# Application Note 2420
# 1-Wire Communication with a Microchip PICmicro Microcontroller

www.maxim-ic.com

## INTRODUCTION

Microchip's PICmicro® microcontroller devices (PICs) have become a popular design choice for low-power and low-cost system solutions. The microcontrollers have multiple general-purpose input/output (GPIO) pins, and can be easily configured to implement Dallas Semiconductor's 1-Wire® protocol. The 1-Wire protocol allows interaction with many Dallas Semiconductor parts including battery and thermal management, memory, iButtons®, and more. This application note will present general 1-Wire routines for a PIC16F628 and explain the timing and associated details. For added simplicity, a 4MHz clock is assumed for all material presented, and this frequency is available as an internal clock on many PICs. Appendix A of this document contains an include file with all 1-Wire routines. Appendix B presents a sample assembly code program designed for a PIC16F628 to read from a DS2761 High-Precision Li+ Battery Monitor. This application note is limited in scope to regular speed 1-Wire communication.

### General Macros

In order to transmit the 1-Wire protocol as a master, only two GPIO states are necessary: high impedance and logic low. The following PIC assembly code snippets achieve these two states. The PIC16F628 has two GPIO ports, PORTA and PORTB. Either of the ports could be setup for 1-Wire communication, but for this example, PORTB is used. Also, the following code assumes that a constant DQ has been configured in the assembly code to indicate which bit in PORTB will be the 1-Wire pin. Throughout the code, this bit number is simply called DQ. Externally, this pin must be tied to a power supply via a pullup resistor.

```
OW_HIZ:MACRO
;Force the DQ line into a high impedance state.
        BSF     STATUS,RP0   ; Select Bank 1 of data memory
        BSF     TRISB, DQ    ; Make DQ pin High Z
        BCF     STATUS,RP0   ; Select Bank 0 of data memory
        ENDM

OW_LO:MACRO
;Force the DQ line to a logic low.
        BCF     STATUS,RP0   ; Select Bank 0 of data memory
        BCF     PORTB, DQ    ; Clear the DQ bit
        BSF     STATUS,RP0   ; Select Bank 1 of data memory
        BCF     TRISB, DQ    ; Make DQ pin an output
        BCF     STATUS,RP0   ; Select Bank 0 of data memory
        ENDM
```

Both of these snippets of code are written as macros. By writing the code as a macro, it is automatically inserted into the assembly source code by using a single macro call. This limits the number of times the code must be rewritten. The first macro, OW_HIZ, forces the DQ line to a high impedance state. The first step is to choose the bank 1 of data memory because the TRISB register is located in bank 1. Next, the DQ output driver is changed to a high impedance state by setting the DQ bit in the TRISB register. The last line of code changes back to bank 0 of data memory. The last line is not necessary, but is used so that all macros and function calls leave the data memory in a known state.

The second macro, OW_LO, forces the DQ line to a logic low. First, bank 0 of data memory is selected, so the PORTB register can be accessed. The PORTB register is the data register, and contains the values that will be forced to the TRISB pins if they are configured as outputs.

*PICmicro is a registered trademark of Microchip Technology Inc.*
*1-Wire and iButton are registered trademarks of Dallas Semiconductor.*

The DQ bit of PORTB is cleared so the line will be forced low. Finally, bank 1 of data memory is selected, and the DQ bit of the TRISB register is cleared, making it an output driver. As always, the macro ends by selecting bank 0 of data memory.

A final macro labeled WAIT is included to produce delays for the 1-Wire signaling. WAIT is used to produce delays in multiples of 5µs. The macro is called with a value of TIME in microseconds, and the corresponding delay time is generated. The macro simply calculates the number of times that a 5µs delay is needed, and then loops within WAIT5U. The routine WAIT5U is shown in the next section. For each instruction within WAIT, the processing time is given as a comment to help understand how the delay is achieved.

```
WAIT:MACRO TIME
;Delay for TIME µs.
;Variable time must be in multiples of 5µs.
        MOVLW       (TIME/5) - 1     ;1µs to process
        MOVWF       TMP0             ;1µs to process
        CALL        WAIT5U           ;2µs to process
        ENDM
```

