INTRODUCTION

The WDT timer is a simple RC timer with a typical time-out period of about 18 ms. This time-out period is dependent on Voltage, Temperature and Silicon process variations. Hence the tolerance on the time-out period is very wide: Min. of 9 ms to a Max. of 33 ms (please refer to appropriate datasheet for device dependent value). There are applications where an additional timer would be useful as an approximate time keeper, hence getting a more precise value of the WDT time-out is useful. This Tech Brief implements an automatic calibration of the WDT time-out period on start-up.

IMPLEMENTATION

The hardware used for this brief is the PICDEM1 board. It is assumed that the main processor oscillator is an accurate crystal or ceramic resonator running at 4 MHz. The program flowchart is depicted in Figure 1. A check is made to see if a certain Codebyte exists, if it does not then a power-up is assumed and the calibration is executed. Note that instead of using the 8-bit wide CodeByte, the PD bit in the STATUS register could also be used for this very same purpose. The calibration uses an internal timed 1 ms interval. To get a better resolution, the WDT is postscaled by 4 in order to measure a longer time period. Every 1 ms interval is kept track by incrementing the WDTValue. The WDT will eventually time-out and cause a reset. The value in WDTValue in then divided by 4 to get the exact WDT time-out period and displayed as a binary number on PORTB. In order to check for repeatability, just press the reset button on the PICDEM1 board and the whole calibration process will be repeated. A power-down, followed by a power-up will also have the same effect.

CONCLUSION

The calibration of the WDT is simple and takes very little overhead in a program. The code in Appendix A is written for a PIC16C84 device, but can be translated to work on any PIC16CXXX product. The code in Appendix B is written for the PIC16CSX family.

RAM Used: 2 Bytes

Rom Used: 50 Words

Execution time: Up to a max. of 132 ms from start-up.

FIGURE 1: PROGRAM FLOWCHART

ST ART

CodeByte = 0xA5?

Yes

No

CodeByte = 0xA5
WDTValue = 0
OPTION = 0B11111010

Wait 1 ms
increment WDTValue

WDT Reset

WDTValue → PORTB

LOOP
APPENDIX A:

/* In this example, a PIC16C84's WDT is calibrated on startup for better accuracy. Accuracy is exact to +/- 0.25mS.
The code written in C works on a PICDEM1 board. The WDT value is written to PORTB at the end of the calibration.
The value is in binary. */

#define MAXROM 1019
#pragma option v
#include <c:\mpc\16c84.h>
#pragma option +1;
03FB
0005 03FB                         #pragma memory ROM [MAXROM]     @ 0x05;
000C 0024                         #pragma memory RAM [36]         @ 0x0C;
#pragma option +1;
#include <c:\mpc\delay14.h>
#pragma option +l;

char WdtValue;
chr CodeByte;

void InitPorts(void)
{
    PORTB = 0; //configure PORTB as output
    CLRF   PORTB
    TRISB = 0;
    CLRF   PORTB
    RETURN
}

void main(void)
{
    InitPorts();
    if(CodeByte != 0xA5)
    {
        WdtValue = 0;
        CLRF   PIR2
        CodeByte = 0xA5;
        MOVWF  TMR1L
        OPTION = 0B11111010;  // set WDT post scaler = 4
        BSF    STATUS,RP0
        MOVWF  TMR0
        CLRWDT
        while(1)
        {
            MOVFWF TMR1L
    }
    DELAY_Ms_4MHz(1);    // delay for 1mS
    WdtValue++;       // count mS
    INCW   PIR2
    GOTO   0018h
}
else
{    CodeByte = 0;
    MOVWF  TMR1L
0020 0811  MOVF  PIR2,W
0021 008D  MOVWF  PIR1
0022 0C8D  RRF  PIR1
0023 0C8D  RRF  PIR1
0024 303F  MOVLW 3Fh
0025 050D  ANDWF PIR1,W
0026 0086  MOVWF  PORTB

