M. Akers Enterprises

3800 Vineyard Ave #E
Pleasanton, California 94566
Voice: +1-925-640-3600
Email: mwakers@home.com

National Semiconductor LM75 Digital Temperature Sensor

Using the LM75 with an 8-bit microcontroller I^2C interface.

Created by Michael W. Akers, M. Akers Enterprises, October 18, 1999

Section

1

The LM75

What exactly is the LM75

Overview

The LM75 was created by National Semiconductor Corporation to fill a crying need in the PC industry for detecting over temperature conditions in a personal computer. This device has a built-in 9 bit ADC that will convert the analog thermal reference to a digital value usable by the PC. But for this document we are going to explore how to access this device using a Atmel 89C4051 flash microcontroller using the MCS Electronics BASCOM-8051 basic language compiler.

Communication with the LM75 is through the I^2C interface (created by the Phillips Corporation). This is a 2-wire communications protocol used to communicate serially with various types of devices similarly configured. The LM75, designed as a 'slave' device, can be configured through the I^2C interface to alert, through various methods, the PC system that an over temperature condition has happened.

At certain times, direct quotations or excerpts of the LM75 Datasheet[1] will be reproduced in this document. The information will be displayed with a gray background, and will contain the page of the LM75 Datasheet. The LM75 Datasheet will, at all times, be the sole authority. If there is a conflict of information, the LM75 Datasheet will be considered the correct source.

Scope

It is the intent of this document to show the relative ease to which the LM75 Digital Temperature Sensor can be utilized. This author has reviewed several programs that use different programming languages to interact with the LM75 DTS. Except for the MCS Electronics BASCOM-8051 Basic Compiler, all other high level languages require a large amount of code to perform a simple task (although the compiled object code is as small as can be). BASCOM-8051 is 99% syntax compatible with Microsoft Qbasic, thus anyone who can program using MS Qbasic can program a microcontroller. MCS Electronics BASCOM-8051 Basic Compiler has built-in commands for handling the I^2C protocol, making the coding process much easier.

The programming examples given in this document will be presented as subroutines that you can use in your project. Appendix A will contain an example program that will read the temperature from the LM75 DTS and display it on a LCD display module.

The LM75

The easiest way to think of the LM75 is as a digital thermal alarm clock. You can read the thermal time, set the thermal alarm, and it's thermal snooze button. You can configure the device to switch on a fan or an audible alarm. As seen in the diagram below the LM75 is a fairly simple device.

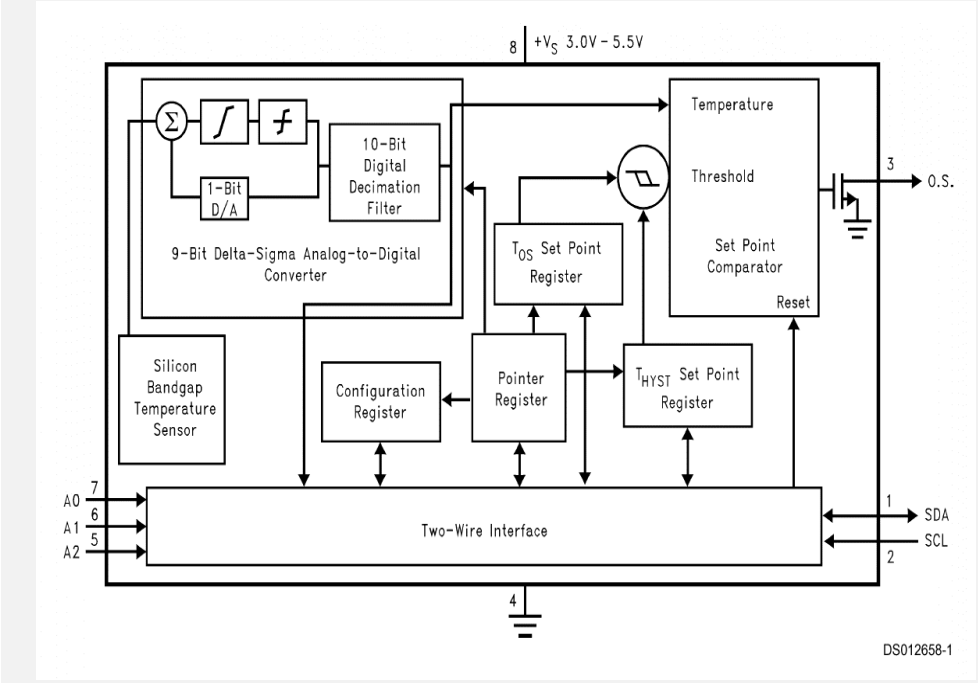


Figure 1. Block diagram of LM75 device. (extracted from page 1 of the LM75 Datasheet)