## General 1-Wire Routines

The 1-Wire timing protocol has specific timing constraints that must be followed in order to achieve successful communication. To aid in making specific timing delays, the routine WAIT5U is used to generate 5µs delays. This routine is shown below.

```
WAIT5U:
;This takes 5µs to complete
        NOP                          ;1µs to process
        NOP                          ;1µs to process
        DECFSZ      TMP0,F           ;1µs if not zero or 2µs if zero
        GOTO        WAIT5U           ;2µs to process
        RETLW       0                ;2µs to process
```

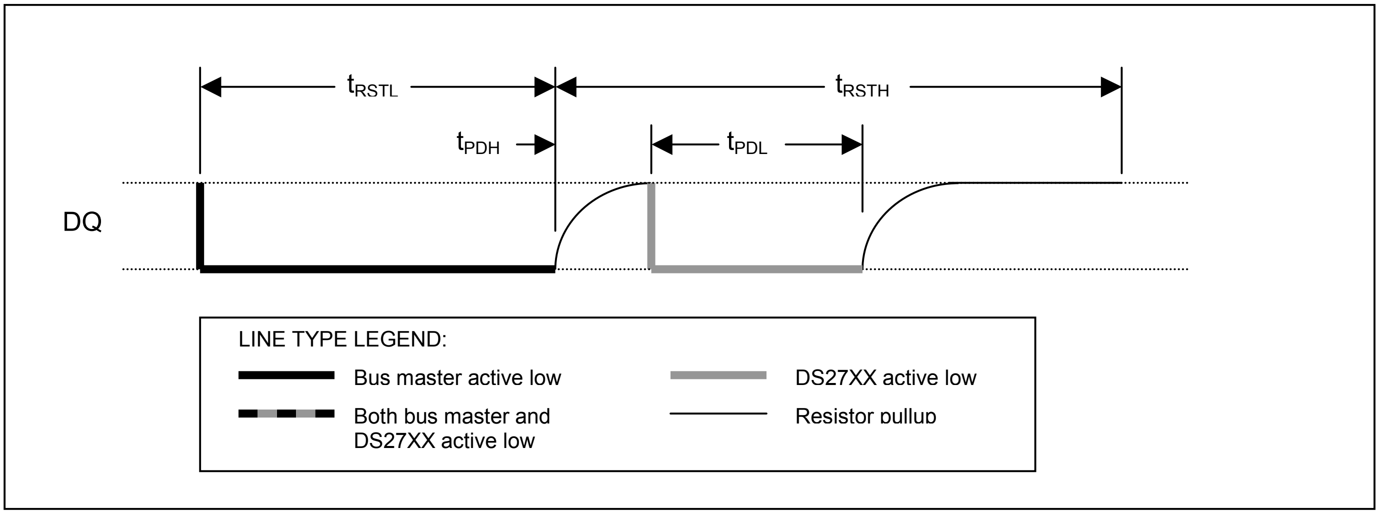
When used in combination with the WAIT macro, simple timing delays can be generated. For example, if a 40µs delay is needed, WAIT 0.40 would be called. This causes the first 3 lines in WAIT to execute resulting in 4µs. Next, the first 4 lines of code in WAIT5U executes in 5µs and loops 6 times for a total of 30µs. The last loop of WAIT5U takes 6µs and then returns back to the WAIT macro. Thus, the total time to process would be 30 + 4 + 6 = 40µs.

## Table 1. Regular Speed 1-Wire Interface Timing

| $2.5V \le V_{DD} \le 5.5V$, $T_A$ = -20°C to 70°C.) | | | | | |
|---|---|---|---|---|---|
| **PARAMETER** | **SYMBOL** | **MIN** | **TYP** | **MAX** | **UNITS** |
| Time Slot | $t_{SLOT}$ | 60 | | 120 | µs |
| Recovery Time | $t_{REC}$ | 1 | | | µs |
| Write 0 Low Time | $t_{LOW0}$ | 60 | | 120 | µs |
| Write 1 Low Time | $t_{LOW1}$ | 1 | | 15 | µs |
| Read Data Valid | $t_{RDV}$ | | | 15 | µs |
| Reset Time High | $t_{RSTH}$ | 480 | | | µs |
| Reset Time Low | $t_{RSTL}$ | 480 | | 960 | µs |
| Presence Detect High | $t_{PDH}$ | 15 | | 60 | µs |
| Presence Detect Low | $t_{PDL}$ | 60 | | 240 | µs |

The start of any 1-Wire transaction begins with a reset pulse from the master device followed by a presence detect pulse from the slave device. Figure 1 illustrates this transaction. This initialization sequence can easily be transmitted via the PIC, and the assembly code is shown below Figure 1. The 1-Wire timing specifications for initialization, reading, and writing are given above in Table 1. These parameters are referenced throughout the rest of the document.

