CHAPTER 4

Microcontroller Program Development

Chapter Outline

Before writing a program, it is always helpful first to think and plan the structure of the program. Although simple programs can easily be developed by writing the code directly without any preparation, the development of complex programs almost always become easier if an algorithm is first derived. Once the algorithm is ready, coding the actual program is not a difficult task.

A program's algorithm can be described in a variety of graphical and text-based methods, such as flow charts, structure charts, data flow diagrams, and program description languages (PDLs). The problem with graphical techniques is that it can be very time consuming to draw shapes with text inside them. Also, it is a tedious task to modify an algorithm described using graphical techniques.

Flow charts can be very useful to describe the flow of control and data in small programs where there are only a handful of diagrams, usually not extending beyond a page or two. The PDL can be useful to describe the flow of control and data in small-to-medium size programs. The main advantage of the PDL description is that it is very easy to modify a given PDL since it only consists of text.

In this book, we will mainly be using the PDL, but flow charts will also be given where it is felt to be useful. The next few sections briefly describe the basic building blocks of the

PIC Microcontroller Projects in C. http://dx.doi.org/10.1016/B978-0-08-099924-1.00004-6

PDL and its equivalent flow charts. It is left to the readers to decide which method to use during the development of their programs.

4.1 Using the PDL and Flow Charts

PDL is a free-format English-like text that describes the flow of control and data in a program. PDL is not a programming language. It is a collection of some keywords that enable a programmer to describe the operation of a program in a stepwise and logical manner. In this section, we will look at the basic PDL statements and their flow chart equivalents. The superiority of the PDL over flow charts will become obvious when we have to develop medium-to-large programs.

4.1.1 BEGIN–END

Every PDL program description should start with a BEGIN and end with an END statement. The keywords in a PDL description should be highlighted to make the reading easier. The program statements should be indented and described between the PDL keywords. An example is shown in Figure 4.1 together with the equivalent flow diagram.

4.1.2 Sequencing

For normal sequencing, the program statements should be written in English text to describe the operations to be performed. An example is shown in Figure 4.2 together with the equivalent flow chart.

4.1.3 IF–THEN–ELSE–ENDIF

IF, THEN, ELSE, and ENDIF should be used to conditionally change the flow of control in a program. Every IF keyword should be terminated with a THEN, and every IF block should be terminated with an ENDIF keyword. The use of the ELSE statement is optional and depends on the application. Figure 4.3 shows an example of using IF–THEN–ENDIF, while Figure 4.4 shows the use of IF–THEN–ELSE–ENDIF statements in a program and their equivalent flow charts.

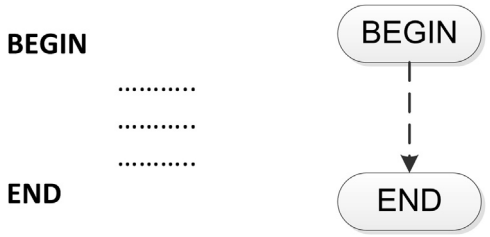

Figure 4.1: BEGIN–END Statement and Equivalent Flow Chart.

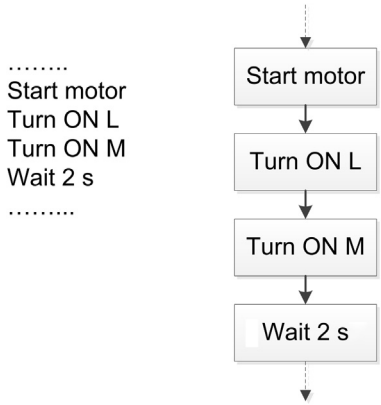

Figure 4.2: Sequencing and Equivalent Flow Chart.

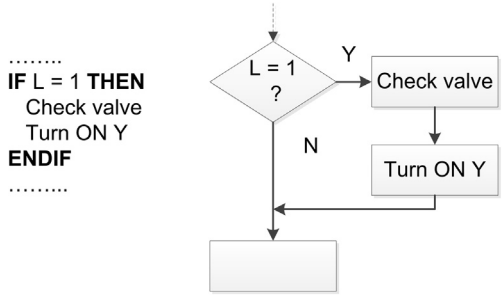

Figure 4.3: Using IF—THEN—ENDIF Statements.

