

Air Flow Control Using Fuzzy Logic

*Author: Robert Schreiber
Microchip Technology Inc.*

INTRODUCTION

Fuzzy logic control can be used to implement a wide variety of intelligent functions including everything from consumer electronic goods and household appliances to auto electronics, process control, and automation.

Typically, fuzzy logic control applications fall into two categories. First, it can be used to enhance existing products with intelligent functions. Second, it can utilize sensors that continuously respond to changing input

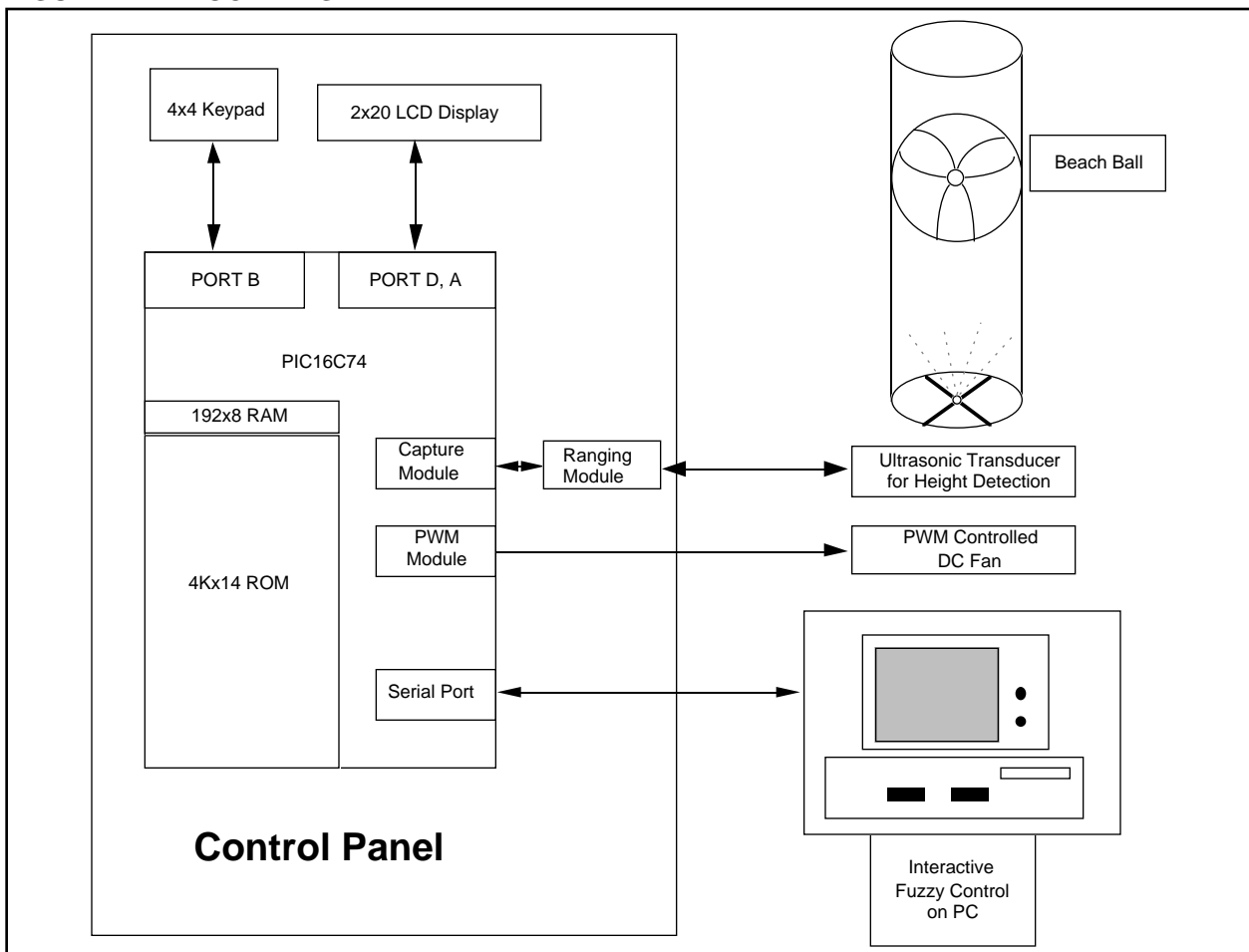
conditions. In addition, fuzzy logic simplifies dealing with non-linearities in systems, and allows for quicker product development cycles.

This application note will step the user through a fuzzy logic control design utilizing sensors. The development tool used is Inform[®] Software's *fuzzyTECH[®]-MP*. The development tool allows for an all-graphical editor, analyzers, and debug capability.

PROJECT DESCRIPTION

The block diagram of the project is shown in Figure 1 and operates as follows.

FIGURE 1: BLOCK DIAGRAM



The control panel prompts the user to enter the desired beach ball height on the 16-key keypad. The keypad input is echoed on the LCD module and the user is prompted for confirmation. Upon confirmation of user input, the control panel initiates a ranging cycle to calculate the current height of the beach ball. The desired height and current height are continually displayed on the LCD module. From the current height, the control panel calculates both the velocity and the delta height (difference in desired height from current height). This information, along with the desired height, is transmitted to the PC via an RS-232 link. The fuzzy logic algorithm, running on the PC, calculates the appropriate duty cycle of the DC fan and transmits this information to the control panel. This emulates a “real world” environment in which system level debug can be done on the PC in real-time. The control panel controls the duty cycle of the DC fan with this input. The above listed ranging process continues indefinitely until interrupted by the user.

The control panel houses an ultrasonic ranging module and the microcontroller. The microcontroller handles all of the peripheral interfaces including the 16-key keypad, the LCD display, the ultrasonic ranging module, and the RS-232 serial link. The project required a microcontroller that could handle the data throughput and all of these peripherals with little or no external components. The microcontroller used was the PIC16C74, which contains 4K of on-chip program memory and 192 bytes of on-chip data memory. Furthermore, the interrupt capabilities, I/O pins, PWM module, capture and compare modules, timer modules, Universal Asynchronous Receiver Transmitter (USART), and A/D converter make it an excellent fit for the application. In addition, the on-chip Pulse Width Modulation (PWM) module allows for a single component (FET) interface for the DC fan control and the ranging module can interface directly to the microcontroller (refer to Application Note AN597, “Implementing Ultrasonic Ranging”).

FUZZY DESIGN

Fuzzy logic first translates the crisp inputs from the sensor into a linguistic description. Then it evaluates the control strategy contained in fuzzy logic rules and translates the result back into a crisp value.

The first step in fuzzy logic control design is system definition. The only possible sources of inputs to the fuzzy logic control algorithm are the ultrasonic transducer, the user, and the DC fan. The key is to decide which of these inputs are significant and which are not. Basically, the behavior of the beach ball was characterized by asking the following questions from the beach ball's perspective:

- Where am I?
- How far am I from where I want to be?
- How fast am I getting there?
- What external force will get me there?

The nice thing about fuzzy logic control is that the linguistic system definition becomes the control algorithm.

The variables were defined as follows:

- **Current Height** [Where am I?]
- **Delta Height** [How far am I from where I want to be?]
- **Velocity** [How fast am I getting there?]
- **Duty Cycle** [What external force will get me there?]