**Figure 1. 1-Wire Initialization Sequence**

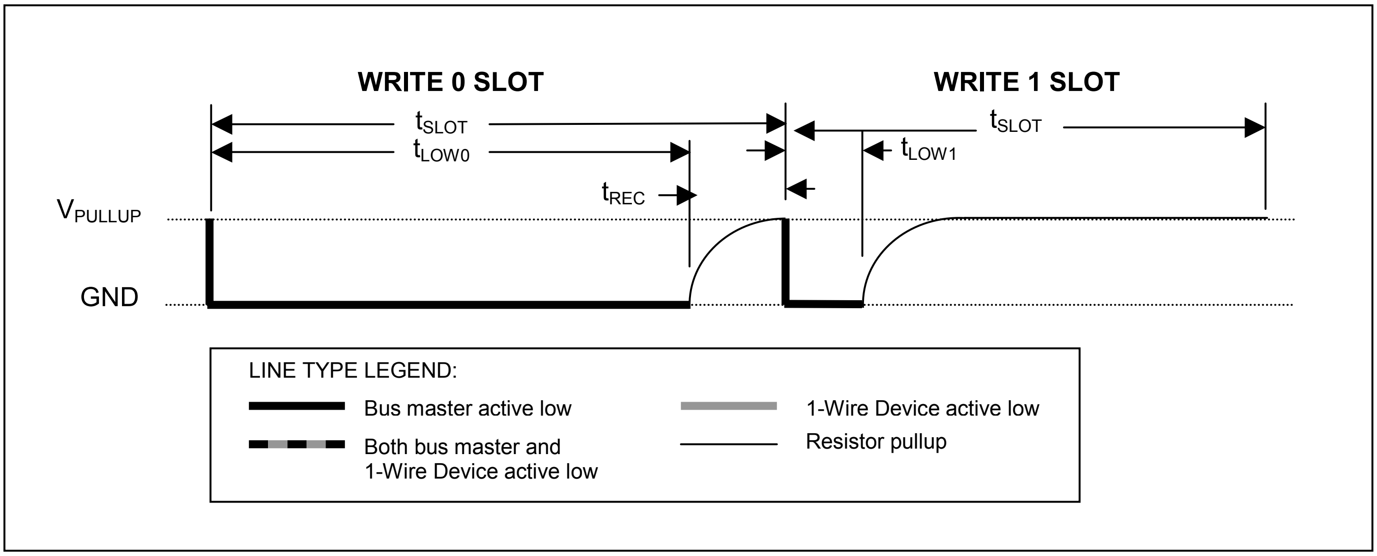



```
OW_RESET:
        OW_HIZ                          ; Start with the line high
        CLRF        PDBYTE              ; Clear the PD byte
        OW_LO
        WAIT        .500                ; Drive Low for 500µs
        OW_HIZ
        WAIT        .70                 ; Release line and wait 70µs for PD Pulse
        BTFSS       PORTB,DQ            ; Read for a PD Pulse
        INCF        PDBYTE,F            ; Set PDBYTE to 1 if get a PD Pulse
        WAIT        .430                ; Wait 430µs after PD Pulse
        RETLW       0
```

The OW_RESET routine starts by ensuring the DQ pin is in a high impedance state so it can be pulled high by the pullup resistor. Next, it clears the PDBYTE register so it is ready to validate the next presence detect pulse. After that, the DQ pin is driven low for 500µs. This meets the tRSTL parameter shown in Table 1, and also provides a 20µs additional buffer. After driving the pin low, the pin is released to a high impedance state and a delay of 70µs is added before reading for the presence detect pulse. Using 70µs ensures that the PIC will sample at a valid time for any combination of tPDL and tPDH. Once the presence detect pulse is read, the PDBYTE register is adjusted to show the logic level read. The DQ pin is then left in a high-impedance state for an additional 430µs to ensure that the tRSTH time has been met, and includes a 20µs additional buffer.

