AppKit:
Using the DS1620 Digital Thermometer/Thermostat

This AppKit shows how to use the Dallas Semiconductor DS1620 Digital Thermometer/ Thermostat chip with PIC microcontrollers and the Parallax BASIC Stamp® single-board computer.

Description

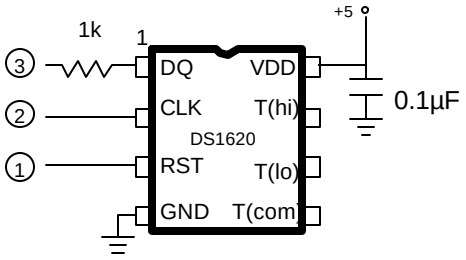
The DS1620 is a complete digital thermometer on a chip, capable of replacing the normal combination of temperature sensor and analog-to-digital converter in most applications. It can measure temperature in units of 0.5° Centigrade (C) from –55° C to +125° C. [In Fahrenheit (°F), units of 0.9° F and a range of –67° F to +257° F.] Temperature measurements are expressed as nine-bit, two's complement numbers. The DS1620 communicates with a microcontroller such as the PIC or Stamp through a three-wire serial connection.

The DS1620 can also operate as a standalone thermostat. A temporary connection to a controller establishes the mode of operation and high/low-temperature setpoints. Thereafter, the chip independently controls three outputs: T(high), which goes active at temperatures above the high-temperature setpoint; T(low), active at temperatures below the low setpoint; and T(com), which goes active at temperatures above the high setpoint, and stays active until the temperature drops below the low setpoint.

Hardware interface

The DS1620 interfaces with controllers through a three-wire connection, consisting of a data input/output line (DQ), a synchronizing clock line (CLK) and a reset/select line (RST).

The figure shows how to connect the DS1620 to the PIC or Stamp for the demo programs. Do not omit the bypass capacitor—not even if you feel that your power supply is solid and well-filtered. Locate that cap as close as practical to the supply leads of the DS1620. Although the 1k resistor is not strictly necessary as long as the firmware is functioning correctly, it's best to leave it in. In the event that both the controller (PIC or Stamp) and the DS1620 try to drive the data line at the same time, the resistor limits the amount of current that can flow between them to a safe value.



	PIC	Stamp
1	ra.0	pin 0
2	ra.1	pin 1
3	ra.2	pin 2

Software interface

From a software standpoint, using the DS1620 boils down to this:

(1) Activate RST by taking it high.
(2) Send an instruction (protocol) to the DS1620 telling it what you want to do.
(3) If you are reading data, shift it into the controller (PIC or Stamp).
(4) If you are writing data, shift it out to the DS1620.
(5) Deactivate RST by taking it low.

The program listings and data sheets show these processes in detail.

Tips for using the DS1620

- Data written to the DS1620 configuration or temperature-setpoint registers is stored in EEPROM. It takes as long as 50 milliseconds (ms) to complete the write. Be sure to program a delay of at least this length before sending further commands to the DS1620.

- The fastest the DS1620 can generate new temperature data is once per second. It does no good to read it at shorter intervals—you'll simply read the value of the previous temperature measurement.

- The DS1620's thermostat outputs can source only 1 mA and sink only 4 mA. You'll need a Darlington-transistor or logic-level MOSFET switch to turn on a decent-sized load.

PIC Program Listing

```
; Program: DS1620.SRC (interface with DS1620 digital thermometer)

; This program presents PIC assembly language subroutines for communicating
; with the DS1620 digital thermometer chip. It also provides tables of
; predefined constants called "protocols" that control the chip's various
; operating modes. This demo configures the DS1620 as a temperature
; sensor slaved to a controller ("CPU" mode) and requests that it perform
; continuous temperature measurements. The '1620 may also be configured as
; a standalone thermostat with setpoint stored in nonvolatile EEPROM.

; ======================== Define I/O Pins ======================
RST             =       ra.0            ; Reset-- 0 = inactive, 1 = active.
CLK             =       ra.1            ; Clock pin.
DQ              =       ra.2            ; Data in/out.
DQin            =       4               ; TRIS ra for DQ = input
DQout           =       0               ; TRIS ra for DQ = output.

; ======================== Define Variables ======================
                org     8               ; RAM above special-function registers.
DSdata          ds      1               ; Byte for I/O with DS1620.
bits            ds      1               ; Number of clock cycles for serial shifts.
flags           ds      1               ; Bit flags.
temp1           ds      1               ; Counter for delays.
temp2           ds      1               ; Counter for delays.
sign            =       flags.0         ; Ninth bit of DSdata for temperature values.
clk9            =       flags.1         ; bit flag: 1= 9-bit xfer, 0= 8-bit xfer.

; ==================== Define DS1620 Constants ==================
; >>> Constants for configuring the DS1620
Rconfig         =       0ACh            ; Protocol for 'Read Configuration.'
Wconfig         =       00Ch            ; Protocol for 'Write Configuration.'
CPU             =       10b             ; Config bit: serial thermometer mode.
NoCPU           =       00b             ; Config bit: standalone thermostat mode.
OneShot         =       01b             ; Config bit: one conversion per start request.
Cont            =       00b             ; Config bit: continuous conversions after start.
; >>> Constants for serial thermometer applications.
StartC          =       0EEh            ; Protocol for 'Start Conversion.'
StopC           =       022h            ; Protocol for 'Stop Conversion.'
Rtemp           =       0AAh            ; Protocol for 'Read Temperature.'
; >>> Constants for programming thermostat functions.
RhiT            =       0A1h            ; Protocol for 'Read High-Temperature Setting.'
WhiT            =       001h            ; Protocol for 'Write High-Temperature Setting.'
RloT            =       0A2h            ; Protocol for 'Read Low-Temperature Setting.'
WloT            =       002h            ; Protocol for 'Write Low-Temperature Setting.'
```

PIC Program Listing (cont.)

```
; ================= Set Up Device and Start Program ===============
                device      pic16c54,xt_osc,wdt_off,protect_off
                reset       start
                org         0

; Main program start.
; To demonstrate communication with the DS1620, we're going to set its
; configuration register to CPU mode and continuous conversion, then
; continuously read the temperature. The results of the temperature readings
; will be written to port RB (eight lsbs) and ra.3 (msb). You can view the readings
; (in binary) by connecting LEDs to these pins through 220-ohm resistors.

start           mov         ra,#0               ; Clear ports
                mov         rb,#0
                mov         !ra,#DQout          ; Port ra: all outputs.
                mov         !rb,#0              ; Output for eight lsbs of temp reading.
                clrb        clk9                ; Clear the ninth-bit clock flag.
                setb        CLK                 ; Set the clock to prepare for comms.
                mov         DSdata,#Wconfig     ; Put write-config protocol into output byte.
                call        Shout               ; Send it to the 1620.
                mov         DSdata,#CPU         ; Configure as thermometer...
                OR          DSdata,#Cont        ; ...continuous conversion.
                call        Shout               ; Send that to the 1620.
                clrb        RST                 ; Deactivate the 1620.
                call        delay               ; Wait for EEPROM programming cycle.
                mov         DSdata,#StartC      ; Start conversions by sending
                call        Shout               ; the start protocol to the 1620.
                clrb        RST                 ; Deactivate the 1620.
:loop           call        delay               ; Short *delay between reads.
                mov         DSdata,#Rtemp       ; Send read-temperature protocol.
                call        Shout
                call        Shin                ; Get the temperature data.
                movb        ra.3,sign           ; Write ninth bit to ra.3.
                mov         rb,DSdata           ; Write lower eight bits to port rb.
                jmp         :loop               ; Repeat forever.

; * Note that the DS1620 can only perform one conversion per second.
; With a 4-MHz clock, this demo routine reads it about five times a
; second. This doesn't do any harm, since the DS1620 provides the
; old reading until a new one is available, but it doesn't do any
; good either! In an actual application, you'll want to read the '1620
; only as often as appropriate.
```

