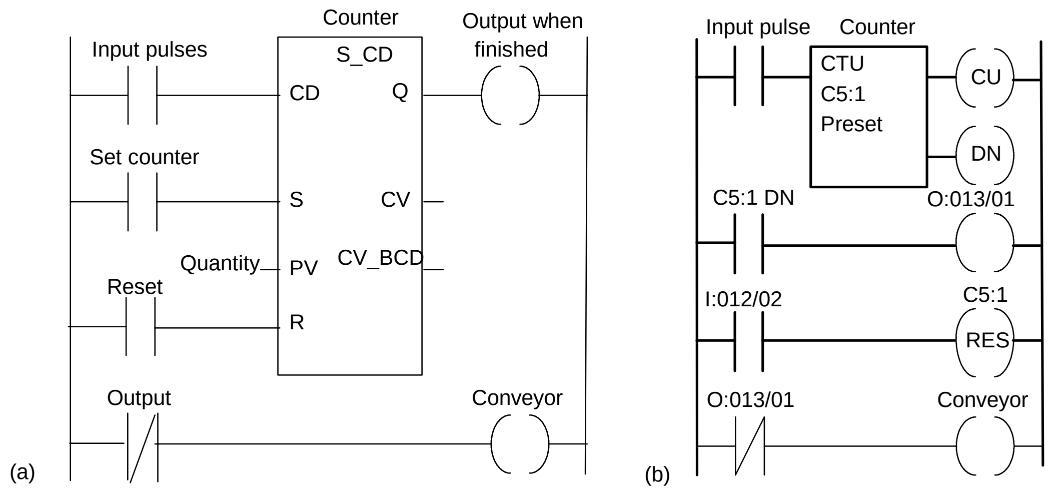


Figure 10.4 *(a) Siemens, (b) Allen-Bradley counting program*

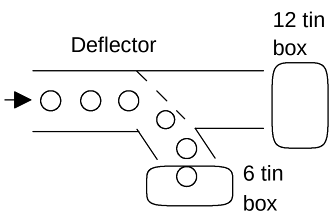
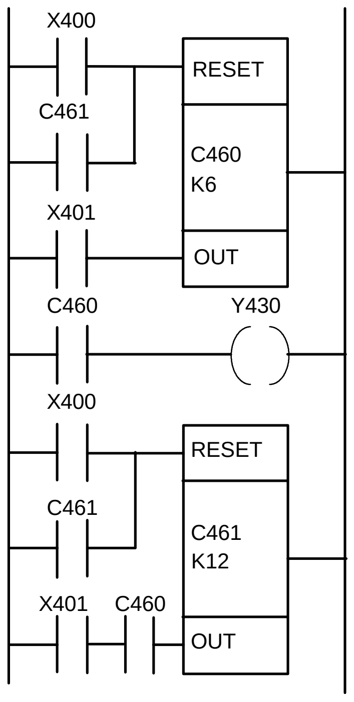


Figure 10.5 *Counting task*

As a further illustration of the use of a counter, consider the problem of the control of a machine which is required to direct 6 tins along one path for packaging in a box and then 12 tins along another path for packaging in another box (Figure 10.5). A deflector plate might be controlled by a photocell sensor which gives an output every time a tin passes it. Thus the number of pulses from the sensor has to be counted and used to control the deflector. Figure 10.6 shows the ladder program that could be used. Mitsubishi notation has been used.



```
LD    X400
OR    C461
RST   C460
K     6
LD    X401
OUT   C460
LD    C460
OUT   Y430
LD    X400
OR    C461
RST   C461
K     12
LD    X401
AND   C460
OUT   C461
```

Figure 10.6 *Ladder program for Figure 10.5 task*

When there is a pulse input to X400, both the counters are reset. The input to X400 could be the push button switch used to start the conveyor moving. The input which is counted is X401. This might be an input from a photocell sensor which detects the presence of tins passing along the conveyor. C460 starts counting after X400 is momentarily closed. When C460 has counted six items, it closes its contacts and so gives an output at Y430. This might be a solenoid which is used to activate a deflector to deflect items into one box or another. Thus the deflector might be in such a position that the first six tins passing along the conveyor are deflected into the 6-pack box, then the deflector plate is moved to allow tins to pass to the 12-pack box. When C460 stops counting it closes its contacts and so allows C461 to start counting. C461 counts for 12 pulses to X401 and then closes its contacts. This results in both counters being reset and the entire process can repeat itself.

Counters can be used to ensure that a particular part of a sequence is repeated a known number of times. This is illustrated by the following program which is designed to enable a three-cylinder, double solenoid-controlled arrangement (Figure 10.7(a)) to give the sequence A+, A–, A+, A–, A+, A–, B+, C+, B–, C–. The A+, A– sequence is repeated three times before B+, C+, B–, C– occur. We can use a counter to enable this repetition. Figure 10.7(b) shows a possible program. The counter only allows B+ to occur after it has received three inputs corresponding to three a– signals.