The Lm75 has four registers that you can read and write to, depending upon what you want the device to do. Mostly though, you will be reading the temperature. Upon power up the device is set to the default mode:

- Comparator Mode
- T_{OS} = 80° C
- T_{HYST} = 75° C
- O.S. Active Low
- Pointer = "00"

The LM75 registers are accessible through the I^2C port. This port is comprised of Pin 1 'SDA' (Serial DAta) and Pin 2 'SCL' (Serial CLock). Also, since this device is addressable (you can have up to eight devices on the I^2C bus), you have three address pins; Pin 7 'A0', Pin 6 'A1', and Pin 5 'A2'. For purposes of simplicity, A0 to A2 are considered tied to

ground. A thorough reading of the LM75 Datasheet will aquatint you with the operational fundamentals of the device.

There are some apparent errors in the LM75 Datasheet. As stated, the errors are only apparent and not actual. The errors, if they are errors, are in the omission of information. The following is how the information should have been presented:

Address word format: (the waveforms in the LM75 Datasheet are correct)

1	0	0	1	A2	A1	A0	R/W
MSB							LSB

- The address for a read function would be 1001XXX1
- The address for a write function would be 1001XXX0

This is the correct way to represent the address word.

Many people who read the LM75 Datasheet often make the mistake of thinking that the registers in the LM75 are accessed as a 16-bit word. Not, as the waveforms clearly delineate, as two 8-bit words, in high byte, low byte, order. So most people make the mistake of using the read word or write word I^2C function when communicating with the LM75. Although the registers are 16-bit, the access method is 8-bit!

Conclusion:

In the next section, I will be going through each register and, with code, explain how to access and interpret the information.

Section

2

Lm75 I/O

Basic Communications

Temperature

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
SGN	MSB	0	0	0	0	0	0	LSB	X	X	X	X	X	X	X
MSB							LSB	MSB							LSB
High Byte								Low Byte							

In the table above is the accurate way to think of the temperature register. Remember that upon power-up the device is in the default mode, (Pointer = '00') read temperature. With this in mind, lets look at the table above.

The temperature register is 16 bits wide. Only bits D7 through D15 mean anything. The High Byte of the temperature word contains the Sign bit (D15) and the whole value temperature, and the Low Byte contains the fractional temperature (D7). The rest of the bits in the Low Byte have no meaning (D0 to D6).

To read the LM75 temperature will require sending the address and read command to the LM75 and then reading the High Byte and then the Low Byte. In addition, the temperature data is in two's compliment form. This means that when the temperature goes below 0°C the sign bit is set and the data is now a two's compliment of the actual temperature value.

Two's compliment simply means that the information is negated. (Real clear isn't it?) As an example, zero is the compliment of one, and one is the compliment of zero. The compliment of falling down is falling up. Get the picture? The math for it though is a little tricky. In doing the math we loose 1 LSB, so it must be replaced when converting back.

Original	00011001
Compliment	11100111

As seen in the example above the compliment to 00011001 (Hex 19) is 11100111 (Hex E7). But if we negate 11100111 we get 00011000 (Hex 18). But we know that we lost 1 LSB when the conversion was performed, so we just add it back. So 00011000 (Hex 18) +

1 gives us 00011001 (Hex19), the original number. (Believe it or not, this gives first year Computer Science students NIGHTMARES!)

