June, 2006

Wakefield Robotics Ltd.

Robotics 101 Online Education

Robotics 101 – An Introduction to PIC Microcontrollers

Chapter 3 – Preparing to Program a Microcontroller

3.1 Objectives

In this chapter, you will learn the steps required to create a program for your microcontroller, which will satisfactorily solve a specified problem.

3.2 Introduction to Program Preparation

If you are an experienced microcontroller programmer, you may be able to skip many of the formal steps that we will define in this chapter. However, unconsciously you will still be going through the same steps as defined here.

The first step is the problem definition. This is a paragraph defining the problem that the microcontroller and its program are supposed to overcome. For instance, a microcontroller may be required to turn a garden sprinkler on and off at specified intervals. The problem definition may be specified by the client, who is hiring you to program the microcontroller, or in many cases, the problem definition is defined by the engineer as he/she works through a design.

If the microcontroller program is fairly complicated, a flowchart may be required to provide a visual representation of the steps that the program must complete to successfully fulfill its purpose. Some programs may be simple enough that a flowchart is not required. However, we will start with representing all our programs with flowcharts, even the simple ones.

In *Chapter 2*, we introduced schematic circuit diagrams. At that time, we said that they were required so that we could correctly assemble our electronic circuits. They are also required so that we can correctly program the microcontroller. Schematic circuit diagrams show which microcontroller pins are connected to which of the inputs and outputs.

This next step is one that I recommend, to help the student to go from the flowchart to the microcontroller program. It requires that you generate an Input / Output Pin Table (IOPT). This table lists which pins are connected the input and output devices, and how they interact with those devices. The input/output pin table is normally generated from the schematic circuit diagram. If you are very familiar with interpreting schematic circuit diagrams, you may be able to skip creating a input / output pin table.

The final step is to generate the high-level program code. In other words, program code written in a language which is easy for us to understand. Usually, the programmer will

follow the steps defined in the flowchart, and with help from the input/output pin table, create the necessary lines of code to satisfy each step.

Let's have a look at each of the program development step in more detail, and create a representative program.

One quick note at this time. There are many ways to solve a problem. So, your program ***will*** *be different from other students. Your program will be based on your experiences, both in programming and from other aspects of life. As you practice programming, your programs will become smaller and more efficient.*

3.3 Problem Definition

Before you begin the microcontroller program writing process, you must have a definition of the problem that the program and microcontroller will solve. I suggest that for a simple problem, a single paragraph be prepared, which describes the overall objective of the microcontroller and its program. However, several paragraphs may be required if the program is very complicated.

Here is a sample problem definition:

Computer Cooling System

You must design a computer cooling system. When the computer is first turned on, the fan is off. As the computer temperature rises, the lower temperature switch (~ 35 °C) closes, but the fan does not run. When the computer temperature reaches 50 °C, the high temperature switch closes and the fan comes on. The fan will now stay on as long the lower temperature switch is closed.

You will notice that the problem definition gives all the information required to solve the problem.

3.4 Flowcharts

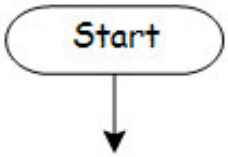
A flowchart provides a visual representation of the steps that the microcontroller program must take to successfully complete its objectives.

The first step is to review the problem definition and break it down into small steps or tasks. This may require that you break the problem down into smaller portions and layout the steps required to solve each portion. Then reassemble the portions into a larger program.

Each task is represented by a block. The blocks are connected with flow lines. The flow lines are lines with arrowheads which indicate the order in which the blocks or tasks are processed.

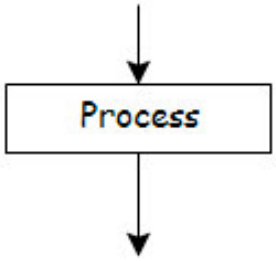
Each flowchart begins with a **Start** block as seen in *Figure 3.1*. Here you can see the start block and the arrowhead on the flow line indicates the next block to be performed.

*Figure 3.1 The **Start** Flowchart Block*



The next block you will probably encounter when programming a microcontroller is a **Process** block. This block is shown in *Figure 3.2*. This block indicates that the microcontroller will perform a known function which may be a single task (such as turn on the light) or a group of tasks (such as flash the light at a regular interval).

