CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

FEATURE ARTICLE

Being Cool is Easy

A X-10 Temperature-Sensing Device

Donald Blake

Skip the expensive sunglasses and red sports car. According to Donald, all you need is a microcontroller, an X-10 temperature sensor, and some interfacing software to control a fan to stay cool on a hot summer day.

My experiments with Microchip's PIC line of microcontrollers began with the '16C84. The PIC16C84 is a good starting device. Its 1-KB EEPROM is quickly reprogrammed and inexpensive programming devices are commercially available or easily built from readily available plans.

The elemental structure of any embedded architecture consists of input, processing, and output. My initial experiments started with the output component. I used a 2-line by 16-character LCD module.

The LCD module, which contains an onboard controller, has a relatively simple interface. I used the Optrex DMC16207, although there are a number of other manufacturers with many sources for new and surplus devices. The interface is documented in Optrex's databook and there's a good technical document available online.

Now that I had a working output device capable of presenting up to 32 characters, I needed an input device. I've always been fascinated with temperature measurement and that's where I focused.

My first temperature sensor was an analog device. An external ADC as well as a voltage reference was required because these functions aren't built into the '16C84. Another downside to the analog device is that a constant current source is required when the sensor is placed at any significant distance from the ADC.

I discovered the Dallas Semiconductor DS1820 1-Wire digital thermometer shortly after implementing the analog sensor. The DS1820 provides a digital interface, eliminates several external components, gives better accuracy, and permits the sensor to be located a considerable distance from the microcontroller.

Using the '16C84, LCD display and the DS1820 digital thermometer, my first complete PIC project was a simple indoor/outdoor thermometer that recorded minimum and maximum temperature, and displayed readings in Fahrenheit and Celsius.

THE X-10 INTERFACE

I've used the X-10 line of home automation (HA) products for many

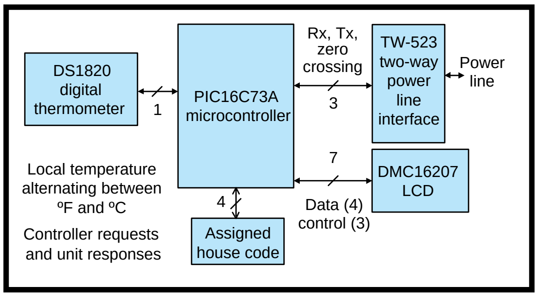
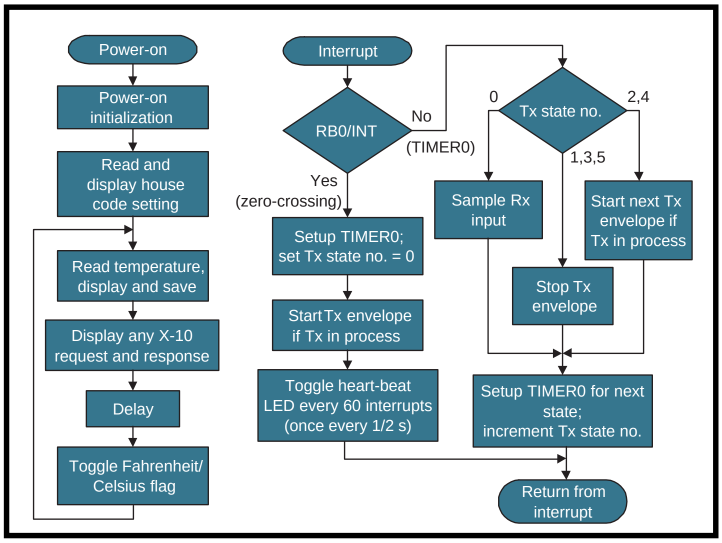

DS1820 digital thermometer
1
PIC16C73A microcontroller
Rx, Tx, zero crossing
3
TW-523 two-way power line interface
Power line
7
DMC16207 LCD
Data (4) control (3)
4
Assigned house code
Local temperature alternating between ºF and ºC
Controller requests and unit responses

Figure 1—*The PIC16C73A reads local temperature from the DS1820 and responds to controller requests via the TW-523. The optional LCD displays local temperature, controller requests, and sensor responses.*

years. I started with simple one-way “dumb” controllers and later migrated to the programmable CP-290 X-10 Powerhouse. My current HA setup uses the Homebase intelligent controller by Home Controls, Inc. I decided on an X-10 interface for the next step in my PIC experiments.