}  
while(1)
0027 0064  CLRWD T
0028 2827  GOTO  0027h
0029 0008  RETURN

}  

void Delay_Ms_4MHz(registerw delay)
0000  

#asm
002A 1283  BCF  STATUS, RP0  ;1
002B 008D  MOVWF  __WImage  ;1

DLMS4M1
RADIX DEC ;Use decimal values
002C 30F9  MOVLW 249  ;1
002D 0084  MOVWF  FSR  ;1

DLMS4M2  ; 4 cycles
002E 0000  NOP  ;1
002F 0B84  DECFSZ  FSR  ;1
0030 282E  GOTO  DLMS4M2  ;2
0031 0B8D  DECFSZ  __WImage  ;1
0032 282C  GOTO  DLMS4M1  ;2

#endasm
0033 0008  RETURN

0000 3000  MOVLW 00h
0001 008A  MOVWF  PCLATH
0002 280A  GOTO  000Ah

ROM USAGE MAP

0000 to 0002  0005 to 0033
Total ROM used 0032

Errors    :    0
Warnings   :    0
/* In this example, a PIC16C54s WDT is calibrated on startup for better accuracy. Accuracy is exact to +/- 0.25 mS. The code written in C works on a PICDEM1 board. The WDT value is written to PORTB at the end of the calibration. The value is in binary. */

#pragma option v
#include <c:\mpc\16c54.h>
#pragma option +l; // Enable header to show branch islands
#define MAXROM 512
#pragma memory ROM [MAXROM] @ 0x00;
#pragma memory RAM [24] @ 0x08; // Reserve byte for __WImage
#pragma option +l;
#include <c:\mpc\delay12.h>
#pragma option +l;

char WdtValue;
char CodeByte;

void InitPorts(void)
{
    CLR PORTB PORTB = 0; //set PORTB as output
    MOVLW 00h __TRIS(0,PORTB);
    TRIS PORTB
    RETLW 00h
}

void main(void)
{
    CALL 0000h InitPorts();
    MOVLW A5h if (CodeByte != 0xA5)
    SUBWF 0A,W
    BTFSC STATUS,Z
    GOTO 0014h
    
    WdtValue = 0;
    CodeByte = 0xA5;
    MOVWF 0A
    MOVLW FAh __OPTION(0B11111010); // set WDT post scaler = 4
    CLRWDT
    while(1)
    
    
    MOVLW 01h
    CALL 001Fh
    Delay_Ms_4MHz(1); // delay for 1 mS
    INCF 09 WdtValue++; // count 1ms delays
    }
    
    if (CodeByte != 0)
    
    PORTB = WdtValue >> 2; //divide by 4 for exact
    // timeout value
    
    MOVWF PORTC
    RRF 07
0018 0327  RRF  07
0019 0C3F  MOVLW  3Fh
001A 0147  ANDWF  07,W
001B 0026  MOVWF  PORTB

}  
while(1)
001C 0004  CLRWD   
CLRWD(); // loop forever
001D 0A1C  GOTO  001Ch
001E 0800  RETLW  00h  
}

void Delay_Ms_4MHz(registerw delay)
0000  
{  
#asm
001F 0000  NOP     ;1
0020 0027  MOVWF  __WImage      ;1
DLMS4M1
RADIX DEC  ;Use decimal values
0021 0CF9  MOVLW  249           ;1
0022 0028  MOVWF  __FSRImage           ;1
DLMS4M2                ;4 cycles
0023 0000  NOP               ;1
0024 02E8  DECFSZ  __FSRImage        ;1
0025 0A23  goto DLMS4M2       ;2
0026 02E7  DECFSZ  __WImage ;1
0027 0A21  goto DLMS4M1       ;2
#endasm
0028 0800  RETLW  00h          }
01FF 0A04  GOTO  0004h

ROM USAGE MAP

0000 to 0028   01FF to 01FF
Total ROM used 002A

Errors             :    0
Warnings           :    0