Defining the variables was the starting point, but for the algorithm to work smoothly, it isn't good enough to say “the beach ball has velocity;” you need to know to what degree the beach ball has velocity. This is accomplished by defining terms that more fully describe the variable. The combination of variables and terms gives a linguistic description of what is happening to the system. From this, the Velocity variable can be described as having a “positive small velocity” or a “positive big velocity,” not just a “velocity.”

There is no fixed rule on how many terms to define per variable. Typically, three to five terms are defined, but more or less may be needed based on the control algorithm. In retrospect, we probably could have reduced Current Height to three terms and Velocity to five terms. Table 1 lists the four variables that are used for the trade show demo and their associated terms.

Once the linguistic variables are defined, data types and values need to be defined. For this application, data types were defined as 8-bit integers (16-bit definition is also possible). After defining the data types, the shell and code values for each variable were specified. A shell value is used within the fuzzy logic development tool and a code value is used when the code is generated.

The best way to describe shell and code values is using the analogy of a D/A converter. If we have a 5.0V, 8-bit D/A converter, the digital input would correspond to the code value and the analog output would correspond to the shell value. This is, if we write (or pass) a value of 128 to the D/A we would get a 2.51V out. Applying this analogy to our project, we would pass a crisp value (digital) to the fuzzy world and the fuzzy world would use the fuzzy value (analog).

Therefore, when we define shell and code values, we are basically defining the "D/A converter." For example, you can define the shell value for Duty Cycle to be a minimum of 0 and a maximum of 100 (percent). Therefore, within the fuzzy logic development tool, Duty Cycle will take on a value between 0 and 100, inclusive.

The code value is limited by the data type, but can take on any or all of the digital range. That is, if the shell value is 0 to 100, the code values could be defined as 0 to 100. But to get full resolution, the code value should be defined over the entire range (i.e., 0 to 255 for 8-bit data types). The code values and shell values were defined as shown in Table 2. Note that for the height and velocity variables, the shell values are scaled by 2 (i.e., a Current Height with a crisp value of 60 would correspond to 30 inches).

TABLE 1: INPUT AND OUTPUT VARIABLES AND TERMS

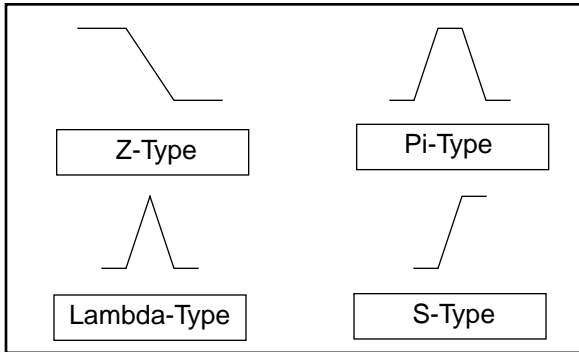
Input Variables			Output Variable
Current Height	Delta Height	Velocity	Duty Cycle
very lo	neg big	neg big	very slo
lo	neg small	neg med	slo
medium	zero	neg small	medium slo
hi	pos small	zero	medium
very hi	pos big	pos small	medium fast
		pos med	fast
		pos big	very fast

TABLE 2: SHELL AND CODE VALUES

Variable	Shell Value		Code Value	
	Min.	Max.	Min.	Max.
Current Height	0	120	0	255
Delta Height	-50	50	0	255
Velocity	-5	5	0	255
Duty Cycle	0	255	0	255

Next, the membership functions were defined to further describe the variables. The fuzzy logic development tool creates the membership functions automatically. This gives a good starting point, but the membership functions still need to be fine-tuned during the debug phase. In this application, only the linear shaped functions (Pi, Z, S and Lambda types) were used as seen in Figure 2.

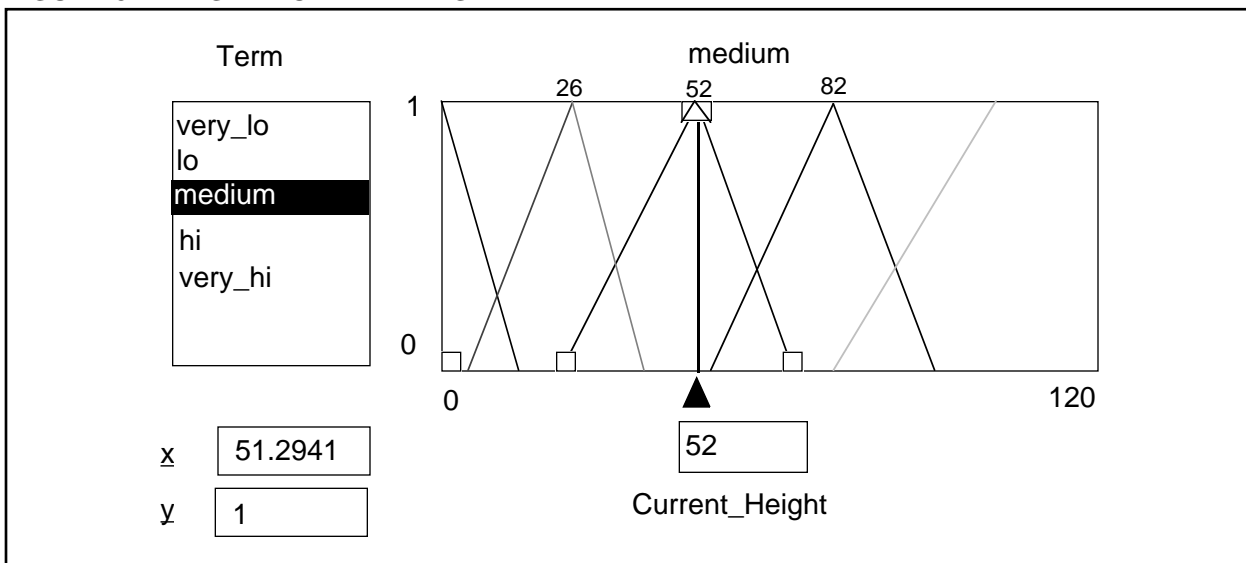
FIGURE 2: STANDARD MEMBERSHIP FUNCTION TYPES



FUZZIFICATION

Fuzzification entails translating a crisp value into a fuzzy value. Once all of the variables have been defined, the interfaces between the variables need to be defined. The interfaces for the input variables contain the fuzzification procedures. In defining the interfaces, the input variable's fuzzification method needs to be defined. The computation of fuzzification is carried out at runtime for code efficiency. The type of fuzzification used in this project is membership function computation. This is largely due to the code space efficiency and accuracy associated with this method. Once fuzzification has taken place, the algorithm is performed in the fuzzy world according to the rule base.

FIGURE 3: DEGREE OF MEMBERSHIP



FUZZY RULE BASE

The entire fuzzy inference is contained within the rule blocks of a system. For example, if the beach ball is near the top of the tube and it was commanded to be near the bottom of the tube, the rule that described the situation would be:

IF CURRENT HEIGHT = VERY HI

AND DELTA HEIGHT = NEGATIVE BIG

THEN DUTY CYCLE = SLOW

The above rule describes one situation, but the rule definition would continue until the system was adequately described. The rule block is the collection of all rules that describe the system.

The rules of the rule block can also be defined in terms of how much a specific rule is supported when calculating inference. The support of a rule, or plausibility, is known as the degree of support for that rule. A plausible rule is defined by a 1.0, a totally implausible rule is defined by 0.0. In this project all rules are fully supported.

