Digitemp Junior – An RS-232 Port-Powered Digital Thermometer

Author: *Michael Kirkhart*
GSE. Inc.
Farmington Hills MI
Email: kirkhart@rust.net

OVERVIEW

Digitemp Jr. is a device that is designed to measure and report ambient temperature. When connected to an RS-232 port on any PC, it will periodically measure and report in ASCII form the ambient temperature in degrees Celsius. These temperature readings can be monitored with any terminal program. If the terminal program supports capture to disk, the temperature readings can be saved to disk for further analysis with a spreadsheet program or other data analysis tool. The simple ASCII output format of Digitemp Jr. makes it relatively easy to write custom software for receiving, recording, and analyzing ambient temperature data. Best of all, Digitemp Jr. requires no external power supplies or batteries. It is powered directly by the RS-232 port.

DATA TRANSMISSION SPECIFICATIONS

When connected to a PC RS-232 port, Digitemp Jr. will obtain an ambient temperature reading and transmit the reading in degrees Celsius approximately once per second. This reading is transmitted in the following format:

```
(-/H)(T)O.D(CR)(LF)
```

The (-/H) character can either be a negative sign or the hundreds digit of the temperature reading. If the temperature reading is between 0°C and 99.5°C, this character will not be transmitted. Realistically, unless an industrial temperature range PIC12C508 is used, you will not be able to measure ambient temperatures below 0°C. Moreover, since an industrial temperature range PIC12C508 can only operate to 85°C, you will not be able to measure ambient temperatures above 85°C.

The T character is tens digit of the temperature reading. If the temperature reading is between -9.5°C and 9.5°C, this character is not transmitted. The O character is the ones digit of the temperature reading, and the D digit is the tenths digit of the temperature reading. Since the DS1620 has a resolution of 0.5°C, the tenths digit will either be a '0' or a '5'.

The CR and LF characters are the ASCII carriage return (ASCII code 0x0d) and line feed (ASCII code 0x0a) characters. These characters are used to force the terminal program to print the next temperature reading on the next line on the screen. They also serve as delimiting characters between temperature readings, allowing easy importing of a captured data into a spreadsheet program.

Examples:

```
0°C will be transmitted as 0.0(CR)(LF)
27.5°C will be transmitted as 27.5(CR)(LF)
-9.5°C will be transmitted as -9.5(CR)(LF)
-15.5°C will be transmitted as -15.5(CR)(LF)
```

The data is transmitted in asynchronous, NRZ format at 9600 bps with one start bit, one stop bit, eight data bits, and no parity. While neither hardware or software flow control is used, the RTS signal from the PC must be asserted because Digitemp Jr. derives its V+ supply from the PC's RTS output.

HARDWARE

Digitemp Jr. consists of a Microchip Technology's PIC12C508 8 pin microcontroller, a Dallas semiconductor DS1620 digital thermometer, an RS-232 output driver circuit, and a simple Zener diode voltage regulator. The data in/out pin (DQ), the clock pin (CLK), and the device select pin (RST) of the DS1620 are connected to the PIC12C508's GP0, GP1, and GP2 I/O pins, respectively. The RS-232 driver circuit is controlled by the PIC12C508's GP4 I/O pin. The PIC12C508 is configured to use the internal 4MHz RC oscillator as a clock source, and to use the GP3 pin as an MCLR input.

The RS-232 driver circuit, which is controlled by the PIC12C508's GP4 I/O pin, consists of a 2N7000 N-channel MOSFET, a 2N4402 PNP transistor, and several resistors. The emitter of the 2N4402 is connected to 10V (V+), and the collector is connected to the RS-232 output. When the PIC12C508's GP4 I/O pin is low, both transistors are off, and the RS-232 output is pulled to approximately -6V via a 3.01KΩ resistor connected to V-. This is the RS-232 MARK level, and is interpreted by the PC's UART as a logic '1'. When the PIC12C508's GP4 I/O pin is high, the 2N7000 MOSFET turns on, which then turns on the 2N4402 PNP transistor. This causes the RS-232 output to go to approximately +6V. This is the RS-232 SPACE level, and is interpreted by the PC's UART as a logic '0'.

The V+ source is supplied by the RTS pin on the PC serial port through a 1N5719 Schottky diode, and the V- source is supplied by the TxD pin on the PC serial port through another 1N5719 Schottky diode. These diodes protect the device against reverse voltage. The +5V source is derived from the V+ source using a 2KΩ resistor, a 1N4733A Zener diode (5.1V ± 5%), and a 1µF monolithic capacitor. Two 0.1µF decoupling capacitors are provided for power supply decoupling for the PIC12C508 and the DS1620.

For the purpose of testing, the prototype circuit was constructed with sockets for both a PIC12C508 and a PIC16C54. The software was initially written for a PIC16C54, as a PICMASTER™ emulator with a 16C5X pod was available for debugging. After debugging the hardware/software with the PICMASTER, the software was converted for use with a PIC12C508 and re-assembled. This required only very minor changes (adding the oscillator trim instructions and changing the port pin equates), which demonstrates the excellent software scalability of the PIC microcontroller family.