The next routine needed for 1-Wire communication is DSTXBYTE, which is used to transmit data to a 1-Wire slave device. The PIC code for this routine is shown below Figure 2. This routine is called with the data to be sent in the WREG register, and it is immediately moved to the IOBYTE register. Next, a COUNT register is initialized to 8 to count the number of bits sent out the DQ line. Starting at the DSTXLP, the PIC starts sending out data. First the DQ pin is driven low for 5µs regardless of what logic level is sent. This ensures the tLOW1 time is met. Next, the lsb of the IOBYTE is shifted into the CARRY bit, and then tested for a one or a zero. If the CARRY is a one, the DQ bit of PORTB is set which changes the pin to a high impedance state and the line is pulled high by the pullup resistor. If the CARRY is a zero, the line is kept low. Next a delay of 60µs is added to allow for the minimum tLOW0 time. After the 60µs wait, the pin is changed to a high impedance state, and then an additional 2µs are added for pullup resistor recovery. Finally, the COUNT register is decremented. If the COUNT register is zero, all eight bits have been sent and the routine is done. If the COUNT register is not zero, another bit is sent starting at DSTXLP. A visual interpretation of the write zero and write one procedure is shown in Figure 2.

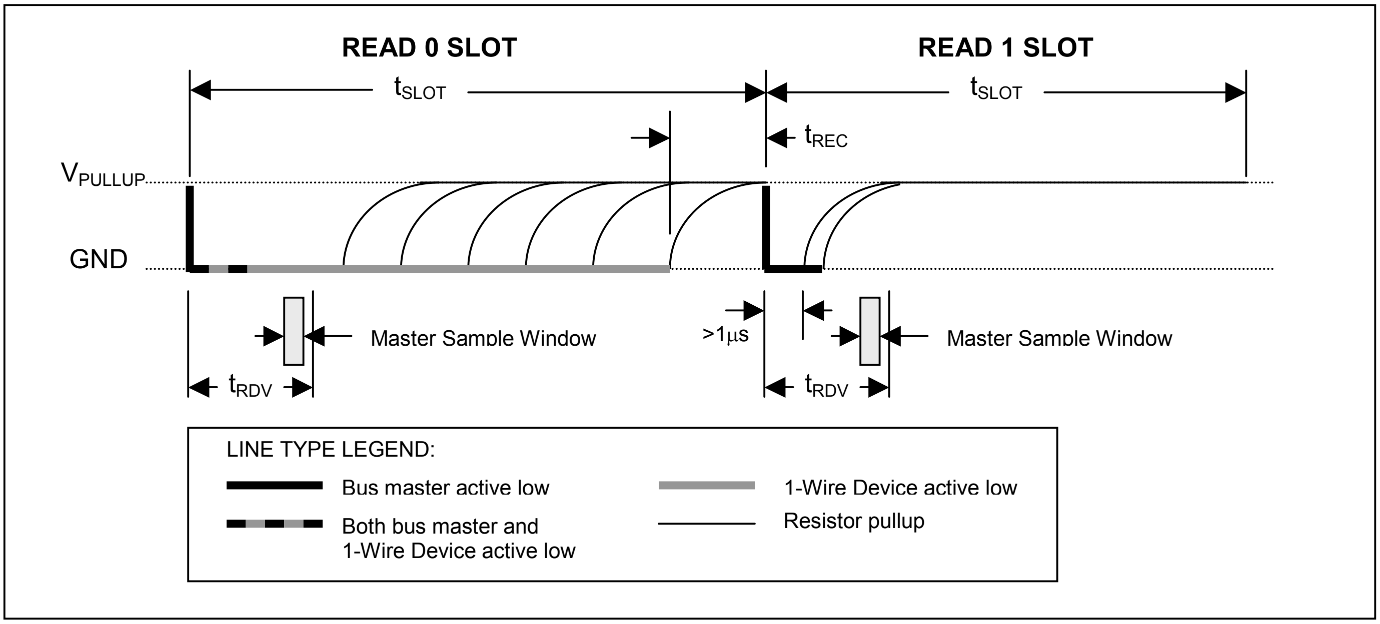
**Figure 2. 1-Wire Write Time Slots**




```
DSTXBYTE:                          ; Byte to send starts in W
        MOVWF       IOBYTE         ; We send it from IOBYTE
        MOVLW       .8
        MOVWF       COUNT          ; Set COUNT equal to 8 to count the bits
DSTXLP:
        OW_LO
        NOP
        NOP
        NOP
        NOP
        NOP                        ; Drive the line low for 5µs
        RRF         IOBYTE,F       ; The data byte
        BTFSC       STATUS,C       ; Check the LSB of IOBYTE for 1 or 0
        BSF         PORTB,DQ       ; Drive the line high if LSB is 1
        WAIT        .60            ; Continue driving line for 60µs
        OW_HIZ                     ; Release the line for pullup
        NOP
        NOP                        ; Recovery time of 2µs
        DECFSZ      COUNT,F        ; Decrement the bit counter
        GOTO        DSTXLP
        RETLW       0
```