PIC Program Listing (cont.)

```
; ================== DS1620 Communication Routines ================
; >> Note that callable subroutines should normally come _before_ the
; body of the program (or be placed there in program memory using "org"
; directives) to ensure that call destinations fall in the lower 256
; addresses of a program-memory page. In example programs like this
; we frequently break this rule since it makes the program more readable.

; Shout shifts out the 8 bits of DSdata and, optionally, the ninth bit
; stored in sign. Eight-bit transfers are required for sending instructions
; (called protocols) to the DS1620. Nine-bit transfers are required for
; storing temperature data to the 1620's thermostat registers. In either
; case, the internal loop executes 9 times. For an eight-bit transfer
; (set up by storing a 0 to clk9) the last clock pulse is skipped. Always
; looping 9 times has the benefit of leaving the value of DSdata unaltered
; after a call to Shout. The bits are rotated back into their original
; positions.

; >> Note: Since all communications with the DS1620 begin with the controller
; sending a protocol, Shout contains the "setb RST" command required to
; activate the DS1620. Since communications can end with either an output
; (Shout) or an input (Shin), Shout does _not_ deactivate the DS1620. Make
; sure to take care of this detail in your code that uses this routine.

Shout
                mov     !ra,#DQout      ; Set to output.
                mov     bits,#9         ; Set up for 8- or 9-bit transfer.
                movb    c,sign          ; Put bit8 (sign bit) into carry.
:begin          setb    RST             ; Activate the 1620.
:loop           rr      DSdata          ; Rotate bit0 of DSdata into carry.
                clrb    CLK             ; Set up for clock pulse
                movb    DQ,c            ; Move carry bit to input of 1620.
                mov     w,--bits        ; Pulse the clock line for each of
                snz                     ; first 8 bits. On the ninth bit,
                snb     clk9            ; pulse the clock only if clk9 = 1.
                setb    CLK             ; Finish the pulse if conditions are met.
                djnz    bits,:loop      ; Loop 9 times.
                ret
```

PIC Program Listing (cont.)

```
; Shin shifts in data from the 1620. If the data to be received is
; a 9-bit temperature, the lower 8 bits are in DSdata and the 9th (sign)
; bit is in the carry bit. If the data is a configuration byte, ignore the
; extra bit in carry.

; >> Note: When a program receives input from the DS1620, it is always the
; end of the communication, so Shin incorporates the command "clrb RST"
; to deactivate the DS1620.

Shin
              mov     !ra,#DQin       ; Set DQ to input.
              mov     bits,#9         ; Nine-bit transfer.
              clc                     ; Clear carry bit.
:loop         rr      DSdata          ; Move carry into bit7, shift bits right.
              clrb    CLK             ; Clock in the bit on falling edge.
              movb    c,DQ            ; Get the bit
              setb    CLK             ; Finish the clock pulse.
              djnz    bits,:loop      ; Loop 9 times.
              clrb    RST             ; Deactivate the 1620.
              ret

; General-purpose delay routine (200+ ms at 4 MHz).
; Not required by DS1620 routines per se, but used to wait for EEPROM
; programming cycles to finish (50 ms max) and to wait an interval
; between temperature readings.

delay         djnz    temp1,delay
              djnz    temp2,delay
              ret
```

BASIC Stamp I (BS1-IC) and BASIC Stamp Ver. D Program Listing: Thermometer

```
' Program: DS1620.BS1
' This program interfaces the DS1620 Digital Thermometer to the
' BASIC Stamp. Input and output subroutines can be combined to
' set the '1620 for thermometer or thermostat operation, read
' or write nonvolatile temperature setpoints and configuration
' data.

' ==================== Define Pins and Variables ================
SYMBOL  DQp = pin2              ' Data I/O pin.
SYMBOL  DQn = 2                 ' Data I/O pin _number_.
SYMBOL  CLKn = 1                ' Clock pin number.
SYMBOL  RSTn = 0                ' Reset pin number.
SYMBOL  DSout = b0              ' Use bit-addressable byte for DS1620 output.
SYMBOL  DSin = w0               ' "   "   "          word "   "      input.
SYMBOL  clocks = b2             ' Counter for clock pulses.

' ==================== Define DS1620 Constants ==================
' >>> Constants for configuring the DS1620
SYMBOL  Rconfig = $AC           ' Protocol for 'Read Configuration.'
SYMBOL  Wconfig = $0C           ' Protocol for 'Write Configuration.'
SYMBOL  CPU = %10               ' Config bit: serial thermometer mode.
SYMBOL  NoCPU = %00             ' Config bit: standalone thermostat mode.
SYMBOL  OneShot = %01           ' Config bit: one conversion per start request.
SYMBOL  Cont = %00              ' Config bit: continuous conversions.
' >>> Constants for serial thermometer applications.
SYMBOL  StartC = $EE            ' Protocol for 'Start Conversion.'
SYMBOL  StopC = $22             ' Protocol for 'Stop Conversion.'
SYMBOL  Rtemp = $AA             ' Protocol for 'Read Temperature.'
' >>> Constants for programming thermostat functions.
SYMBOL  RhiT = $A1              ' Protocol for 'Read High-Temperature Setting.'
SYMBOL  WhiT = $01              ' Protocol for 'Write High-Temperature Setting.'
SYMBOL  RloT = $A2              ' Protocol for 'Read Low-Temperature Setting.'
SYMBOL  WloT = $02              ' Protocol for 'Write Low-Temperature Setting.'

' ==================== Begin Program ============================
' Start by setting initial conditions of I/O lines.
low RSTn                        ' Deactivate the DS1620 for now.
high CLKn                       ' Initially high as shown in DS specs.
pause 100                       ' Wait a bit for things to settle down.
```