Homebase uses the X-10 TW-523 two-way power line interface module to send and receive X-10 commands. My one and only TW-523 module failed a short time back and left me to rely on my old faithful CP-290 for several days. I ordered a replacement TW-523 along with a few spares. These spare TW-523 modules enabled me to further my PIC experimentation.



Power-on
Power-on initialization
Read and display house code setting
Read temperature, display and save
Display any X-10 request and response
Delay
Toggle Fahrenheit/ Celsius flag
Interrupt
RB0/INT
No
(TIMER0)
Yes
(zero-crossing)
Setup TIMER0; set Tx state no. = 0
Start Tx envelope if Tx in process
Toggle heart-beat LED every 60 interrupts (once every 1/2 s)
Tx state no.
0
2,4
1,3,5
Sample Rx input
Start next Tx envelope if Tx in process
Stop Tx envelope
Setup TIMER0 for next state; increment Tx state no.
Return from interrupt

Figure 2—*These are the two main elements of the temperature sensor software. The foreground loop on the left manages the DS1820 and the LCD. The background interrupt-driven task on the right receives and transmits X-10 commands via the TW-523.*

PUTTING IT ALL TOGETHER

I now had to decide what to do with an X-10 interface. I certainly could make use of some sort of motion-sensing device. After some thought, I decided that a temperature sensing device would be a logical extension to my first project.

I also had a real need for such a device. My daughter’s bedroom on the second floor tends to get hot during sunny summer days. I had been using the Homebase controller to turn on a fan on summer afternoons.

Most of the time this solution was adequate. On occasion, however, when we had a cool, rainy day (not uncommon in central New York), the bedroom would get too cool.

The obvious solution was to turn the fan on in the afternoon only if the temperature was above a certain level. However, I’d also like to be able to manually override the fan control when I’m at home.

The more I thought about the requirements for this fan controller, it became obvious that an autonomous device didn’t make sense. What I needed was a “dumb” sensor device, which provided input to the Homebase controller.

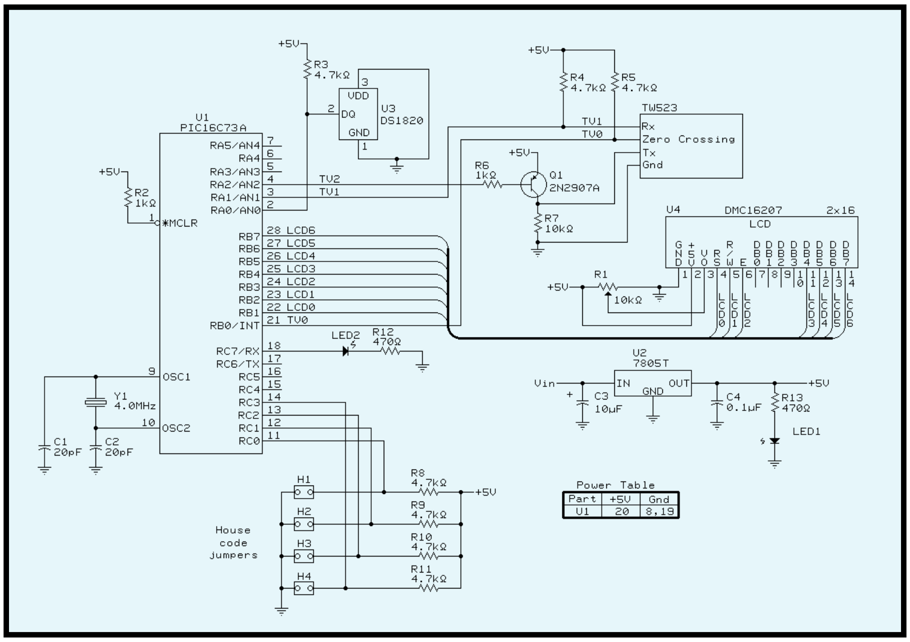
My first thought was to implement a remote temperature-sensing device that would send an X-10 command to the Homebase controller whenever the temperature transitioned through a predefined threshold.

There were two problems with this approach. First, the threshold would have to be set in the remote device. Second, I already had enough X-10 transmission devices and adding one (or more) additional device(s) that transmitted on their own would only increase the probability of a collision.

What I finally decided on was a device that would speak only when spoken to. The Homebase controller sends a query to the remote temperature sensor. This query specifies the temperature threshold. The remote temperature sensor then, and only then, responds (transmits) to indicate if the current temperature is below, at, or above the specified threshold.

The sensor is programmed for a single house code and responds to queries from the Homebase controller. These queries consist of two consecutive X-10 unit On commands. The sensor decodes these On commands to determine the temperature threshold value to use for the comparison.

