Application Note 171

Using Programmable Oscillators to Generate and Control System Clocks

www.maxim-ic.com

INTRODUCTION

The DS1077 is a solid-state, CMOS, oscillator capable of generating frequencies from 8kHz up to 133.33MHz. It can be used as a fixed frequency stand-alone oscillator, or as a processor-controlled frequency generator. The frequency of the two, synchronous, oscillator outputs are user adjustable in sub-multiples of the master frequency through the use of two on-chip programmable prescalers and divider. The frequency and mode settings are configurable "on-the-fly" and stored in EEPROM using a 2-wire serial interface which can address up to eight DS1077s on a single 2-wire bus. Two digital control inputs, CTRL0 and CTRL1 are also capable of controlling the frequency or mode. Also available is the DS1077L, which is a 3V version of the DS1077 and is capable of generating frequencies from 4.87kHz up to 66.66MHz. Throughout the remainder of this document, unless otherwise specified, both the 3V and 5V versions of the DS1077 will be referred to as the DS1077.

This application note will show several examples of using a DS1077. It will show how a DS1077 can be used in place of a crystal to clock an 8051 microcontroller and a Microchip PIC™ microcontroller. This application note will also show how to use multiple DS1077s on a single 2-wire bus. Finally, example 8051 firmware is included to show how to implement a 2-wire master, along with the lower layer communication routines to communicate with each DS1077 on the bus.

SYSTEM OVERVIEW

The reference schematic in Figure 1 shows two systems that are independent of each other. Both systems operate at a frequency generated by the DS1077. In this example, although each system is independent of each other, they are both controlled by a common 2-wire master. This 2-wire master can be part of a larger system that needs control of its subsystems. The master can decide to reduce the operating frequencies of the subsystems that are not currently needed in order to conserve power, reduce heat in some cases, or maybe even reduce EMI radiation. It is sometimes advantageous to use a DS1077 to avoid running high frequency clocks throughout a large system, like over a backplane. It is much easier and safer to distribute a relatively low frequency 2-wire bus. Likewise, one could also conceive of using the DS1077 to reduce the system operating frequency instead of the microcontroller (many micros can reduce its speed or even go into a low current sleep mode), since the DS1077 can reduce the frequency of the entire system, and not just the micro alone. All this stated, applications using the DS1077 can be large and complex, but for the purpose of this application note, the example schematic has been greatly simplified and is intended to make several points. It shows multiple DS1077s on a common bus, and it also shows that the DS1077 can directly drive some popular microcontrollers.

