Introduction

Measuring temperature is not always a trivial task. Most sensors used to read temperature transduce the reading to an electrical signal. These sensors provide a voltage level relative to the temperature reading. This voltage is converted to a digital number using an analog-to-digital converter where it can then be processed by a microprocessor or microcontroller.

The DS1620 Digital Thermometer from Dallas Semiconductor provides a single-chip solution that reads temperature and converts it to a 9-bit digital value. This data is then read from the DS1620 via a serial interface to a microcontroller (MCU). The device also provides three thermal alarm outputs for thermostatic control.

This application note describes the interface between the MC68HC705J1A (J1A) and the DS1620 that is used to measure temperature in the range of –55 °C to +125 °C. Since the J1A does not have a serial module on chip, a software driver is created to provide the appropriate serial bus signals to the DS1620. Circuitry and example
code are included here to demonstrate the interface between the two parts.

**Features**

The DS1620 provides these features:

- No external components required
- Supply voltage range is 2.7 to 5.5 volts.
- Measures temperature from –55 °C to +125 °C in 0.5 °C increments. The equivalent Fahrenheit range is –67 °F to +257 °F in 0.9 °F increments.
- Temperature is read as a 9-bit value.
- Conversion time is 1 second (max).
- Thermostatic settings are user-definable and non-volatile (EEPROM).
- Data is transceived via a 3-wire serial bus.
- Available in 8-pin DIP or SOIC packages

**Description**

The DS1620 provides 9-bit temperature data which indicates the temperature of the chip. All data is communicated via the 3-wire serial interface. User-defined temperature settings are stored in non-volatile memory.

Three thermal alarm outputs act as a thermostat, signifying user-defined thresholds.

- The pin $T_{\text{High}}$ is driven high if the DS1620’s temperature is greater than or equal to the user-defined temperature, TH.
- The $T_{\text{Low}}$ pin is driven high if the DS1620’s temperature is less than or equal to a user-defined temperature, TL.
The $T_{\text{COM}}$ pin is used to derive hysteresis between the $T_{\text{High}}$ and $T_{\text{Low}}$ pins. It is driven high when the temperature exceeds $\text{TH}$ and stays high until the temperature falls below that of $\text{TL}$.

**DS1620 Hardware Interface**

**Pinout and Pin Descriptions**  
*Figure 1* and *Table 1* illustrate and describe the DS1620 pinout.

![Figure 1. DS1620 Pinout](image)

**Table 1. DS1620 Pin Descriptions**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Symbol</th>
<th>Name</th>
<th>I/O/PWR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DQ</td>
<td>Data input/output</td>
<td>I/O</td>
<td>3-wire port data</td>
</tr>
<tr>
<td>2</td>
<td>CLK/CONV</td>
<td>Clock</td>
<td>I</td>
<td>3-wire port clock</td>
</tr>
<tr>
<td>3</td>
<td>RST</td>
<td>Reset</td>
<td>I</td>
<td>3-wire port reset</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>Ground</td>
<td>PWR</td>
<td>System ground</td>
</tr>
<tr>
<td>5</td>
<td>$T_{\text{COM}}$</td>
<td>High/low combination trigger</td>
<td>O</td>
<td>Goes high when temperature exceeds $\text{TH}$; will reset to low when temperature falls below $\text{TL}$</td>
</tr>
<tr>
<td>6</td>
<td>$T_{\text{Low}}$</td>
<td>Low temp trigger</td>
<td>O</td>
<td>Goes high when temperature falls below $\text{TL}$</td>
</tr>
<tr>
<td>7</td>
<td>$T_{\text{High}}$</td>
<td>High temp trigger</td>
<td>O</td>
<td>Goes high when temperature exceeds $\text{TH}$</td>
</tr>
<tr>
<td>8</td>
<td>$V_{\text{DD}}$</td>
<td>Supply voltage</td>
<td>PWR</td>
<td>System power range is 2.7 V to 5.5 V</td>
</tr>
</tbody>
</table>
The temperature sensor shown in Figure 2 uses oscillators that have particular temperature coefficients to derive a temperature reading. For detailed information on this process, consult the DS1620 data sheet.

Figure 2. DS1620 Block Diagram

Read and write data transfer timing is shown in Figure 3 and Figure 4. Only logic levels are shown here. Consult the DS1620 data sheet if detailed AC electrical characteristics are needed.