BASIC Stamp I and Ver. D Program Listing: Thermometer (cont.)

```
' Configure the DS1620 for thermometer operation. The configuration
' register is nonvolatile EEPROM. You only need to configure
' the DS1620 once. It will retain those configuration settings
' until you change them--even with power removed. To conserve
' Stamp program memory, you can preconfigure the DS1620, then
' remove the configuration code from your final program. (You'll
' still have to issue a start-conversion command, though.)

let DSout=Wconfig                ' Put write-config command into output byte.
gosub Shout                      ' And send it to the DS1620.
let DSout=CPU+Cont               ' Thermometer, continuous conversion.
gosub Shout                      ' Send to DS1620.
low RSTn                         ' Deactivate '1620.
pause 50                         ' Wait 50ms for EEPROM programming cycle.
let DSout=StartC                 ' Now, start the conversions by
gosub Shout                      ' sending the start protocol to DS1620.
low RSTn                         ' Deactivate '1620.

' The loop below continuously reads the latest temperature data from
' the DS1620. The '1620 performs one temperature conversion per second.
' If you read it more frequently than that, you'll get the result
' of the most recent conversion. The '1620 data is a 9-bit number
' in units of 0.5 deg. C. See the ConverTemp subroutine below.

Again:
  pause 1000                     ' Wait 1 second for conversion to finish.
  let DSout=Rtemp                ' Send the read-temperature opcode.
  gosub Shout
  gosub Shin                     ' Get the data.
  low RSTn                       ' Deactivate the DS1620.
  gosub ConverTemp               ' Convert temperature reading to absolute.
  gosub DisplayF                 ' Display in degrees F.
  gosub DisplayC                 ' Display in degrees C.
goto Again
```

BASIC Stamp I and Ver. D Program Listing: Thermometer (cont.)

```
' ==================== DS1620 I/O Subroutines ==================
' Subroutine: Shout
' Shift bits out to the DS1620. Sends the 8 bits stored in DSout
' (b0). Note that Shout activates the DS1620, since all trans-
' actions begin with the Stamp sending a protocol (command). It does
' not deactivate the DS1620, though, since many transactions either
' send additional data, or receive data after the initial protocol.
' Note that Shout destroys the contents of DSout in the process of
' shifting it. If you need to save this value, copy it to another
' register.
Shout:
high RSTn                       ' Activate DS1620.
output DQn                      ' Set to output to send data to DS1620.
for clocks = 1 to 8             ' Send 8 data bits.
  low CLKn                      ' Data is valid on rising edge of clock.
  let DQp = bit0                ' Set up the data bit.
  high CLKn                     ' Raise clock.
  let DSout=DSout/2             ' Shift next data bit into position.
next                            ' If less than 8 bits sent, loop.
return                          ' Else return.

' Subroutine: Shin
' Shift bits in from the DS1620. Reads 9 bits into the lsbs of DSin
' (w0). Shin is written to get 9 bits because the DS1620's temperature
' readings are 9 bits long. If you use Shin to read the configuration
' register, just ignore the 9th bit. Note that DSin overlaps with DSout.
' If you need to save the value shifted in, copy it to another register
' before the next Shout.
Shin:
input DQn                       ' Get ready for input from DQ.
for clocks = 1 to 9             ' Receive 9 data bits.
  let DSin = DSin/2             ' Shift input right.
  low CLKn                      ' DQ is valid after falling edge of clock.
  let bit8 = DQp                ' Get the data bit.
  high CLKn                     ' Raise the clock.
next                            ' If less than 9 bits received, loop.
return                          ' Else return.
```

BASIC Stamp I and Ver. D Program Listing: Thermometer (cont.)

```
' ================= Data Conversion/Display Subroutines ===============
' Subroutine: ConverTemp
' The DS1620 has a range of -55 to +125 degrees C in increments of 1/2
' degree. It's awkward to work with negative numbers in the Stamp's
' positive-integer math, so I've made up a temperature scale called
' DSabs (DS1620 absolute scale) ranging from 0 (-55C) to 360 (+125C).
' Internally, your program can do its math in DSabs, then convert to
' degrees F or C for display.

ConverTemp:
if bit8 = 0 then skip                   ' If temp > 0 skip "sign extension" procedure.
  let w0 = w0 | $FE00                   ' Make bits 9 through 15 all 1s to make a
                                        ' 16-bit two's complement number.
skip:
  let w0 = w0 + 110                     ' Add 110 to reading and return.
return

' Subroutine: DisplayF
' Convert the temperature in DSabs to degrees F and display on the
' PC screen using debug.

DisplayF:
let w1 = w0*9/10                        ' Convert to degrees F relative to -67.
if w1 < 67 then subzF                   ' Handle negative numbers.
  let w1 = w1-67
  Debug #w1, " F",cr
return
subzF:
  let w1 = 67-w1                        ' Calculate degrees below 0.
  Debug "-",#w1," F",cr                 ' Display with minus sign.
return
```

BASIC Stamp I and Ver. D Program Listing: Thermometer (cont.)

```
' Subroutine: DisplayC
' Convert the temperature in DSabs to degrees C and display on the
' PC screen using debug.

DisplayC:
let w1 = w0/2                          ' Convert to degrees C relative to -55.
if w1 < 55 then subzC                  ' Handle negative numbers.
  let w1 = w1-55
  Debug #w1, " C",cr
return
subzC:
  let w1 = 55-w1                       ' Calculate degrees below 0.
  Debug "-",#w1," C",cr                ' Display with minus sign.
return
```

BASIC Stamp I (BS1-IC) and BASIC Stamp Ver. D Program Listing: Thermostat

```
' Program: DS_STAT.BS1
' This program interfaces the DS1620 Digital Thermometer to the
' BASIC Stamp to configure it for thermostat operation. A PC
' running terminal software, should be connected to the Stamp
' with data out to pin 4 (through a 22k resistor) and data in
' to pin 3:

'      Function      DB25 Pin      DB9 Pin       Stamp Pin
'        GND           7             5            GND
'      Transmit        2             3             4
'      Receive         3             2             3

' Hardware handshaking must be disabled in the terminal software.
' Communication format is 2400 baud, no parity, 8 data bits,
' 1 stop bit. The Stamp prompts the user for the high and low
' temperature setpoints, then copies these to the registers of the
' DS1620. The setpoints _must_ be preceded by the appropriate sign
' (+ or -). The program does not assume that "78" means "+78."
' Once the DS1620 is programmed, the circuit should be turned
' off, and the DS1620 removed for installation in its standalone
' thermostat application.

' ==================== Define Pins and Variables ================
SYMBOL  ComIn = 4                ' Serial communication input pin.
SYMBOL  ComOut = 3               ' Serial communication output pin.
SYMBOL DQp = pin2                ' Data I/O pin.
SYMBOL  DQn = 2                  ' Data I/O pin _number_.
SYMBOL  CLKn = 1                 ' Clock pin number.
SYMBOL  RSTn = 0                 ' Reset pin number.
SYMBOL  DSout = w0               ' Use bit-addressable word for DS1620 output.
SYMBOL  DSin = w0                ' "   "   "          word "   "      input.
SYMBOL  clocks = b2              ' Counter for clock pulses.
SYMBOL  config = b3              ' Copy of the DS1620 configuration bits.
SYMBOL  setTemp = w2             ' Copy of the temperature setting.
SYMBOL  comData = b6             ' Serial input data.
SYMBOL  index = b7               ' Temporary counter used in prompts.

' ==================== Define DS1620 Constants ==================
' >>> Constants for configuring the DS1620
SYMBOL  Rconfig = $AC            ' Protocol for 'Read Configuration.'
SYMBOL  Wconfig = $0C            ' Protocol for 'Write Configuration.'
SYMBOL  CPU = %10                ' Config bit: serial thermometer mode.
SYMBOL  NoCPU = %00              ' Config bit: standalone thermostat mode.
SYMBOL  OneShot = %01            ' Config bit: one conversion per start request.
SYMBOL  Cont = %00               ' Config bit: continuous conversions after start.
```