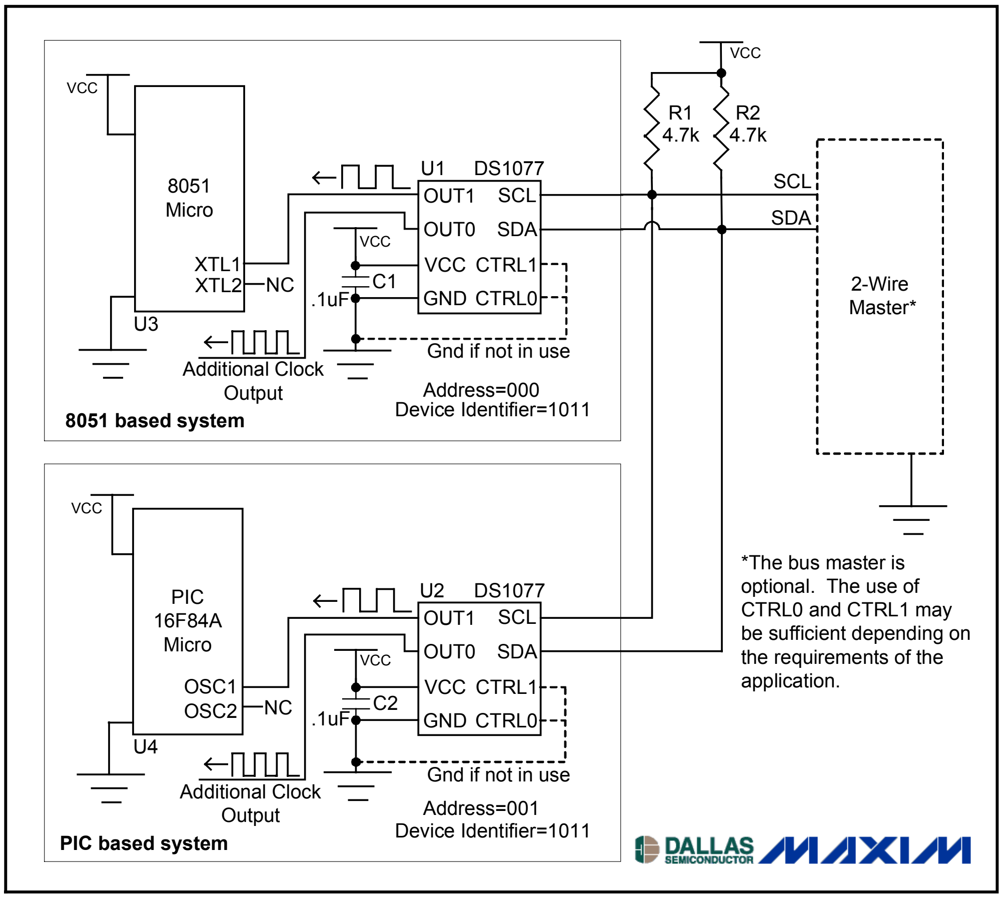
USING THE DS1077 AS A SYSTEM CLOCK

In the reference schematic, U1, the DS1077Z-125, is used to generate the clock for U3, an 8051 microcontroller. The particular 8051 that was used to develop the firmware found in *Appendix A* was the DS87C520. The DS87C520 is able to operate up to a frequency of 33MHz. Since U1 is the 125MHz version of the DS1077, OUT0 can only generate 125MHz, 62.5MHz, 31.025MHz, and 15.625MHz. While this would be fine if our application only needed full-speed and half-speed (using the 66MHz version of the DS1077), but again for the development of the firmware it was desirable to run the microcontroller all the way down to the kHz range. OUT0 does not have a divider, only a prescaler which divides by 1, 2, 4, or 8. OUT1 on the

PIC is a trademark of Microchip Semiconductor.

other hand, has in addition to the prescaler, a divider that can further divide the frequency by 2 through 1025. Therefore, OUT1 was chosen to clock the 8051. OUT0 can then still be used to clock other components of the system. For the purpose of simplicity, CTRL0 and CTRL1 were not used and were tied to GND. However, note that caution must be taken if CTRL0 and CTRL1 inputs are used to either disable the output or enter power-down mode because floating the microcontrollers' crystal input can cause unwanted oscillations or false clocking.

REFERENCE SCHEMATIC Figure 1



The second system shown in the reference schematic shows U2, a DS1077 clocking the OSC1 pin of U4, a PIC microcontroller. Again, the DS1077 is used in place of a fixed-frequency crystal. The PIC used in this example can operate up to 20MHz, so like with the 8051 example, OUT1 is used in order to take advantage of the 2-1025 divider, which is only provided by OUT1. OUT0 can be used by other components of the system. However, if either output is not used, and therefore not connected, it would be wise to disable the unused output by using the corresponding CTRL pin. This will cause a noticeable reduction in the supply current, as well as reduce the probability of unwanted EMI radiation. Although, as with the 8051 system, for simplicity CTRL0 and CTRL1 were not used and tied to ground.

Like always, it is important that sufficient decoupling be provided. Likewise, it is important that the decoupling capacitors C1 and C2 have good high-frequency performance and are physically located as close as possible to each DS1077 using short PCB traces.

ADVANTAGES OF USING A DS1077 OVER A CRYSTAL

There are several important reasons why a DS1077 might be preferred over a crystal. First, the DS1077 frequency can be changed. In fact, it can be changed on-the-fly, or even disabled. Second, the DS1077 offers dual, synchronous, individually controllable outputs. Also, the DS1077 eliminates the need for messy tank circuits when operating a crystal at one of its harmonic frequencies (for crystals above 30MHz). Finally, the DS1077 is much less susceptible to vibration than crystals.