Figure 3. Serial Data Read Timing
Thermostat Operation

The DS1620 has three thermal alarms that trigger the output pins $T_{\text{High}}$, $T_{\text{Low}}$, and $T_{\text{COM}}$. These pins can be used to control closed-loop heating and cooling systems by activating and deactivating a system dependent on the defined temperature boundaries.

The $T_{\text{High}}$ pin is set to 1 when the temperature exceeds the $TH$ value. Likewise, the $T_{\text{Low}}$ pin is set to 1 when the temperature falls below the $TL$ value.

To control oscillation in a thermostatic system, the $T_{\text{COM}}$ pin can be used to provide hysteresis.

Figure 4. Serial Data Write Timing

Figure 5. Thermostat Outputs
DS1620 Software Interface

Configuration and Status Register

The configuration and status register configures the DS1620 for different modes of operation and to provide status information on the device.

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONE</td>
<td>THF</td>
<td>TLF</td>
<td>NVB</td>
<td>1</td>
<td>0</td>
<td>CPU</td>
<td>1SHOT</td>
</tr>
</tbody>
</table>

**Figure 6. Configuration and Status Register**

**DONE — Conversion Complete Flag**
1 = Temperature conversion is complete.
0 = Temperature conversion is in progress.

**THF — High Temperature Flag**
1 = The temperature is greater than or equal to the value of the TH register. It remains 1 until it is reset by writing a 0 to this bit or until power is removed from the device. This allows the user to determine if the device has ever exceeded the TH limit.
0 = The temperature is less than the value of the TH register.

**TLF — Low Temperature Flag**
1 = The temperature is less than or equal to the value of the TL register. It remains 1 until it is reset by writing a 0 to this bit or until power is removed from the device. This allows the user to determine if the device has ever fallen below the TL limit.
0 = The temperature is greater than the value of the TL register.

**NVB — EEPROM Busy Flag**
1 = A write to an EEPROM cell is in progress. This process could take up to 50 ms. Write to the EEPROM memory only within the 0 °C to 70 °C temperature range.
0 = The EEPROM is not busy.
CPU — CPU Use Bit

1 = The operation of the CLK/CONV pin acts as a normal clock. This bit is stored in an EEPROM cell.
0 = The CLK/CONV pin is used to control a conversion start when RST is low. The DS1620 is shipped with CPU = 0.

1SHOT — One-Shot Mode

1 = The DS1620 will execute one temperature conversion after the start convert T command is received. This bit is stored in an EEPROM cell.
0 = The DS1620 continuously executes the temperature conversion process. The DS1620 is shipped with 1SHOT = 0.

Command Set

The DS1620 command set is given in Table 2, which is followed by an explanation of each command. Not all DS1620 commands are shown in Table 2 since the commands to receive a more accurate temperature reading are not covered in this application note.

<table>
<thead>
<tr>
<th>Command</th>
<th>Protocol</th>
<th>Data After Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read temperature</td>
<td>$AA</td>
<td>Read 9-bit data</td>
</tr>
<tr>
<td>Start convert T</td>
<td>$EE</td>
<td>Idle</td>
</tr>
<tr>
<td>Stop convert T</td>
<td>$22</td>
<td>Idle</td>
</tr>
<tr>
<td>Write TH</td>
<td>$01</td>
<td>Write 9-bit data</td>
</tr>
<tr>
<td>Write TL</td>
<td>$02</td>
<td>Write 9-bit data</td>
</tr>
<tr>
<td>Read TH</td>
<td>$A1</td>
<td>Read 9-bit data</td>
</tr>
<tr>
<td>Read TL</td>
<td>$A2</td>
<td>Read 9-bit data</td>
</tr>
<tr>
<td>Write config</td>
<td>$0C</td>
<td>Write 8-bit data</td>
</tr>
<tr>
<td>Read config</td>
<td>$AC</td>
<td>Read 8-bit data</td>
</tr>
</tbody>
</table>