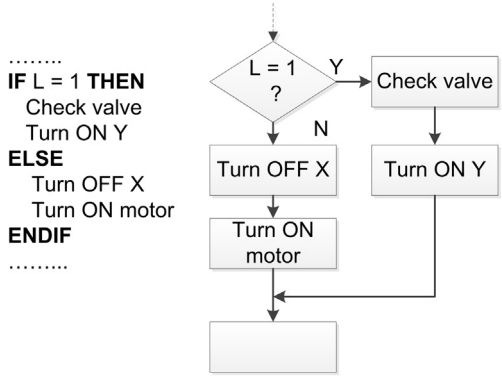

Figure 4.4: Using IF—THEN—ELSE—ENDIF Statements.

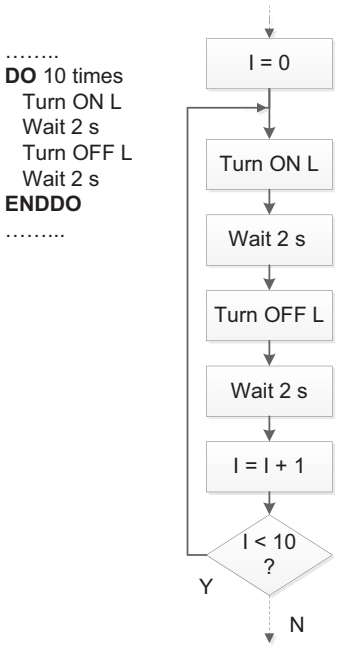

Figure 4.5: Using DO−ENDDO Statements.

4.1.4 DO−ENDDO

The DO−ENDDO statements should be used when it is required to create iterations, or conditional or unconditional loops in programs. Every DO statement should be terminated with an ENDDO. Other keywords, such as FOREVER or WHILE, can be used after the DO statement to indicate an endless loop or a conditional loop, respectively. Figure 4.5 shows an example of a DO−ENDDO loop executed 10 times. Figure 4.6 shows an endless loop created using the FOREVER statement. The flow chart equivalents are also shown in the figures.

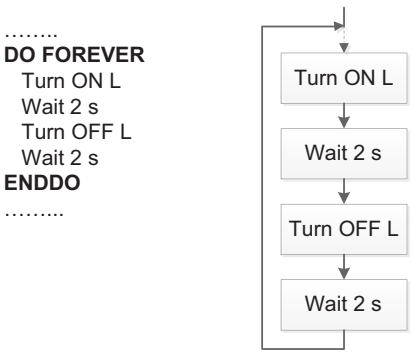

Figure 4.6: Using DO−FOREVER Statements.

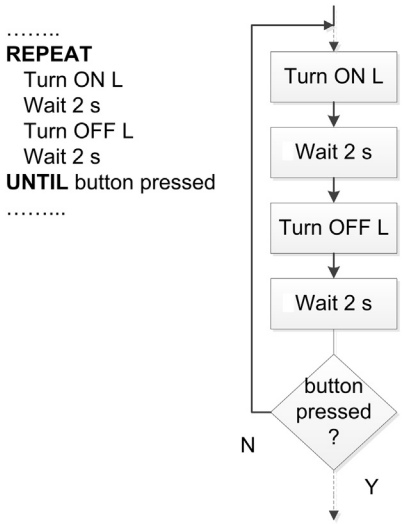

Figure 4.7: Using REPEAT−UNTIL Statements.

4.1.5 REPEAT−UNTIL

REPEAT−UNTIL is similar to DO−WHILE, but here the statements enclosed by the REPEAT−UNTIL block are executed at least once, while the statements enclosed by DO−WHILE may not execute at all if the condition is not satisfied just before entering the DO statement. An example is shown in Figure 4.7, with the equivalent flow chart.