SOFTWARE

The software consists of an initialization block, a main loop, and subroutines for:

- Interfacing with the DS1620
- Asynchronous transmission of one byte of data
- Math routines
- Delay routines

Initialization block

Invoked on a CPU reset, this code trims the on-board RC oscillator, sets up the OPTION register, initializes the GPIO register as well as the TRIS register, and sets up the DS1620 configuration register for one-shot, CPU controlled temperature conversion operation.

Main loop

The main loop, which is executed approximately once per second, starts by sending a START CONVERSION command to the DS1620. It then monitors the DS1620 CONVERSION DONE bit in the configuration register to determine when the conversion is complete. Once the conversion is complete, the results of the temperature conversion are read out of the DS1620. The results of this conversion are converted to ASCII and transmitted via the RS-232 output one byte at a time. The value transmitted is in degrees Celsius. After transmission of the temperature data, a 760 millisecond delay is executed. This delay is used to "pad" the main loop execution time to approximately one second. Finally, the watchdog timer is reset, and the main loop is executed again.

Subroutines

- `TSWrInst` - this subroutine writes an 8-bit instruction to the DS1620, where the instruction is passed in the w register.
- `TSWrData` - this subroutine writes a 9-bit data value to the DS1620, where the 9-bit value is passed in the temp and temp1 file registers.
- `TSRdData` - this subroutine reads a 9 bit data value from the DS1620. The 9-bit value is returned in the temp and temp1 file registers.
- `asyncTx` - this subroutine, which is a modified version of example code from Microchip AN510, is used to transmit one byte of data in asynchronous, NRZ format at 9600 bps. Because of the level inversion that takes place from the PIC's TxD pin (GP4) to the PC UART's RxD pin, this routine sets the GP4 output if the bit to be transmitted is a 0, and clears the GP4 output if the bit to be transmitted is a 1.
- `mpy8X8, B2_BCD` - these subroutines, which were obtained from Microchip AN526, are used when converting the temperature result from the DS1620 into ASCII.
- `delay, L_delay` - these subroutines are used to generate the various time delays needed. The delay subroutine, which was written by Philip Doucet and published in an Electronics Design magazine "Software Ideas for Design" section, generates a programmable delay. The delay duration, which is specified in instruction cycles, is passed in the `delay_h` and `delay_l` file registers. The `L_delay` subroutine, is used to generate delays in multiples of 10 milliseconds. For this routine, the length of the delay, in units of 10 milliseconds, is passed in the w register.