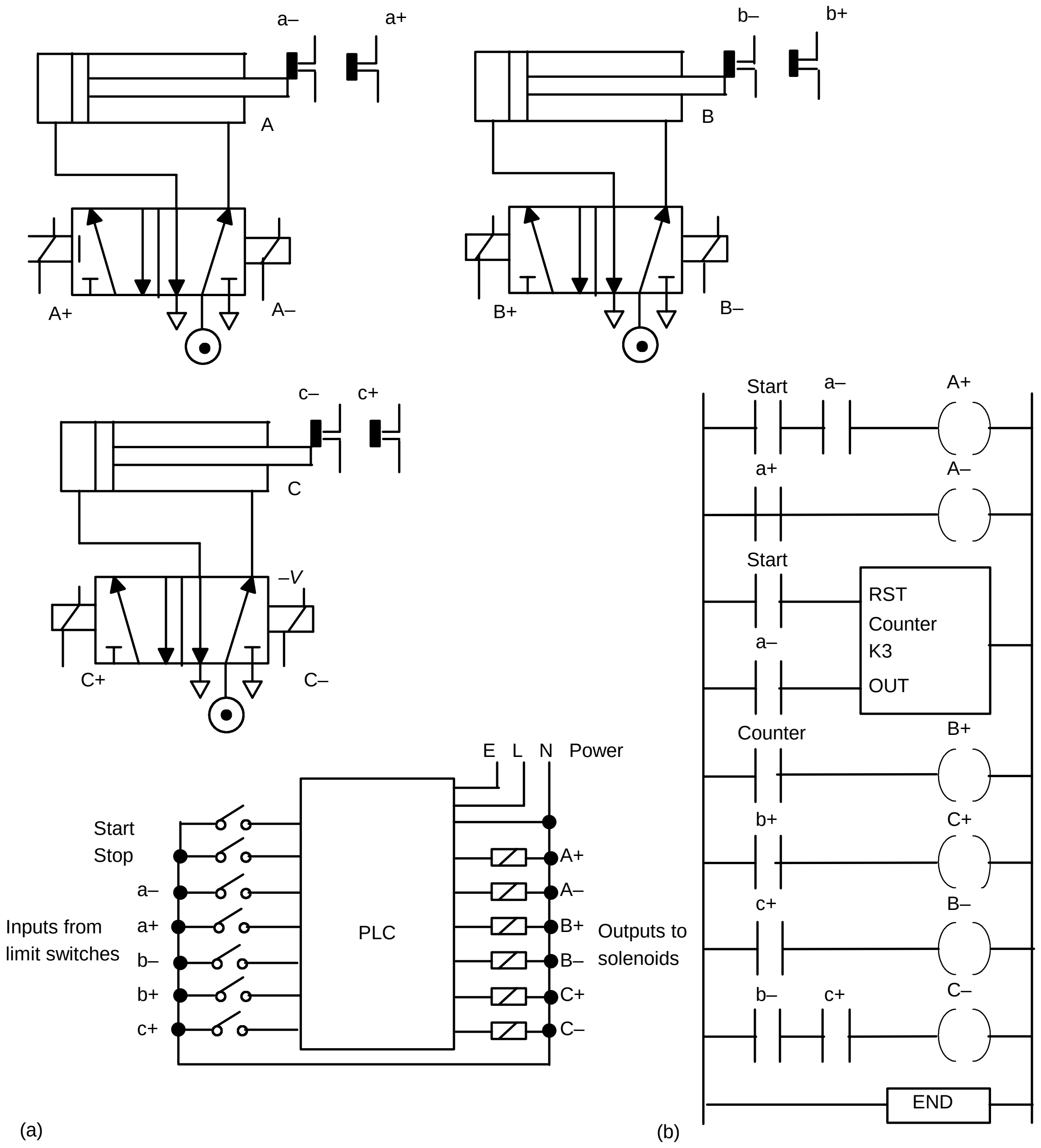


Figure 10.7 *(a) Three-cylinder system, (b) program*

10.3 Up and down counting

It is possible to program up- and down-counters together. Consider the task of counting products as they enter a conveyor line and as they leave it, or perhaps cars as they enter a multi-storage parking lot and as they leave it. An output is to be triggered if the number of items/cars entering is some number greater than the number leaving, i.e. the number in the parking lot has reached a 'saturation' value. The output might be to illuminate a 'No empty spaces' sign. Suppose we use the up-counter for items entering and the count down for items leaving. Figure 10.8(a) shows the basic form a ladder program for such an application can take. When an item enters it gives a pulse on input In 1. This increases the count by one. Thus each item entering increases the accumulated count by 1. When an item leaves it gives an input to In 2. This reduces the number by 1. Thus each item leaving reduces the accumulated count by 1. When the accumulated value reaches the preset value, the output Out 1 is switched on. Figure 10.8(b) shows the implementation of this program with an Allen-Bradley program.