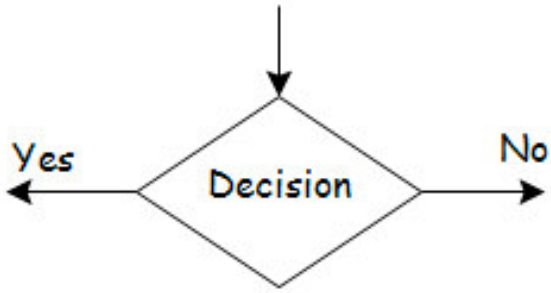
*Figure 3.2 The **Process** Flowchart Block*



With microcontrollers, one of the first steps you must program is to initialize the registers and variables. In other words, the microcontroller hardware (registers) and RAM (variables) must be correctly set up.

Since microcontrollers typically react to events that occur in the real world, it must be able to make decisions. For example, if a microcontroller's job is to turn on the lights when it gets dark out, it must check the ambient light intensity and then decide if it is dark enough to turn on the lights. The *Decision* flowchart block is shown in *Figure 3.3*.

*Figure 3.3 The **Decision** Flowchart Block*



Following our previous example, the decision could be labeled *Is It Dark Enough?*. Then if the answer is *Yes* - we follow the left path and turn on the lights, or else, if it is *No* - we follow the right path and ensure the lights are off.

If the microcontroller interacts with the real world, it must read in from the input pins and output to the output pins. The input / output flowchart block, as shown in *Figure 3.4*, indicates this type of tasks. For example, preceding the earlier decision block there would have to be an input block to read the level of the ambient light and after the same decision block, there would have to be two output blocks, one to turn the light on and another to turn the light off. See *Figure 3.5* for a flowchart describing this example.

*Figure 3.4 The **Input** / **Output** Flowchart Block*

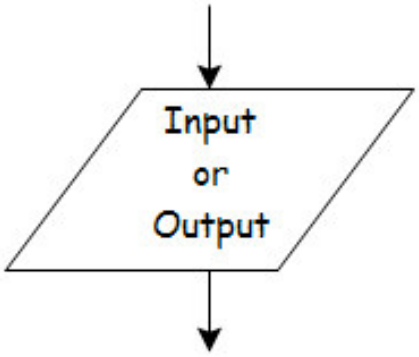
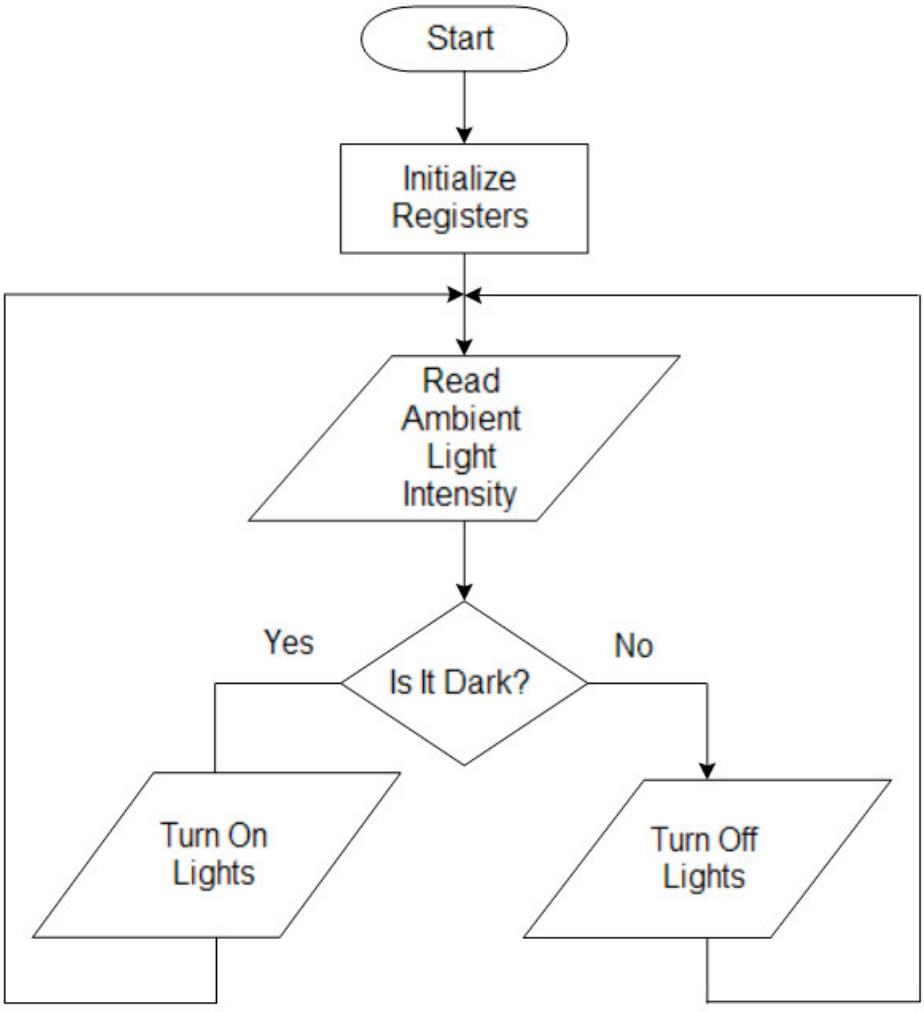