FIGURE 1: OVERALL PROGRAM FLOWCHART

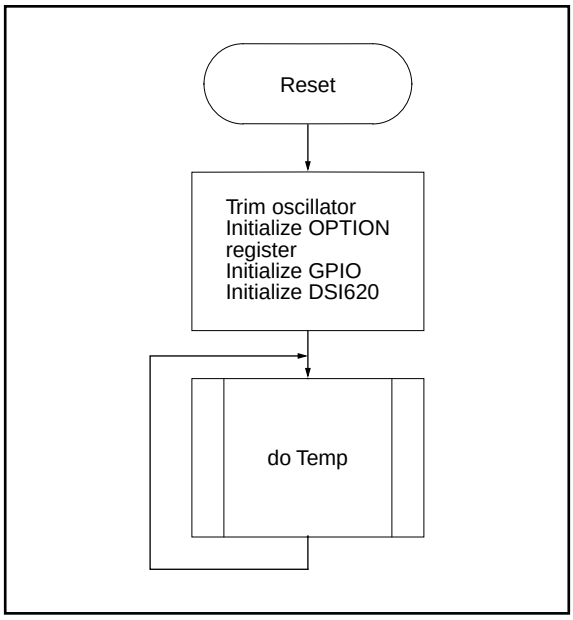


FIGURE 2: MAIN LOOP (DOTEMP) FLOWCHART

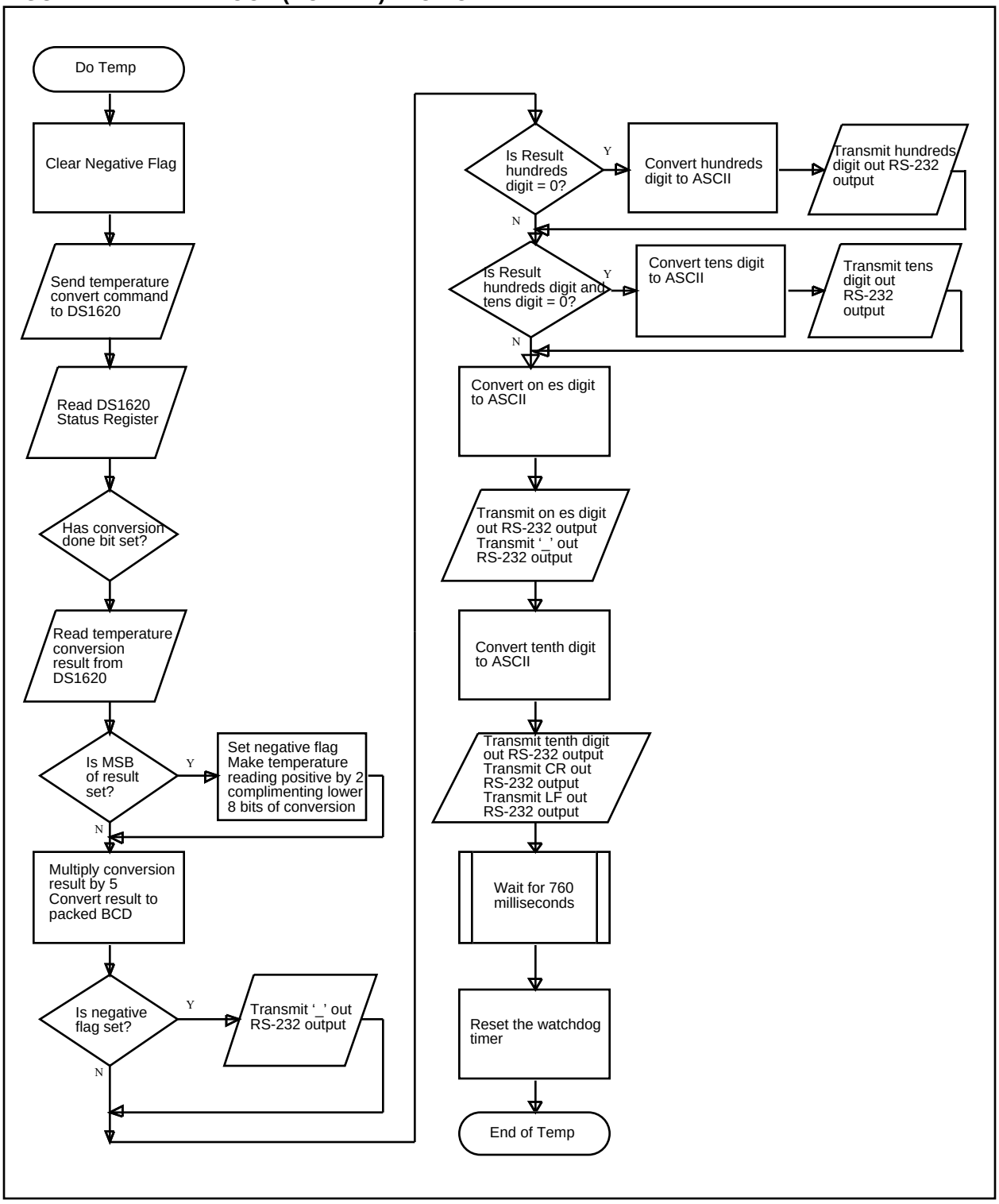
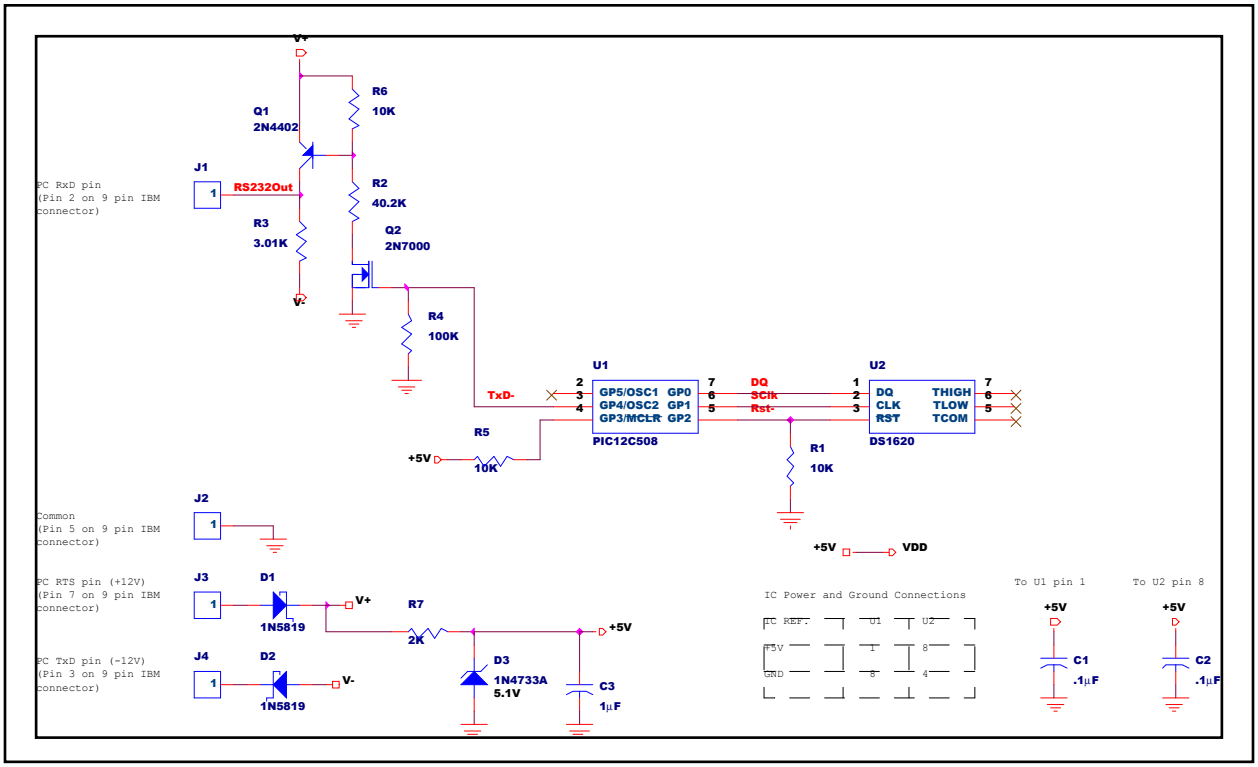