4.1.6 Calling Subprograms

In some applications, a program consists of a main program and a number of subprograms (or functions). A subprogram activation in PDL should be shown by adding the CALL statement before the name of the subprogram. In flow charts, a rectangle with vertical lines at each side should be used to indicate the invocation of a subprogram. An example call to a subprogram is shown in Figure 4.8 for both a PDL description and a flow chart. Optionally, the input−output data to a function can be listed if desired. The following example shows how the temperature can be passed to function DISPLY as an input:

```
CALL DISPLY(I: temperature)
```

In the following function call the temperature is passed to the function called CONV. The function formats the temperature for display and returns it to the calling program:

```
CALL CONV(I: temperature, O: formatted temperature)
```

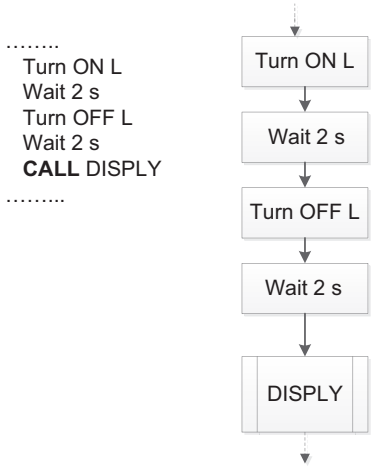

Figure 4.8: Calling a Subprogram.

4.1.7 Subprogram Structure

A subprogram should begin and end with the keywords BEGIN/name and END/name, respectively, where *name* is the name of the subprogram. In flow chart representation, a horizontal line should be drawn inside the BEGIN box, and the name of the subprogram should be written at the lower half of the box. An example subprogram structure is shown in Figure 4.9 for both a PDL description and a flow chart.

Interrupt service routines can be shown using the same method, but the keyword ISR can be inserted in front of the function name to identify that the function is actually an

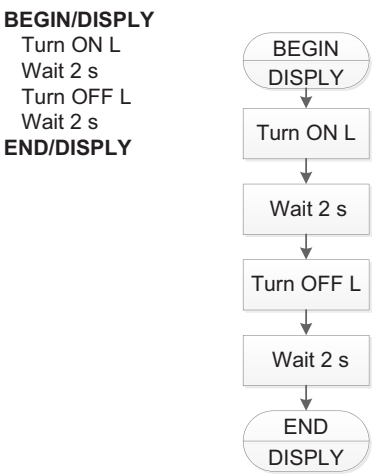

Figure 4.9: Subprogram Structure.

interrupt service routine. For example, in Figure 4.9, assuming that function DISPLY is an interrupt service routine, the function body can be written as

```
BEGIN/ISR:DISPLY
   Turn ON L
   Wait 2 s
   Turn OFF L
   Wait 2 s
END/ISR:DISPLY
```

4.2 Examples

Some examples are given in this section to show how the PDL and flow charts can be used in program development.

Example 4.1

It is required to write a program to convert hexadecimal numbers "A" to "F" into the decimal format. Show the algorithm using a PDL and also draw the flow chart. Assume that the number to be converted is called HEX_NUM, and the output number is called DEC_NUM.

Solution 4.1

The required PDL is

```
BEGIN
   IF HEX_NUM = "A" THEN
     DEC_NUM = 10
   ELSE IF HEX_NUM = "B" THEN
     DEC_NUM = 11
   ELSE IF HEX_NUM = "C" THEN
     DEC_NUM = 12
   ELSE IF HEX_NUM = "D" THEN
     DEC_NUM = 13
   ELSE IF HEX_NUM = "E" THEN
     DEC_NUM = 14
   ELSE IF HEX_NUM = "F" THEN
     DEC_NUM = 15
   ENDIF
END
```

The required flow chart is shown in Figure 4.10. Note that it is much easier to write PDL statements than it is to draw flow chart shapes and write text inside them.