The decoded value is compared to the actual sensed temperature. The sensor responds with a unit On command if the temperature is greater than or equal to the queried temperature value, or with an Off if not. The homebase controller then



Part	+5V	Gnd
U1	20	8,19

Figure 3—*The PIC16C73A is wired directly to the DS1820, LCD, and two of the three TW-523 signals. The TW-523 Tx input is controlled indirectly through Q1.*

takes the appropriate action based on the response received from the sensor.

An example of the protocol is illustrated in Table 1. The controller sends a 75°F query and the temperature sensor responds with a unit On command to indicate that the current temperature is equal to or greater than 75°F.

Note that an On command consists of two X-10 transmissions. Each transmission contains a House Code and a Key or Function Code. The Key Code of the first transmission identifies the unit (1 through 16) and the Function Code of the second transmission identifies the Command Code (On in this case).

OVERVIEW OF THE SENSOR

Refer to the block diagram of the X-10 temperature sensor in Figure 1. The PIC16C73A microcontroller is central to the sensor and controls its other elements. The PIC's on-chip program memory contains the X-10 temperature sensor software. There are two main elements of the software—a background component and a foreground component (see Figure 2).

The main foreground loop is entered on powerup after completing initialization. In this main loop, the temperature is read from the DS1820 digital thermometer once every 2 s and displayed on the LCD. The temperature display alternates between Fahrenheit and Celsius. The temperature is also saved for use by the background component.

The PIC's interrupt drives the background component, which controls the X-10 input and output interface via the TW-523 two-way power line interface module. There are two interrupt sources: an external interrupt generated from the TW-523 zero-crossing signal and an internal interrupt generated from PIC's Timer0.

The zero-crossing signal is used to synchronize the sampling of the TW-523 Rx output and control of the TW-523 Tx input. The software sets up the PIC's internal Timer0 to create a sequence of precise internal interrupts synchronized with the zero-crossing signal for this purpose.

The background component of the software monitors the TW-523 Rx output. X-10 transmissions are received bit-by-bit and reassembled. When a query request is recognized, it is compared to the temperature reading that was saved in the main foreground loop. The appropriate response is generated and subsequently transmitted bit-by-bit through control of the TW-523 Tx input (see Figure 5).

THE MICROCONTROLLER

When I started the X-10 temperature sensor, I had two different PIC microcontrollers on-hand—the '16C84 and the '16C73A (see Figure 3). The only choice was the '16C73A because the '16C84 had insufficient I/O.

The software for the X-10 temperature sensor was developed using Microchip's MPLAB Integrated Development Environment (IDE) in combination with Microchip's PICSTART Plus programmer. The MPLAB IDE software, information on the PICSTART Plus, and datasheets for the PIC microcontrollers are available from Microchip's web site.

THE LCD

The LCD is an Optrex DMC16207 2-line by 16-character display. The device operates with either a 4- or 8-bit bidirectional parallel data interface and three control lines.