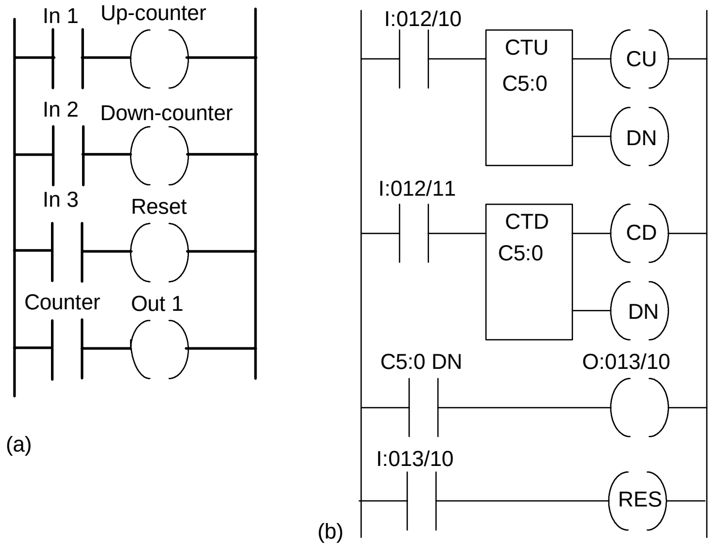


Figure 10.8 *(a) Using up- and down-counters, (b) Allen-Bradley program*

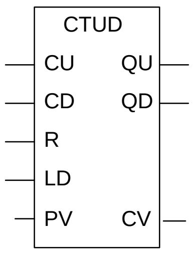


Figure 10.9 *IEC 1131-3 standard symbol for up-down counter*

Up-down counters are available as single entities. Figure 10.9 shows the IEC 1131-3 standard symbol. The counter has two inputs CU and CD and counts up the number of pulses detected at the input CU and counts down the number of pulses detected at input CD. If the counter input reaches zero, the QD output is set on and the counting down stops. If the count reaches the maximum value PV, the QU output is set on and the counting up stops. CV is the count value. LD can be used to preset the counter output CV with the value PV. The reset R clears the counter input to zero.

Figure 10.10 shows how the above system might appear for a Siemens PLC and the associated program instruction list. CU is the count up input and CD the count down. R is the reset. The set accumulator value is loaded via F0.0, this being an internal relay.

11 Shift registers

The term *register* is used for an electronic device in which data can be stored. An internal relay, see Chapter 7, is such a device. The *shift register* is a number of internal relays grouped together which allow stored bits to be shifted from one relay to another. This chapter is about shift registers and how they can be used where a sequence of operations is required or to keep track of particular items in a production system.

11.1 Shift registers

A register is a number of internal relays grouped together, normally 8, 16 or 32. Each internal relay is either effectively open or closed, these states being designated as 0 and 1. The term *bit* is used for each such binary digit. Therefore, if we have eight internal relays in the register we can store eight 0/1 states. Thus we might have:

Internal relays

1	2	3	4	5	6	7	8

and each relay might store an on-off signal such that the state of the register at some instant is:

1	0	1	1	0	0	1	0

i.e. relay 1 is on, relay 2 is off, relay 3 is on, relay 4 is on, relay 5 is off, etc. Such an arrangement is termed an 8-bit register. Registers can be used for storing data that originate from input sources other than just simple, single on-off devices such as switches.

With the *shift register* it is possible to shift stored bits. Shift registers require three inputs, one to load data into the first location of the register, one as the command to shift data along by one location and one to reset or clear the register of data. To illustrate this, consider the following situation where we start with an 8-bit register in the following state:

1	0	1	1	0	0	1	0

Suppose we now receive the input signal 0. This is an input signal to the first internal relay.

Input 0

→	1	0	1	1	0	0	1	0

If we also receive the shift signal, then the input signal enters the first location in the register and all the bits shift along one location. The last bit overflows and is lost.

0	1	0	1	1	0	0	1	→

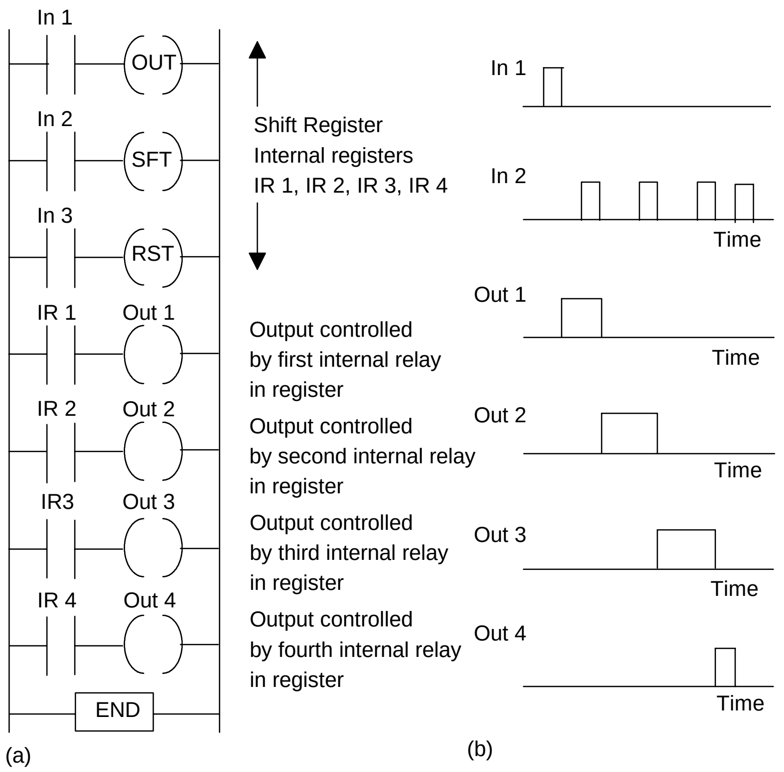
Overflow 0

Thus a set of internal relays that were initially on, off, on, on, off, off, on, off are now off, on, off, on, on, off, off, on.