NOTE: Writing to the EEPROM memory cells typically requires 10 ms at room temperature. The maximum time specified is 50 ms. The test code in this application note is written for a 50-ms wait period.
<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Temperature</td>
<td>$AA</td>
<td>Reads the contents of the temperature. The next nine clocks will transmit the 9-bit value on the serial bus to the host.</td>
</tr>
<tr>
<td>Start Convert T</td>
<td>$EE</td>
<td>Begins the temperature conversion process. No data is read or written after this command. In continuous mode, the part will continually cycle through the conversion process. In single-shot mode, the part will convert one temperature reading and then remain idle.</td>
</tr>
<tr>
<td>Stop Convert T</td>
<td>$22</td>
<td>Stops the temperature conversion process. No data is read or written after this command. After the command is issued, the current conversion process is finished and the DS1620 remains idle. Until a start convert T command is issued, the DS1620 will remain in its idle state.</td>
</tr>
<tr>
<td>Write TH</td>
<td>$01</td>
<td>Writes to the high-temperature register (TH). The next nine clock cycles will transmit the 9-bit value on the serial bus to the DS1620. This sets the threshold level for operation of the T\text{High} output pin.</td>
</tr>
<tr>
<td>Write TL</td>
<td>$02</td>
<td>Writes to the low-temperature register (TL). The next nine clock cycles will transmit the 9-bit value on the serial bus to the DS1620. This sets the threshold level for operation of the T\text{Low} output pin.</td>
</tr>
<tr>
<td>Read TH</td>
<td>$A1</td>
<td>Reads the value of the TH register. The next nine clock cycles will transmit the 9-bit value on the serial bus to the host. This 9-bit value is the temperature limit for the T\text{High} output pin.</td>
</tr>
<tr>
<td>Read TL</td>
<td>$A2</td>
<td>Reads the value of the TL register. The next nine clock cycles will transmit the 9-bit value on the serial bus to the host. This 9-bit value is the temperature limit for the T\text{Low} output pin.</td>
</tr>
<tr>
<td>Write Config</td>
<td>$0C</td>
<td>Writes to the configuration register. The next eight clock cycles will transmit the 8-bit value on the serial bus to the DS1620.</td>
</tr>
<tr>
<td>Read Config</td>
<td>$AC</td>
<td>Reads the config register. The next eight clocks will transmit the 8-bit value on the serial bus to the host.</td>
</tr>
</tbody>
</table>
The temperature reading is provided in a two’s complement 9-bit value. **Table 3** illustrates the relationship between temperature and the 9-bit reading. For Fahrenheit, a table lookup or conversion factor must be used.

**Table 3. Temperature/Data Relationship**

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Digital Output, Hex</th>
<th>Digital Output, Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>+125 °C</td>
<td>$00FA</td>
<td>0 11111010</td>
</tr>
<tr>
<td>+25 °C</td>
<td>$0032</td>
<td>0 00110010</td>
</tr>
<tr>
<td>+0.5 °C</td>
<td>$0001</td>
<td>0 00000001</td>
</tr>
<tr>
<td>0 °C</td>
<td>$0000</td>
<td>0 00000000</td>
</tr>
<tr>
<td>–0.5 °C</td>
<td>$01FF</td>
<td>1 11111111</td>
</tr>
<tr>
<td>–25 °C</td>
<td>$01CE</td>
<td>1 11001110</td>
</tr>
<tr>
<td>–55 °C</td>
<td>$0192</td>
<td>1 10010010</td>
</tr>
</tbody>
</table>

The 9-bit temperature value and thermostat settings are stored as two 8-bit values in memory. This is illustrated in **Figure 7**.

**Figure 7. Memory Configuration of 9-Bit Data**
With only 20 pins, the J1A is one of the smaller members of the HC05 Family. It has a total of 1240 bytes of erasable programmable read-only memory (EPROM) and includes 14 I/O pins.

The pins used to drive the DS1620 on the J1A are:

- Port A, Bit 0 — This I/O pin (DQ) is used to transmit and receive data on the DQ pin of the DS1620.
- Port A, Bit 1 — This I/O pin (CLK) is configured as an output to drive the serial clock pin, CLK/CONV, of the DS1620.
- Port A, Bit 2 — This I/O pin (RST) is configured as an output to drive the reset pin, RST, of the DS1620.

The schematic used for testing the J1A-to-DS1620 interface on the MMEVS development system is shown in Figure 8.

For more information on the HC705J1A, consult the *MC68HC705J1A Technical Databook*, Motorola document order number MC68HC705J1A/D.
I/O driving or manipulation is the process of toggling I/O pins with software instructions to create a certain hardware peripheral. The HC05 CPU provides special instructions to specifically manipulate single I/O pins.

The serial transmission driver has been put into two subroutines called TXD for transmitting eight bits of data and RXD for receiving eight bits of data.