The degree to which a crisp value belongs to a term is known as the degree of membership. For example, the terms Medium and Hi for the variable Current Height were defined as a Lambda-type membership function centered around the crisp values 52 (26 inches) and 82 (41 inches), respectively, as shown in Figure 3.

Therefore, if the beach ball was at 26 inches, the degree of membership would be 1.0 for Medium and 0.0 for Hi. However, as the beach ball rises in height, the degree of membership for the term Medium would decrease and the degree of membership for the term Hi would increase. The interplay of these linguistic variable terms is controlled by the rule base. The rule base defines not only the relationship between the terms, but also how much each rule is supported, as described previously.

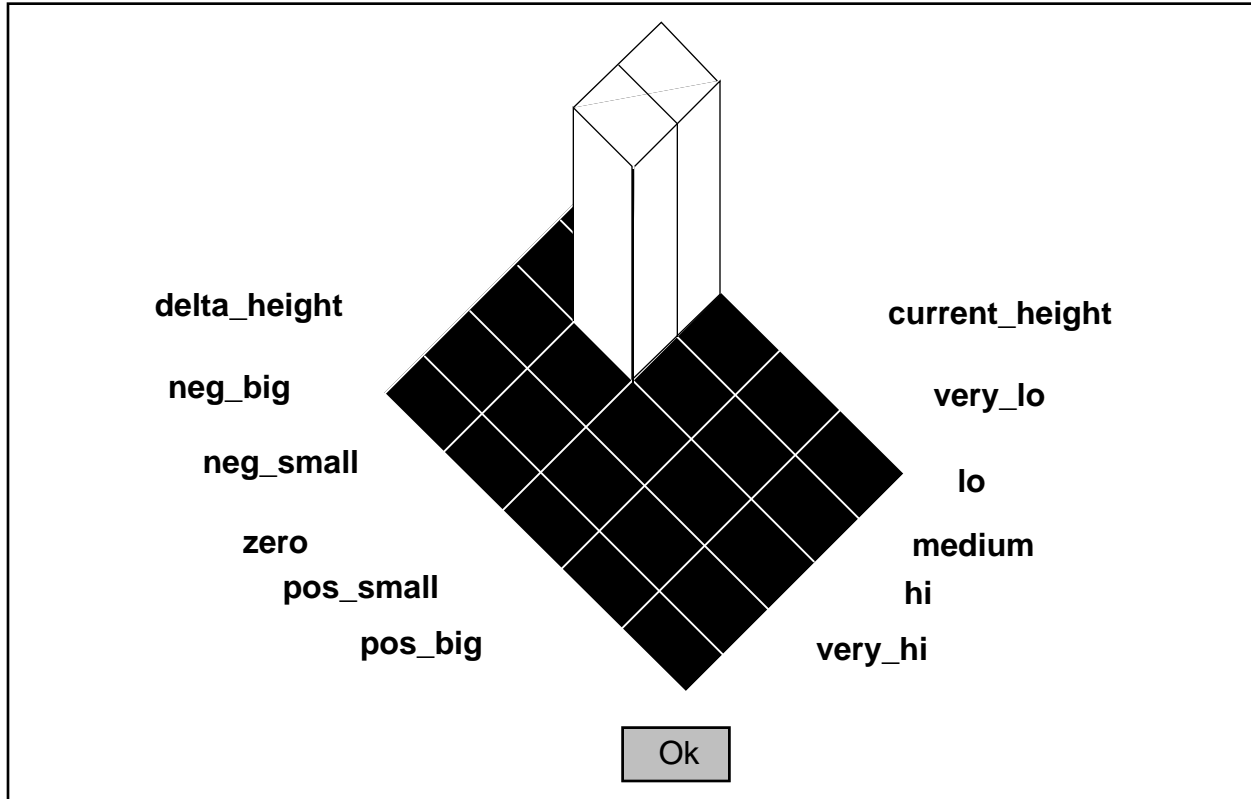
From the list of rules, a Fuzzy Associative Map (FAM) is constructed (see below). The FAM shows the plausibility (degree of support) of each rule as seen in Figure 4 and Figure 5.

FIGURE 4: MATRIX RULE EDITOR WITH FAM RULES

The screenshot displays the Matrix Rule Editor interface. At the top left, a grid is defined with 'delta_height' and 'current_height' as columns and 'neg_big', 'neg_small', 'zero', 'pos_small', and 'pos_big' as rows. A small icon is visible in the top-left cell of the grid. To the right of the grid is a 'Show ...' menu with three radio button options: 'Degree of Support' (selected), 'Input Aggregation', and 'Composition with Degree of Support'. Below the menu is a 'Degree of Support' slider control, currently set to 0.000. The bottom section is divided into 'IF' and 'THEN' parts. The 'IF' part contains three vertical lists of linguistic terms: 'current_height' (very_lo, lo, medium, hi, very_hi), 'delta_height' (neg_big, neg_small, zero, pos_small, pos_big), and 'velocity' (neg_big, neg_med, neg_small, zero, pos_small). Each list has a vertical scrollbar and a small icon. The 'THEN' part contains a vertical list of linguistic terms: 'duty_cycle' (very_slow, slow, med_slow, medium, med_fast), also with a vertical scrollbar and a small icon.

AN600

FIGURE 5: 3-D RULE DISPLAY



DEFUZZIFICATION

Defuzzification entails translating a fuzzy value to a crisp value. The interface for the output variables contains the defuzzification procedures. For most control applications (and this project), the center-of-maximum (CoM) method is used for defuzzification. CoM evaluates more than one output term as valid and compromises between them by computing a weighted mean of the term membership maxima. Example 1 and Figure 6 show the defuzzification of the linguistic variable Duty Cycle using CoM.

EXAMPLE 1: DEFUZZIFICATION OF DUTY CYCLE

The crisp values of the three input variables are as follows:

Current Height: 30

Delta Height: 0

Velocity: 0

The crisp value can be calculated using the CoM method with the following equation.