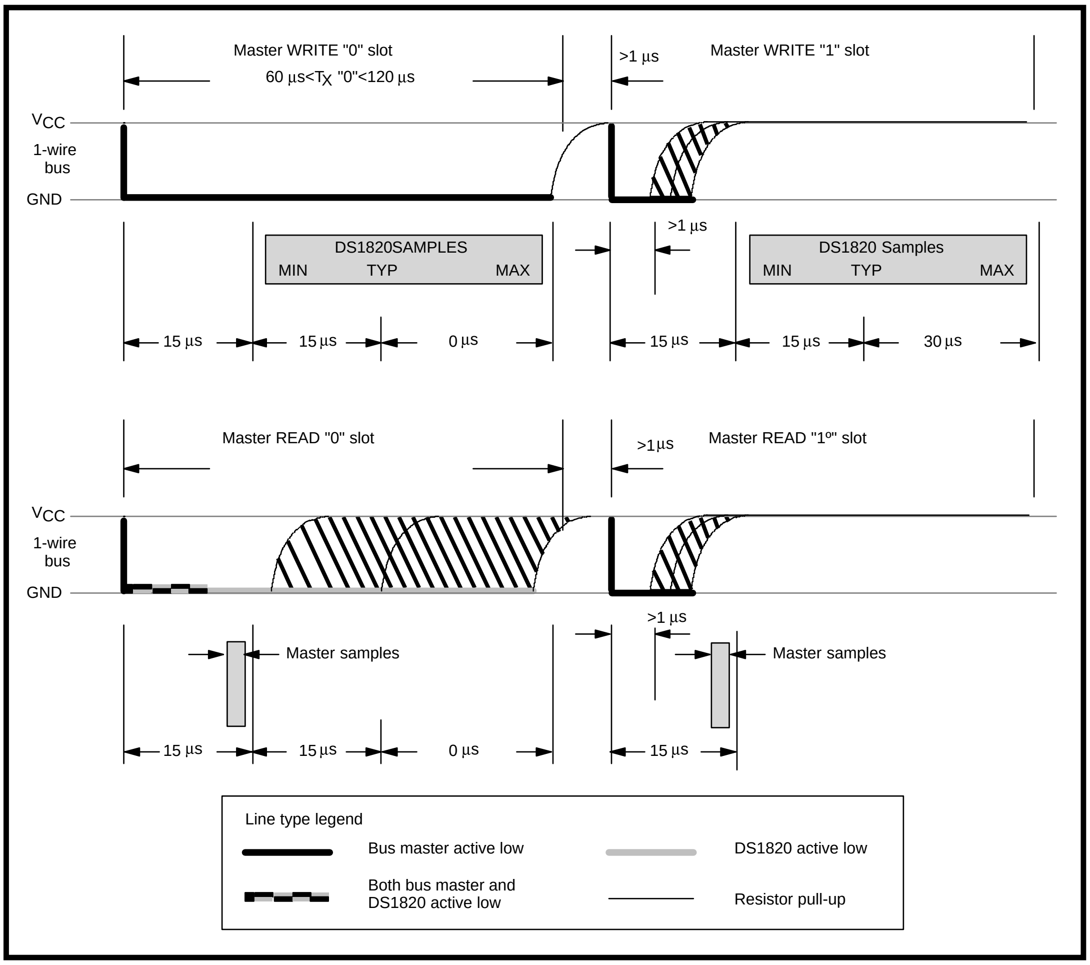


Figure 4— *Command bytes are written to and data bytes are read from the DS1820 a bit at a time. Each bit read or write slot takes 60 µs with at least 1 µs between slots.*

The X-10 temperature sensor initializes the LCD to its 4-bit mode. Only seven microcontroller I/O lines are required in 4-bit mode instead of 11, which would be required in 8-bit mode. An external 10 kΩ potentiometer controls LCD contrast.

The LCD accepts 8-bit ASCII data and control characters. In 4-bit mode, two consecutive output operations are necessary to transfer each ASCII or control character to the display. The four data lines are bidirectional and are also used to read busy status from the LCD.

The LCD is not necessary to the operation of the X-10 temperature sensor. It can be eliminated with no other modification to the hardware or software. When present, the LCD displays the selected house code at power-on, current local temperature (alternating between Fahrenheit and Celsius), received requests, and transmitted responses.

The local temperature is displayed on the first line of the LCD and the second line is displayed on the selected House Code or the X-10 controller query and transmitted responses.

The selected House Code is displayed for about 60 s following power-on. A query received from the X-10 controller and the response transmitted by the X-10 temperature sensor are displayed for about 120 s. The second line is blanked after the specified time period has elapsed. The LCD is controlled by the foreground software component.

Controller
- Transmit 75°F query:
- Unit A, Key Code 7
- Unit A, ON Command
- Unit A, Key Code 5
- Unit A, ON Command

Temperature sensor
- Compare current temperature to 75°F
- Transmit response:
- Unit A, Key Code 16
- Unit A, ON Command

Controller
- Take appropriate action for temperature at or above 75°F.

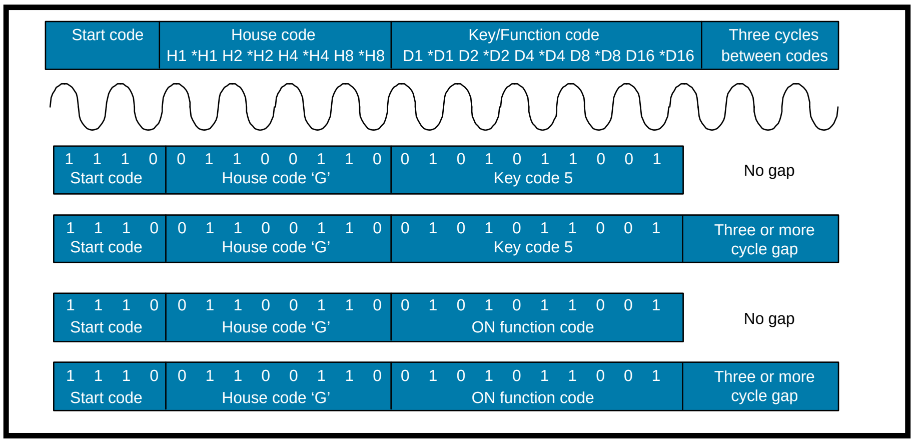
Table 1—*In this command protocol example, the X-10 controller sends a 75° query to the temperature sensor. The sensor responds to indicate that the temperature is at or above this value.*

The LCD was also invaluable during debug of the X-10 temperature sensor software. I used the second line of the display to output information to help troubleshoot problems.