So with the above in mind, lets get to the code! In BASCOM-8051 syntax, you must declare the Subroutines and the variables associated with them.

```
Declare Sub Readlm75(Lm75addr as Byte)
Dim Lm75read As Byte                    ' Read address base
Dim Lm75write As Byte                   ' Write address base
Dim Lm75addr As Byte                    ' Lm75 Address
Dim Lm75high As Byte                    ' Lm75 High Byte
Dim Lm75low As Byte                     ' Lm75 Low Byte
Dim Lm75sign As Bit                     ' Lm75 Sign Bit
```

Now setup the Read and write addresses.

```
Lm75read = &B10010001
Lm75write = &B10010000
```

And now the subroutine!

```
Sub Readlm75(Lm75Addr)
    Call Setaddr Lm75addr, 0
    I2cstart                            ' Start the I2C process.
    I2cwbyte Lm75read, 8                ' Send the LM75 address and read Info.
    I2crbyte Lm75high , 8               ' Get 8 bits and get ACK from LM75
    I2crbyte Lm75low , 9                ' Get 8 bits and send NACK to LM75
    I2Cstop                             ' Stop the I2C system
    If Lm75high > 127 Then
            Lm75high = Not Lm75high     ' Flip the bits Bob!
            Incr Lm75high               ' Add 1 to the value
            Lm75sign = 1                ' Yup, we be negative!
    End If
    Lm75low = Lm75low And &B10000000    ' Mask out the lower 7 bits.
    If Lm75Low = &B10000000 Then
            Lm75low = 5
    End If
    Lm75read = &B10010001               ' Reset Lm75read
End Sub
```

This leaves the variables Lm75high with the whole temperature value, Lm75low with the fractional temperature value, and the Lm75sign bit denoting the polarity of the temperature value.

Pointer

P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	Register Select	

As seen above the Pointer register is 8 bits in length. The Pointer register is used to Point to the register that you would like to read from or write to. The only bits that we need to concern ourselves with are P0 and P1. Bits P2 to P7 are for test purposes only, and a write operation into these registers could damage or destroy the Lm75. So, leave them alone.

The breakdown of the Pointer register is as follows:

P0	P1	Register Pointed To
0	0	Temperature (Read only)(Power-up default)
0	1	Configuration (Read/Write)
1	0	T_{HYST} (Read/Write)
1	1	T_{OS} (Read/Write)

As it is unnecessary to specifically set the pointer before performing a Read/Write operation on a different register as a separate process, I will not create a subroutine to access this register.

Configuration

D7	D6	D5	D4 D3	D2	D1	D0
0	0	0	Fault Queue	O.S. Polarity	Cmp/Int	Shutdown

The Configuration register controls the O.S polarity, whether or not the output is in Comparator or Interrupt mode, fault queue (for noisy environments), and a way to put the device to sleep.

Shutdown Mode:

To put the device in shutdown mode D0 is set. The device goes into a quiescent but nominally active state and will draw around 1 μA. The I^2C bus remains active and you can still Read/Write the Configuration, T_{HYST}, and T_{OS} registers. Resetting theis bit will bring the device back up to full operation.

Comparator/Interrupt Mode:

With D1 = 0 the device is in comparator mode. When the temperature goes above the T_{OS} value, the O.S. will go low. When the temperature falls below the T_{HYST} value then the O.S. goes high. This assumes that the Polarity bit is low. With D1=1, the device is in Interrupt mode. When the temperature goes above the T_{OS} value, the O.S. will start

pulsing low. To reset the Interrupt the temperature must be below T_{OS} and any device read function will then reset the Interrupt.

O.S. Polarity:

With D2=0 (default) the O.S. output is Active Low, when D2=1 then O.S. output is Active High.

Fault Queue:

D4	D3	Number of Faults
0	0	1 (default)
0	1	2
1	0	4
1	1	6

As the temperature nears the TOS level, any noise will cause the device to false trigger. The Fault Queue allows you to determine how many times a false trigger happens before the device triggers for real.