CONTROLLING THE DS1077

The DS1077 can be used in fixed-frequency applications as well as variable-frequency applications. In fixed-frequency applications, the 2-wire master is not needed and the CTRL inputs are optional. But for applications that require control of the frequency or mode, the CTRL inputs and/or the 2-wire master must be used depending on the amount of flexibility required.

CTRL0 and CTRL1 Inputs

If an application needs the ability to instruct the DS1077 to enter power-down mode to conserve energy, or if the application would like to turn off (tri-state) the oscillator outputs, then CTRL0 and CTRL1 must be used. While there are no specific 2-wire commands to enter power-down mode or to disable the outputs, there are some tricks that can be done in firmware to achieve the same results. For example, while the CTRL inputs are at a known state, the corresponding bit in the MUX register can be set or cleared to turn on and off the desired function.

2-Wire Interface

When an application needs to generate frequencies other than divide by 1, 2, 4, or 8, then the 2-wire interface is required. Interfacing to an existing 2-wire bus is simple. Just connect SDA and SCL (and GND). Be sure that somewhere the bus contains the bus pull-up resistors. These are R1 and R2 in the reference schematic. Although 4.7kΩ resistors were used in the example, these may need to be tweaked depending on the bus capacitance, the number of devices on the bus, and the desired communication speed. However, 4.7kΩ will work for the majority of applications. If there is not an existing bus, then one can be created with a micro. Example firmware for an 8051 microcontroller is given in *Appendix A*. Also, if the 2-wire interface will not be used at all in an application, be sure that SDA and SCL are tied to well-defined logic levels and not left floating.

Notice that in the example there are more than one DS1077s on the same bus. In order for the 2-wire master to communicate with each DS1077 individually, each needs to have a unique address. Three bits in the BUS register allow up to 8 DS1077s on the bus at once. U1 was programmed to have an address of '000', while U2 was programmed to have an address of '001'. For examples of 2-wire communication, refer to the DS1077 or DS1077L data sheet as well as the firmware listed in *Appendix A*.

DS1077 CONTROLLING THE 2-WIRE MASTER

Having read this far, one may wonder why an additional 2-wire master was used in the example application when there were two fine microcontrollers quite capable of generating the 2-wire protocol needed to communicate with the DS1077. While it can be done, it is dangerous and extra caution must be taken. First of all, when the frequency of the microcontroller is changed, so will the timing of 2-wire routines. Also, just as with the 8051 example and PIC examples, some micros may have a minimum operating frequency spec in addition to a maximum frequency. Finally, disabling the outputs are out of the question since the micro would no longer be clocking, and therefore, no longer be able to issue a command to re-enable the oscillator. But if it must be done, it is very important to check the data sheet of the microcontroller being used.

FIRMWARE

Firmware for an 8051-based system is included in *Appendix A*. It is intended to show an example of the lower layer routines needed to talk to the DS1077s. However, note that the firmware implements an open-loop system. Closing the loop is very application-specific. But for the purpose of illustrating examples of communicating with the DS1077, a menu-based, open-loop example is beneficial. A PC terminal program is used to give the DS1077s commands. The commands can then be looked up in the firmware to see exactly what is being performed. The basic menu commands are as follows:

1) Read the selected DS1077 and display the registers
2) Edit the MUX word
3) Edit the DIV word
4) Edit the BUS word
5) Write to the DS1077 at address '000'
6) Write to the DS1077 at address '000'
7) Write to the DS1077 at address '001'
8) Write to the DS1077 at address '001'
9) Change firmware 2-wire communication address

Since the DS1077 settings are stored in EEPROM, the DS1077 powers up to the values stored in EEPROM. The values of the MUX, DIV, and BUS registers can be written to using the corresponding menu commands. The firmware for menu commands 2 to 8 show examples how to perform 2-wire writes, while menu command 1 shows a 2-wire read. Finally, menu command 9 shows how to change the 2-wire address that will be addressed.