BASIC Stamp I and Ver. D Program Listing: Thermostat (cont.)

```
' >>> Constants for serial thermometer applications.
SYMBOL  StartC  = $EE                  ' Protocol for 'Start Conversion.'
SYMBOL  StopC   = $22                  ' Protocol for 'Stop Conversion.'
SYMBOL  Rtemp   = $AA                  ' Protocol for 'Read Temperature.'
' >>> Constants for programming thermostat functions.
SYMBOL RhiT = $A1                      ' Protocol for 'Read High-Temperature Setting.'
SYMBOL  WhiT = $01                     ' Protocol for 'Write High-Temperature Setting.'
SYMBOL  RloT = $A2                     ' Protocol for 'Read Low-Temperature Setting.'
SYMBOL  WloT = $02                     ' Protocol for 'Write Low-Temperature Setting.'

' ==================== Begin Program ============================
' Start by setting initial conditions of I/O lines.
Start:
low RSTn                               ' Deactivate the DS1620 for now.
high CLKn                              ' Initially high as shown in DS specs.

' Next, send prompts and gather data for configuration.
promptUser:
Serout comOut,N2400,(13,"Hi:")         ' Prompt for the high setpoint.
For index = 0 to 1                                  ' Prepare to get both setpoints.
  let setTemp = 0                      ' Clear the high- and low-temp registers.
  Serin comIn,N2400,comData            ' Get the sign of the temperature (+/-).
  if comData = "+" then skip1          ' Prepare for positive numbers.
  if comData <> "-" then Err           ' If not negative then what? Error..
  let setTemp = 256                    ' If negative, set the sign bit.
skip1:
  Serin comIn,N2400,#comData           ' Now get the numeric part of the entry.
  let comData = comData * 2            ' Times 2 for 1620's 0.5-degree scale.
  if setTemp = 0 then skip2            ' Skip next line if positive.
  let comData=comData^255+1            ' If negative, take two's complement.
skip2:
  let setTemp = setTemp | comData      ' Merge entry value into setTemp.
  let DSout = WloT                     ' First entry is hi temp, 2nd is lo temp.
  if index = 1 then skip3:
  let DSout = WhiT
skip3:
  gosub Shout                          ' Send protocol..
  let DSout = setTemp                  ' ..and the temperature setting..
  gosub Shout                          ' ..to the DS1620. (Two Shouts are needed to
  gosub Shout                          ' send 9-bit temp data to the DS1620.)
  low RSTn                             ' Deactivate '1620.
  pause 100                            ' Let EEPROM self-program.
  let DSout = RloT                     ' Read back the value written to '1620.
```

BASIC Stamp I and Ver. D Program Listing: Thermostat (cont.)

```
  if index = 1 then skip4:
  let DSout = RhiT
skip4:
  gosub Shout                          ' Send read protocol to the DS1620.
  gosub Shin                           ' Get the data.
  low RSTn                             ' Deactivate '1620.
  if DSin = setTemp then skip5         ' Compare to the value written.
  goto Fail                            ' If they're different, print "fail" msg.
skip5:
  if index <> 0 then skip6             ' Now set up to prompt for lo temp.
  Serout comOut,N2400,(13,"Lo:")
next index
skip6:
  let config = NoCPU+OneShot           ' Temps done: configure the '1620.
skip7:
  let DSout = Wconfig
  gosub Shout
  let DSout = config
  gosub Shout
  low RSTn
  pause 100
  let DSout = Rconfig                  ' Read back the configuration and..
  gosub Shout
  gosub Shin
  low RSTn
  let DSin = DSin & %11                ' ..compare it to value sent.
  if DSin = config then skip8
  goto Fail                            ' If they're different, print "fail" msg.
skip8:
  Serout comOut,N2400,(" OK")          ' Else, print "OK".
end                                    ' Stop until power cycles.

Err:
  Serout comOut,N2400,(" ERR")         ' Signal data-entry error.
  end

Fail:
  Serout comOut,N2400,(13," FAIL")     ' Signal 1620 write error.
  end
```

BASIC Stamp I and Ver. D Program Listing: Thermostat (cont.)

```
' ==================== DS1620 I/O Subroutines ==================
' Subroutine: Shout
' Shift bits out to the DS1620. Sends the 8 bits stored in DSout
' (b0). Note that Shout activates the DS1620, since all trans-
' actions begin with the Stamp sending a protocol (command). It does
' not deactivate the DS1620, though, since many transactions either
' send additional data, or receive data after the initial protocol.
' Note that Shout destroys the contents of DSout in the process of
' shifting it. If you need to save this value, copy it to another
' register.
Shout:
high RSTn                       ' Activate DS1620.
output DQn                      ' Set to output to send data to DS1620.
for clocks = 1 to 8             ' Send 8 data bits.
  low CLKn                      ' Data is valid on rising edge of clock.
  let DQp = bit0                ' Set up the data bit.
  high CLKn                     ' Raise clock.
  let DSout=DSout/2             ' Shift next data bit into position.
next                            ' If less than 8 bits sent, loop.
return                          ' Else return.

' Subroutine: Shin
' Shift bits in from the DS1620. Reads 9 bits into the lsbs of DSin
' (w0). Shin is written to get 9 bits because the DS1620's temperature
' readings are 9 bits long. If you use Shin to read the configuration
' register, just ignore the 9th bit. Note that DSin overlaps with DSout.
' If you need to save the value shifted in, copy it to another register
' before the next Shout.
Shin:
input DQn                       ' Get ready for input from DQ.
for clocks = 1 to 9             ' Receive 9 data bits.
  let DSin = DSin/2             ' Shift input right.
  low CLKn                      ' DQ is valid after falling edge of clock.
  let bit8 = DQp                ' Get the data bit.
  high CLKn                     ' Raise the clock.
next                            ' If less than 9 bits received, loop.
return                          ' Else return.
```