FIGURE 3: SCHEMATIC



MICROCHIP TOOLS USED

Hardware Development Tools Used:

The PICMASTER emulator with a 16C54 pod was used to debug the PIC16C54 test version.

Assembler/Compiler version:

MPLAB 3.22.00 development software with MPASM version 1.50

APPENDIX A: SOURCE CODE

```
;***********************************************
;*    Program for the Digitemp Jr. RS-232*
;*    powered PC thermometer       *
;*                                 *
;*    PIC12C508 Version 1.0 written*
;*    7/20/1997 by Michael Kirkhart*
;*                                              *
;***********************************************
         list           p=12C508   ;specifies 12C508 microcontroller
         list           r=DEC      ;specifies decimal radix as default
         list           x=ON       ;specifies to expand macros in listing
         errorlevel     1          ;print warnings and errors only in list file

;************************
;* General system info  *
;************************
;
;Instruction clock frequency = 4.000 MHz
;Non-branching instruction execution time = 1 microsecond
;Fuse settings: Watchdog timer = ON
;               Code Protect = OFF
;               Oscillator = INTRC

         __config   0xff6

;************************
;* CPU Register equates*
;************************
IND0     equ        00             ;indirect file register
RTCC     equ        01             ;real time clock/counter
PC       equ        02             ;program counter
STATUS   equ        03             ;status register
FSR      equ        04             ;file select register (pointer)
OSCCAL   equ        05             ;internal oscillator fine trim register
GPIO     equ        06             ;general purpose I/O register

;**************************************
;* Status register bit definitions *
;**************************************

CARRY    equ        0              ;carry/!borrow flag
DCARRY   equ        1              ;BCD carry/!borrow flag
ZERO     equ        2              ;zero flag
PDOWN    equ        3              ;powerdown flag
TIMEOUT  equ        4              ;watchdog timeout flag

;*******************************
;* GPIO bit definitions *
;* (GPIO port pins are used  *
;*  to communicate to      *
;*  DS1620 digital temp    *
;*  sensor and the external)*
;*  PC's serial port)*
;*******************************

TSDQ     equ        0              ;DS1620 serial data in/out pin (I/O)
TSCLK    equ        1              ;DS1620 serial data clock pin (O)
TSCS     equ        2              ;DS1620 chip select pin (O)
PCTXD    equ        4              ;RS-232 TxD pin (active low)
                                   ;Pin = 1: RxD at PC = 0
                                   ;Pin = 0: RxD at PC = 1
```

```
;****************************************
;* Equates for register files (variables)*
;****************************************

xmtReg    equ        0x07          ;asychronous TxD buffer

flags     equ        0x08          ;register file used as a flag register
                                   ;(see equates section for bit defines)

delay_l   equ        0x09          ;register file for delay value LSB
delay_h   equ        0x0a          ;register file for delay value MSB
dly_tmp   equ        0x0b          ;temp value for delay routine

temp      equ        0x0c          ;temporary register file 0
temp1     equ        0x0d          ;temporary register file 1
eye       equ        0x0e          ;used for loop counting

value     equ        0x0f          ;temperature value
value1    equ         0x10         ;  storage register files


;register files used by math routines
op1_H     equ        0x11          ;16 bit operand1 MSB
op1_L     equ        0x12          ;16 bit operand1 LSB

op2       equ        0x13          ;8 bit operand2

res1_H    equ        0x14          ;16 bit result1 MSB
res1_L    equ        0x15          ;16 bit result1 LSB

count     equ        0x16          ;math routine loop counter
mathtmp   equ        0x17          ;math routine temp register file

R0        equ        0x18          ;file registers
R1        equ        0x19          ;  used by 16 bit binary to
R2        equ        0x1a          ;    packed BCD routine


;****************************************
;* Miscelaneous equates (constants)     *
;****************************************

;Port A, B initialization values
GPINIT    equ         0x01         ;GPIO TRIS initialization value
TSDQOUT   equ        0x00          ;GPIO TRIS value for writing to DS1620
TSDQIN    equ        0x01          ;GPIO TRIS value for reading from DS1620

;DS1620 command values
RDTEMP    equ        0xaa          ;DS1620 read temperature command
WRTH      equ        0x01          ;DS1620 write TH register command
WRTL      equ        0x02          ;DS1620 write TL register command
RDTH      equ        0xa1          ;DS1620 read TH register command
RDTL      equ        0xa2          ;DS1620 read TL register command
STRTCNV   equ        0xee          ;DS1620 start temperature conversion command
STOPCNV   equ        0x22          ;DS1620 stop temperature conversion command
WRCONFG   equ        0x0c          ;DS1620 write config register command
RDCONFG   equ        0xac          ;DS1620 read config register command

;DS1620 config register initialization value
TSCFG     equ        0x03          ;DS1620 config register initialization value
                                   ;(CPU, 1SHOT bits set)

;bit defines for ‘flags’ register file