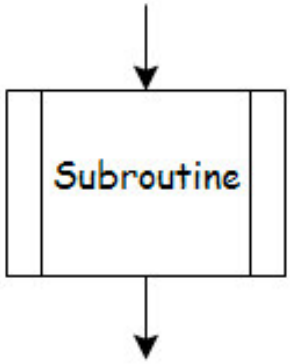
Figure 3.5 Flowchart for the Automatic Night Lights Example



Note that the flowchart shown in *Figure 3.5* does not have an end. It just keeps looping forever. You may have noticed that the lights may be turned on (or off), even though they are already on (or off). This should not cause any problems with the operation of the program as you will see in a following chapter.

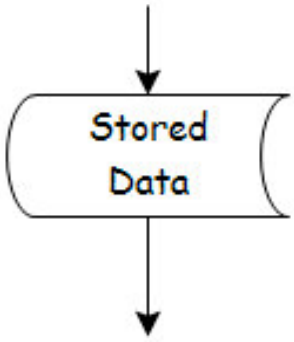
Later, we will be also introducing subroutines. Subroutines are small programs which can be called from anywhere within the main program. This enables program code to be shared, so that the same code does not have to be repeated throughout the program. This helps to keep the program size small, so it will fit in the available Flash memory. The sequence of tasks in the subroutine, are usually indicated by its own flowchart.

*Figure 3.6 The **Subroutine** Flowchart Block*



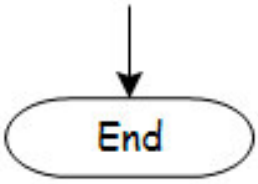
In *Chapter 11* of this book, we will be discussing how to store and read data from the microcontroller's EEPROM. Since this is non-volatile memory, whatever is stored in the EEPROM can be retrieved later, even if the power has been removed from the microcontroller. The *Stored Data* flowchart block is shown in *Figure 3.7*.

*Figure 3.7 The **Stored Data** Flowchart Block*



Although the program example shown in *Figure 3.5* does not have an end, some programs do. If the program has an end, it is indicated with an *End* flowchart block, as shown in *Figure 3.8*. A program with an end, would come to a stop and no longer run any lines of code, if the correct conditions occur. For instance, consider a microcontroller controlling a fireworks exhibit. After the show has completed, the equipment is placed in a safe mode and the microcontroller shuts down. The matching flowchart would finish with an End block.,

*Figure 3.8 The **End** Flowchart Block*



3.4 Schematic Diagrams

In this course, we will be using schematic symbols to create circuit diagrams that represent components in our electronic circuits. They may not represent the actual look of the devices, but they provide us with enough information to physically create the circuits.

For more information on schematic circuit diagrams and schematic symbols, see *Chapter 2* of this course.