The grouping together of internal relays to form a shift register is done automatically by a PLC when the shift register function is selected. With the Mitsubishi PLC, this is done by using the programming code SFT (shift) against the internal relay number that is to be the first in the register array. This then causes a block of relays, starting from that initial number, to be reserved for the shift register.

11.2 Ladder programs

Consider a 4-bit shift register and how it can be represented in a ladder program (Figure 11.1(a)). The input In 3 is used to reset the shift register, i.e. put all the values at 0. The input In 1 is used to input to the first internal relay in the register. The input In 2 is used to shift the states of the internal relays along by one. Each of the internal relays in the register, i.e. IR 1, IR 2, IR 3 and IR 4, is connected to an output, these being Out 1, Out 2, Out 3 and Out 4.

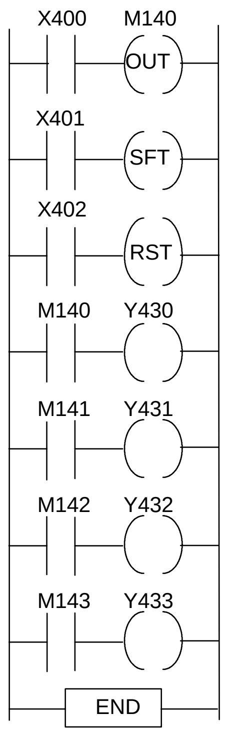


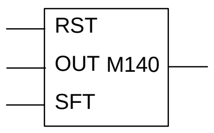
In 1
OUT
In 2
SFT
In 3
RST
IR 1
Out 1
IR 2
Out 2
IR3
Out 3
IR 4
Out 4
END
Shift Register
Internal registers
IR 1, IR 2, IR 3, IR 4
Output controlled by first internal relay in register
Output controlled by second internal relay in register
Output controlled by third internal relay in register
Output controlled by fourth internal relay in register
In 1
In 2
Out 1
Out 2
Out 3
Out 4
Time
(a)
(b)

Figure 11.1 *The shift register*

Suppose we start by supplying a momentary input to In 3. All the internal relays are then set to 0 and so the states of the four internal relays IR 1, IR 2, IR 3 and IR 4 are 0, 0, 0, 0. When In 1 is momentarily closed there is a 1 input into the first relay. Thus the states of the internal relays IR 1, IR 2, IR 3 and IR 4 are now 1, 0, 0, 0. The IR 1 contacts close and we thus end up with an output from Out 1. If we now supply a momentary input to In 2, the 1 is shifted from the first relay to the second. The states of the internal relays are now 0, 1, 0, 0. We now have no input from Out 1 but an output from Out 2. If we supply another momentary input to In 2, we shift the states of the relays along by one location to give 0, 0, 1, 0. Outputs 1 and 2 are now off but Out 3 is on. If we supply another momentary input to In 2 we again shift the states of the relays along by one and have 0, 0, 0, 1. Thus now, outputs 1, 2 and 3 are off and output 4 has been switched on. When another momentary input is applied to In 2, we shift the states of the relays along by one and have 0, 0, 0, 0 with the 1 overflowing and being lost. All the outputs are then off. Thus the effect of the sequence of inputs to In 2 has been to give a sequence of outputs Out 1, followed by Out 2, followed by Out 3, followed by Out 4. Figure 11.1(b) shows the sequence of signals.

Figure 11.2 shows the Mitsubishi version of the above ladder program and the associated instruction list. Instead of the three separate outputs for reset, output and shift, the Mitsubishi shift register might appear in a program as a single function box, as shown in the Figure. With the Mitsubishi shift register, the M140 is the address of the first relay in the register.





Representation of the three shift register elements in a single box

```
LD    X400
OUT   M140
LD    X401
SFT   M140
LD    X402
RST   M140
LD    M140
OUT   Y430
LD    M141
OUT   Y431
LD    M142
OUT   Y432
LD    M143
OUT   Y433
END
```

Figure 11.2 *Mitsubishi program*

13 Designing systems

This chapter considers how programs are designed and how they and a PLC system can be tested and faults found. This involves consideration of both the hardware and the software.

13.1 Program development

Whatever the language in which a program is to be written, a systematic approach to the problem can improve the chance of high quality programs being generated in as short a time as possible. A systematic design technique is likely to involve the following steps:

1 A definition of what is required with the inputs and outputs specified.
2 A definition of the algorithm to be used. An algorithm is a step-by-step sequence which defines a method of solving the problem. This can often be shown by a flow chart or written in pseudocode, this involving the use of the words BEGIN, DO, END, IF-THEN-ELSE, WHILE-DO.
3 The algorithm is then translated into instructions that can be inputted to the PLC. Because programs are often very long and can end up difficult to write as a long single block and even more difficult to later follow for fault finding and maintenance, it is advisable to break the program down into areas which are then further subdivided until manageable size blocks of program occur. This is termed *top-down design*.
4 The program is then tested and debugged.
5 The program is documented so that any person using or having to modify the program at a later date understands how the program works.

13.1.1 Flow charts and pseudocode

Figure 13.1(a) shows the symbols used in flow charts. Each step of an algorithm is represented by one or more of these symbols and linked by lines to represent the program flow, Figure 13.1(b) illustrating this. Pseudocode is a way of describing the steps in an algorithm in an informal way.