BASIC Stamp II (BS2-IC) Program Listing: Thermometer

```
' Program: DS1620.BS2 (interface DS1620 digital thermometer to BS2)

' This program interfaces the DS1620 Digital Thermometer to
' the BS2. Input and output routines can be combined to set
' the '1620 for thermometer or thermostat operation, read
' or write nonvolatile temperature setpoints and configuration
' data. In addition to using the BS2's new Shiftin and Shiftout
' instructions to communicate with the 1620, this program uses
' new math and display operators that work with signed integers.
' This makes it relatively easy to convert between degrees C and
' F and to display both positive and negative temperature
' readings. Note that after math operations on negative numbers
' it's necessary to "extend the sign bits." All this means is
' setting all of the bits to the left of actual value to 1s.

' Also note the use of the new */ (pronounced 'star-slash')
' operator. This works like multiplying by an integer (0-255)
' and a fraction (in units of 1/256). For example, to multiply
' 17 by Pi (approx 3.1416) would be written "17 */ $0324."
' The second value is written in hex to emphasize that it's being
' split into bytes: $03 is the integer and $24/$100 is the fraction.
' In the C-to-F conversion below, we multiply the C value by 1.8
' with "*/ $01CC" since $CC/$FF (204/256) = 0.8.

' ===================== Define Pins and Variables ================
DQ              con     2               ' Pin 2 <=> DQ.
CLK             con     1               ' Pin 1  => CLK.
RST             con     0               ' Pin 0  => RST (high = active).
DSdata          var     word            ' Word variable to hold 9-bit data.
Sign            var     DSdata.bit8     ' Sign bit of raw temperature data.
T_sign          var     bit             ' Saved sign bit for converted temperature.

' ===================== Define DS1620 Constants ==================
' >>> Constants for configuring the DS1620
Rconfig         con     $AC             ' Protocol for 'Read Configuration.'
Wconfig         con     $0C             ' Protocol for 'Write Configuration.'
CPU             con     %10             ' Config bit: serial thermometer mode.
NoCPU           con     %00             ' Config bit: standalone thermostat mode.
OneShot         con     %01             ' Config bit: one conversion per start request.
Cont            con     %00             ' Config bit: continuous conversions after start.
' >>> Constants for serial thermometer applications.
StartC          con     $EE             ' Protocol for 'Start Conversion.'
StopC           con     $22             ' Protocol for 'Stop Conversion.'
Rtemp           con     $AA             ' Protocol for 'Read Temperature.'
' >>> Constants for programming thermostat functions.
RhiT            con     $A1             ' Protocol for 'Read High-Temperature Setting.'
WhiT            con     $01             ' Protocol for 'Write High-Temperature Setting.'
RloT            con     $A2             ' Protocol for 'Read Low-Temperature Setting.'
WloT            con     $02             ' Protocol for 'Write Low-Temperature Setting.'
```

BASIC Stamp II Program Listing: Thermometer (cont.)

```
' ==================== Begin Program ==========================

low RST                         ' Deactivate '1620 for now.
high CLK                        ' Put clock in starting state.
pause 100                       ' Let things settle down a moment.
high RST                        ' Activate the '1620 and set it for continuous..
shiftout DQ,CLK,lsbfirst,[Wconfig,CPU+Cont]          ' ..temp conversions.
low RST                         ' Done--deactivate.
pause 50                        ' Wait for the EEPROM to self-program.
high RST                        ' Now activate it again and send the..
shiftout DQ,CLK,lsbfirst,[StartC]               ' Send start-conversion protocol.
low RST                         ' Done--deactivate.

' The loop below continuously reads the latest temperature from
' the DS1620.
again:
  pause 1000                                    ' Wait a second between readings.
  high RST                                      ' Activate the '1620.
  shiftout DQ,CLK,lsbfirst,[Rtemp]              ' Request to read temperature.
  shiftin DQ,CLK,lsbpre,[DSdata\9]              ' Get the temperature reading.
  low RST
  T_sign = Sign                                 ' Save the sign bit of the reading.
  DSdata = DSdata/2                             ' Scale reading to whole degrees C.
  if T_sign = 0 then no_neg1
    DSdata = DSdata | $FF00                     ' Extend sign bits for negative temps.
no_neg1:
  debug SDEC DSdata," degrees C",cr             ' Show signed temperature in C.
    DSdata = (DSdata */ $01CC)                  ' Multiply by 1.8.
  if T_sign = 0 then no_neg2                    ' If negative, extend sign bits.
    DSdata = DSdata | $FF00
no_neg2:
    DSdata = DSdata + 32                                ' Complete the conversion.
    debug SDEC DSdata," degrees F",cr           ' Show signed temperature in F.
goto again                                      ' Repeat forever.
```

DS1620

PRELIMINARY

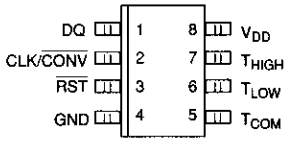
DALLAS SEMICONDUCTOR

DS1620
Digital Thermometer and Thermostat

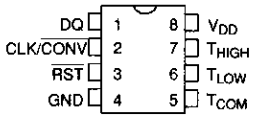
FEATURES

- Requires no external components
- Measures temperatures from –55°C to +125°C in 0.5°C increments. Fahrenheit equivalent is –67°F to 257°F in 0.9°F increments
- Temperature is read as a 9–bit value
- Converts temperature to digital word in 1 second
- Thermostatic settings are user–definable and non–volatile
- Data is read from/written via a 3–wire serial interface (CLK, DQ, $\overline{RST}$)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system
- 8–pin DIP or SOIC package

PIN ASSIGNMENT



DS1620S
8–PIN SOIC (200 MIL)
See Mech. Drawing
Pg. 483



DS1620
8–PIN DIP
See Mech. Drawing
Pg. 480

PIN DESCRIPTION

DQ	–	3–Wire Input/Output
CLK/$\overline{CONV}$	–	3–Wire Clock Input and Standalone Convert Input
$\overline{RST}$	–	3–Wire Reset Input
GND	–	Ground
T_{HIGH}	–	High Temperature Trigger
T_{LOW}	–	Low Temperature Trigger
T_{COM}	–	High/Low Combination Trigger
V_{DD}	–	Power Supply Voltage (+5V)

DESCRIPTION

The DS1620 Digital Thermometer and Thermostat provides 9–bit temperature readings which indicate the temperature of the device. With three thermal alarm outputs, the DS1620 can also act as a thermostat. T_{HIGH} is driven high if the DS1620's temperature is greater than or equal to a user–defined temperature TH. T_{LOW} is driven high if the DS1620's temperature is less than or equal to a user–defined temperature TL. T_{COM} is driven high when the temperature exceeds TH and stays high until the temperature falls below that of TL.

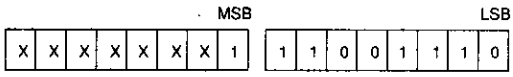
User–defined temperature settings are stored in non–volatile memory, so parts can be programmed prior to insertion in a system, as well as used in standalone applications without a CPU. Temperature settings and temperature readings are all communicated to/from the DS1620 over a simple 3–wire interface.