$$C = \frac{\sum_i [I \cdot \max_x (M) \cdot \arg (\max_x (M))]}{\sum_i I}$$

c = crisp output value

i = linguistic term

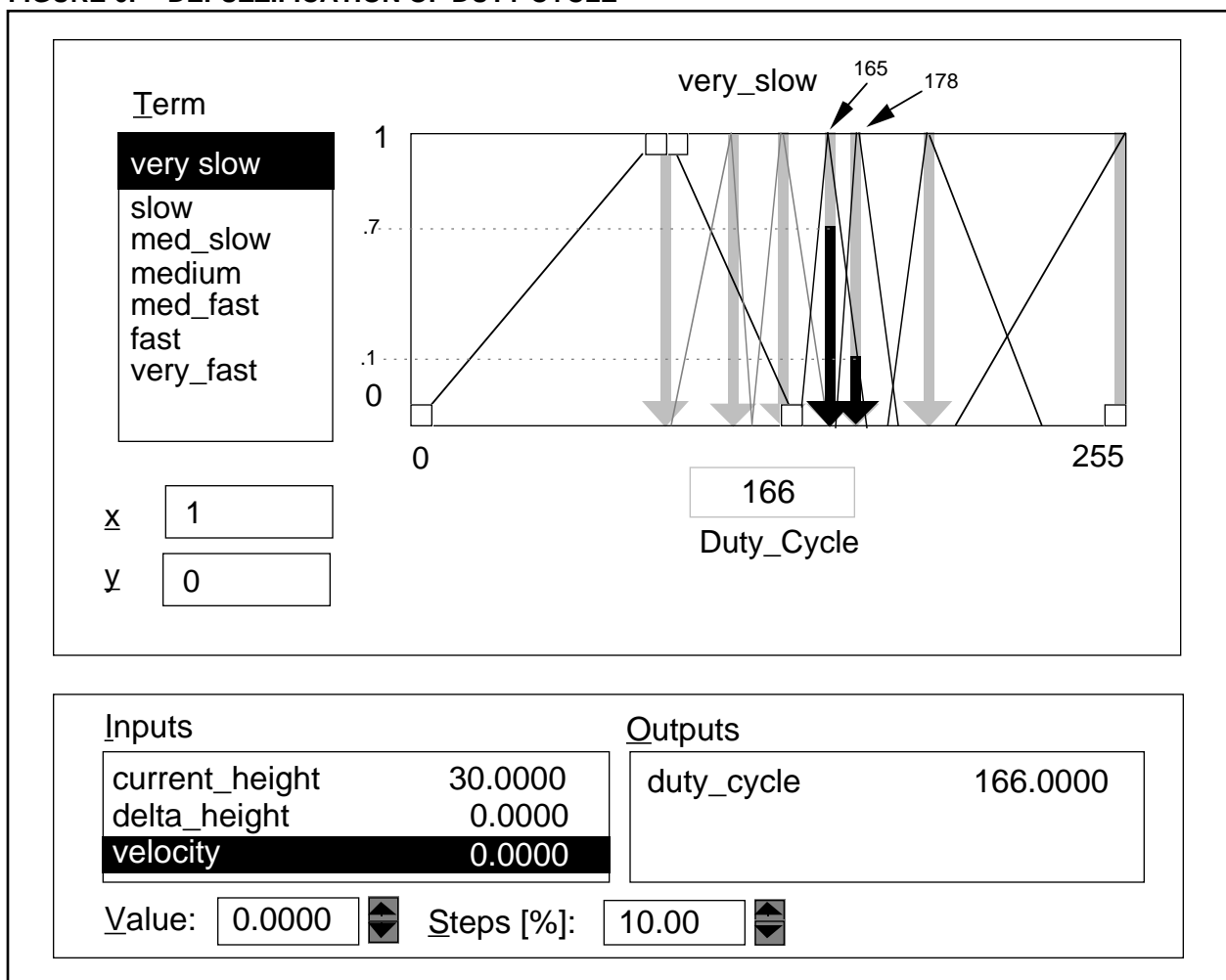
I = inference result

M = membership function of linguistic term

For this example, when the crisp values are fuzzified, the Duty Cycle variable is defined to be mostly "medium" (degree of membership of 0.7) and somewhat "medium fast" (degree of membership 0.1). The arguments for the "medium" and "medium fast" term membership maxima are 165 and 178, respectively.

$$\frac{((0.7 \cdot 1.0 \cdot 165) + (0.1 \cdot 1.0 \cdot 178))}{(0.7 + 0.1)} = 166$$

FIGURE 6: DEFUZZIFICATION OF DUTY CYCLE



DEBUGGING

In serial debug mode, one can graphically adjust the variable terms and see the results in "real time." On this project, the first variable adjusted was the Duty Cycle variable. Duty Cycle was adjusted so that the beach ball reached 30 inches (Figure 7). The Delta Height terms were fine-tuned -- negative small, zero, and positive small were bunched together -- and the beach ball stabilized at 30 inches (Figure 8). There was virtually no fluctuation in the height. In order for the system to self-correct for environmental (external) changes, the Velocity variable was used. The velocity variable is calculated by the difference in height between consecutive height calculations. A few rules were added that used the Velocity variable to nudge the ball into place when the environmental conditions changed (Figure 9).

Another advantage of fuzzy logic is that it simplifies dealing with non-linearities of the system. The system was highly non-linear, so it was tested at the extremes and moving the beach ball at different rates from one extreme to the other. The Current Height variable needed almost no adjustment (Figure 10). The variable that required the most work was the Duty Cycle variable, but in less than a day, the algorithm was working well within specifications. The beach ball could go from a resting position, with the DC fan off, to the maximum allowable height of 42 inches in less than 8 seconds with no overshoot. Operation between the minimum and maximum height was much quicker, also with no overshoot.

The final graphical representation of the linguistic variables are shown in Figure 7 through Figure 10.