Now for the code!

```
Sub Getconfig(lm75addr)
  Call Setaddr Lm75addr , 2
  I2cstart
  I2cwbyte Lm75write                ' Send write address
  I2cwbyte &B00000001 , 8           ' Set Pointer to point to Configuration register
  I2cstart                          ' Restart I2C
  I2cwbyte Lm75read                 ' Send read address
  I2crbyte Lm75config , 9           ' Read the config register
  I2cstop
  Resetpointer                      ' Reset the pointer to '00'
End Sub

Sub Setconfig(lm75addr)
  Call Setaddr Lm75addr , 1
  I2cstart                          ' Start I2C
  I2cwbyte Lm75write                ' Send write address
  I2cwbyte &B00000001 , 8           ' Set Pointer to point to Configuration register
  I2cwbyte Lm75config               ' Send the Lm75config byte to write
  I2cstop
  Resetpointer                      ' Reset the pointer to '00'
End Sub
```

T_{OS} and T_{HYST}

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
SGN	MSB	0	0	0	0	0	0	LSB	X	X	X	X	X	X	X
MSB							LSB	MSB							LSB
High Byte								Low Byte							

T_{OS} Register

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
SGN	MSB	0	0	0	0	0	0	LSB	X	X	X	X	X	X	X
MSB							LSB	MSB							LSB
High Byte								Low Byte							

T_{HYST} Register

The T_{OS} and T_{HYST} registers are exactly like the Temperature register except that you can write to them. Everything that pertains to the temperature register pertains to the T_{OS} and T_{HYST} registers.

On power-up the TOS register contains &B0101000000000000 (Hex 5000) or 80°C. And the THYST register contains &B0100101100000000 (Hex 4800) or 75°C. The following four subroutines will show how to set and get data on these registers.

```
Sub Gettos(lm75addr)
  Call Setaddr Lm75addr , 2
  I2cstart
  I2cwbyte Lm75write                    ' Send write address
  I2cwbyte &B00000011 , 8               ' Set Pointer to Tos
  I2cstart
  I2cwbyte Lm75read                     ' Send read address
  I2crbyte Lm75toshi , 8                ' Read Tos high byte
  I2crbyte Lm75toslo , 9                ' Read Tos low byte
  I2cstop
  Resetpointer                          ' Reset the pointer to '00'
End Sub

Sub Settos(lm75addr)
  Call Setaddr Lm75addr , 1
  I2cstart
  I2cwbyte Lm75write                    ' Send write address
  I2cwbyte &B00000011                   ' Set Pointer to Tos
  I2cwbyte Lm75toshi                    ' Send Tos high byte
  I2cwbyte Lm75toslo                    ' Send Tos low byte
  I2cstop
```

```
  Resetpointer                          ' Reset the pointer to '00'
End Sub

Sub Getthyst(lm75addr)
  Call Setaddr Lm75addr , 2
  I2cstart
  I2cwbyte Lm75write                    ' Send write address
  I2cwbyte &B00000010 , 8               ' Set pointer to Thyst
  I2cstart
  I2cwbyte Lm75read                     ' Send read address
  I2crbyte Lm75thysthi , 8              ' Read Thyst high byte
  I2crbyte Lm75thystlo , 9              ' Read Thyst low byte
  I2cstop
  Resetpointer                          ' Reset the pointer to '00'
End Sub

Sub Setthyst(lm75addr)
  Call Setaddr Lm75addr , 1
  I2cstart
  I2cwbyte Lm75write                    ' Send write address
  I2cwbyte &B00000010                   ' Set pointer to Thyst
  I2cwbyte Lm75thysthi                  ' Send Thyst high byte
  I2cwbyte Lm75thystlo                  ' Send Thyst low byte
  I2cstop
  Resetpointer                          ' Reset the pointer to '00'
End Sub
```

Conclusion:

Well, that's it! Not as hard as you were thinking was it. Using the I^2C interface code in BASCOM-8051 makes accessing and controlling the LM75 a snap.

Appendix A contains the full core code file, along with a schematic of a circuit that will work with the code. All you need to do now is complete the program to read and write to the LM75.