The flowcharts for the DS1620 serial I/O drivers are shown in Figure 9 through Figure 11. These routines were written especially for the DS1620 and may not be able to properly drive other MCU peripherals with serial buses.

Figure 11 shows the flowchart for the main test routine. The step-by-step sequence of testing is:

1. Write $00$ to the configuration register. This sets the DS1620 for continuous conversion mode.
2. Write to the TH register. The value is set at $3C = 30 \, ^\circ C = 86 \, ^\circ F$.
3. Read the TH register. Store the reading in RAM locations TH_MSB and TH_LSB.
4. Write to the TL register. The value is set at $28 = 20 \, ^\circ C = 68 \, ^\circ F$.
5. Read the TL register. Store the reading in RAM locations TL_MSB and TL_LSB.
6. Send the start conversion command.
7. Stop the code from running on the emulator to allow 1 second of time for the temperature reading.
8. Restart the code. The temperature is read and placed in RAM locations TEMP_MSB and TEMP_LSB.

After the test sequence is finished, the TH, TL, and temperature values are verified. To get a temperature reading again, restart the code at step 8.
To test the thermostat outputs, increase the temperature higher than 86 °F. The T\text{High} pin should read as 1. Decrease the temperature below 68 °F and the T\text{Low} pin should read as 1. Since the DS1620 is configured for continuous conversion, no software is needed to output the thermostatic outputs. This is an inherent function of the DS1620.

The assembly code for the test routine is provided in Code Listing.

Development Tools

The interface was created and tested using these development tools:

- M68MMPFB0508 — Motorola MMEVS platform board
- M68EM05J1A — Motorola J1A emulation module
- Win IDE Version 1.02 — Editor, assembler, and debugger by P&E Microcomputer Systems
Flowcharts for the Serial Drivers

Figure 9. TXD Subroutine Flowchart
Figure 10. RXD Subroutine Flowchart
Figure 11. Flowchart for Main Test Routine
Code Listing

************************************************************************************
* File name: DS1620.ASM
* Example Code for the MC68HC705J1A Interface to the
*     Dallas DS1620 Digital Thermometer
* Ver: 1.0
* Date: June 5, 1998
* Author: Mark Glenewinkel
*         Motorola Field Applications
*         Consumer Systems Group
* Assembler: P&E IDE ver 1.02
*
* For code explanation and flow charts,
* please consult Motorola Application Note
*     "Interfacing the MC68HC705J1A to the DS1620 Digital Thermometer"
*     Literature # AN1754/D
*
***********************************************************************************

*** SYSTEM DEFINITIONS AND EQUATES ***********************************************
*** Internal Register Definitions
PORTA EQU $00 ;PortA
DDRA EQU $04 ;data direction for PortA

*** Application Specific Definitions
SER_ PORT EQU $00 ;PORTA is SER_PORT
CLK EQU 1T ;PORTA, bit 1, clock signal
DQ EQU 0T ;PORTA, bit 0, data signal
RST EQU 2T ;PORTA, bit 2, reset signal
DQ_DIR EQU 0T ;PortA Data Dir for DQ signal
READ_TEMP EQU $AA ;instr for reading temperature
START_CONV EQU $EE ;instr for starting temperature conv
STOP_CONV EQU $22 ;instr for stopping temperature conv
WRITE_TH EQU $01 ;instr for writes high temp limit to TH reg
WRITE_TL EQU $02 ;instr for writes low temp limit to TL reg
READ_TH EQU $A1 ;instr for reads high temp limit from TH reg
READ_TL EQU $A2 ;instr for reads high temp limit from TL reg
WRITE_CONFIG EQU $0C ;instr for writes to config reg
READ_CONFIG EQU $AC ;instr for reads from config reg

*** Memory Definitions
EPROM EQU $300 ;start of EPROM mem
RAM EQU $C0 ;start of RAM mem
RESET EQU $7FE ;vector for reset

AN1754

MOTOROLA
*** RAM VARIABLES  ******************************************************************
ORG RAM
TEMP_MSB DB 1 ;temperature reading MSB
TEMP_LSB DB 1 ;temperature reading MSB
TH_MSB DB 1 ;High temp trigger MSB
TH_LSB DB 1 ;High temp trigger LSB
TL_MSB DB 1 ;Low temp trigger MSB
TL_LSB DB 1 ;Low temp trigger LSB

*** MAIN ROUTINE  ******************************************************************
ORG EPROM ;start at beginning of EPROM
*** Initialize Ports
START lda #$07 ;init SER_PORT
sta SER_PORT
lda #$07 ;make SER_PORT pins outputs
sta DDRA