Consider how the following program operations can be represented by flow charts and pseudocode and then programmed using ladder and sequential function chart programming:

1 *Sequential*
Consider a sequence when event A has to be followed by event B. Figure 13.2(a) shows how this can be represented by a flow chart.

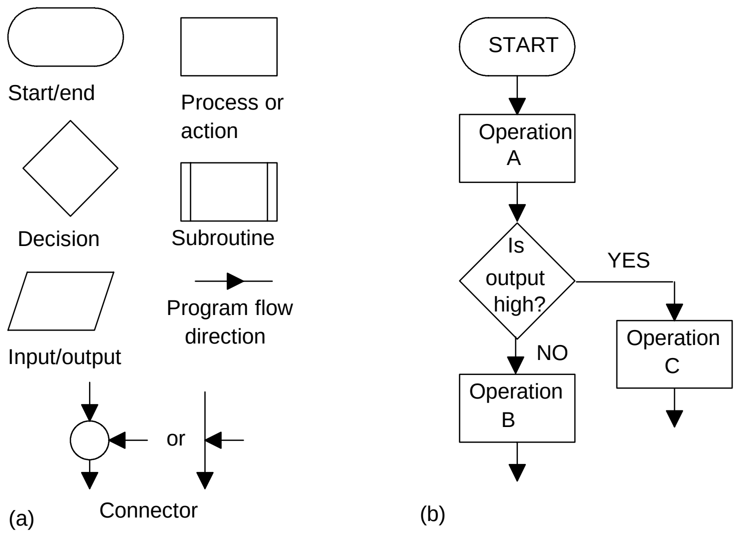


Figure 13.1 *(a) Flow chart symbols, (b) example of a simple flow chart*

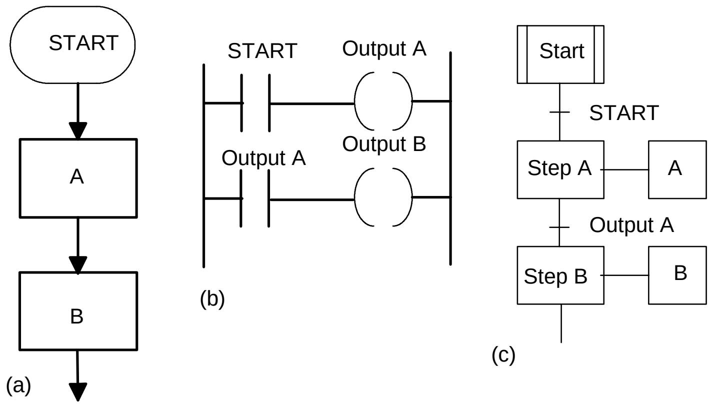


Figure 13.2 *Sequence*

In pseudocode this is written as:

```
BEGIN A
    DO A
END A
BEGIN B
    DO B
END B
```

A sequence can be translated into a ladder program in the way shown in Figure 13.2(b). When the start input occurs then output A happens. When action A happens it operates output A relay and results in output B occurring. Figure 13.2(c) shows the sequential function chart representation of a sequence.

2 *Conditional*

Figure 13.3(a) shows the flow chart for when A or B is to happen if a particular condition X being YES or NO occurs. The pseudocode to describe this involves the words IF-THEN-ELSE-ENDIF.

```
IF X
THEN
    BEGIN A
    DO A
    END A
ELSE
    BEGIN B
    DO B
    END B
ENDIF X
```

Such a condition can be represented by the ladder diagram shown in Figure 13.3(b). When the start input occurs, the output will be A if there is an input to X, otherwise the output is B. Figure 13.3(c) shows the sequential function chart for such selective branching.

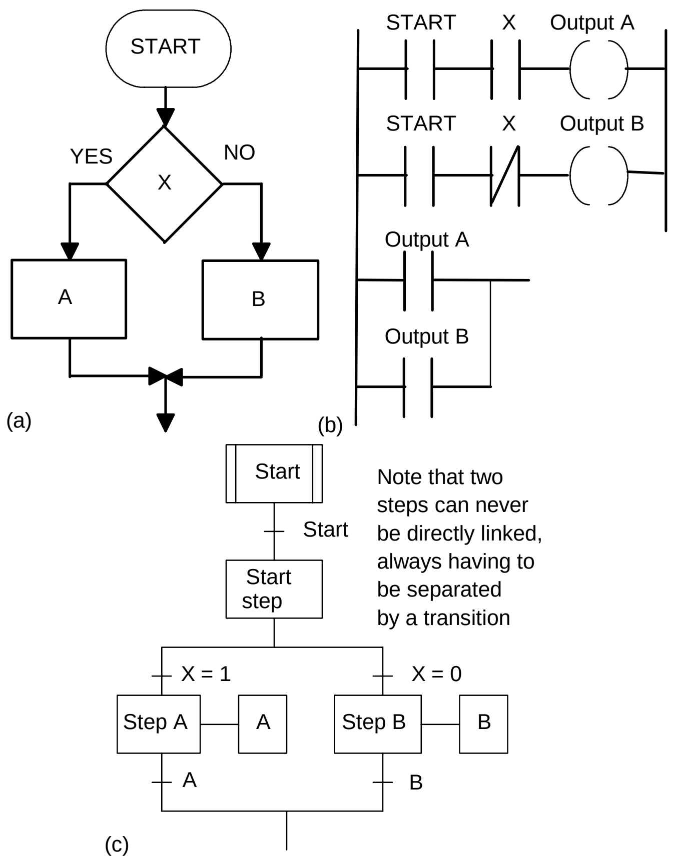


Figure 13.3 *Conditional*

3 *Looping*

A loop is a repetition of some element of a program, the element being repeated as long as some condition prevails. Figure 13.4(a) shows how this can be represented by a flow chart. As long as condition X is realised then the sequence A followed by B occurs and is repeated. When X is no longer realised then the program continues and the looping through A and B ceases.

