Stamp Applications no. 2 (April '95):

Thermometer-on-a-Chip Simplifies Temperature Measurement

Using the DS1620, by Scott Edwards

DESIGNING a temperature-measurement application for the BASIC Stamp is a lot like voting; you end up selecting the lesser of three evils, shown in figure 1. Cost, calibration, or too many components make these solutions less than ideal.

Now, there's a fourth candidate. It's the Dallas Semiconductor DS1620 digital thermometer/thermostat chip, shown in figure 2. The DS1620 measures temperature in units of 0.5 degrees Centigrade from -55° to +125° C. The DS1620 is calibrated at the factory for exceptional accuracy: ±0.5° C from 0 to +70° C, which, is the Stamp's operating temperature range.

(For fans of the familiar Fahrenheit scale, those °C temperatures convert to: range, -67° to +257° F; resolution, 0.9° F; accuracy, ±0.9° F from 32° to 158° F.)

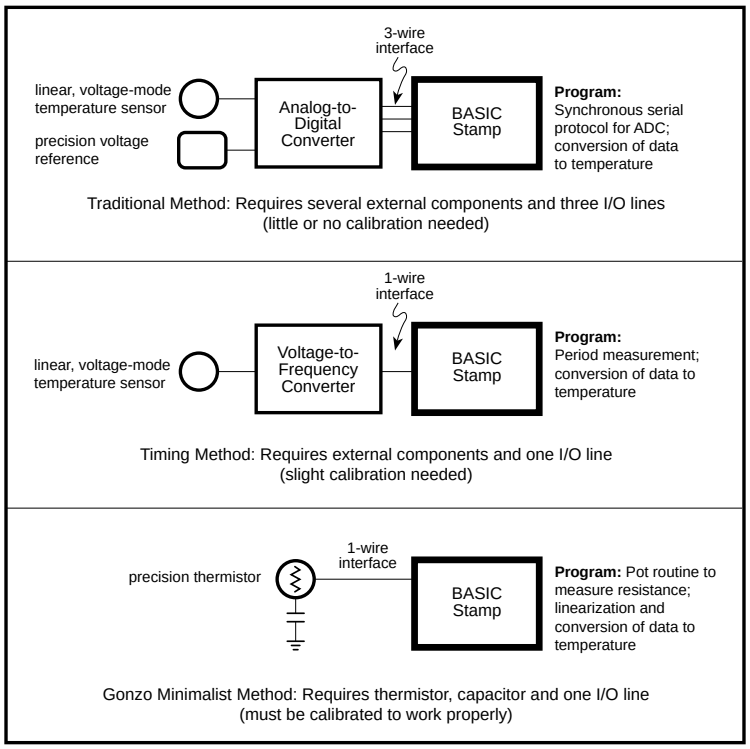


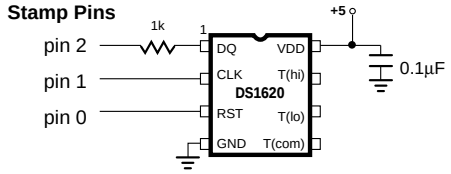
Figure 1. Three methods for measuring temperature with the Stamp.

The chip makes its temperature data available as a 9-bit number conveyed over a three-wire serial interface. The DS1620 can be set to operate continuously, taking one temperature measurement per second, or intermittently, conserving power by measuring only when told to.

The DS1620 can also operate as a standalone thermostat. A temporary connection to a Stamp or other 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.

Let's concentrate on applications using the DS1620 as a Stamp peripheral, as shown in the listing. Later, we'll talk about adapting the code for configuring it as a thermostat.

Using the DS1620 requires sending a command (what Dallas Semi calls a "protocol") to the chip, then listening for a response (if applicable). The code under "DS1620 I/O Subroutines" in the listing shows how this is done. In a typical temperature-measurement application, the program will set the DS1620 to thermometer mode, configure it for continuous conversions, and tell it to start. Thereafter, all the program must do is request a temperature reading, then shift it in, as shown in the listing's *Again* loop.