NEGTEMP   equ        0             ;flag that indicates temperature is negative
NEGSIGN   equ        1             ;flag that indicates negative sign is to be
                                   ;  displayed
```

```
;delay constants for 1 millisecond delay using delay routine
ONEMS_H   equ        0x03          ;
ONEMS_L   equ        0xe8          ;

;delay constants for 10 millisecond delay using delay routine
TENMS_H   equ        0x27          ;
TENMS_L   equ        0x10          ;

;delay constants for one asynchronous bit time at 9600 baud
;using delay routine (104 microseconds - 4 cycles to load
; constants in delay registers - 8 cycles to set output = 92 microseconds)
B9600_H   equ        0x00          ;
B9600_L   equ        0x5c          ;

;************************
;* Macro definitions    *
;************************

CLC     macro                  ;this macro will clear the C flag
        bcf     STATUS,CARRY
        endm

SEC     macro                  ;this macro will set the C flag
        bsf     STATUS,CARRY
        endm

SCC     macro                  ;used after an instruction that affects the C
        btfsc   STATUS,CARRY   ; flag, this macro will skip the next
        endm                   ; instruction if the C flag is clear

SCS     macro                  ;used after an instruction that affects the C
        btfss   STATUS,CARRY   ; flag, this macro will skip the next
        endm                   ; instruction if the C flag is set

SLT     macro                  ;used after a subtract instruction, this macro
        btfsc   STATUS,CARRY   ; will skip the next instruction if the result
        endm                   ; of the subtraction is < 0

SGE     macro                  ;used after a subtract instruction, this macro
        btfss   STATUS,CARRY   ; will skip the next instruction if the result
        endm                   ; of the subtraction is >= 0

SEQ     macro                  ;used after an instruction that affects the Z
        btfss   STATUS,ZERO    ; flag, this macro will skip the next
        endm                   ; instruction if a result is zero

SNE     macro                  ;used after an instruction that affects the Z
        btfsc   STATUS,ZERO    ; flag, this macro will skip the next
        endm                   ; instruction if a result is non-zero


;************************
;* Start of program*
;************************
; actual reset vector - instruction at address 0x1ff was movlw XX, where
; XX is the calibration value to be copied into the OSCCAL register

         org        0             ;start of program memory
         movwf      OSCCAL        ;calibrate on-chip oscillator
         goto       start         ;jump to start of program

;*******************************
;* Subroutines                  *
;* These must be located in the*
;* lower 256 bytes of program*
```

```
;* memory                          *
;**********************************

;************************
;* DS1620 Routines*
;************************

;**********************************
;* Routine to write instruction *
;* (8 bit) to DS1620            *
;*                              *
;* Calling convention:          *
;*                              *
;* w = instruction to be        *
;* written to DS1620            *
;*                              *
;* Returns: 0 in W register     *
;*                              *
;* Routine modifies W,eye,temp  *
;*                              *
;* NOTE: This routine switches  *
;* the DS1620 chip select from  *
;* low to high, but leaves it   *
;* in a high state so that a    *
;* read or write access to the  *
;* part can be completed        *
;* properly.  If a start or     *
;* stop conversion instruction  *
;* is written (these 2          *
;* instructions require no      *
;* additional read or write     *
;* access), the calling routine *
;* MUST clear the DS1620 chip   *
;* select line by doing a:      *
;* bcf GPIO,TSCS        *
;**********************************

TSWrInst
         movwf      temp           ;save instruction in temp register
         movlw      TSDQOUT        ;set TSDQ pin
         tris       GPIO           ;  as output
         bcf        GPIO,TSCLK     ;clear TSCLK pin and
         nop                       ;  wait one instruction cycle
         bsf        GPIO,TSCS      ;set TSCS pin
         movlw      8              ;initialize
         movwf      eye            ;  loop counter
TSWrI1   rrf        temp           ;rotate LSB into carry bit
         btfss      STATUS,CARRY   ;is it set?
         goto       TSWrI2         ;  no - goto TSWrI2
         bsf        GPIO,TSDQ      ;set TSDQ pin
         goto       TSWrI3         ;  and goto TSWrI3
TSWrI2   bcf        GPIO,TSDQ      ;clear TSDQ pin
TSWrI3   bsf        GPIO,TSCLK     ;toggle
         bcf        GPIO,TSCLK     ;  TSCLK pin
         decfsz     eye            ;have we shifted all 8 bits out?
         goto       TSWrI1         ;  no - go thru loop again
         retlw      0              ;return from subroutine


;**********************************
;* Routine to write data        *
;* (9 bit) to DS1620            *
;*                              *
;* Calling convention:          *
;*                              *
;* temp = LSB of data to be     *
```

```
;* written to DS1620            *
;*                              *
;* temp1 = MSB of data to be    *
;* written to DS1620 (only      *
;* least significant bit is     *