OPERATION–READING TEMPERATURE

The DS1620 measures temperatures through the use of an onboard, proprietary temperature measurement technique. The temperature reading is provided in a 9–bit, two's complement format. Table 1 describes the exact relationship of output data to measured temperature. The data is transmitted serially through the 3–wire serial interface, LSB first. The DS1620 can measure temperature over the range of –55°C to +125°C in 0.5°C increments. For Fahrenheit usage, a lookup table or conversion factor must be used.

Since data is transmitted over the 3–wire bus LSB first, temperature data can be written to/read from the DS1620 as either a 9–bit word (taking $\overline{RST}$ low after the 9th (MSB) bit), or as two transfers of 8–bit words, with the most significant 7 bits being ignored or set to zero, as illustrated in Table 1. After the MSB, the DS1620 will output 0's.

Note that temperature is represented in the DS1620 in terms of a $^1/_2$°C LSB, yielding the following 9–bit format:

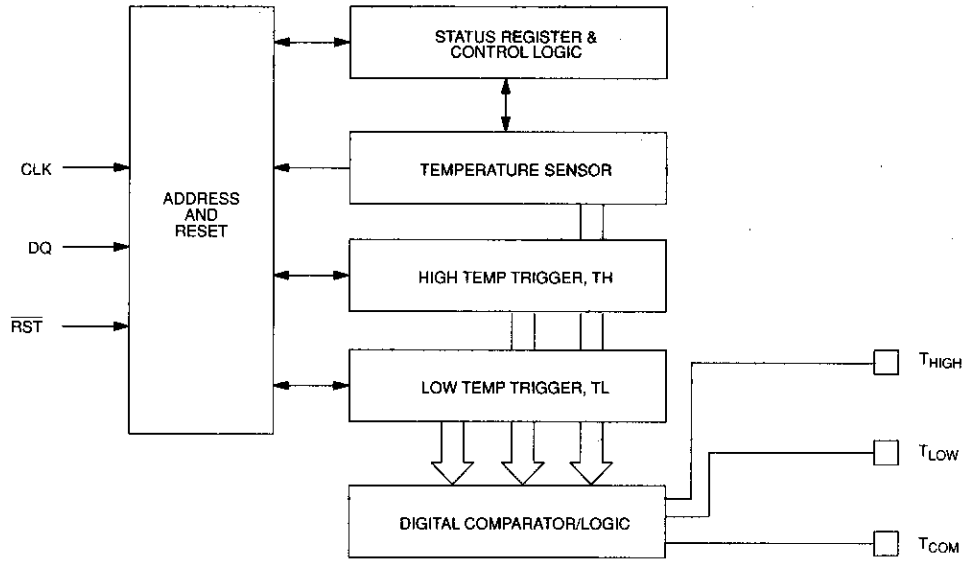


T = –25°C

TEMPERATURE/DATA RELATIONSHIPS Table 1

TEMP	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0 11111010	00FA
+25°C	0 00110010	0032h
$^1/_2$°C	0 00000001	0001h
0°C	0 00000000	0000h
–$^1/_2$°C	1 11111111	01FFh
–25°C	1 11001110	01CEh
–55°C	1 10010010	0192h

DS1620 FUNCTIONAL BLOCK DIAGRAM Figure 1



DETAILED PIN DESCRIPTION Table 2

PIN	SYMBOL	DESCRIPTION
1	DQ	**Data Input/Output pin** for 3–wire communication port.
2	CLK/$\overline{CONV}$	**Clock Input pin** for 3–wire communication port. When the DS1620 is used in a standalone application with no 3–wire port, this pin can be used as a convert pin. Temperature conversion will begin on the falling edge of $\overline{CONV}$.
3	$\overline{RST}$	**Reset input pin** for 3–wire communication port.
4	GND	**Ground pin.**
5	T_{COM}	**High/Low Combination Trigger.** Goes high when temperature exceeds TH; will reset to low when temperature falls below TL.
6	T_{LOW}	**Low Temperature Trigger.** Goes high when temperature falls below TL.
7	T_{HIGH}	**High Temperature Trigger.** Goes high when temperature exceeds TH.
8	V_{DD}	**Supply Voltage.** 5V input power pin.

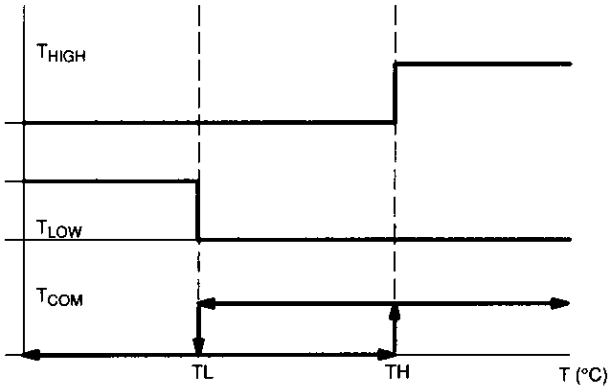
OPERATION–THERMOSTAT CONTROLS

Three thermally triggered outputs, T_{HIGH}, T_{LOW}, and T_{COM}, are provided to allow the DS1620 to be used as a thermostat, as shown in Figure 2. When the DS1620's temperature meets or exceeds the value stored in the high temperature trip register, the output T_{HIGH} becomes active (high) and remains active until the DS1620's measured temperature becomes less than the stored value in the high temperature register, TH. The T_{HIGH} output can be used to indicate that a high temperature tolerance boundary has been met or exceeded, or as part of a closed loop system can be used to activate a cooling system and to deactivate it when the system temperature returns to tolerance.

The T_{LOW} output functions similarly to the T_{HIGH} output. When the DS1620's measured temperature equals or falls below the value stored in the low temperature register, the T_{LOW} output becomes active. T_{LOW} remains active until the DS1620's temperature becomes greater than the value stored in the low temperature register, TL. The T_{LOW} output can be used to indicate that a low temperature tolerance boundary has been met or exceeded, or as part of a closed loop system, can be used to activate a heating system and to deactivate it when the system temperature returns to tolerance.

The T_{COM} output goes high when the measured temperature meets or exceeds TH, and will stay high until the temperature equals or falls below TL. In this way, any amount of hysteresis can be obtained.

THERMOSTAT OUTPUT OPERATION Figure 2



OPERATION AND CONTROL

The DS1620 must have temperature settings resident in the TH and TL registers for thermostatic operation. A configuration/status register is also used to determine the method of operation that the DS1620 will use in a particular application, as well as indicating the status of the temperature conversion operation. The configuration register is defined as follows:

CONFIGURATION/STATUS REGISTER

DONE	THF	TLF	X	X	X	CPU	1SHOT

where

X = Don't Care

DONE = Conversion Done bit. 1=conversion complete, 0=conversion in progress.

THF = Temperature High Flag. This bit will be set to 1 when the temperature is greater than or equal to the value of TH. It will remain 1 until reset by writing 0 into this location or by removing power from the device. This feature provides a method of determining if the DS1620 has ever been subjected to temperatures above TH while power has been applied.