*** Write $00 to Config reg, setup for cont conv
lda #WRITE_CONFIG ;load Acca with instruction
jsr TXD ;transmit instruction
lda #$00 ;load Acc with data
jsr TXD ;transmit data
bclr RST,SER_PORT ;toggle RST
bset RST,SER_PORT
jsr NV_WAIT ;wait ~50 ms for NV memory operation

*** Set the TH reg to $3C = 30C = 86F
lda #WRITE_TH ;load Acca with instruction
jsr TXD ;transmit instruction
lda #$3C ;load Acc with data
jsr TXD ;transmit data
lda #$00 ;load Acc with data
jsr TXD ;transmit data
bclr RST,SER_PORT ;toggle RST
bset RST,SER_PORT
jsr NV_WAIT ;wait ~50 ms for NV memory operation

*** Read the TH reg to verify
lda #READ_TH ;load Acca with instruction
jsr TXD ;transmit instruction
jsr RXD ;receive data
sta TH_LSB ;store away result
jsr RXD ;receive data
sta TH_MSB ;store away result
bclr RST,SER_PORT ;toggle RST
bset RST,SER_PORT

*** Set the TL reg to $28 = 20C = 68F
lda #WRITE_TL ;load Acca with instruction
jsr TXD ;transmit instruction
lda #$28 ;load Acc with data
jsr TXD ;transmit data
lda #$00 ;load Acc with data
jsr TXD ;transmit data
bclr RST,SER_PORT ;toggle RST
bset RST,SER_PORT
jsr NV_WAIT ;wait ~50 ms for NV memory operation

*** Read the TL reg to verify
lda #READ_TL ;load Acca with instruction
jsr TXD ;transmit instruction
jsr RXD ;receive data
sta TL_LSB ;store away result
jsr RXD ;receive data
sta TL_MSB ;store away result
bclr RST,SER_PORT ;toggle RST
bset RST,SER_PORT

*** Start temperature conversion
lda #START_CONV ;load Acca with instruction
jsr TXD ;transmit instruction
bclr RST,SER_PORT ;toggle RST
bset RST,SER_PORT

*** Read current temperature
lda #READ_TEMP ;load Acca with instruction
jsr TXD ;transmit instruction
jsr RXD ;receive data
sta TEMP_LSB ;store away result
jsr RXD ;receive data
sta TEMP_MSB ;store away result
bclr RST,SER_PORT ;toggle RST
bset RST,SER_PORT

DUMMY bra DUMMY ;test sequence is over

*** SUBROUTINES

*** Routine takes contents of AccA and transmits it serially to
*** the DS1620, LSB first

TXD
ldx #8T ;set counter

WRITE
asra ;Carry bit = LSB
bcc J1
bset DQ,SER_PORT ;DQ=1
bra CLOCK_IT ;branch to clock_it

J1
bclr DQ,SER_PORT ;DQ=0
brn J1 ;evens it out

CLOCK_IT
bclr CLK,SER_PORT ;CLK=0
bset CLK,SER_PORT ;CLK=1
deecx ;decrement counter
bne WRITE ;return from sub

rts
*** Routine clocks the DS1620 to read data from DQ, LSB first
*** 8 bit contents are put in AccA
RXD
bclr DQ_DIR,DDRA ; make the DQ pin on J1A input
ldx #8T ; set counter
READ
bclr CLK, SER_PORT ; CLK=0
brclr DQ, SER_PORT, J2 ; carry bit = DQ
J2
ror a ; put carry bit into AccA LSB
bset CLK, SER_PORT ; CLK=1
decx ; decrement counter
bne READ
bset DQ_DIR, DDRA ; make the DQ pin on J1A output
rts ; return from sub

*** Routine creates a ~50 ms routine with a 2MHz MCU internal bus for
*** NV memory to be set correctly
NV_WAIT
ldx #66T
J3
lda #255T
J4
deca ; 3
bne J4 ; 3
decx
bne J3
rts

*** VECTOR TABLE
ORG RESET
DW START
References

MC68HC705J1A Technical Data, document order number MC68HC705J1A/D, Motorola, 1996.

M68HC05 Applications Guide, document order number M68HC05AG/AD, Motorola, 1996.


HC05/08 Website

http://design-net.com/csic/welcome.htm

Development Tools Website