;* used)                        *
;*                              *
;* Returns: 0 in W register     *
;*                              *
;* Routine modifies W,eye,temp, *
;* temp1                        *
;******************************


TSWrData
         movlw      TSDQOUT         ;set TSDQ pin
         tris       GPIO            ;  as output
         movlw      9               ;initialize
         movwf      eye             ;  loop counter
TSWrD1   rrf        temp1           ;rotate 16 bit value (temp1 & temp) right
         rrf        temp            ;  once, placing LSB into carry bit
         btfss      STATUS,CARRY    ;is it set?
         goto       TSWrD2          ;  no - goto TSWrI2
         bsf        GPIO,TSDQ       ;set TSDQ pin
         goto       TSWrD3          ;  and goto TSWrI3
TSWrD2   bcf        GPIO,TSDQ       ;clear TSDQ pin
TSWrD3   bsf        GPIO,TSCLK      ;toggle
         bcf        GPIO,TSCLK      ;  TSCLK pin
         decfsz     eye             ;have we shifted all 9 bits out?
         goto       TSWrD1          ;  no - go thru loop again
         bcf        GPIO,TSCS       ;clear TSCS pin
         retlw      0               ;return from subroutine


;******************************
;* Routine to read data         *
;* (9 bit) from DS1620          *
;*                              *
;* Calling convention:          *
;*                              *
;* Returns: 0 in W register     *
;* temp = LSB of data           *
;* temp1 = MSB of data          *
;*                              *
;* Routine modifies W,eye,temp, *
;* temp1                        *
;******************************

TSRdData
         movlw      TSDQIN          ;set TSDQ pin
         tris       GPIO            ;  as an input
         movlw      9               ;initialize
         movwf      eye             ;  loop counter
         clrf       temp            ;clear out temp and
         clrf       temp1           ;  temp1
TSRdD1   bcf        GPIO,TSCLK      ;clear TSCLK
         bcf        STATUS,CARRY    ;clear out carry bit before doing right shift
         rrf        temp1           ;rotate 16 bit value (temp1 & temp) right
         rrf        temp            ;  once
         btfsc      GPIO,TSDQ       ;is TSDQ pin high?
         bsf        temp1,0         ;  yes - set LSB of temp1
         bsf        GPIO,TSCLK      ;set TSCLK pin
         decfsz     eye             ;have we shifted all 9 bits in?
         goto       TSRdD1          ;  no - go thru loop again
         bcf        GPIO,TSCS       ;clear TSCS pin
         retlw      0               ;return from subroutine
```

```
;*************************************
;* Asynchronous serial transmit*
;* byte routine                      *
;*                                   *
;* Calling convention:               *
;* xmtReg = data to be transmitted*
;*                                   *
;* Returns: 0 in W register*
;*                                   *
;* Modifies: eye, xmitReg*
;*                                   *
;* Modified version of transmit code*
;* found in Microchip AN510*
;*************************************

asyncTx
         movlw      9              ;Assume XmtReg contains data to be Xmted
         movwf      eye            ;1 start bit + 8 data bits = 9 bits
         bsf        GPIO,PCTXD     ;Send Start Bit (active low)
X_next   movlw      B9600_L        ;delay
         movwf      delay_l        ; for
         movlw      B9600_H        ;  one
         movwf      delay_h        ;   bit
         call       delay          ;    time
         rrf        xmtReg         ;Rotate next bit to be sent into carry bit
         SCC                        ;Test the bit to be transmitted
         bcf        GPIO,PCTXD     ;Bit is a one (active low)
         SCS                       ;
         bsf        GPIO,PCTXD     ;Bit is zero (active low)
         decfsz     eye            ;If count = 0, then transmit a stop bit
         goto       X_next         ;transmit next bit

X_Stop   bcf        GPIO,PCTXD     ;Send Stop Bit (active low)
         movlw      B9600_L        ;delay
         movwf      delay_l        ; for
         movlw      B9600_H        ;  one
         movwf      delay_h        ;   bit
         call       delay          ;    time
         retlw      0              ;return from subroutine

;************************
;*   Math routines      *
;************************

;***********************************************
;* 8X8 Multiply routine              *
;*                                   *
;* Input: op1_L = 8 bit multiplicand            *
;*        op2 = 8 bit multiplier                *
;*                                   *
;* Output: res1_H = MSB of 16 bit result        *
;*         res1_L = LSB of 16 bit result        *
;*                                   *
;* Obtained from Microchip AN526*
;***********************************************

mpy8X8   clrf    res1_H
         clrf    res1_L
         movlw   8
         movwf   count
         movf    op1_L,w
         bcf     STATUS,CARRY    ; Clear the carry bit in the status Reg.
loop     rrf     op2
         btfsc   STATUS,CARRY
```

```
         addwf   res1_H
         rrf     res1_H
         rrf     res1_L
         decfsz  count
         goto    loop
         retlw   0


;*********************************************
;* 16 bit binary to 5 digit packed BCD routine  *
;*                                   *
;* Input: res1_H = MSB of 16 bit value          *