TEMPERATURE SENSOR

The DS1820 1-Wire digital thermometer from Dallas Semiconductor is available in both a PR35 (3-pin) and a 16-pin SSOP package. The DS1820 provides a 9-bit digital value that represents the device's temperature in 0.5°C increments over a –55°C to +125°C range.

I used the PR35 package for the X-10 temperature sensor. There are three connections with either package: power, ground, and data in/out. An unusual feature of the device is that the power connection is optional. The power and ground pins



Start code
House code H1 *H1 H2 *H2 H4 *H4 H8 *H8
Key/Function code D1 *D1 D2 *D2 D4 *D4 D8 *D8 D16 *D16
Three cycles between codes
1 1 1 0 Start code
0 1 1 0 0 1 1 0 House code 'G'
0 1 0 1 0 1 1 0 0 1 Key code 5
No gap
1 1 1 0 Start code
0 1 1 0 0 1 1 0 House code 'G'
0 1 0 1 0 1 1 0 0 1 Key code 5
Three or more cycle gap
1 1 1 0 Start code
0 1 1 0 0 1 1 0 House code 'G'
0 1 0 1 0 1 1 0 0 1 ON function code
No gap
1 1 1 0 Start code
0 1 1 0 0 1 1 0 House code 'G'
0 1 0 1 0 1 1 0 0 1 ON function code
Three or more cycle gap

Figure 5—*Transmission of each X-10 command requires 11 power line cycles. Each command is transmitted twice with three or more cycles between pairs.*

can be tied together to operate the device in its parasite power mode.

In this mode, which I use in the X-10 temperature sensor, the DS1820 steals power from the data in/out pin when it's high. The data in/out line is wired to the 16C73A Port A bit 0 (RA0) pin.

The DS1820 temperature conversion cycle requires 500 ms. During this time, up to 1 mA is required and the data in/out pin must be held high by the PIC. The 4.7-kΩ pull-up resistor, used when reading from the DS1820, will not supply sufficient current during the conversion. The X-10 temperature sensor software performs the DS1820 read temperature sequence shown in Table 2.

The sequence begins with a device reset performed by pulling the data in/out line low for 720 µs (minimum 480 µs, maximum 960 µs). The DS1820 responds with a "device present" indication 15 to 60 µs following the release of the data in/out line. The DS1820 pulls the data in/out line low for 60 to 240 µs.

The X-10 temperature sensor software checks for the "device present" response. If no device is detected, a software flag is set to indicate that a temperature reading is not available, and the remainder of the read temperature sequence is skipped.

The sensor uses three of the six DS1820 commands. Each command consists of 8-bits, which are written serially a bit at a time. The bit write timing is illustrated in Figure 4. To write a one, the PIC drives the data in/out line low for at least 1 µs and then drives the line high.

The DS1820 samples the data in/out line between 15 and 45 µs after the PIC first drives it low. To write a zero, the PIC drives the data in/out line low and keeps it low for at least 60 µs. The PIC drives the data in/out line high for at least 1 µs between bits.

Each DS1820 has a unique serial number in built-in ROM that enables several devices to be wired together on the same 1-wire bus. A `match ROM` command identifies the serial number of the selected device.

The X-10 temperature sensor uses the DS1820 in a single-drop configuration. The `skip ROM` command is used to select the device in a single-drop configuration. It works like the `match ROM` command without needing to provide the serial number.

The `convert T` command instructs the DS1820 to begin a temperature conversion cycle. The conversion requires a maximum of 500 ms during which the data in/out pin must be held high when using the parasite power mode.

The temperature conversion is read from the DS1820's scratchpad memory by the last four steps in the sequence. The DS1820 sends one CRC byte following the eight bytes of scratchpad memory. This byte can be used to validate the data transfer.

The sensor software reads these nine bytes, one bit at a time. The software saves the first two bytes, which contain the 9-bit temperature value. The remaining six bytes and the CRC value are discarded. The DS1820 datasheet further details the reset, read, and write cycles, and has information pertaining to additional features, including the content of the remaining scratchpad memory locations.