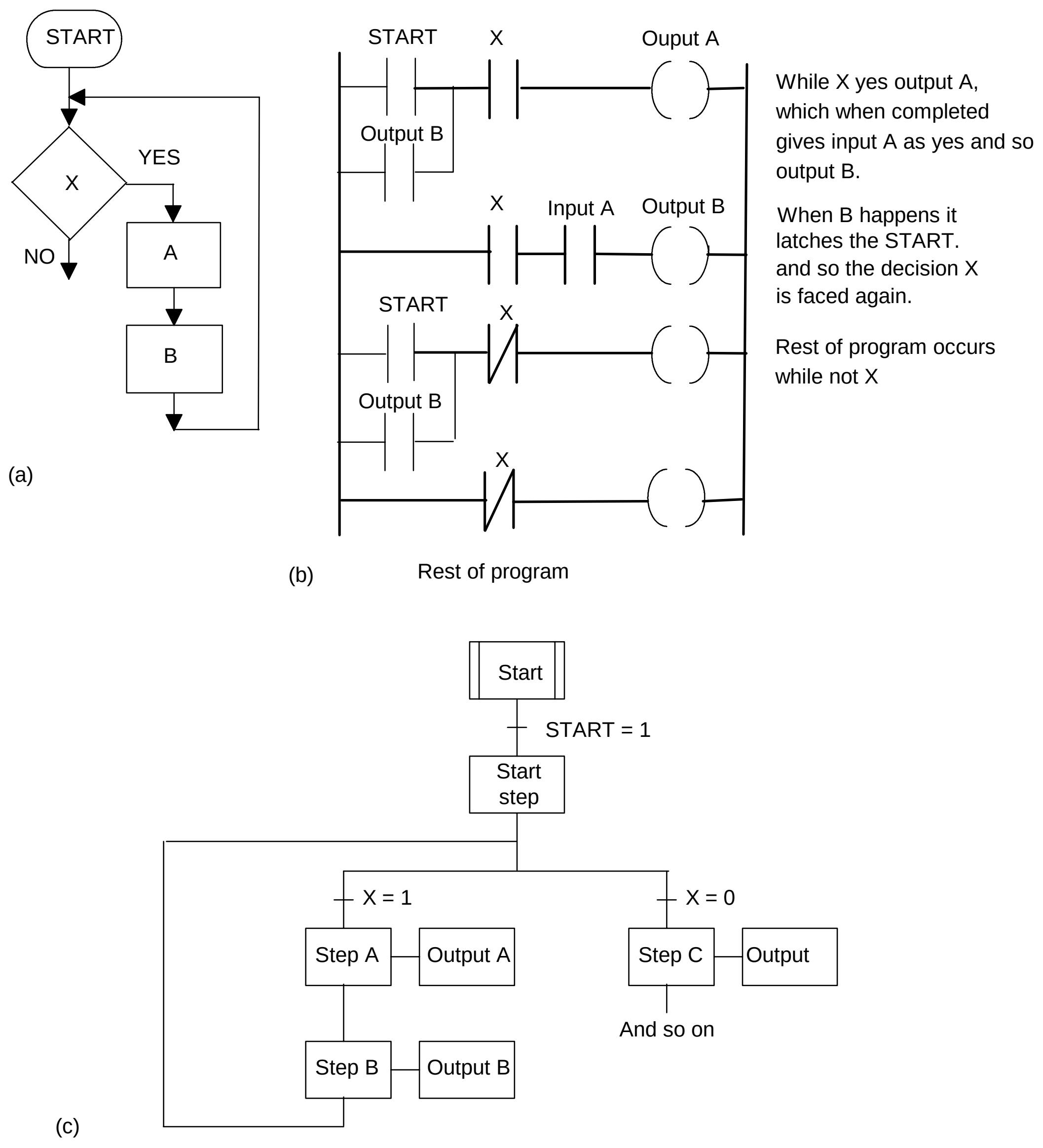

START
X
YES
NO
A
B
(a)
START
X
Ouput A
Output B
While X yes output A, which when completed gives input A as yes and so output B.
X
Input A
Output B
When B happens it latches the START. and so the decision X is faced again.
START
X
Output B
Rest of program occurs while not X
X
(b)
Rest of program
Start
START = 1
Start step
X = 1
X = 0
Step A
Output A
Step C
Output
And so on
Step B
Output B
(c)

Figure 13.4 *Looping*

In pseudocode this can be represented by using the words WHILE-DO -ENDWHILE:

```
WHILE X
```

```
        BEGIN A
        DO A
        END A
        BEGIN B
        DO B
        END B
    ENDWHILE X
```

Figure 13.4(b) shows how this can be represented by a ladder diagram and using an internal relay. Figure 13.4(c) shows the sequential flow chart.