3.5 Input / Output Pin Tables (IOPTs)

I recommend that the student use Input / Output Pin Tables, in conjunction with the program flowchart, to create the program code for the microcontroller. People very familiar with electronics, may be able to skip the creation of the IOPT, and create the program code from the schematic circuit diagram and the program flowchart.

The purpose of the IOPT is to translate the schematic circuit diagram into a form that is easier to convert into program code.

The IOPT performs the following tasks:

1. Lists the I/O pins on the microcontroller that will be used and whether they are inputs or outputs. This helps us correctly set up the registers – especially the TRIS registers. *Remember: unused pins should be set up as outputs.*
2. Briefly describes the purpose of each I/O pin.
3. Describes the operation of each pin. For instance, high voltage turns the LED on and low voltage turns the LED off.

Table 3.1 Sample Input / Output Pin Table

Pins	Input or Output	Use	Mode of Operation
RA0	Input	Low Temperature Switch	High Voltage => Temperature > 35 °C Low Voltage => Temperature < 35 °C
RA1	Input	Higher Temperature Switch	High Voltage => Temperature > 50 °C Low Voltage => Temperature < 50 °C
RA2	Output	Fan Motor	High Voltage => Fan On High Voltage => Fan Off

Let's have a quick look at the IOPT shown in *Table 3.1*. Only 3 pins in Port A are being used. The remaining pins can all be set as outputs. For example:

```
TRISA = %00000011        ' Pins RA0 and RA1 set as inputs, the rest as outputs
```

If the register bit containing Pin 0 of Port A is a one, then the input voltage is high and the temperature is greater than 35 °C. Otherwise, if the register bit for this pin is a zero, the temperature is less than 35 °C.

Having this information provided to you, simplifies the programming process.

3.6 Writing Program code

Now for the tough part, let's start writing program code. I will work through an example to show you how it is done. We will then spend the remainder of this course learning how to program certain tasks.

Figure 3.9 Programming Flowchart

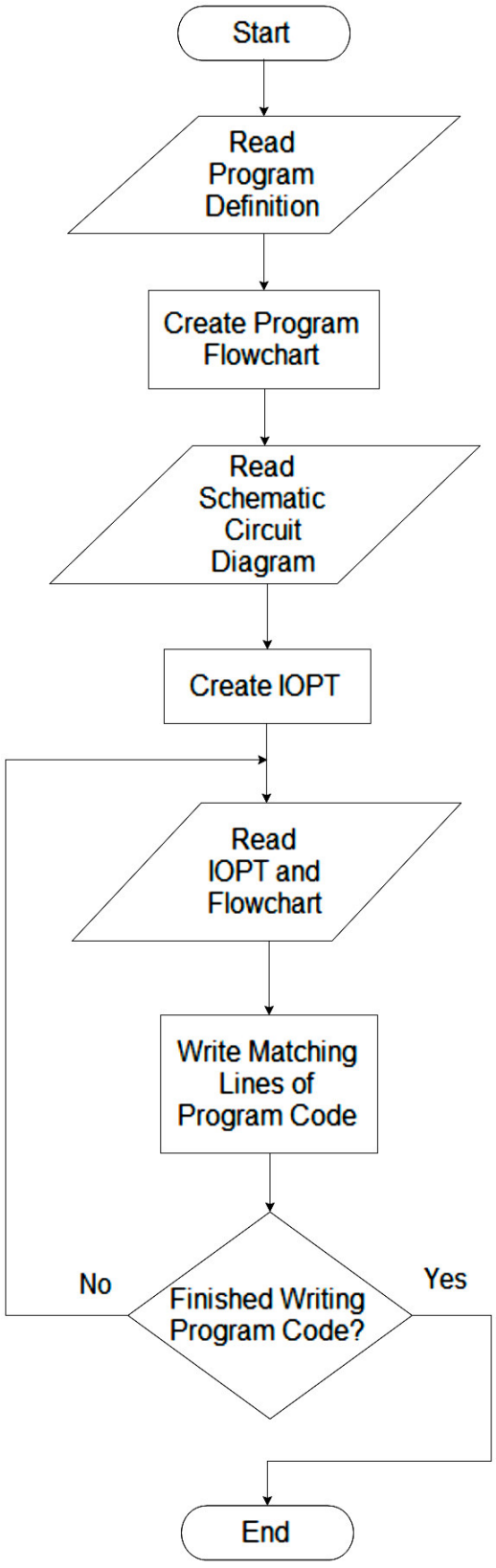


Figure 3.9 shows the flowchart which represents the creation of the program code for a microcontroller. Let's follow the steps shown in the flowchart.