FIGURE 7: DUTY CYCLE VARIABLE

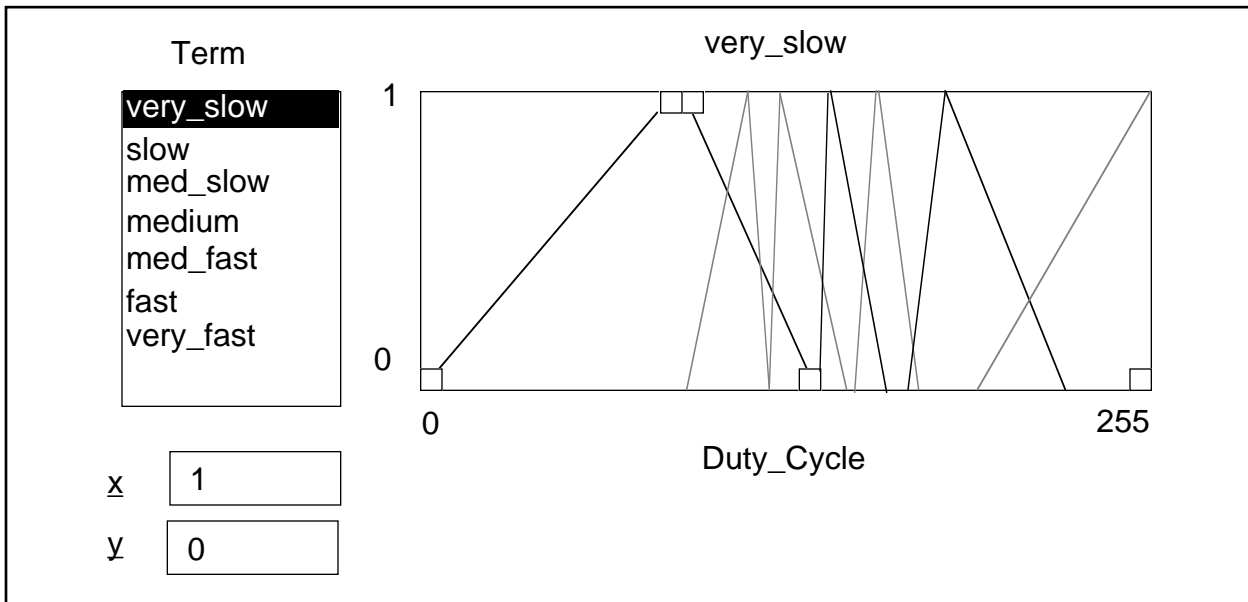


FIGURE 8: DELTA HEIGHT VARIABLE

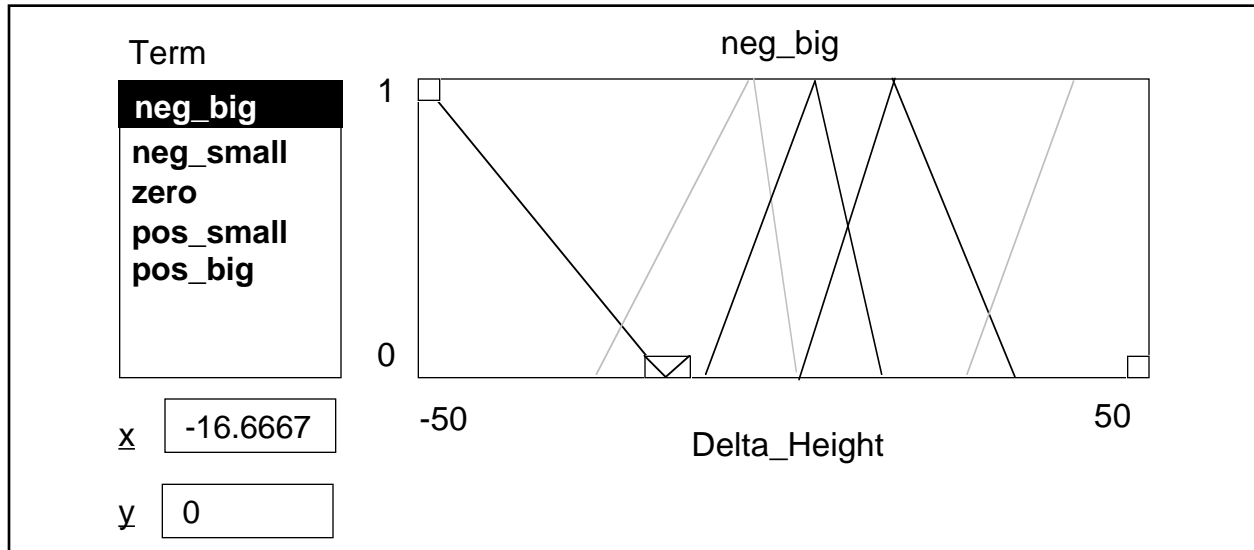


FIGURE 9: VELOCITY VARIABLE

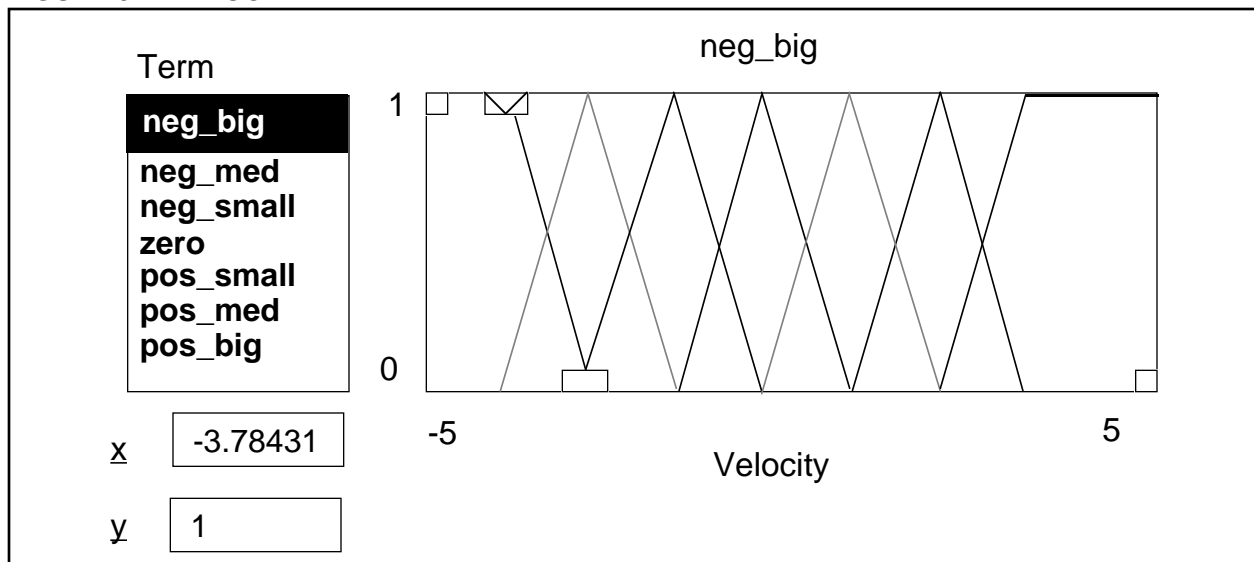
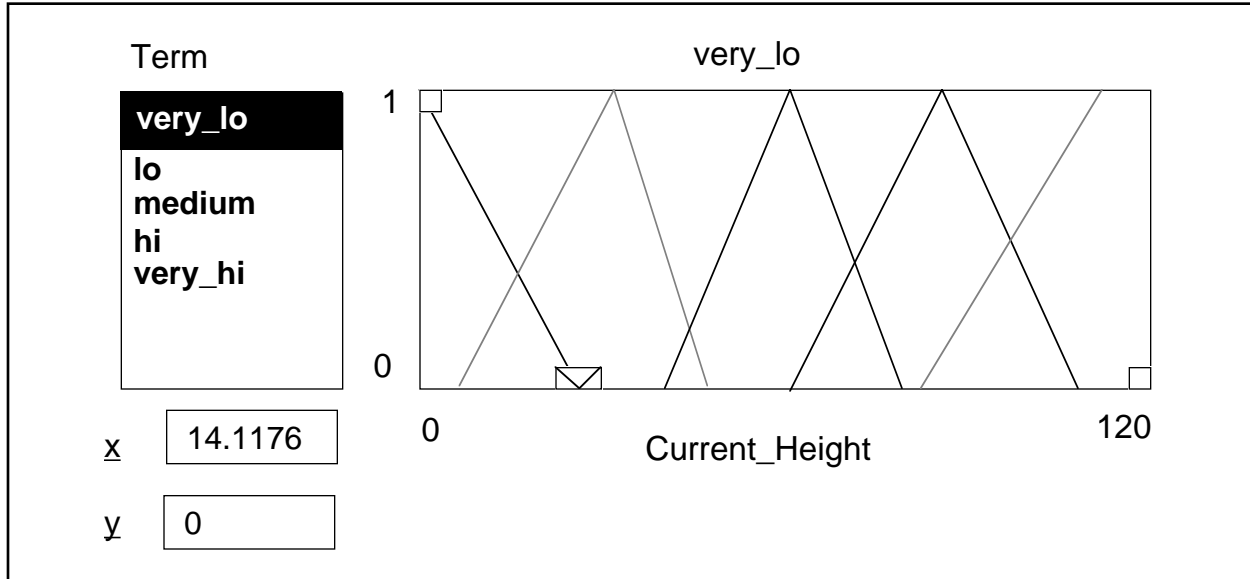


FIGURE 10: CURRENT HEIGHT VARIABLE



INTEGRATION

The system parameters and graphical variable representations are captured in a Fuzzy Technology Language (FTL) file. The FTL file is a vendor and hardware independent language which defines the fuzzy logic based system. The FTL file for this project can be seen in Appendix A.

The FTL file is used to generate the public variable definitions and code which can be embedded in the microcontroller. The appropriate device family from the pre-assembler code are generated by simply selecting the compile pull-down menu. Once the pre-assembler file is generated, the "hooks" to the main program must be added.

The best way to embed the code is to use the template MYMAIN.ASM. The template for each of the families of devices (PIC16C5X, PIC16CXXX and PIC17CXX) is included in the *fuzzyTECH[®]-MP* development kit. The template shown in Appendix B is for the PIC16CXXX family.