TLF = Temperature Low Flag. This bit will be set to 1 when the temperature is less than or equal to the value of TL. It will remain 1 until reset by writing 0 into this location or by removing power from the device. This feature provides a method of determining if the DS1620 has ever been subjected to temperatures below TL while power has been applied.

CPU = CPU use bit. If CPU=0, the CLK/$\overline{\text{CONV}}$ pin acts as a conversion start control, when $\overline{\text{RST}}$ is low. If CPU is 1, the DS1620 will be used with a CPU communicating to it over the 3–wire port, and the operation of the CLK/$\overline{\text{CONV}}$ pin is as a normal clock in concert with DQ and $\overline{\text{RST}}$.

1SHOT = One–Shot Mode. If 1SHOT is 1, the DS1620 will perform one temperature conversion upon reception of the Start Convert T protocol. If 1SHOT is 0, the DS1620 will continuously perform temperature conversion.

For typical thermostat operation, the DS1620 will operate in continuous mode. However, for applications where only one reading is needed at certain times, and to conserve power, the one–shot mode may be used. Note that the thermostat outputs (T_{HIGH}, T_{LOW}, T_{COM}) will remain in the state they were in after the last valid temperature conversion cycle when operating in one–shot mode.

OPERATION IN STANDALONE MODE

In applications where the DS1620 is used as a simple thermostat, no CPU is required. Since the temperature limits are nonvolatile, the DS1620 can be programmed prior to insertion in the system. In order to facilitate operation without a CPU, the CLK/$\overline{\text{CONV}}$ pin (pin 3) can be used to initiate conversions. Note that the CPU bit must be set to 0 in the configuration register to use this mode of operation.

To use the CLK/$\overline{\text{CONV}}$ pin to initiate conversions, $\overline{\text{RST}}$ must be low and CLK/$\overline{\text{CONV}}$ must be high. If CLK/$\overline{\text{CONV}}$ is driven low and then brought high in less than 10 ms, one temperature conversion will be performed and then the DS1620 will return to an idle state. If CLK/$\overline{\text{CONV}}$ is driven low and remains low, continuous conversions will take place until CLK/$\overline{\text{CONV}}$ is brought high again. With the CPU bit set to 0, the CLK/$\overline{\text{CONV}}$ will override the 1–shot bit if it is equal to 1. This means that even if the part is set for one–shot mode, driving CLK/$\overline{\text{CONV}}$ low will initiate conversions.

3–WIRE COMMUNICATIONS

The 3–wire bus is comprised of three signals. These are the $\overline{\text{RST}}$ (reset) signal, the CLK (clock) signal, and the DQ (data) signal. All data transfers are initiated by driving the $\overline{\text{RST}}$ input high. Driving the $\overline{\text{RST}}$ input low terminates communication. (See Figures 3 and 4.) A clock cycle is a sequence of a falling edge followed by a rising edge. For data inputs, the data must be valid during the rising edge of a clock cycle. Data bits are output on the falling edge of the clock, and remain valid through the rising edge.

When reading data from the DS1620, the DQ pin goes to a high impedance state while the clock is high. Taking $\overline{RST}$ low will terminate any communication and cause the DQ pin to go to a high impedance state.

Data over the 3–wire interface is communicated LSB first. The command set for the 3–wire interface as shown in Table 3 is as follows; only these protocols should be written to the DS1620, as writing other protocols to the device may result in permanent damage to the part.

Read Temperature [AAh]
This command reads the contents of the register which contains the last temperature conversion result. The next nine clock cycles will output the contents of this register.

Write TH [01h]
This command writes to the TH (HIGH TEMPERATURE) register. After issuing this command, the next nine clock cycles clock in the 9–bit temperature limit which will set the threshold for operation of the T_{HIGH} output.

Write TL [02h]
This command writes to the TL (LOW TEMPERATURE) register. After issuing this command, the next nine clock cycles clock in the 9–bit temperature limit which will set the threshold for operation of the T_{LOW} output.

Read TH [A1h]
This command reads the value of the TH (HIGH TEMPERATURE) register. After issuing this command, the next nine clock cycles clock out the 9–bit temperature limit which sets the threshold for operation of the T_{HIGH} output.

Read TL [A2h]
This command reads the value of the TL (LOW TEMPERATURE) register. After issuing this command, the next nine clock cycles clock out the 9–bit temperature limit which sets the threshold for operation of the T_{LOW} output.

Start Convert T [EEh]
This command begins a temperature conversion. No further data is required. In one–shot mode, the temperature conversion will be performed and then the DS1620 will remain idle. In continuous mode, this command will initiate continuous conversions.

Stop Convert T [22h]
This command stops temperature conversion. No further data is required. This command may be used to halt a DS1620 in continuous conversion mode. After issuing this command, the current temperature measurement will be completed, and then the DS1620 will remain idle until a Start Convert T is issued to resume continuous operation.

Write Config [0Ch]
This command writes to the configuration register. After issuing this command, the next eight clock cycles clock in the value of the configuration register.

Read Config [ACh]
This command reads the value in the configuration register. After issuing this command, the next eight clock cycles output the value of the configuration register.

DS1620 COMMAND SET Table 3

INSTRUCTION	DESCRIPTION	PROTOCOL	3–WIRE BUS DATA AFTER ISSUING PROTOCOL	NOTES
Read Temperature	Reads last converted temperature value from temperature register.	AAh	<read data>	
Start Convert T	Initiates temperature conversion.	EEh	idle	1
Stop Convert T	Halts temperature conversion.	22h	idle	1
Write TH	Writes high temperature limit value into TH register.	01h	<write data>	2
Write TL	Writes low temperature limit value into TL register.	02h	<write data>	2
Read TH	Reads stored value of high temperature limit from TH register.	A1h	<read data>	2
Read TL	Reads stored value of low temperature limit from TL register.	A2h	<read data>	2
Write Config	Writes configuration data to configuration register.	0Ch	<write data>	2
Read Config	Reads configuration data from configuration register.	ACh	<read data>	2

NOTES:

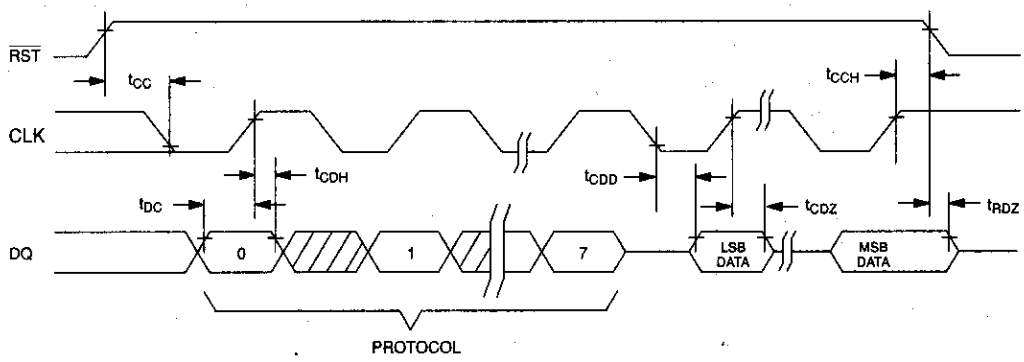
1. In continuous conversion mode, a Stop Convert T command will halt continuous conversion. To restart, the Start Convert T command must be issued. In one–shot mode, a Start Convert T command must be issued for every temperature reading desired.
2. Writing to the E^2 typically requires 10 ms at room temperature. After issuing a write command, no further reads or writes should be requested for at least 10 ms.