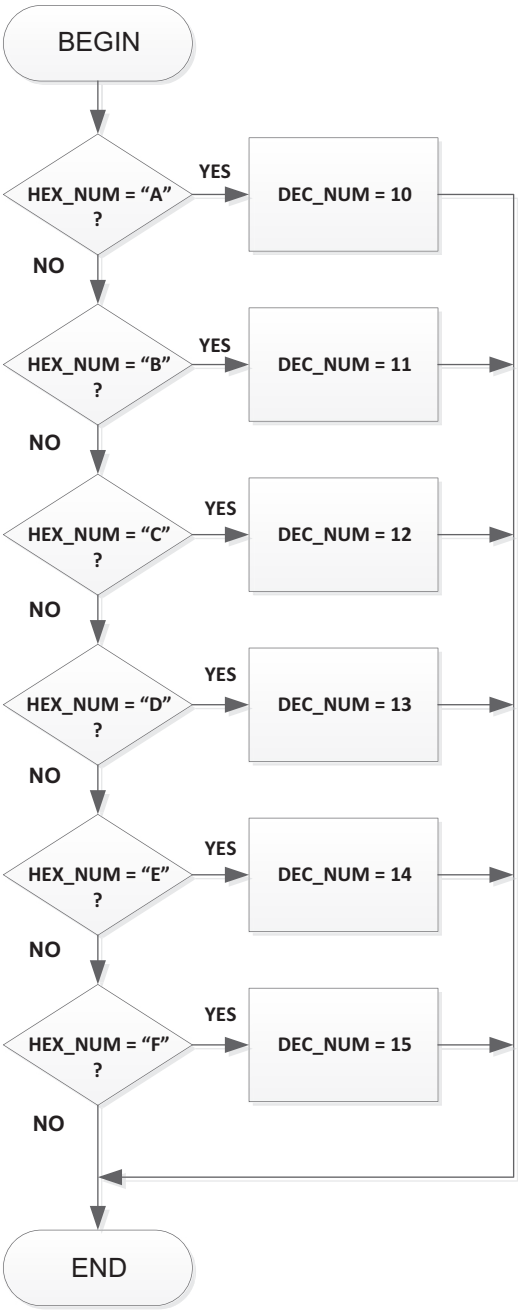

Figure 4.10: Flow Chart Solution.

Example 4.2

The PDL of part of a program is given as follows:

```
J = 0
M = 0
DO WHILE J < 10
   DO WHILE M < 20
      Flash the LED
      Increment M
   ENDDO
   Increment J
ENDDO
```

Show how this PDL can be implemented by a flow chart.

Solution 4.2

The required flow chart is shown in Figure 4.11. Here again, note how complicated the flow chart can be even for a simple nested DO WHILE loop.

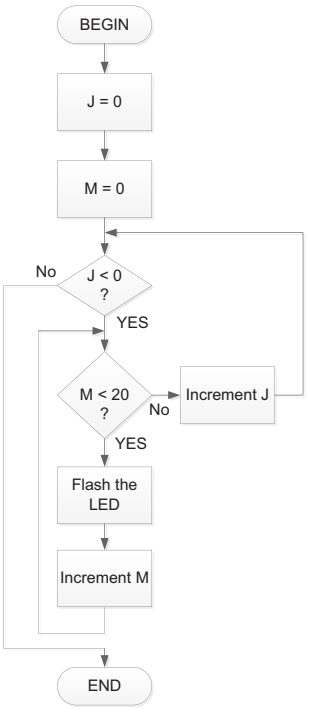

Figure 4.11: Flow Chart Solution.

Example 4.3

It is required to write a program to calculate the sum of integer numbers between 1 and 100. Show the algorithm using a PDL and also draw the flow chart. Assume that the sum will be stored in a variable called SUM.

Solution 4.3
The required PDL is

```
BEGIN
   SUM = 0
   I = 1
   DO 100 TIMES
      SUM = SUM + I
      Increment I
   ENDDO
END
```

The required flow chart is shown in Figure 4.12.

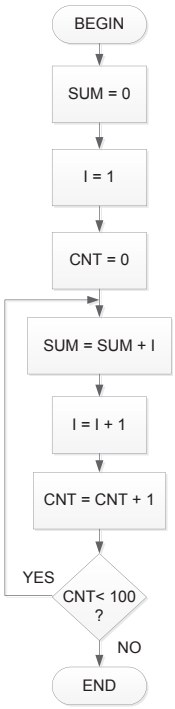

Figure 4.12: Flow Chart Solution.

Example 4.4

It is required to write a program to calculate the sum of all the even numbers between 1 and 10 inclusive. Show the algorithm using a PDL and also draw the flow chart. Assume that the sum will be stored in a variable called SUM.