;*        res1_L = LSB of 16 bit value          *
;*                                   *
;* Output: R0 contains (0), 10^4 digit          *
;*         R1 contains the 10^3, 10^2 digits    *
;*         R2 contains the 10^1, 10^0 digits    *
;*                                              *
;* Obtained from Microchip AN526*
;*********************************************


B2_BCD   bcf     STATUS,CARRY    ; clear the carry bit
         movlw   16
         movwf   count
         clrf    R0
         clrf    R1
         clrf    R2
loop16   rlf     res1_L
         rlf     res1_H
         rlf     R2
         rlf     R1
         rlf     R0

         decfsz  count
         goto    adjDEC
         retlw   0

adjDEC   movlw   R2
         movwf   FSR
         call    adjBCD
         movlw   R1
         movwf   FSR
         call    adjBCD
         movlw   R0
         movwf   FSR
         call    adjBCD
         goto    loop16

adjBCD   movlw   3
         addwf   IND0,w
         movwf   mathtmp
         btfsc   mathtmp,3       ; test if result > 7
         movwf   IND0
         movlw   0x30
         addwf   IND0,w
         movwf   mathtmp
         btfsc   mathtmp,7       ; test if result > 7
         movwf   IND0            ; save as MSD
         retlw   0

;*********************************************
;* Routine to generate a time delay in *
;* multiples of 10 milliseconds from 1 ms to*
;* 2.55s                             *
;*                                   *
```

```
;* Input: W = delay length in tens of*
;*             tens of milliseconds*
;*                                  *
;* Output: W = 0                    *
;*                                  *
;* Calls: delay                     *
;*                                  *
;* Uses: delay_h, delay_l, eye, W*
;************************************************

L_delay
         movwf      eye             ;set loop count
Ldloop   clrwdt                     ;clear the watchdog timer
         movlw      TENMS_L         ;set
         movwf      delay_l         ;  delay
         movlw      TENMS_H         ;    constants for
         movwf      delay_h         ;      10 millisecond delay
         call       delay           ;call delay routine
         decfsz     eye             ;have we gone thru the loop 200 times?
         goto       Ldloop          ;  if not, do it again!
         retlw      0               ;return from subroutine

;************************************************
;* Routine for generating a programmable delay  *
;* (routine written by Philip Doucet - obtained *
;* from Electronics Design - August 8, 1994,    *
;* page 26ES)                                   *
;************************************************

delay
        movlw   0x14            ;subtract minimum # of instructions to
        subwf   delay_l         ;   execute this routine from requested delay
        btfss   STATUS,CARRY    ;check for borrow
        decf    delay_h         ;   and decrement high byte if there was one
        bcf     STATUS,CARRY    ;divide by 4
        rrf     delay_l         ;   to determine how many times to
        bcf     STATUS,CARRY    ;      execute
        rrf     delay_l         ;         delay_l loop
        movf    delay_h         ;check to see if
        btfsc   STATUS,ZERO     ;   delay_h = 0 and
        goto    dly_30          ;      skip delay_h loop if it is
        nop                     ;nop equalizes timing between paths

;delay_h setup and loop

dly_10  movlw   0x3e            ;since each delay_h loop needs 256 cycle, or
        movwf   dly_tmp         ;   40h times thru inner loop of cycles, minus
        nop                     ;      cycle setup, so 40h - 2 = 3eh
        goto    dly_20          ;add a 2 cycle delay
dly_20  nop                     ;inner
        decfsz  dly_tmp         ;   loop
        goto    dly_20          ;      for
        nop                     ;         delay_h
        decfsz  delay_h         ;outer loop
        goto    dly_10          ;  for
        nop                     ;      delay_h

;delay_l setup and loop

dly_30  movf    delay_l         ;if delay_l
        btfsc   STATUS,ZERO     ;   = 0,
        goto    dly_end         ;      skip loop
        nop                     ;
dly_40  nop                     ;loop for
        decfsz  delay_l         ;   delay_l
        goto    dly_40          ;
```

```
        nop                     ;
dly_end retlw   0               ;return from subroutine


;********************
;* Start of program *
;********************

start   movlw   0x0f            ;assign prescaler to WDT, set prescaler to
        option                  ;   128, use internal clock for RTCC
        clrf    GPIO            ;clear GPIO outputs
        movlw   GPINIT          ;set GP0 - GP3 pins
        tris    GPIO            ;   as inputs
        clrf    flags           ;initialize flags register file

        movlw   10              ;wait for 100 milliseconds for
        call    L_delay         ;  power to stablize

; initialize DS1620 temp sensor

        movlw   RDCONFG         ;read
        call    TSWrInst        ;  the DS1620
        call    TSRdData        ;    configuration register into temp
        movf    temp,w          ;read temp into W register, mask out
        andlw   0x03            ;  everything except CPU and ONESHOT bits
        xorlw   0x03            ;are the CPU and ONESHOT bits set?
        SNE                     ;  if so, we don't need to write the config