FUNCTION EXAMPLE

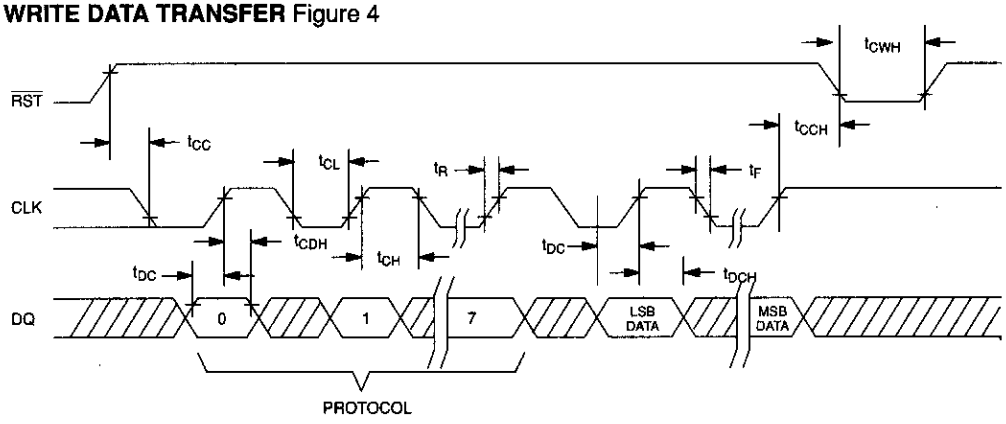
Example: CPU sets up DS1620 for continuous conversion and thermostatic function.

CPU MODE	DS1620 MODE (3–WIRE)	DATA (LSB FIRST)	COMMENTS
TX	RX	0Ch	CPU issues Write Config command.
TX	RX	00h	CPU sets DS1620 up for continuous conversion.
TX	RX	01h	CPU issues Write TH command.
TX	RX	0050h	CPU sends data for TH limit of +40°C.
TX	RX	02h	CPU issues Write TL command.
RX	TX	0014h	CPU sends data for TL limit of +10°C.
TX	RX	A1h	CPU issues Read TH command.
RX	TX	0050h	DS1620 sends back stored value of TH for CPU to verify.
TX	RX	A2h	CPU issues Read TL command.
RX	TX	0014h	DS1620 sends back stored value of TL for CPU to verify.
TX	RX	EEh	CPU issues Start Convert T command.

READ DATA TRANSFER Figure 3



WRITE DATA TRANSFER Figure 4



NOTE: t_{CL}, t_{CH}, t_R, and t_F apply to both read and write data transfer.

ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.5V to +7.0V
Operating Temperature	–55°C to +125°C
Storage Temperature	–55°C to +125°C
Soldering Temperature	260°C for 10 seconds

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

RECOMMENDED DC OPERATING CONDITIONS

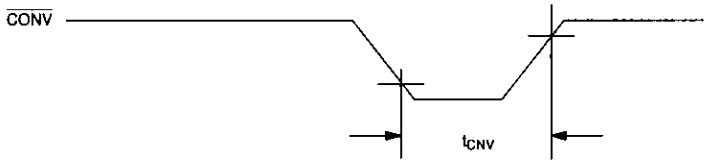
PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply	V_{DD}	4.5	5.0	5.5	V	1
Logic 1	V_{IH}	2.0		V_{CC}+0.3	V	1
Logic 0	V_{IL}	–0.3		+0.8	V	1

DC ELECTRICAL CHARACTERISTICS

(–55°C to +125°C; V_{DD}=4.5V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	MAX	UNITS	NOTES
Thermometer Error	T_{ERR}	0°C to +70°C		±1/2	°C	
		–40°C to +0°C and +70°C to +85°C		±1	°C	
		–55°C to –40°C and +85°C to +125°C		±2	°C	
Logic 0 Output	V_{OL}			0.4	V	3
Logic 1 Output	V_{OH}		2.4		V	2
Input Resistance	R_I	$\overline{RST}$ to GND DQ,CLK to V_{DD}		2 2	mΩ mΩ	
Active Supply Current	I_{CC}	0°C to +70°C		1	mA	4, 5
Standby Supply Current	I_{STBY}	0°C to +70°C		1	µA	4, 5

SINGLE CONVERT TIMING DIAGRAM (STAND–ALONE MODE)



AC ELECTRICAL CHARACTERISTICS

(–55°C to +125°C; V_{DD}=4.5V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Temperature Conversion Time	T_{TC}			1	s	
Data to CLK Setup	t_{DC}	35			ns	6
CLK to Data Hold	t_{CDH}	40			ns	6
CLK to Data Delay	t_{CDD}			100	ns	6, 7, 8
CLK Low Time	t_{CL}	250			ns	6
CLK High Time	t_{CH}	250			ns	6
CLK Frequency	f_{CLK}	DC		2.0	MHz	6
CLK Rise and Fall	t_R, t_F			500	ns	
$\overline{RST}$ to CLK Setup	t_{CC}	100			ns	6
CLK to $\overline{RST}$ Hold	t_{CCH}	40			ns	6
$\overline{RST}$ Inactive Time	t_{CWH}	125			ns	6, 9
CLK High to I/O High Z	t_{CDZ}			50	ns	6
$\overline{RST}$ Low to I/O High Z	t_{RDZ}			50	ns	6
Convert Pulse Width	t_{CNV}	250 ns		500 ms		
NV Write Cycle Time	t_{WR}		10	50	ms	

AC ELECTRICAL CHARACTERISTICS

(–55°C to +125°C; V_{DD}=4.5V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Input Capacitance	C_I		5		pF	
I/O Capacitance	$C_{I/O}$		10		pF	

NOTES:

1. All voltages are referenced to ground.
2. Logic one voltages are specified at a source current of 1 mA.
3. Logic zero voltages are specified at a sink current of 4 mA.
4. I_{CC} specified with DQ pin open.
5. I_{CC} specified with V_{CC} at 5.0V and $\overline{RST}$=GND.
6. Measured at V_{IH} = 2.0V or V_{IL} = 0.8V.
7. Measured at V_{OH} = 2.4V or V_{OL} = 0.4V.
8. Load capacitance = 50 pF.
9. t_{CWH} must be 10 ms minimum following any read or write command that involves the E^2 memory.