Where a loop has to be repeated for a particular number of times, a counter can be used, receiving an input pulse each time a loop occurs and switching out of the loop sequence when the required number of loops has been completed (Figure 13.5).

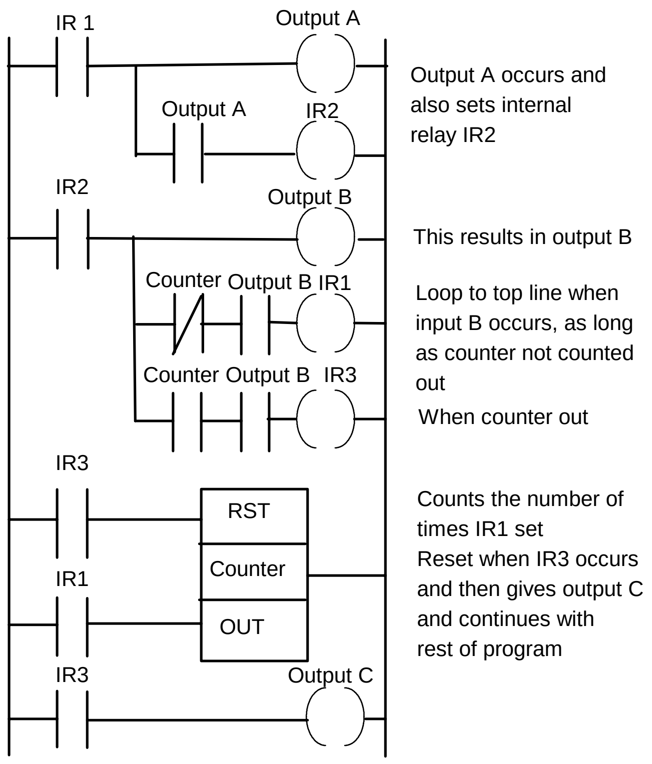

IR 1
Output A
Output A occurs and also sets internal relay IR2
Output A
IR2
IR2
Output B
This results in output B
Counter
Output B
IR1
Loop to top line when input B occurs, as long as counter not counted out
Counter
Output B
IR3
When counter out
IR3
RST
Counter
OUT
Counts the number of times IR1 set
Reset when IR3 occurs and then gives output C and continues with rest of program
IR1
IR3
Output C

Figure 13.5 *Looping*

13.2 Safe systems

Modern safety legislation charges employers with duties which include making the workplace safe and without risks to health and ensuring that plant and machinery are safe and that safe systems of work are set and followed. There is thus a need to assess the risks in the workplace. This means looking for hazards, i.e. anything that can cause harm, deciding who might be harmed and how, evaluating the risks that somebody will be harmed by a hazard and whether existing precautions are adequate or

14 Programs

This chapter extends the examples given in previous chapters to show programs developed to complete specific tasks. These include tasks which involve temperature control and a number involving pneumatic valves.

14.1 Temperature control

Consider the task of using a PLC as an on-off controller for a heater in the control of temperature in some enclosure. The heater is to be switched on when the temperature falls below the required temperature and switched off when the temperature is at or above the required temperature. The basic algorithm might be considered to be:

```
IF temperature below set value
THEN
    DO switch on heater
ELSE
    DO switch off heater
ENDIF
```

The sensor used for the temperature might be a thermocouple, a thermistor or an integrated chip (see Section 2.1.5). When connected in an appropriate circuit, the sensor will give a suitable voltage signal related to the temperature. This voltage can be compared, using an operational amplifier, with the voltage set for the required temperature so that a high output signal is given when the temperature is above the required temperature and a low output signal when it is below. Thus when the temperature falls from above the required temperature to below it, the signal switches from a high to a low value. This transition can be used as the input to a PLC. The PLC can then be programmed to give an output when there is a low input and this output used to switch on the heater.

Figure 14.1 shows the arrangement that might be used and a Mitsubishi ladder program. The input from the operational amplifier has been connected to the input port with the address X400. This input has contacts which are normally closed. When the input goes high, the contacts open. The output is taken from the output port with the address Y430. Thus there is an output when the input is low and no output when the input is high.

Figure 14.2 shows the key sequence that would be used with a graphic programmer to enter the part of the program given in Figure 14.1.