The bit-read timing is also illustrated in Figure 4. To read each bit, the PIC first drives the data in/out line low for at least 1 µs and then puts the RA0 pin in the high-impedance mode.

The DS1820 will drive the data in/out line low to output a zero or put it in a high-impedance mode to output a one. The pull-up resistor causes the data in/out line to go high when it's in the high-impedance mode. The PIC samples RA0 15 µs from the time it drives the line low.

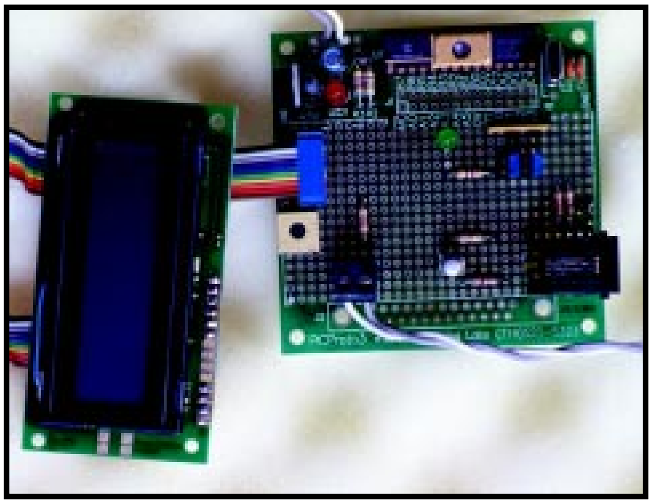


Photo 1—*All components except the LCD, DS1820 digital thermometer, and TW-523 two-way power line interface are mounted on the printed circuit board. The TW-523 cable plugs into the modular jack at lower right. The DS1820 is connected via the twisted pair running from the terminal block at the lower left.*

POWER LINE INTERFACE

The TW-523 interface is used to send and receive X-10 codes via the AC power line. A technical note is available online from X-10. For those of you who are long-time subscribers to *Circuit Cellar*, the X-10 protocol and theTW-523 module were described by Ken Davidson in issues 3 and 5.

An X-10 command consists of a start code, a House Code, and a Key or Function Code. Transmission of each command requires 11 power line cycles: two for the Start Code, four for the House Code, and five for the Key or Function Code.

Each command is transmitted twice with at least three power line cycles between pairs. The four House Code bits and the five Key or Function Code bits are transmitted in true and complement form on alternating half cycles of the power line.

Here's an example. To turn on unit 5 of House Code G requires the following X-10 commands:

Start Code (1110), House Code G (0101), Key Code 5 (00010)
Start Code (1110), House Code G (0101), ON Function Code (00101)

Figure 5 illustrates the transmission of these two X-10 commands. The House Code, Key and Function Code encoding is defined in the X-10 technical note.

ZERO CROSSING DETECT

The X-10 transmission must be synchronized with zero crossing on each half cycle of the power line. This is the point when the AC voltage goes from positive to negative or from negative to positive. The TW-523 provides a zero-crossing output. This output is a 60-Hz square wave and is connected to the 16C73A RB0/INT pin.

The X-10 sensor software configures the RB0/INT pin to generate an interrupt. RB0/INT can be configured to select either the rising edge or the falling edge. An interrupt on both edges is desirable and is achieved by toggling the edge select (INTEDG bit in the '16C73A OPTION register) just after each zero-crossing interrupt.

The software uses the '16C73A Timer0 in combination with the zero-crossing interrupt to provide the precise timing necessary for sampling of incoming and gating of outgoing X-10 commands. This timing sequence is illustrated in Figure 6 and described in more detail in the following sections.

The zero-crossing interrupt is also used to blink the heartbeat LED (LED2) at a 1-Hz rate, which provides a warm fuzzy indication that the temperature sensor and TW-523 module are alive and well.

RECEIVING REQUESTS

Each X-10 command is transmitted twice. The TW-523 only provides the second of the two transmissions via its Rx output. The Rx output is valid between 500 and 700 µs after zero crossing. The software sets up Timer0 to generate six interrupts after each zero crossing. The first Timer0 interrupt occurs at 600 µs after zero crossing and the Rx output is sampled at this time.