Appendix

A

LM75 Core Code

```
'--------------------------------------------------------------------------------
' Program Name           : Lm75Full.bas
' Program Date           : October 15,1999
' Program Written By     : M. Akers Enterprises
'                          Michael W. Akers
'                          3800 Vineyard Ave. #E
'                          Pleasanton, California 94566
'                          Voice: +1 925 484 4750
'                          Email: mwakers@home.com
' Program Purpose        : This program will demonstrate how to interface to,
'                          and communicate With The National Semiconductor LM75
'                          Digital Temperature Sensor.
' Target Processor       : Atmel 89C52
'--------------------------------------------------------------------------------
' Programmer           Date        Comments
' -------------------  ----------  ------------------------------------------
' Michael Akers        10/15/99    Initial creation of program.
' -------------------------------------------------------------------------------
' Define the processor (the regfile goes here!)
   $regfile = "8052.DAT"
' Define all meta-commands that must beinserted before all other commands.
' Define all subroutines
   Declare Sub Readlm75(lm75addr As Byte)
   Declare Sub Getconfig(lm75addr As Byte)
   Declare Sub Setconfig(lm75addr As Byte)
   Declare Sub Gettos(lm75addr As Byte)
   Declare Sub Settos(lm75addr As Byte)
   Declare Sub Gethyst(lm75addr As Byte)
   Declare Sub Sethyst(lm75addr As Byte)
   Declare Sub Setaddr(lm75addr As Byte , Flagrw As Byte)
   Declare Sub Resetpointer()
   Declare Sub Val2temp(lm75tmphi As Byte , Lm75tmplo As Byte , Lm75tmpsign As Bit)
' Define all variables and constants
   Dim Lm75read As Byte                        ' Read address base
   Dim Lm75write As Byte                       ' Write address base
   Dim Lm75addr As Byte                        ' Lm75 Address
   Dim Lm75high As Byte                        ' Lm75 Temperature High Byte
   Dim Lm75low As Byte                         ' Lm75 Temperature Low Byte
   Dim Lm75sign As Bit                         ' Lm75 Temperature Sign Bit
   Dim Lm75config As Byte                      ' Lm75 Configuration
   Dim Lm75toshi As Byte                       ' Lm75 Tos high byte
   Dim Lm75toslo As Byte                       ' Lm75 Tos low byte
   Dim Lm75tossign As Bit                      ' Lm75 Tos sign bit
   Dim Lm75thysthi As Byte                     ' Lm75 Thyst high byte
   Dim Lm75thystlo As Byte                     ' Lm75 Thyst low byte
   Dim Lm75thystsign As Bit                    ' Lm75 Thyst sign bit
```