Solution 4.4
The required PDL is

```
BEGIN
   SUM = 0;
   CNT = 1
   REPEAT
      IF CNT is even number THEN
         SUM = SUM + CNT
      ENDIF
      INCREMENT CNT
      UNTIL CNT > 10
END
```

The required flow chart is shown in Figure 4.13. Note how complicated the flow chart can be for a very simple problem such as this.

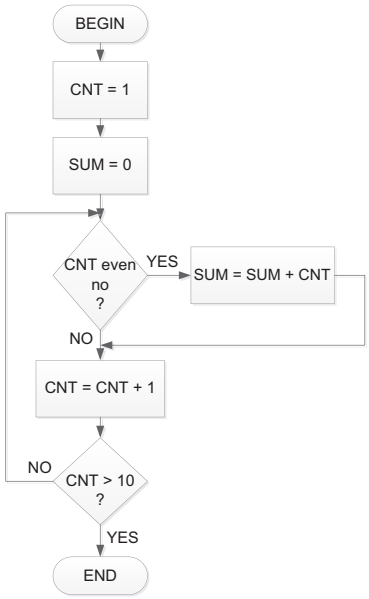

Figure 4.13: Flow Chart Solution.

Example 4.5

It is required to write a program to turn ON a light emitting diode (LED) when a button is pressed and to turn it OFF when the button is released. Assuming that initially the LED is to be OFF, write the PDL statements for this example.

Solution 4.5
The required PDL statements are as follows:

```
BEGIN
   Turn OFF LED
   DO FOREVER
      IF Button is pressed THEN
        Turn ON LED
      ELSE
        Turn OFF LED
      ENDIF
   ENDDO
END
```

Example 4.6

A temperature sensor is connected to the A/D input of a microcontroller. It is required to write a program to read the temperature every second and display it on a liquid crystal display (LCD). Use function DISPLAY to format and display the temperature. Show the PDL statements for this example.

Solution 4.6
The required PDL statements are

```
BEGIN
   Configure the A/D port
   DO FOREVER
      Read Temperature from A/D port
      CALL DISPLAY
      Wait 1 s
   ENDDO
END
BEGIN/DISPLAY
   Format temperature for LCD display
   Display temperature on LCD
END/DISPLAY
```

4.3 Representing for Loops in Flow Charts

Most programs include some form of iteration or looping. One of the easiest ways to create a loop in a C program is by using the *for* statement. This section shows how a *for* loop can be represented in a flow chart. As shown below, there are several methods of representing a *for* loop in a flow chart.

Suppose that we have a *for* loop as shown below and we wish to draw an equivalent flow chart.

```
for(m = 0; m < 10; m++)
{
  Cnt = Cnt + 2* m;
}
```

4.3.1 Method 1

Figure 4.14 shows one of the methods for representing the above *for* loop as with a flow chart. Here, the flow chart is drawn using the basic primitive components.

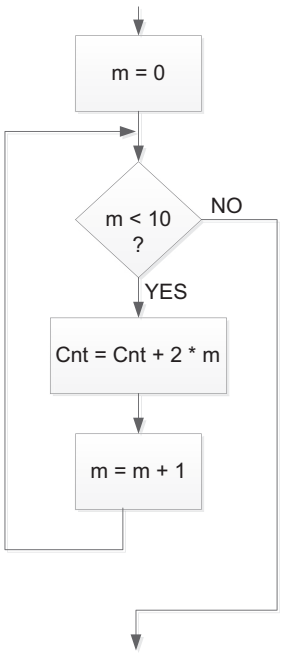

Figure 4.14: Method 1 for Representing a *for* Loop.