The file MYMAIN.ASM should contain your program in the "main_loop" section. The only other modifications required to the template are listed below and are specified in the left hand column of Appendix B.

1. Processor Type definition
2. Code Start Address
3. Fuzzy RAM Start Address
4. Include Public Variable Definition file (myproj.var), which was created by *fuzzyTECH[®]-MP*
5. Include Pre-Assembler Code (myproj.asm) which was created by *fuzzyTECH[®]-MP*
6. Call Initialization (initmyproj) which was created by *fuzzyTECH[®]-MP*
7. Set Crisp Input Value(s)
8. Call Fuzzy Logic System (myproj)
9. Read Crisp Output Value(s)

For this project, the fuzzy logic algorithm assembled to 704 words of program memory and 41 bytes of data memory.

SUMMARY

This project demonstrates many aspects of fuzzy logic control - quick development cycle, real-time debug, sensor integration, and non-linear system control. The total development time for the application took less than a week and performed well within system specifications.

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX A: FUZZY TECHNOLOGY LANGUAGE FILE

```
PROJECT {
  NAME = B_BALL.FTL;
  AUTHOR = ROBERT SCHREIBER;
  DATEFORMAT = M.D.YY;
  LASTCHANGE = 9.16.94;
  CREATED = 9.14.94;
  SHELL = MP;
  COMMENT {
} /* COMMENT */
SHELLOPTIONS {
  ONLINE_REFRESHTIME = 55;
  ONLINE_TIMEOUTCOUNT = 0;
  ONLINE_CODE = OFF;
  TRACE_BUFFER = (OFF, PAR(10000));
  BSUM_AGGREGATION = OFF;
  PUBLIC_IO = ON;
  FAST_CMBF = ON;
  FAST_COA = OFF;
  SCALE_MBF = OFF;
  FILE_CODE = OFF;
  BTYPE = 8_BIT;
} /* SHELLOPTIONS */
MODEL {
  VARIABLE_SECTION {
    LVAR {
      NAME = current_height;
      BASEVAR = Current_Height;
      LVRANGE = MIN(0.000000), MAX(120.000000),
              MINDEF(0), MAXDEF(255),
              DEFAULT_OUTPUT(120.000000);
      RESOLUTION = XGRID(0.000000), YGRID(1.000000),
                  SHOWGRID (ON), SNAPTOGRID(ON);

      TERM {
        TERMNAME = very_lo;
        POINTS = (0.000000, 1.000000),
                (14.117647, 0.000000),
                (120.000000, 0.000000);
        SHAPE = LINEAR;
        COLOR = RED (255), GREEN (0), BLUE (0);
      }

      TERM {
        TERMNAME = lo;
        POINTS = (0.000000, 0.000000),
                (5.176471, 0.000000),
                (24.941176, 1.000000),
                (40.941176, 0.000000),
                (120.000000, 0.000000);
        SHAPE = LINEAR;
        COLOR = RED (0), GREEN (255), BLUE (0);
      }

      TERM {
        TERMNAME = medium;
        POINTS = (0.000000, 0.000000),
                (27.294118, 0.000000),
                (51.294118, 1.000000),
                (66.352941, 0.000000),
                (120.000000, 0.000000);
        SHAPE = LINEAR;
        COLOR = RED (0), GREEN (0), BLUE (255);
      }
    }
  }
}
```

AN600

```
TERM {
  TERMNAME = hi;
  POINTS = (0.000000, 0.000000),
           (55.529412, 0.000000),
           (82.352941, 1.000000),
           (106.352941, 0.000000),
           (120.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (128), GREEN (0), BLUE (0);
}
TERM {
  TERMNAME = very_hi;
  POINTS = (0.000000, 0.000000),
           (73.411765, 0.000000),
           (113.411765, 1.000000),
           (120.000000, 1.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (128), BLUE (0);
}
/* LVAR */
LVAR {
  NAME      = delta_height;
  BASEVAR   = Delta_Height;
  LVRANGE   = MIN(-50.000000), MAX(50.000000),
             MINDEF(0), MAXDEF(255),
             DEFAULT_OUTPUT(-50.000000);
  RESOLUTION = XGRID(0.000000), YGRID(1.000000),
              SHOWGRID (ON), SNAPTOGRID(ON);
}
TERM {
  TERMNAME = neg_big;
  POINTS = (-50.000000, 1.000000),
           (-16.666667, 0.000000),
           (50.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (255), GREEN (0), BLUE (0);
}
TERM {
  TERMNAME = neg_small;
  POINTS = (-50.000000, 0.000000),
           (-21.764706, 0.000000),
           (-6.470588, 1.000000),
           (-0.588235, 0.000000),
           (50.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (255), BLUE (0);
}
TERM {
  TERMNAME = zero;
  POINTS = (-50.000000, 0.000000),
           (-12.352941, 0.000000),
           (0.196078, 1.000000),
           (13.529412, 0.000000),
           (50.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (0), BLUE (255);
}
TERM {
  TERMNAME = pos_small;
  POINTS = (-50.000000, 0.000000),
           (0.196078, 0.000000),
           (10.000000, 1.000000),
           (10.392157, 1.000000),
           (32.745098, 0.000000),
           (50.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (128), GREEN (0), BLUE (0);
}
```

```

}
TERM {
    TERMNAME = pos_big;
    POINTS = (-50.000000, 0.000000),
             (26.470588, 0.000000),
             (39.803922, 1.000000),
             (50.000000, 1.000000);

    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (128), BLUE (0);
}
} /* LVAR */
LVAR {
    NAME      = duty_cycle;
    BASEVAR   = Duty_Cycle;
    LVRANGE   = MIN(0.000000), MAX(255.000000),
             MINDEF(0), MAXDEF(255),
             DEFAULT_OUTPUT(0.000000);
    RESOLUTION = XGRID(0.000000), YGRID(1.000000),
             SHOWGRID (ON), SNAPTOGRID(ON);
}
TERM {
    TERMNAME = very_slow;
    POINTS = (0.000000, 0.000000),
             (1.000000, 0.000000),
             (103.000000, 1.000000),
             (113.000000, 1.000000),
             (147.000000, 0.000000),
             (255.000000, 0.000000);

    SHAPE = LINEAR;
    COLOR = RED (255), GREEN (0), BLUE (0);
}
TERM {
    TERMNAME = slow;
    POINTS = (0.000000, 0.000000),
             (108.000000, 0.000000),
             (127.000000, 1.000000),
             (131.000000, 0.000000),
             (255.000000, 0.000000);

    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (255), BLUE (0);
}
TERM {
    TERMNAME = med_slow;
    POINTS = (0.000000, 0.000000),
             (133.000000, 0.000000),
             (142.000000, 1.000000),
             (162.000000, 0.000000),
             (255.000000, 0.000000);

    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (128), BLUE (128);
}
TERM {
    TERMNAME = medium;
    POINTS = (0.000000, 0.000000),
             (151.000000, 0.000000),
             (164.000000, 1.000000),
             (166.000000, 1.000000),
             (174.000000, 0.000000),
             (255.000000, 0.000000);

    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (0), BLUE (255);
}
TERM {
    TERMNAME = med_fast;
    POINTS = (0.000000, 0.000000),
             (166.000000, 0.000000),
             (178.000000, 1.000000),