        goto    noWrCFG         ;    register, so jump to noWrCFG

        movlw   WRCONFG         ;otherwise, send write configuration
        call    TSWrInst        ;    register command to DS1620
        clrf    temp1           ;write
        movlw   TSCFG           ;  configuration register
        movwf   temp            ;    value to
        call    TSWrData        ;      DS1620

noWrCFG  movlw      100             ;delay for
         call       L_delay         ;  1 second (100 * 10ms)

;*********************************************
;* Main program loop (doTemp)      *
;* From this point on, we merely obtain a *
;* temperature reading from the DS1620, convert*
;* it to ASCII, and transmitt it.*
;* This will occur once every second until*
;* power is lost.                  *
;*********************************************

doTemp  bcf     flags,NEGTEMP   ;initialize negative flag
        movlw   STRTCNV         ;initiate temperature
        call    TSWrInst        ;  conversion
        bcf     GPIO,TSCS     ;clear DS1620 chip select line
waitcnv movlw   RDCONFG         ;read
        call    TSWrInst        ;  config
        call    TSRdData        ;    register
        btfss   temp,7          ;is conversion DONE bit set?
        goto    waitcnv         ;  -if not, loop thru again

        movlw   RDTEMP          ;read
        call    TSWrInst        ;  temperature
        call    TSRdData        ;    value

        btfss   temp1,0         ;is MSB of temperature value set?
        goto    posTemp         ;  -if not, temperature is positive - jump
        bsf     flags,NEGTEMP   ;set negative temperature flag
        bsf     flags,NEGSIGN   ;set display negative sign flag
```

```
        decf    temp            ;determine absolute value of temperature
        comf    temp            ;  (convert from 2's complement form)
posTemp movf    temp,w          ;copy temperature value from temporary
        movwf   value           ;  register to value

;since temperature value obtained from the DS1620 is 2 * temp (in degrees C),
;we must multiply the value by 5 to get a value that is 10 * temp in degrees
;C.
        movwf   op1_L           ;move temperature value to multiplicand
        movlw   5               ;move 5 to
        movwf   op2             ;  multiplier
        call    mpy8X8          ;do multiplication
        movf    res1_H,w        ;copy
        movwf   value1          ;  results
        movf    res1_L,w        ;    to
        movwf   value           ;      value, value1
        call    B2_BCD          ;convert value to BCD

dispTmp btfss   flags,NEGSIGN   ;should we display negative sign?
        goto    noNeg           ;  if no, display 10^3 digit
        movlw   '-'             ;otherwise, write negative sign
        movwf   xmtReg          ;transmit negative sign
        call    asyncTx         ; out RS-232 port
        goto    digit2          ;go to digit2

noNeg   swapf   R1,w            ;get MSN from
        andlw   0x0f            ;  R1 (10^3 digit)
        SNE                     ;is it = 0?
        goto    digit2          ;  if yes - skip to 10^2 digit
        iorlw   0x30            ;ASCII value for digit = digit value + 0x30
        movwf   xmtReg          ;transmit digit
        call    asyncTx         ; out RS-232 port

digit2  movf    R1,w            ;load both the 10^3, 10^2 digits
        SNE                     ;are they both 0?
        goto    digit3; if yes - skip to 10^1 digit
        andlw   0x0f            ;mask out 10^3 digit, leaving 10^2 digit
        iorlw   0x30            ;ASCII value for digit = digit value + 0x30
        movwf   xmtReg          ;transmit digit
        call    asyncTx         ; out RS-232 port

digit3  swapf   R2,w            ;get the
        andlw   0x0f            ;  10^1 digit
        iorlw   0x30            ;ASCII value for digit = digit value + 0x30
        movwf   xmtReg          ;transmit digit
        call    asyncTx         ; out RS-232 port

decipt  movlw   '.'             ;
        movwf   xmtReg          ;transmit decimal point
        call    asyncTx         ; out RS-232 port

digit4  movf    R2,w            ;get the
        andlw   0x0f            ;  10^0 digit
        iorlw   0x30            ;ASCII value for digit = digit value + 0x30
        movwf   xmtReg          ;transmit digit
        call    asyncTx         ; out RS-232 port

        movlw   0x0d            ;transmit
        movwf   xmtReg          ; carriage return
        call    asyncTx         ;  out RS-232 port
        movlw   0x0a            ;transmit
        movwf   xmtReg          ; line feed
        call    asyncTx         ;  out RS-232 port

;wait for 760 milliseconds before displaying next temperature
;Since it takes approximately 240 milliseconds to execute the code in the loop
```

```
;to this point, delaying 760 milliseconds will give us one temperature reading
;transmitted per second
        movlw   76              ;delay for
        call    L_delay         ;  760 milliseconds (76 * 10 milliseconds)

        clrwdt                  ;clear the watchdog timer
        goto    doTemp          ;do the whole loop over again

;****************
;* Reset vector *
;****************
; For 12C508, this location contains movlw XX, where XX is the calibration value
; for the on-board oscillator - thus the real reset vector is at address 0
        org     0x1ff   ;location of “reset” vector

;******************
;* End of program *
;******************
        end

?
```