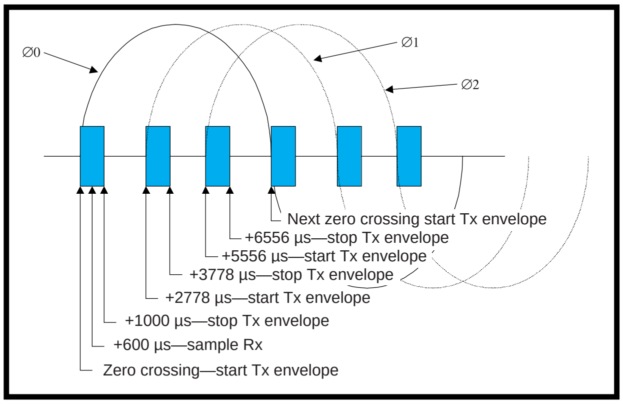
Remember that it takes 11 power line cycles or 22 zero-crossing events for a complete X-10 transmission. Four samples are required for the Start Code, eight samples for the House Code, and 10 for the Key or Function Code.

The temperature sensor looks for a consistent X-10 transmission which consists of:

- valid start code: 1 1 1 0
- four House Code bits in true and

complement form
- five Key or Function Code bits in true or complement form

The incoming X-10 command and any earlier received commands are discarded if any inconsistencies are detected and the software starts over looking for a valid start code.



Ø0
Ø1
Ø2
Next zero crossing start Tx envelope
+6556 µs—stop Tx envelope
+5556 µs—start Tx envelope
+3778 µs—stop Tx envelope
+2778 µs—start Tx envelope
+1000 µs—stop Tx envelope
+600 µs—sample Rx
Zero crossing—start Tx envelope

Figure 6—*Transmission of each bit of an X-10 command starts at the zero crossing and lasts for 1000 µs. There are three transmissions of each bit, one for each phase of a three-phase setup. Received data is sampled at 600 µs after zero crossing.*

PROCESSING REQUESTS

The X-10 sensor recognizes a request when it receives two consecutive On commands for the selected House Code. Remember from the example in Table 1 that one On command requires two X-10 transmissions. The first provides the Key Code and the second provides the Function Code (On in this case). Combining the two Key Codes forms the query value. In the example of Table 1, the Key Codes of 7 and 5 represent a query value of 75°F.

The query value is compared to the last local temperature read from the DS1820, if any. If the local temperature is greater than or equal to the query value, a response of Unit 16 On is created. Otherwise, a response of Unit 16 Off is created. An internal software flag is set indicating that a response is ready to be transmitted.

If the sensor detects an error during DS1820 communication, all queries are ignored and no response is generated or transmitted.

TRANSMITTING RESPONSES

As in receiving X-10 commands, transmission must be synchronized to the zero crossing event. The TW-523 Tx input controls when a high-frequency (120 KHz) carrier is superimposed onto the power line. A "1" is represented by a 120-KHz burst and a "0" is represented by no burst. The TW-523 Tx input is driven by the 16C73A RA2 output through Q1.

For single-phase residential power, this burst begins at zero crossing and lasts for 1 ms. A three-phase setup requires three 1-ms bursts during each half cycle. The temperature sensor is designed to work in a three-phase setup.

The first burst begins at zero crossing. Subsequent Timer0 interrupts control when that burst ends or when the remaining two bursts begin and end. See Figure 6 for the exact timing.

The TW-523 doesn't handle the duplicate transmissions on outgoing commands as it does for incoming transmissions. The X-10 software must handle transmitting each command twice with a gap of at least three cycles between pairs.

I use 10 half cycles between groups so it takes a total of 108 half cycles (zero crossing interrupts) to transmit a complete response. Breaking down the 108 figure, you get 44 half cycles per group of two (11 cycles × 2 zero crossings/cycle × 2 transmissions), 10 half cycle gaps between codes, × 2 codes (Unit/Key Code and Unit/Function Code).

Reset device, check device present response
Send `skip ROM` command
Send `convert T` command
Delay 500 milliseconds (while holding data in/out high)
Reset device, check device present response
Send `skip ROM` command
Send `read scratchpad` command
Read scratch pad (8-bytes plus one CRC byte)

Table 2—*The temperature sensor uses 3 of the 6 DS1820 commands to read the temperature. The skip ROM command is used to select the device in a single-device configuration.*

SYNCHRONIZATION OF TEMPERATURE MONITORING

Remember from the high-level software flow in Figure 2 that DS1820 communication takes place in the main foreground loop and the interrupt-driven background loop deals with the TW-523. The DS1820 communication requires precise timing. A zero crossing or Timer0 interrupt during DS1820 communication would disturb this timing.