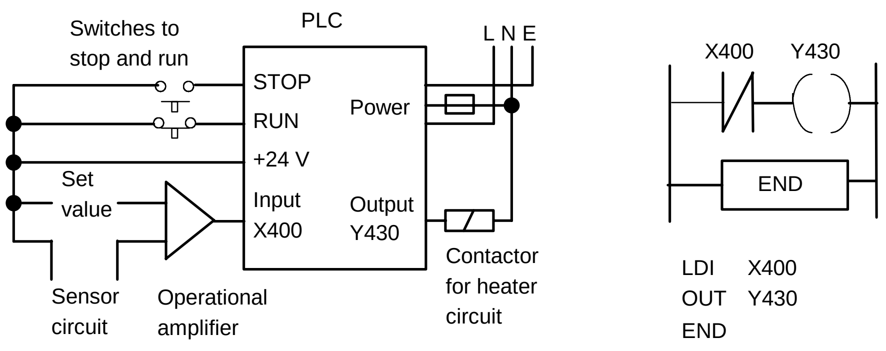


Figure 14.1 *Temperature control*

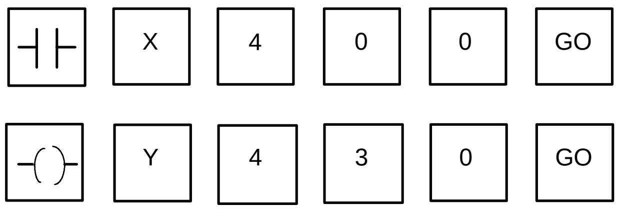


Figure 14.2 *Entering program graphically*

In Siemens format the program given in Figure 14.1 could be as shown in Figure 14.3 and in Allen-Bradley form as shown in Figure 14.4. To illustrate how such a program might be entered using a computer, Figure 14.5 also shows the function keys that would be used to enter that part of the program with the Allen-Bradley software loaded. At each instant in the program a screen displays prompts and lists the function keys and their significance.

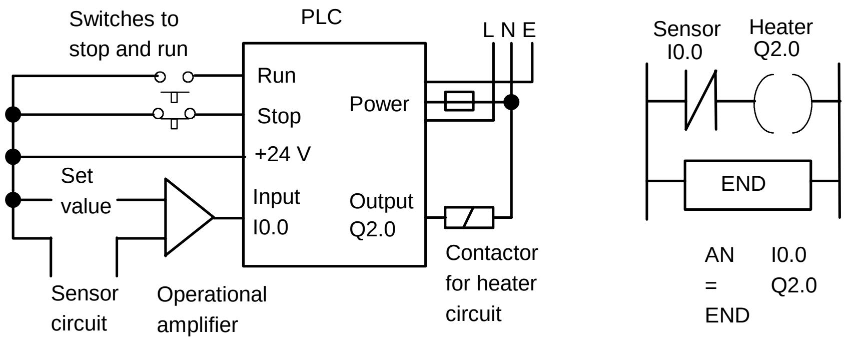
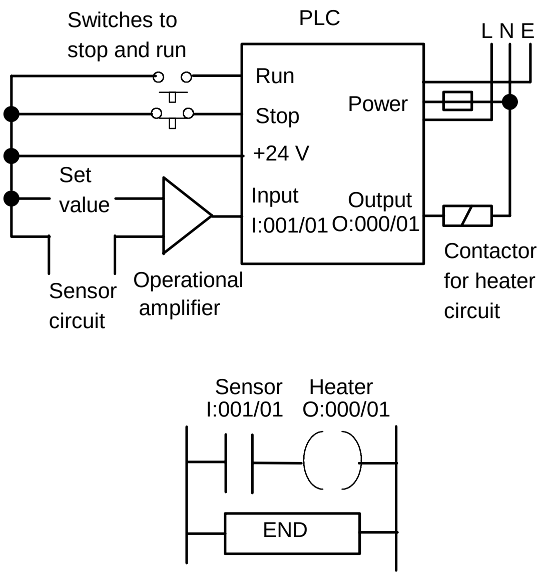


Figure 14.3 *Temperature control*



Offline programming selected F1
Mode selected: Create file F6
File name entered: TEMP, enter key pressed
Select the processor: F2 pressed until relevant one on screen.
Create file F1
Monitor file F8. This shows the file as just consisting of the END rung.
Edit F10. This is to enable the file to be modified.
Append rung and append instruction F3 pressed twice.
-||- F1 key pressed.
I;001/01 entered, enter key pressed.
-( )- F3 key pressed.
O;000/01 entered, enter key pressed.
To accept the program Esc is pressed.

Figure 14.4 *Allen-Bradley program*

Consider a more complex temperature control task involving a domestic central heating system (Figure 14.5). The central heating boiler is to be thermostatically controlled and supply hot water to the radiator system in the house and also to the hot water tank to provide hot water from the taps in the house. Pump motors have to be switched on to direct the hot water from the boiler to either, or both the radiator and hot water systems according to whether the temperature sensors for the room temperature and the hot water tank indicate that the radiators or tank need heating. The entire system is to be controlled by a clock so that it only operates for certain hours of the day. Figure 14.6(a) shows how a Mitsubishi PLC, and Figure 14.6(b) a Siemens PLC, might be used.

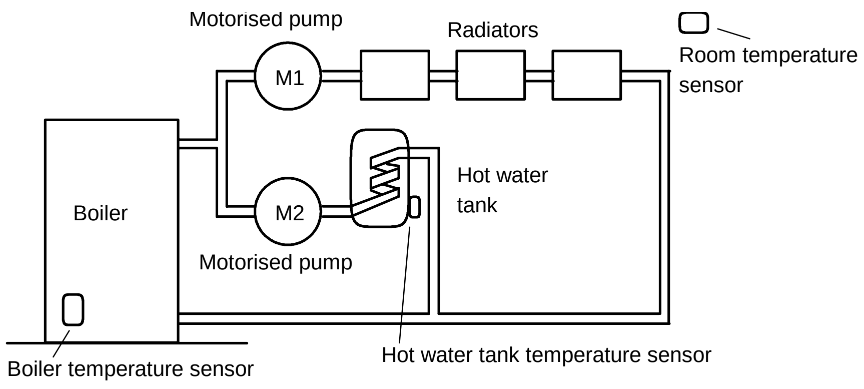


Figure 14.5 *Central heating system*