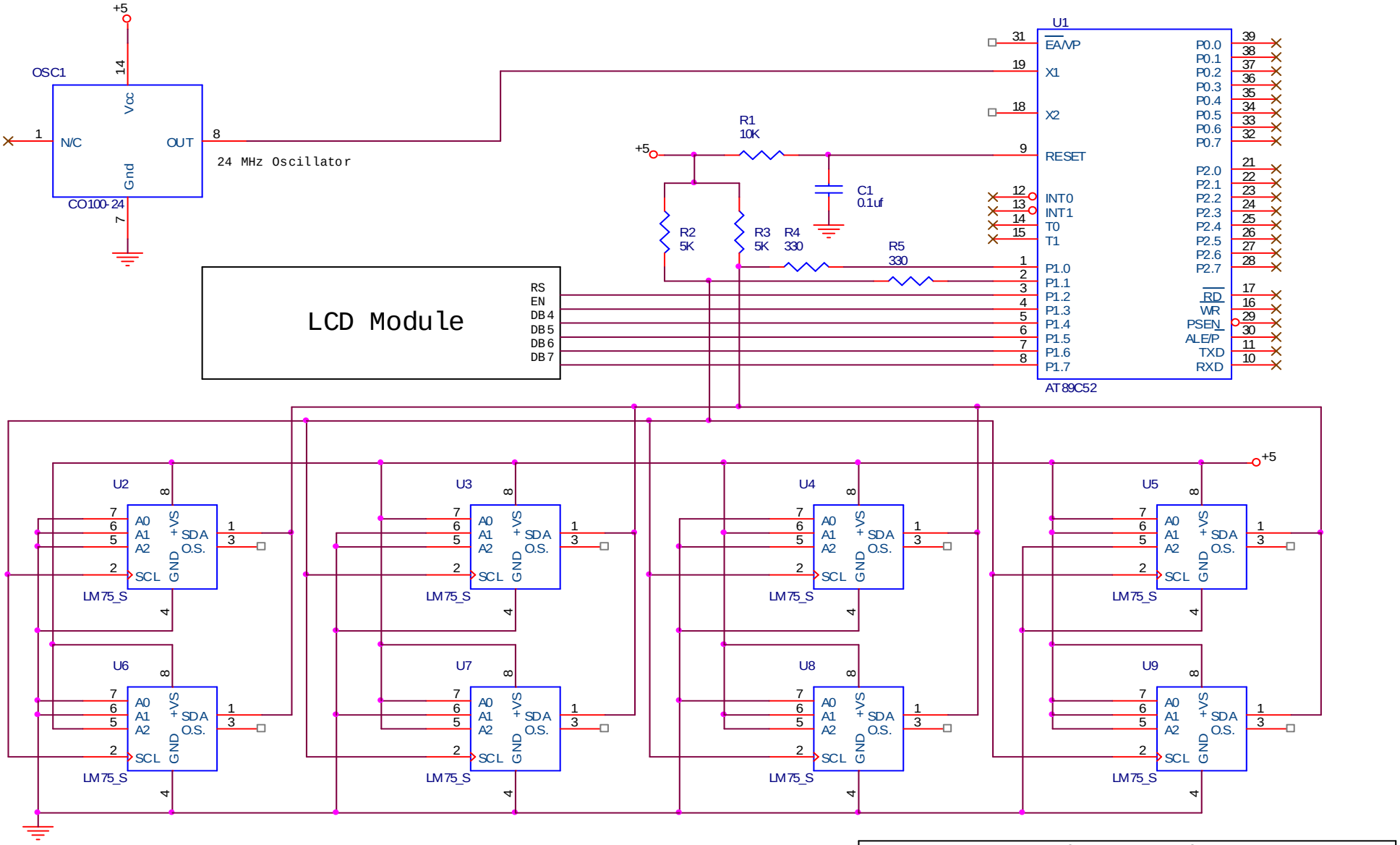
```
   Dim Flagrw As Byte                          ' Read/Write Flag
   Dim Lm75tmphi As Byte                       ' Lm75 Temp hi byte
   Dim Lm75tmplo As Byte                       ' Lm75 Temp lo byte
   Dim Lm75tmpsign As Bit                      ' Lm75 Temp sign bit
' Define all Configurations and Pin assignments
   Config Sda = P1.0
   Config Scl = P1.1
   Config I2cdelay = 1
   Config Lcd = 40 * 4
   Config Lcdpin , Db4 = P1.4 , Db5 = P1.5 , Db6 = P1.6 , Db7 = P1.7 , E = P1.3 , Rs = P1.2
' Initialize variables as needed.
   Lm75read = &B10010001
   Lm75write = &B10010000
' Program start.
Start:



Goto Start
'Program end.
' Begin subroutine section.
Sub Readlm75(lm75addr)
   Call Setaddr Lm75addr , 0
   I2cstart                                    ' Start the I2C process.
   I2cwbyte Lm75read                           ' Send the LM75 address and read Info.
   I2crbyte Lm75high , 8                       ' Get 8 bits and get ACK from LM75
   I2crbyte Lm75low , 9                        ' Get 8 bits and send NACK to LM75
   I2cstop                                     ' Stop the I2C system
   Call Val2temp Lm75high , Lm75low , Lm75sign ' Convert the value (if needed)
   Lm75high = Lm75tmphi
   Lm75low = Lm75tmplo
   Lm75sign = Lm75tmpsign
   Lm75read = &B10010001                       ' Reset Lm75read
End Sub
Sub Getconfig(lm75addr)
   Call Setaddr Lm75addr , 2
   I2cstart
   I2cwbyte Lm75write                          ' Send write address
   I2cwbyte &B00000001 , 8                     ' Set pointer register to point
                                               ' at the configuration register.
   I2cstart                                    ' Restart I2C
   I2cwbyte Lm75read                           ' Send read address
   I2crbyte Lm75config , 9                     ' Read the config register
   I2cstop
   Resetpointer                                ' Reset the pointer to '00'
End Sub
Sub Setconfig(lm75addr)
   Call Setaddr Lm75addr , 1
   I2cstart                                    ' Start I2C
   I2cwbyte Lm75write                          ' Send write address
   I2cwbyte &B00000001 , 8                     ' Set Pointer to point to Config register
   I2cwbyte Lm75config                         ' Send the Lm75config byte to write
   I2cstop
```

```
   Resetpointer                              ' Reset the pointer to '00'
End Sub
Sub Gettos(lm75addr)
   Call Setaddr Lm75addr , 2
   I2cstart
   I2cwbyte Lm75write                        ' Send write address
   I2cwbyte &B00000011 , 8                   ' Set Pointer to Tos
   I2cstart
   I2cwbyte Lm75read                         ' Send read address
   I2crbyte Lm75toshi , 8                    ' Read Tos high byte
   I2crbyte Lm75toslo , 9                    ' Read Tos low byte
   I2cstop
   Resetpointer                              ' Reset the pointer to '00'
End Sub
Sub Settos(lm75addr)
   Call Setaddr Lm75addr , 1
   I2cstart
   I2cwbyte Lm75write                        ' Send write address