As discussed earlier, we begin with the Start block. This block really doesn't tell us anything, except where to begin reading the flowchart.

Before we begin the programming process, we have to have our Program Definition prepared. When we read it, we will familiarize ourselves with the problem to be solved.

Based on the Program Definition, we will prepare a flowchart representing the tasks required to complete a solution. This may require that you reread the Program Definition several times, to ensure that you correctly interpret it and that your flowchart represents a true solution to the problem.

Now, using the schematic circuit diagram, prepare an Input / Output Pin Table as shown in *Section 3.5*. If you have trouble understanding the electronics, it would be a good idea to get assistance. Errors in the IOPT will result in errors in your final microcontroller program.

With the flowchart and IOPT readily available, it is time to start writing the program code. Read the task from the flowchart, and using the IOPT as a reference, translate the task into the appropriate programming language. When the task has been converted into microcontroller commands, look up the next task in the flowchart and repeat the process. When you finish the last step in the flowchart, finish the program code with "end.". This indicates the end of the program code.

The process of writing microcontroller programming code sounds very complicated, but with practice, it will become second nature.

3.7 Examples

Let's work through an example problem. Here is a sample problem definition that we looked at in *Section 3.3*.

Computer Cooling System

You must design a computer cooling system. When the computer is first turned on, the fan is off. As the computer temperature rises, the lower temperature switch (~ 35 °C) closes, but the fan does not run. When the computer temperature reaches 50 °C, the high temperature switch closes and the fan comes on. The fan will now stay on as long the lower temperature switch is closed.

Our next step is to create a flowchart to describe our solution to this problem. Remember, there are an infinite number of solutions to any problem. Do not be upset if your solution does not match that of other students or mine.

The flowchart representing my solution is shown in *Figure 3.10*. The next step is to create an IOPT. However, we can't do that without the use of a schematic circuit diagram. The schematic circuit diagram is shown in *Figure 3.11*.