```

AN600

```
        (193.000000, 0.000000),
        (255.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (255), GREEN (0), BLUE (128);
}
TERM {
    TERMNAME = fast;
    POINTS = (0.000000, 0.000000),
             (189.000000, 0.000000),
             (202.000000, 1.000000),
             (232.000000, 0.000000),
             (255.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (128), GREEN (0), BLUE (0);
}
TERM {
    TERMNAME = very_fast;
    POINTS = (0.000000, 0.000000),
             (206.000000, 0.000000),
             (255.000000, 1.000000);
    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (128), BLUE (0);
}
} /* LVAR */
LVAR {
    NAME = velocity;
    BASEVAR = Velocity;
    LVRANGE = MIN(-5.000000), MAX(5.000000),
             MINDEF(0), MAXDEF(255),
             DEFAULT_OUTPUT(0.000000);
    RESOLUTION = XGRID(0.000000), YGRID(1.000000),
                SHOWGRID (OFF), SNAPTOGRID(ON);
    TERM {
        TERMNAME = neg_big;
        POINTS = (-5.000000, 1.000000),
                 (-3.784314, 1.000000),
                 (-2.529412, 0.000000),
                 (5.000000, 0.000000);
        SHAPE = LINEAR;
        COLOR = RED (255), GREEN (0), BLUE (0);
    }
    TERM {
        TERMNAME = neg_med;
        POINTS = (-5.000000, 0.000000),
                 (-3.784314, 0.000000),
                 (-2.529412, 1.000000),
                 (-1.274510, 0.000000),
                 (5.000000, 0.000000);
        SHAPE = LINEAR;
        COLOR = RED (0), GREEN (255), BLUE (0);
    }
    TERM {
        TERMNAME = neg_small;
        POINTS = (-5.000000, 0.000000),
                 (-2.568627, 0.000000),
                 (-1.313725, 1.000000),
                 (-0.058824, 0.000000),
                 (5.000000, 0.000000);
        SHAPE = LINEAR;
        COLOR = RED (0), GREEN (0), BLUE (255);
    }
    TERM {
        TERMNAME = zero;
        POINTS = (-5.000000, 0.000000),
                 (-1.000000, 0.000000),
                 (-0.019608, 1.000000),
```

```

        (0.960784, 0.000000),
        (5.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (128), GREEN (0), BLUE (0);
}
TERM {
    TERMNAME = pos_small;
    POINTS = (-5.000000, 0.000000),
            (-0.137255, 0.000000),
            (1.117647, 1.000000),
            (2.372549, 0.000000),
            (5.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (128), BLUE (0);
}
TERM {
    TERMNAME = pos_med;
    POINTS = (-5.000000, 0.000000),
            (1.078431, 0.000000),
            (2.333333, 1.000000),
            (3.588235, 0.000000),
            (5.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (0), BLUE (128);
}
TERM {
    TERMNAME = pos_big;
    POINTS = (-5.000000, 0.000000),
            (2.294118, 0.000000),
            (3.549020, 1.000000),
            (5.000000, 1.000000);
    SHAPE = LINEAR;
    COLOR = RED (255), GREEN (0), BLUE (128);
}
} /* LVAR */
} /* VARIABLE_SECTION */

OBJECT_SECTION {
    INTERFACE {
        INPUT = (current_height, FCMBF);
        POS = -213, -137;
        RANGECHECK = ON;
    }
    INTERFACE {
        INPUT = (delta_height, FCMBF);
        POS = -216, -83;
        RANGECHECK = ON;
    }
    INTERFACE {
        OUTPUT = (duty_cycle, COM);
        POS = 158, -79;
        RANGECHECK = ON;
    }
    RULEBLOCK {
        INPUT = current_height, delta_height, velocity;
        OUTPUT = duty_cycle;
        AGGREGATION = (MIN_MAX, PAR (0.000000));
        COMPOSITION = (GAMMA, PAR (0.000000));
        POS = -39, -113;
        RULES {
            IF    current_height = very_lo
            AND  delta_height = neg_big
            THEN duty_cycle = slow    WITH 1.000;
            IF    current_height = very_lo
            AND  delta_height = neg_small
            THEN duty_cycle = med_slow WITH 1.000;
        }
    }
}

```

AN600

```
IF    current_height = very_lo
AND  delta_height = zero
THEN duty_cycle = medium WITH 1.000;
IF    current_height = very_lo
AND  delta_height = pos_small
THEN duty_cycle = fast WITH 1.000;
IF    current_height = very_lo
AND  delta_height = pos_big
THEN duty_cycle = very_fast WITH 1.000;
IF    current_height = lo
AND  delta_height = neg_big
THEN duty_cycle = slow WITH 1.000;
IF    current_height = lo
AND  delta_height = neg_small
THEN duty_cycle = med_slow WITH 1.000;
IF    current_height = lo
AND  delta_height = zero
THEN duty_cycle = medium WITH 1.000;
IF    current_height = lo
AND  delta_height = pos_small
THEN duty_cycle = fast WITH 1.000;
IF    current_height = lo
AND  delta_height = pos_big
THEN duty_cycle = very_fast WITH 1.000;
IF    current_height = medium
AND  delta_height = neg_big
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = medium
AND  delta_height = neg_small
THEN duty_cycle = med_slow WITH 1.000;
IF    current_height = medium
AND  delta_height = zero
THEN duty_cycle = med_fast WITH 1.000;
IF    current_height = medium
AND  delta_height = pos_small
THEN duty_cycle = fast WITH 1.000;
IF    current_height = medium
AND  delta_height = pos_big
THEN duty_cycle = very_fast WITH 1.000;
IF    current_height = hi
AND  delta_height = neg_big
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = hi
AND  delta_height = neg_small
THEN duty_cycle = med_slow WITH 1.000;
IF    current_height = hi
AND  delta_height = zero
THEN duty_cycle = med_fast WITH 1.000;
IF    current_height = hi
AND  delta_height = pos_small
THEN duty_cycle = fast WITH 1.000;
IF    current_height = hi
AND  delta_height = pos_big
THEN duty_cycle = very_fast WITH 1.000;
IF    current_height = very_hi
AND  delta_height = neg_big
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = very_hi
AND  delta_height = neg_small
THEN duty_cycle = slow WITH 1.000;
IF    current_height = very_hi
AND  delta_height = zero
THEN duty_cycle = med_slow WITH 1.000;
IF    current_height = very_hi
AND  delta_height = pos_small
THEN duty_cycle = medium WITH 1.000;
```



```
IF    current_height = very_hi
AND  delta_height = pos_big
THEN duty_cycle = very_fast WITH 1.000;
IF    current_height = very_lo
AND  delta_height = neg_small
AND  velocity = zero
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = very_lo
AND  delta_height = neg_small
AND  velocity = pos_small
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = very_lo
AND  delta_height = neg_small
AND  velocity = pos_med
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = very_lo
AND  delta_height = neg_small
AND  velocity = pos_big
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = very_lo
AND  delta_height = pos_small
AND  velocity = zero
THEN duty_cycle = fast WITH 1.000;
IF    current_height = very_lo
AND  delta_height = pos_small
AND  velocity = neg_small
THEN duty_cycle = fast WITH 1.000;
IF    current_height = very_lo
AND  delta_height = pos_small
AND  velocity = neg_med
THEN duty_cycle = fast WITH 1.000;
IF    current_height = very_lo
AND  delta_height = pos_small
AND  velocity = neg_big
THEN duty_cycle = fast WITH 1.000;
IF    current_height = lo
AND  delta_height = neg_small
AND  velocity = zero
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = lo
AND  delta_height = neg_small
AND  velocity = pos_small
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = lo
AND  delta_height = neg_small
AND  velocity = pos_med
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = lo
AND  delta_height = neg_small
AND  velocity = pos_big
THEN duty_cycle = very_slow WITH 1.000;
IF    current_height = lo
AND  delta_height = pos_small
AND  velocity = zero
THEN duty_cycle = fast WITH 1.000;
IF    current_height = lo
AND  delta_height = pos_small
AND  velocity = neg_small
THEN duty_cycle = fast WITH 1.000;
IF    current_height = lo
AND  delta_height = pos_small
AND  velocity = neg_med
THEN duty_cycle = fast WITH 1.000;
IF    current_height = lo
AND  delta_height = pos_small
AND  velocity = neg_big
```