The final routine for 1-Wire communication is DSRXBYTE, which allows the PIC to receive information from a slave device. The code is shown below Figure 3. The COUNT register is initialized to 8 before any DQ activity begins and its function is to count the number of bits received. The DSRXLP begins by driving the DQ pin low to signal to the slave device that the PIC is ready to receive data. The line is driven low for 6µs, and then released by putting the DQ pin into a high impedance state. Next, the PIC waits an additional 4µs before sampling the data line. There is 1 line of code in OW_LO after the line is driven low, and 3 lines of code within OW_HIZ. Each line takes 1µs to process. Adding up all the time results in 1 + 6 + 3 + 4 = 14µs which is just below the tRDV spec of 15µs. After the PORTB register is read, the DQ bit is masked off, and then the register is added to 255 to force the CARRY bit to mirror the DQ bit. The CARRY bit is then shifted into IOBYTE where the incoming byte is stored. Once the byte is stored a delay of 50µs is added to ensure that tSLOT is met. The last check is to determine if the COUNT register is zero. If it is zero, 8 bits have been read, and the routine is exited. Otherwise, the loop is repeated at DSRXLP. The read zero and read one transactions are visually shown in Figure 3.

**Figure 3. 1-Wire Read Time Slots**




```
DSRXBYTE:                           ; Byte read is stored in IOBYTE
        MOVLW       .8
        MOVWF       COUNT           ; Set COUNT equal to 8 to count the bits
DSRXLP:
        OW_LO
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP                         ; Bring DQ low for 6µs
        OW_HIZ
        NOP
        NOP
        NOP
        NOP                         ; Change to HiZ and Wait 4µs
        MOVF        PORTB,W         ; Read DQ
        ANDLW       1<<DQ           ; Mask off the DQ bit
        ADDLW       .255            ; C = 1 if DQ = 1: C = 0 if DQ = 0
        RRF         IOBYTE,F        ; Shift C into IOBYTE
        WAIT        .50             ; Wait 50µs to end of time slot
        DECFSZ      COUNT,F         ; Decrement the bit counter
        GOTO        DSRXLP
RETLW       0
```

## SUMMARY

Dallas Semiconductor's 1-Wire communication protocol can easily be implemented on Microchip's PICmicro line of microcontrollers. In order to complete 1-Wire transactions, only two GPIO states are needed, and the multiple GPIOs on a PIC are easily configured for this task. There are three basic routines necessary for 1-Wire communication: Initialization, Read Byte, and Write Byte. These three routines have been presented and thoroughly detailed to provide accurate 1-Wire regular speed communication. This allows a PIC to interface with any of the many Dallas Semiconductor 1-Wire devices. Appendix A of this document has all three routines in a convenient include file. Appendix B contains a small assembly program meant to interface a PIC16F628 to a DS2761 High-Precision Li+ Battery Monitor.