DQ—Data input/output
CLK—Clock for shifting data in/out (active-low conversion start in thermostat/ 1-shot mode)
RST—Reset; high activates chip, low disables it
GND—Ground connection
VDD—Supply voltage; +4.5 to 5.5 Vdc
T(hi)—In thermostat mode, outputs a 1 when temp is above high setpoint
T(lo)—In thermostat mode, outputs a 1 when temp is below low setpoint
T(com)—In thermostat mode, outputs a 1 when temp exceeds high setpoint and remains high until temp drops below low setpoint

Figure 2. DS1620 pinout and connection.

The DS1620 delivers temperature data in a nine-bit, two's complement format, shown in the table. Each unit represents 0.5° C, so a reading of 50 translates to +25°C. Negative values are expressed as two's complement numbers. In two's complement, values with a 1 in their leftmost bit position are negative. The leftmost bit is often called the *sign* bit, since a 1 means - and a 0 means +.

To convert a negative two's complement value to a positive number, you must invert it and add 1. If you want to display this value, remember to put a minus sign in front of it.

Rather than mess with two's complement negative numbers, the program converts DS1620 data to an absolute scale called DSabs, with a range of 0 to 360 units of 0.5° C each. The Stamp can perform calculations in this all-positive system, then readily convert the results for display in °C or °F, as shown in the listing.

Going Further. Once you have configured the DS1620, you don't have to reconfigure it unless you want to change a setting. The DS1620 stores its configuration in EEPROM (electrically erasable, programmable read-only memory), which retains data even with the power off. In memory-tight Stamp applications, you might want to run the full program once for configuration, then strip out the configuration stuff to make more room for your final application.

If you want to use the DS1620 in its role as a standalone thermostat, the Stamp can help here, too. The listing includes protocols for putting the DS1620 into thermostat (*NoCPU*) mode, and for reading and writing the temperature setpoints. You could write a Stamp program to accept temperature data serially, convert it to nine-bit, two's complement format, then write it to the DS1620 configuration register. An example program that does this is available from the source listed below.

Be aware of the DS1620's drive limitations in thermostat mode; it sources just 1 mA and sinks 4 mA. This isn't nearly enough to drive a relay—it's barely enough to light an LED. You'll want to buffer this output with a Darlington

Nine-Bit Format for DS1620 Temperature Data

-Temperature-		-DS1620 Data-		
°F	°C	Binary	Hexadecimal	Decimal
+257	+125	0 11111010	00FA	250
+77	+25	0 00110010	0032	50
+32.9	+0.5	0 00000001	0001	1
+32	0	0 00000000	0000	0
+31.1	-0.5	1 11111111	01FF	511
-13	-25	1 11001110	01CE	462
-67	-55	1 10010010	0192	402

Example conversion of a negative temperature:

-25°C = 1 11001110 in binary. The 1 in the leftmost bit indicates that this is a negative number. Invert the lower eight bits and add 1: 11001110 -> 00110001 + 1 = 00110010 = 50. Units are 0.5°C, so divide by 2. Converted result is -25°C.

or MOSFET switch in serious applications.

For Your Convenience

Dallas Semiconductor offers data and samples of the DS1620 at reasonable cost. Call them at 214-450-0448 or contact your distributor. For a copy of the program listing (plus a thermostat programming example and PIC assembly language source code) on disk, a sample DS1620, and DS1620 documentation, you may order the DS1620 App Kit from Scott Edwards Electronics, 964 Cactus Wren Lane, Sierra Vista, AZ 85635; phone, 520-459-4802; fax 520-459-0623. Price is $20 postpaid. Visa, Mastercard, American Express, checks, and qualified purchase orders accepted.

```
' Program Listing For Interfacing To DS1620 Digital Thermometer
' Program: DS1620.BAS
' 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 = w0          ' 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 after start.
' >>> 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.

' Now 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 need 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          ' Configure as 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 the temperature reading to absolute.
  gosub DisplayF           ' Display in degrees F.
  gosub DisplayC           ' Display in degrees C.
goto Again
' ==================== DS1620 I/O Subroutines =================
' Subroutine: Shout
' Shift bits out to the DS1620. Sends the lower 8 bits stored in
' DSout (w0). 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.
```

```
' ================= 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) that ranges from 0 (-55 C) to 360 (+125 C).
' 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

' 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
```