   I2cwbyte &B00000011                       ' Set Pointer to Tos
   I2cwbyte Lm75toshi                        ' Send Tos high byte
   I2cwbyte Lm75toslo                        ' Send Tos low byte
   I2cstop
   Resetpointer                              ' Reset the pointer to '00'
End Sub
Sub Getthyst(lm75addr)
   Call Setaddr Lm75addr , 2
   I2cstart
   I2cwbyte Lm75write                        ' Send write address
   I2cwbyte &B00000010 , 8                   ' Set pointer to Thyst
   I2cstart
   I2cwbyte Lm75read                         ' Send read address
   I2crbyte Lm75thysthi , 8                  ' Read Thyst high byte
   I2crbyte Lm75thystlo , 9                  ' Read Thyst low byte
   I2cstop
   Resetpointer                              ' Reset the pointer to '00'
End Sub
Sub Setthyst(lm75addr)
   Call Setaddr Lm75addr , 1
   I2cstart
   I2cwbyte Lm75write                        ' Send write address
   I2cwbyte &B00000010                       ' Set pointer to Thyst
   I2cwbyte Lm75thysthi                      ' Send Thyst high byte
   I2cwbyte Lm75thystlo                      ' Send Thyst low byte
   I2cstop
   Resetpointer                              ' Reset the pointer to '00'
End Sub
Sub Setaddr(lm75addr , Flagrw)
   If Lm75addr <> 0 Then
      Lm75addr = Lm75addr * 2                ' Shift the address 1 bit left
      Select Case Flagrw
         Case 0 : Lm75read = Lm75read + Lm75addr       ' Add the offset to the read address
         Case 1 : Lm75write = Lm75write + Lm75addr     ' Add the offset to the write address
         Case 2 :
```

```
            Lm75read = Lm75read + Lm75addr      ' Add the offset to read and write address
            Lm75write = Lm75write + Lm75addr
         Case Else
      End Select
   End If
End Sub
Sub Resetpointer()
' Only call this routine from within a subroutine that has called Setaddr()!
   I2cstart
   I2cwbyte Lm75write                          ' Send write address
   I2cwbyte &B00000000 , 9                     ' Set pointer to '00'
   I2cstop
   Lm75read = &B10010001
   Lm75write = &B10010000
End Sub
Sub Val2temp(lm75tmphi , Lm75tmplo , Lm75tmpsign)
' This routine will convert the hi and lo bytes into a temperature value
   If Lm75tmphi > 127 Then
      Lm75tmphi = Not Lm75tmphi
      Incr Lm75tmphi
      Lm75tmpsign = 1
   End If
   Lm75tmplo = Lm75tmplo And &B10000000
   If Lm75tmplo = &B10000000 Then
      Lm75tmplo = 5
   End If
End Sub
Sub Temp2val(lm75tmphi , Lm75tmplo , Lm75tmpsign)
' This routine will convert a temperature value into the prober hi and lo byte values.
   If Lm75tmpsign = 1 Then
      Lm75tmphi = Not Lm75tmphi
      Incr Lm75tmphi
   End If
   If Lm75tmplo <> 0 Then
      Lm75tmplo = &B10000000
   End If
End Sub
' Insert include files here.
' End of subroutines and actual end of program.
```



M. Akers Enterprises		
Title: LM75 Demo (handles 8 LM75 devices)		
Size: A	Document Number: MAE-LM75DEMO-001	Rev: --
Date: Monday, October 18, 1999	Sheet 1 of 1	