One way to deal with this would be to disable interrupts during the periods of time-critical DS1820 communication. However, this technique disturbs the precise timing required for the TW-523 interface.

The solution is to synchronize the time-critical DS1820 code with the zero crossing and Timer0 interrupts. Note in Figure 6 that there are three intervals in each half cycle between transmission bursts where no interrupts occur. The intervals are 1.778 ms long—more than enough time to deal with time-critical DS1820 communication.

An interrupt synchronization subroutine is called just before all time-critical DS1820 code. This subroutine sets an internal flag then waits for that flag to be reset.

The '16C73A interrupt handler resets this flag at the start of each 1.778-ms interval. The interrupt synchronization subroutine has a 3-ms timeout for protection against a situation where the zero crossing interrupt has been lost.

PROTOTYPE CONSTRUCTION

I built the X-10 temperature sensor on a microEngineering Labs PICProto3 prototyping board. The main components of the sensor are mounted on the PICProto3. This PCB provides traces and pads for the standard PIC components such as power supply circuitry, microcontroller and the oscillator circuitry.

The remaining temperature sensor components (with the exception of the LCD, digital thermometer, and TW-523 module) are mounted in the prototyping area of the PICProto3. Direct point-to-point wiring using 26-AWG solid wire was implemented.

The LCD is connected to the PICProto3 by a 10-wire ribbon cable and header connector. The digital thermometer is wired to a twisted pair cable connected via a terminal block. The TW-523 is connected by a modular cable, which plugs into a jack mounted on the PICProto3.

APPLICATIONS

The X-10 temperature sensor was designed to aid in control of a house fan in a remote location. This device can also be used in other applications where it's desirable to determine temperature in relation to a threshold. Applications that come to mind are:

- monitoring freezer temperature to generate an alarm if the temperature is too high
- monitoring plumbing temperature to generate an alarm and/or turn on a heater if the temperature gets too low

For applications where it's desirable to obtain a remote temperature, not just the relationship to a given threshold, the software could be easily modified to transmit the local temperature upon request. The temperature could be encoded using standard X-10 Key and Function Codes. The X-10 controller could then, for example, record hourly temperatures and daily high and low temperatures, as desired.

Although the X-10 sensor provides feedback for a closed-loop environment, care should be exercised as with any X-10 application. Because X-10 control is not 100% reliable, it shouldn't be employed in a situation where safety might be compromised. For example, an electric heater should never be controlled by X-10 alone if it could cause a fire if left on too long.

COOLING DOWN

The temperature sensor provides remote temperature measurement without wires, using only standard X-10 Key and Function Codes. The Extended Code/Data capabilities of the X-10 standard are not required. In addition, the device uses only a single X-10 House Code.

Many other possibilities exist for this temperature sensor. The protocol could be expanded to share a single House Code among multiple sensors. The LCD could be left out to reduce the number of I/O required and permit the sensor to be used with an inexpensive microcontroller such as a PIC16C84. This technique could even be applied to motion detectors, light-level sensors, or other types of sensors. ◼

Don Blake has 30 years experience in avionics embedded systems and diagnostic software development. He is a senior programmer at Lockheed Martin Federal Systems in Owego, NY. You may reach him at donblake@worldnet.att.net.

SOFTWARE

The complete source code for this project is available for download via the *Circuit Cellar* web site.

REFERENCES

David Tait's 16C84 programmer plans, www.man.ac.uk/~mbhstdj/files/pic84v05.zip

LCD FAQ, ftp.ee.ualberta.ca /pub/cookbook/faq/lcd.doc

TW-523 manual, www.x10.com/support/support_manuals.htm

K. Davidson, "Power-Line-Based Computer Control", *Circuit Cellar* 3, May/June 1998.

K. Davidson, "The X-10 TW523 Two-Way Power Line Interface" *Circuit Cellar* 5, Sept/Oct 1988.

SOURCES

PIC16C84
Microchip Technology, Inc.
(888) 628-6247
(480) 786-7200
Fax: (480) 899-9210
www.microchip.com

DMC16207
Optrex
(313) 471-6220
Fax: (313) 471-4767

DS-1820
Dallas Semiconductor
(972) 371-4448
Fax: (972) 371-3715
www.dalsemi.com

TW-253
X-10 USA
(800) 675-3044
(201) 784-9700
Fax: (201) 784-9464
www.x10.com

PICProto3
microEngineering Labs, Inc.
(719) 520-5323
Fax: (719) 520-1867
www.melabs.com

PIC development tools
Dontronics
(613) 9338-6286
Fax: (613) 9338-2935
www.dontronics.com