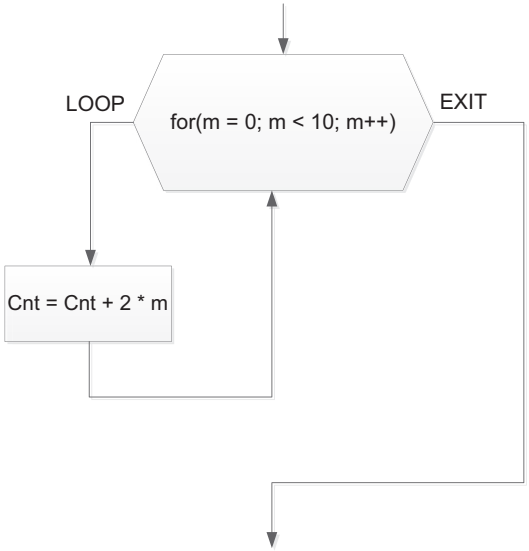

Figure 4.15: Method 2 for Representing a *for* Loop.

4.3.2 Method 2

Figure 4.15 shows the second method for representing the *for* loop with a flow chart. Here, a hexagon-shaped flow chart symbol is used to represent the *for* loop, and the complete *for* loop statement is written inside this symbol.

4.3.3 Method 3

Figure 4.16 shows the third method for representing the for loop with a flow chart. Here, again a hexagon-shaped flow chart symbol is used to represent the *for* loop, and the symbol is divided into three to represent the initial condition, the increment, and the terminating condition.

4.4 Summary

This chapter has described the program development process using the PDL and flow charts as tools. The PDL is commonly used as it is a simple and convenient method of describing the operation of a program. The PDL consists of several English-like keywords. Although the flow chart is also a useful tool, it can be very tedious in large programs to draw shapes and write text inside them.

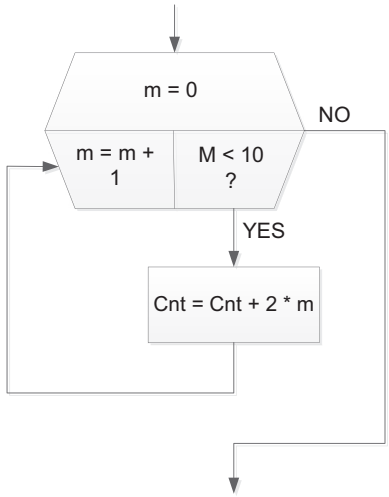

Figure 4.16: Method 3 for Representing a *for* Loop.

4.5 Exercises

1. Describe the various shapes used in drawing flow charts.
2. Describe how the various keywords used in PDL can be used to describe the operation of a program.
3. What are the advantages and disadvantages of flow charts?
4. It is required to write a program to calculate the sum of numbers from 1 to 10. Draw a flow chart to show the algorithm for this program.
5. Write the PDL statements for question (4) above.
6. It is required to write a program to calculate the roots of a quadratic equation, given the coefficients. Draw a flow chart to show the algorithm for this program.
7. Write the PDL statements for question (6) above.
8. Draw the equivalent flow chart for the following PDL statements:

```
DO WHILE count < 10
  Increment J
  Increment count
ENDDO
```

9. It is required to write a function to calculate the sum of numbers from 1 to 10. Draw a flow chart to show how the function subprogram and the main program can be implemented.
10. Write the PDL statements for question (9) above.

11. It is required to write a function to calculate the cube of a given integer number and then call this function from a main program. Draw a flow chart to show how the function subprogram and the main program can be implemented.

12. Write the PDL statements for question (8) above.

13. Draw the equivalent flow chart for the following PDL statements:

```
BEGIN
  J = 0
  K = 0
  REPEAT
    Flash LED A
    Increment J
    REPEAT
      Flash LED B
      Increment K
    UNTIL K = 10
  UNTIL J > 15
END
```

14. It is required to write a function to convert meters into inches and then call this function from a main program. Draw a flow chart to show how the function subprogram and the main program can be implemented.

15. Write the PDL statements for question (14) above.

16. Draw the equivalent flow chart for the following PDL statements:

```
BEGIN
  Configure I/O ports
  Turn OFF motor
  Turn OFF buzzer
  DO FOREVER
    IF button 1 is pressed THEN
      Turn ON motor
      IF button 2 is pressed THEN
        Turn ON buzzer
        Wait 3 s
        Turn OFF buzzer
      ENDIF
    ELSE
      Wait for 10 s
      Turn OFF motor
    ENDIF
  ENDDO
END
```