Figure 3.10 Flowchart for Computer Cooling System

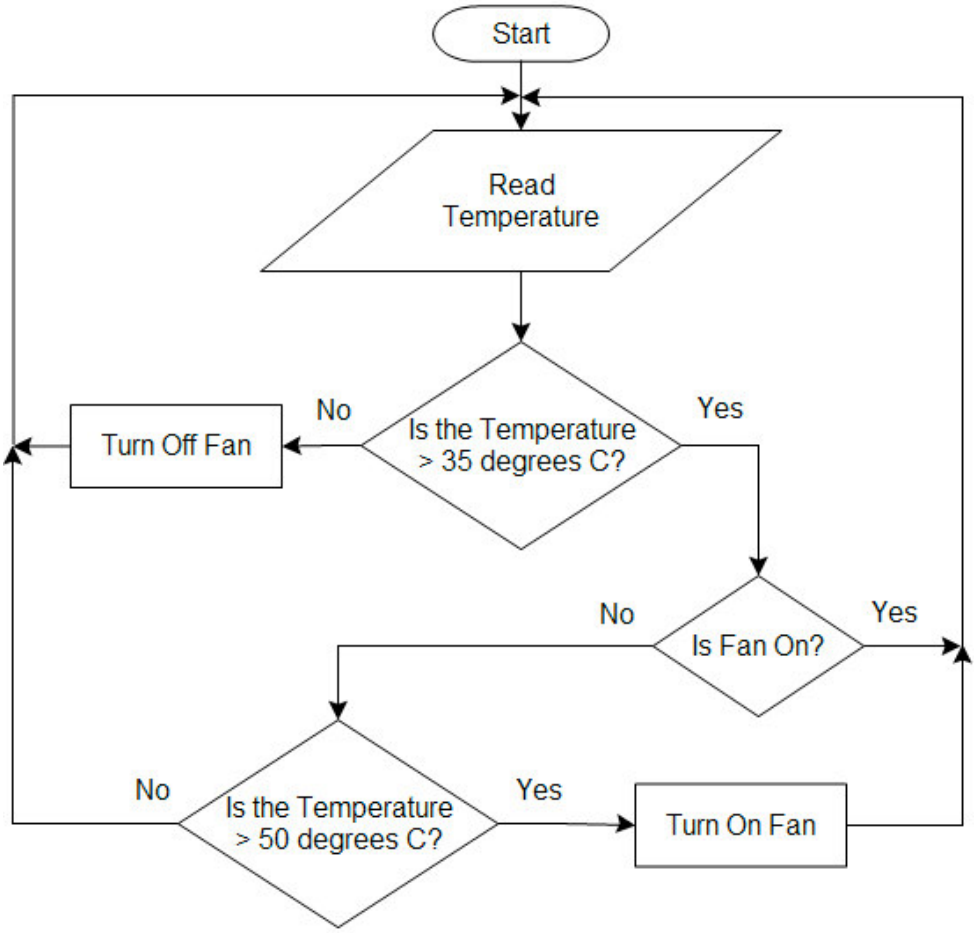
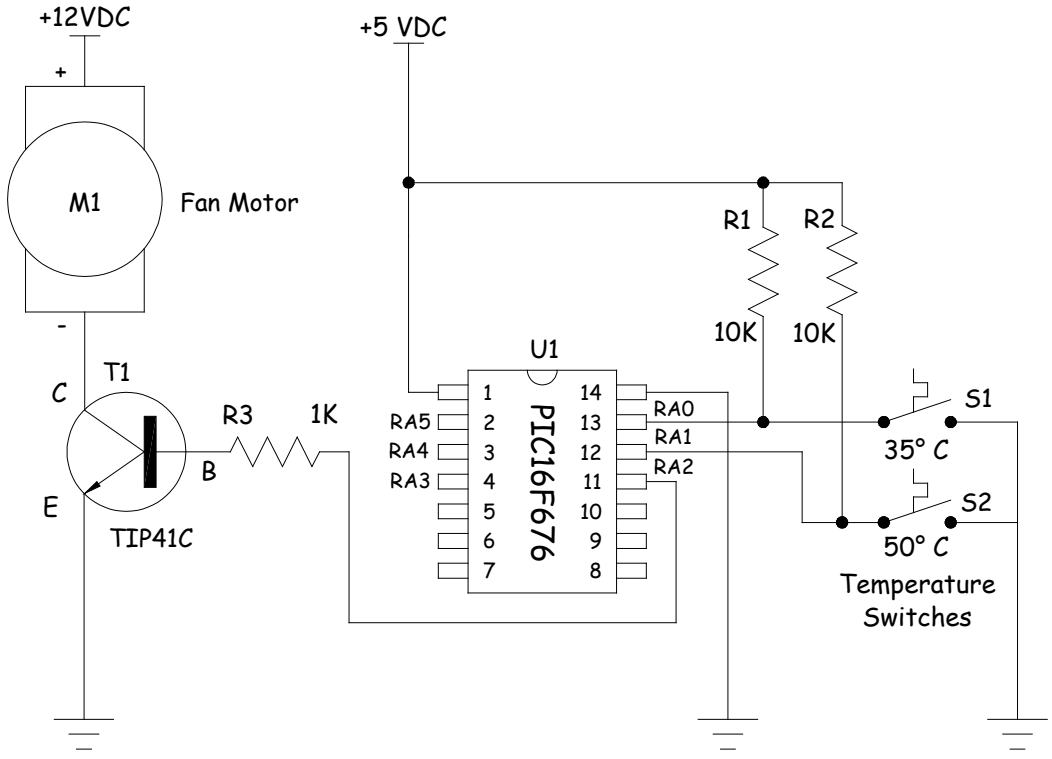


Figure 3.11 Schematic Circuit Diagram for Computer Cooling System



Notice the interesting schematic symbol for a thermal or temperature switch. The switches are normally open (NO) and when the temperature reaches a preset value, the contacts close.