# APPENDIX A: 1-WIRE INCLUDE FILE (1W_16F6X.INC)

```
; *******************************************************
;
;       Dallas 1-Wire Support for PIC16F628
;
;       Processor has 4MHz clock and 1µs per instruction cycle.
;
; *******************************************************


; *******************************************************
;       Dallas Semiconductor 1-Wire MACROS
; *******************************************************
OW_HIZ:MACRO
        BSF         STATUS,RP0     ; Select Bank 1 of data memory
        BSF         TRISB, DQ      ; Make DQ pin High Z
        BCF         STATUS,RP0     ; Select Bank 0 of data memory
        ENDM
; -------------------------------------------------------
OW_LO:MACRO
        BCF         STATUS,RP0     ; Select Bank 0 of data memory
        BCF         PORTB,  DQ     ; Clear the DQ bit
        BSF         STATUS,RP0     ; Select Bank 1 of data memory
        BCF         TRISB, DQ      ; Make DQ pin an output
        BCF         STATUS,RP0     ; Select Bank 0 of data memory
        ENDM
; -------------------------------------------------------
WAIT:MACRO TIME
;Delay for TIME µs.
;Variable time must be in multiples of 5µs.
        MOVLW       (TIME/5)-1     ;1µs
        MOVWF       TMP0           ;1µs
        CALL        WAIT5U         ;2µs
        ENDM

; *******************************************************
;       Dallas Semiconductor 1-Wire ROUTINES
; *******************************************************
WAIT5U:
;This takes 5uS to complete
        NOP                        ;1µs
        NOP                        ;1µs
        DECFSZ      TMP0,F         ;1µs or 2µs
        GOTO        WAIT5U         ;2µs
        RETLW       0              ;2µs
; -------------------------------------------------------
OW_RESET:
        OW_HIZ                     ; Start with the line high
        CLRF        PDBYTE         ; Clear the PD byte
        OW_LO
        WAIT        .500           ; Drive Low for 500µs
        OW_HIZ
        WAIT        .70            ; Release line and wait 70µs for PD Pulse
        BTFSS       PORTB,DQ       ; Read for a PD Pulse
        INCF        PDBYTE,F       ; Set PDBYTE to 1 if get a PD Pulse
```

```
        WAIT        .400          ; Wait 400µs after PD Pulse
        RETLW       0
; -------------------------------------------------------
DSRXBYTE:                         ; Byte read is stored in IOBYTE
        MOVLW       .8
        MOVWF       COUNT         ; Set COUNT equal to 8 to count the bits
DSRXLP:
        OW_LO
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP                       ; Bring DQ low for 6µs
        OW_HIZ
        NOP
        NOP
        NOP
        NOP                       ; Change to HiZ and Wait 4µs
        MOVF        PORTB,W       ; Read DQ
        ANDLW       1<<DQ         ; Mask off the DQ bit
        ADDLW       .255          ; C=1 if DQ=1:  C=0 if DQ=0
        RRF         IOBYTE,F      ; Shift C into IOBYTE
        WAIT        .50           ; Wait 50µs to end of time slot
        DECFSZ      COUNT,F       ; Decrement the bit counter
        GOTO        DSRXLP
        RETLW       0
; -------------------------------------------------------
DSTXBYTE:                         ; Byte to send starts in W
        MOVWF       IOBYTE        ; We send it from IOBYTE
        MOVLW       .8
        MOVWF       COUNT         ; Set COUNT equal to 8 to count the bits
DSTXLP:
        OW_LO
        NOP
        NOP
        NOP
        NOP
        NOP                       ; Drive the line low for 5µs
        RRF         IOBYTE,F      ; The data byte
        BTFSC       STATUS,C      ; Check the LSB of IOBYTE for 1 or 0
        BSF         PORTB,DQ      ; Drive the line high if LSB is 1
        WAIT        .60           ; Continue driving line for 60µs
        OW_HIZ                    ; Release the line for pullup
        NOP
        NOP                       ; Recovery time of 2µs
        DECFSZ      COUNT,F       ; Decrement the bit counter
        GOTO        DSTXLP
        RETLW       0
; -------------------------------------------------------
```