AN600

```
THEN duty_cycle = fast WITH 1.000;
IF current_height = medium
  AND delta_height = neg_small
  AND velocity = zero
THEN duty_cycle = slow WITH 1.000;
IF current_height = medium
  AND delta_height = neg_small
  AND velocity = pos_small
THEN duty_cycle = slow WITH 1.000;
IF current_height = medium
  AND delta_height = neg_small
  AND velocity = pos_med
THEN duty_cycle = slow WITH 1.000;
IF current_height = medium
  AND delta_height = neg_small
  AND velocity = pos_big
THEN duty_cycle = slow WITH 1.000;
IF current_height = medium
  AND delta_height = pos_small
  AND velocity = zero
THEN duty_cycle = fast WITH 1.000;
IF current_height = medium
  AND delta_height = pos_small
  AND velocity = neg_small
THEN duty_cycle = fast WITH 1.000;
IF current_height = medium
  AND delta_height = pos_small
  AND velocity = neg_med
THEN duty_cycle = fast WITH 1.000;
IF current_height = medium
  AND delta_height = pos_small
  AND velocity = neg_big
THEN duty_cycle = fast WITH 1.000;
IF current_height = hi
  AND delta_height = neg_small
  AND velocity = zero
THEN duty_cycle = med_slow WITH 1.000;
IF current_height = hi
  AND delta_height = neg_small
  AND velocity = pos_small
THEN duty_cycle = med_slow WITH 1.000;
IF current_height = hi
  AND delta_height = neg_small
  AND velocity = pos_med
THEN duty_cycle = med_slow WITH 1.000;
IF current_height = hi
  AND delta_height = neg_small
  AND velocity = pos_big
THEN duty_cycle = med_slow WITH 1.000;
IF current_height = hi
  AND delta_height = pos_small
  AND velocity = zero
THEN duty_cycle = very_fast WITH 1.000;
IF current_height = hi
  AND delta_height = pos_small
  AND velocity = neg_small
THEN duty_cycle = very_fast WITH 1.000;
IF current_height = hi
  AND delta_height = pos_small
  AND velocity = neg_med
THEN duty_cycle = very_fast WITH 1.000;
IF current_height = hi
  AND delta_height = pos_small
  AND velocity = neg_big
THEN duty_cycle = very_fast WITH 1.000;
IF current_height = very_hi
```

```
        AND delta_height = neg_small
        AND velocity = zero
    THEN duty_cycle = medium WITH 1.000;
    IF current_height = very_hi
        AND delta_height = neg_small
        AND velocity = pos_small
    THEN duty_cycle = medium WITH 1.000;
    IF current_height = very_hi
        AND delta_height = neg_small
        AND velocity = pos_med
    THEN duty_cycle = medium WITH 1.000;
    IF current_height = very_hi
        AND delta_height = neg_small
        AND velocity = pos_big
    THEN duty_cycle = medium WITH 1.000;
    IF current_height = very_hi
        AND delta_height = pos_small
        AND velocity = zero
    THEN duty_cycle = very_fast WITH 1.000;
    IF current_height = very_hi
        AND delta_height = pos_small
        AND velocity = neg_small
    THEN duty_cycle = very_fast WITH 1.000;
    IF current_height = very_hi
        AND delta_height = pos_small
        AND velocity = neg_med
    THEN duty_cycle = very_fast WITH 1.000;
    IF current_height = very_hi
        AND delta_height = pos_small
        AND velocity = neg_big
    THEN duty_cycle = very_fast WITH 1.000;
} /* RULES */
}
INTERFACE {
    INPUT = (velocity, FCMBF);
    POS = -211, -29;
    RANGECHECK = ON;
}
} /* OBJECT_SECTION */
} /* MODEL */
} /* PROJECT */
TERMINAL {
    BAUDRATE = 9600;
    STOPBITS = 1;
    PROTOCOL = NO;
    CONNECTION = PORT1;
    INPUTBUFFER = 4096;
    OUTPUTBUFFER = 1024;
} /* TERMINAL */
```

APPENDIX B: MYMAIN.ASM TEMPLATE FOR THE PIC16CXXX FAMILY

```

1 PROCESSOR 16C71
;-----
;- USER MAIN FILE
;-----
2 CODE_START      EQU    0x100  ;code startadr for 16C71
  RESET_ADR      EQU    0x000  ;reset vector
3 FUZZY_RAM_START EQU    0x00C  ;first free RAM location for 16C71
4 include         "myproj.var" ;include preassembler variables
  CBLOCK
    user1        ;starts after fuzzy ram locations
                ;reserve 1 byte (example)
  ENDC
  ORG CODE_START ;example start address for code

mymain
6 call          initmyproj  ;call init once
main_loop
  movlw        000          ;example
7 movwf        lv0_Input_1 ;set 1st crisp input
  movlw        0A0          ;example
7 movwf        lv1_Input_2 ;set 2nd crisp input
8 call         myproj       ;call preassembler code
  movf         invalidflags,W
  btfss        Z            ;test if the project is completely defined
  goto         case_no_fire

case_fire
  ;proj OK
9 movf         lv2_Output,W ;fetch crisp output
  ;user code
  goto         main_loop

case_no_fire
  ;no rule defined for this input combination
  ;call default_handling_routine
  ;user code
  goto         main_loop
5 INCLUDE     "myproj.asm"  ;include preassembler code
;-----
;- RESET VECTOR
;-----
  ORG         RESET_ADR
  goto        mymain      ;jump to program code
  END          ;end for assembler (only here)

```

Note: Refer to the "Integration" section for the number descriptions.

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks


The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

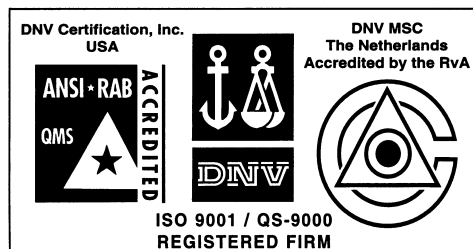
dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.



MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: <http://www.microchip.com>

Rocky Mountain

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta

500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo

2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York

150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia

Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing

Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu

Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou

Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai

Microchip Technology Consulting (Shanghai)
Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen

Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong

Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India

Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan

Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan

Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark

Microchip Technology Nordic ApS
Regus Business Centre
Lautrup høj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France

Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom

Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/18/02