Now let's construct the IOPT for this problem. From the circuit schematic, we can see that only input / output pins RA0, RA1 and RA2 are used. So only these pins are listed in the first column of the IOPT.

Pins RA0 and RA1 are inputs, as they read switches, and RA2 is an output, as it controls the transistor. This information is stored in the second column.

RA0 connects to the lower temperature switch and RA1 connects to the higher temperature switch. RA2 connects to the fan motor transistor. This information goes in the third column.

Finally, the fourth column contains the information about what the microcontroller sees when events occur in the real world. For instance, when the computer system temperature is greater than 35 °C then the thermal switch closes and the voltage to RA0 is low. Otherwise, the voltage to RA0 is high.

Table 3.2 Input / Output Pin Table for Computer Cooling System

Pins	Input or Output	Use	Mode of Operation
RA0	Input	Low Temperature Switch	High Voltage => Temperature > 35 °C Low Voltage => Temperature < 35 °C
RA1	Input	Higher Temperature Switch	High Voltage => Temperature > 50 °C Low Voltage => Temperature < 50 °C
RA2	Output	Fan Motor	High Voltage => Fan On High Voltage => Fan Off

Now it is time to start writing the microcontroller program. The information stored in the flowchart defines the steps that the program must perform, and the information in the IOPT refines those program steps to match the microcontroller and the electronic circuit.

The completed program listing for this problem is shown in *Program Listing 3.1*. Let's briefly consider the steps required to create this program.

At the top of program listing is the program file name ComputerCooling, which means that this file is stored on your computer hard drive as ComputerCooling.pbas . Next I list some important facts about the program, such as the purpose of the program, the I/O pins used, and perhaps the date, microcontroller and author. Notice that all comments begin with an apostrophe to tell the compiler to ignore these lines. They are only useful to the people reading the program code.

The program code itself begins at the line label main:. First, we initialize the microcontroller hardware by setting the registers. Note the comments after each command while initializing the registers. This helps us to understand the purpose of each command.

Program Listing 3.1

```
Program ComputerCooling
‘ Computer Cooling System
‘ Fan comes on when computer temperature reaches 50 ℃. and stays on until
‘ computer temperature drops below 35 ℃.
‘ PORTA.0 - Lower Temperature Switch: High if < 35 ℃
‘ PORTA.1 - Upper Temperature Switch: High if < 50 ℃
‘ PORTA.2 - Fan: High when Fan Is On

‘ Main Program

main:
        ‘ Initialize Registers
        CMCON = 7               ‘ Disable comparators
        ANSEL = 0               ‘ Disable ADCs
        TRISA = %00000011       ‘ Set RA0 & RA1 as inputs, rest as outputs
        TRISC = 0               ‘ Set Port C as outputs
        PORTA = 0               ‘ Set Port A register low
        PORTC = 0               ‘ Set Port C register low

        MainLoop:
                if PORTA.0 = 0 then             ‘ If Temp > 35 ℃ then go to HighTemp loop
                        goto HighTemp
                else                            ‘ Else turn off fan
                        PORTA.2 = 0
                end if
        goto MainLoop

        HighTemp:
                if PORTA.2 = 1 then             ‘ If Temp < 50 ℃ then go to MainLoop
                        goto MainLoop
                end if
                if PORTA.1 = 0 then             ‘ If Temp > 50 ℃ then turn on fan
                        PORTA.2 = 1
                end if
        goto MainLoop

end.
```

Although we show a block in the flowchart to read the input temperature, we find that the input pins to the microcontroller are automatically updated and hence do not require a separate command. Therefore, to check if the temperature is greater than 35 degrees Celsius, we simply check if the register bit connected to RA0 has a zero in it. If it is a one, then the voltage is high, the switch is open, and the temperature is less than 35 °C. Similarly, if it is zero, then the voltage is low, the switch is closed, and the temperature is greater than 35 °C.

See if you can follow the rest of the program logic. The nice thing about the Basic programming language is that it has English-like commands.

This finishes the third chapter of this book. In Chapter Four, we will look at different types of programming languages and how it is converted from the form that we create to something that the microcontroller can run.