## APPENDIX B: PIC16F628 TO DS2761 ASSEMBLY CODE (PIC_2_1W.ASM)

```
; *******************************************
;
; Dallas Semiconductor PIC code
;
; This code will interface a PIC16F628 microcontroller to
; a DS2761 High-Precision Li+ Battery Monitor
;
; *******************************************;
;
;                         VCC
;                          ^
;                          |
;                          |
;                          /
;                          \ Rpup
;                          /
;                          \
;                          |
; 16F628                   |                DS2761
; RB1 (pin 7) ------------------------------ DQ (pin 7)
;
; *******************************************;

;---------------------------------------------------------
; List your processor here.

        list p=16F628

; Include the processor header file here.

        #include <p16F628.inc>
;---------------------------------------------------------
; Assign the PORTB with Constants

        constant DQ=1                       ; Use RB1 (pin7) for 1-Wire
;---------------------------------------------------------
; These constants are standard 1-Wire ROM commands

        constant SRCHROM=0xF0
        constant RDROM=0x33
        constant MTCHROM=0x55
        constant SKPROM=0xCC
;---------------------------------------------------------
; These constants are used throughout the code

        cblock          0x20
                        IOBYTE
                        TMP0                ; Address 0x23
                        COUNT               ; Keep track of bits
                        PICMSB              ; Store the MSB
                        PICLSB              ; Store the LSB
                        PDBYTE              ; Presence Detect Pulse
        endc
;---------------------------------------------------------
; Setup your configuration word by using __config.
```

```
; For the 16F628, the bits are:
; CP1,CP0,CP1,CP0,N/A, CPD, LVP, BODEN, MCLRE, FOSC2, PWRTE, WDTE, FOSC1, FOSC0
; CP1 and CP0 are the Code Protection bits
; CPD: is the Data Code Protection Bit
; LVP is the Low Voltage Programming Enable bit
; PWRTE is the power-up Timer enable bit
; WDTE is the Watchdog timer enable bit
; FOSC2, FOSC1 and FOSC0 are the oscillator selection bits.

; CP disabled, LVP disabled, BOD disabled, MCLR enabled, PWRT disabled, WDT disabled, INTRC I/O oscillator
; 11111100111000

        __config 0x3F38
;------------------------------------------------------------
; Set the program origin for subsequent code.

        org 0x00
        GOTO        SETUP
        NOP
        NOP
        NOP
        GOTO        INTERRUPT           ; PC 0x04...INTERRUPT VECTOR!
;------------------------------------------------------------
INTERRUPT:
        SLEEP
;------------------------------------------------------------
; Option Register bits
; ____
; RBPU,INTEDG,TOCS,TOSE,PSA,PS2,PS1,PS0
; 7=PORTB Pullup Enable, 6=Interrupt Edge Select, 5=TMR0 Source,
; 4=TMR0 Source Edge, 3=Prescaler Assign, 2-0=Prescaler Rate Select

; 11010111
; PORTB pullups disabled,rising edge,internal,hightolow,TMR0,1:256

SETUP:
        BCF         STATUS,RP1
        BSF         STATUS,RP0          ; Select Bank 1 of data memory
        MOVLW       0xD7
        MOVWF       OPTION_REG
        BCF         STATUS,RP0          ; Select Bank 0 of data memory
;------------------------------------------------------------

        BCF         INTCON,7            ; Disable all interrupts.

;------------------------------------------------------------
        GOTO        START
;------------------------------------------------------------
; Include the 1-Wire communication routines and macros

        #INCLUDE 1w_16f6x.inc
;------------------------------------------------------------
START:
;------------------------------------------------------------
GET_TEMP:
        CALL        OW_RESET            ; Send Reset Pulse and read for Presence Detect Pulse
        BTFSS       PDBYTE,0            ; 1 = Presence Detect Detected
```

```
        GOTO        NOPDPULSE
        MOVLW       SKPROM
        CALL        DSTXBYTE            ; Send Skip ROM Command (0xCC)
        MOVLW       0x69
        CALL        DSTXBYTE            ; Send Read Data Command (0x69)
        MOVLW       0x0E
        CALL        DSTXBYTE            ; Send the DS2761 Current Register MSB address (0x0E)
        CALL        DSRXBYTE            ; Read the DS2761 Current Register MSB
        MOVF        IOBYTE,W
        MOVWF       PICMSB              ; Put the Current MSB into file PICMSB
        CALL        DSRXBYTE            ; Read the DS2761 Current Register LSB
        MOVF        IOBYTE,W
        MOVWF       PICLSB              ; Put the Current LSB into file PICLSB
        CALL        OW_RESET

NOPDPULSE:                              ; Add some error processing here!
        SLEEP                           ; Put PIC to sleep
;---------------------------------------------------------
        end
```