Here is an example of writing 1234h to the DIV register of DS1077 at address '000':

```
      LCALL START2WIRE        ; 2-WIRE START

      MOV   A,#0B0H           ; DEVICE IDENTIFIER, SLAVE ADDRESS, WRITE
      LCALL WRITEBITS         ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE     ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,#01H            ; ACCESS DIV COMMAND
      LCALL WRITEBITS         ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE     ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,#12H
      LCALL WRITEBITS         ; SEND MSB
      LCALL ACKSLAVEWRITE     ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,#34H
      LCALL WRITEBITS         ; SEND LSB
      LCALL ACKSLAVEWRITE     ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL STOP2WIRE         ; 2-WIRE STOP
```

To write the same to the DS1077 at address '001', simply replace:

```
      MOV   A,#0B0H           ; DEVICE IDENTIFIER, SLAVE ADDRESS, WRITE
```

with the following:

```
      MOV   A,#0B2H           ; DEVICE IDENTIFIER, SLAVE ADDRESS, WRITE
```

And here is an example of reading the DS1077 at address ‘000’ DIV register.

```
LCALL START2WIRE         ; 2-WIRE START
MOV   A,#0B0H            ; DEVICE IDENTIFIER, SLAVE ADDRESS, READ
LCALL WRITEBITS          ; SEND COMMAND BYTE
LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE
MOV   A,#01H             ; READ DIVREG COMMAND
LCALL WRITEBITS          ; SEND COMMAND BYTE
LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE
LCALL START2WIRE         ; REPEATED 2-WIRE START
MOV   A,#0B1H            ; DEVICE IDENTIFIER, SLAVE ADDRESS, READ
LCALL WRITEBITS          ; SEND COMMAND BYTE
LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

LCALL READBITS           ; READ DATA FROM DS1077 A
MOV   DIVDATAMSB,A       ; SAVE MUX MSB TO A VARIABLE
LCALL ACKSLAVEREAD       ; ACK SLAVE SO WE CAN READ NEXT BYTE
LCALL READBITS           ; READ DATA FROM DS1077 A
MOV   DIVDATALSB,A       ; SAVE MUX LSB TO A VARIABLE
;LCALL      ACKSLAVEREAD; NO ACK - NO MORE TO READ
LCALL STOP2WIRE          ; 2-WIRE STOP
```

The firmware found in *Appendix A*, as well as additional information can be found on our ftp site listed at the end of this application note under *Contact Information*.

CONCLUSION

The DS1077 can often be used as a replacement for a crystal when more flexibility and control are required. Furthermore, the DS1077 can be used to generate a wide range frequencies from the kHz range up to 133MHz. Depending on the amount of control needed, it can operate stand-alone, or processor-controlled using the 2-wire interface. While the example circuit was simple, it demonstrated several key features of the DS1077. Consider using a DS1077 in your next application.

MAXIM INTEGRATED PRODUCTS / DALLAS SEMICONDUCTOR CONTACT INFORMATION

Dallas Semiconductor
4401 S. Beltwood Parkway
Dallas, TX 75244
Tel: 972-371-4448

Maxim Integrated Products, Inc
120 San Gabriel Drive
Sunnyvale, CA 94086
Tel: 408-737-7600

Product Literature / Samples Requests:
(800) 998-8800

Sales and Customer Service:
(408) 737-7600

World Wide Web Site:
www.maxim-ic.com

Product Information:
http://www.maxim-ic.com/MaximProducts/products.htm

Ordering Information:
http://www.maxim-ic.com/BuyMaxim/Sales.htm

FTP Site:
ftp://ftp.dalsemi.com

APPENDIX A

```
;***************************************************************************
;* DS1077/1077L APPNOTE FIRMWARE                            *
;* Copyright (c) 1999,2000,2001 Dallas Semiconductor/MAXIM          *
;***************************************************************************
;*                                        *
;* This program is used to show an example of how to use the DS1077 in a   *
;* real application.                               *
;*                                        *
;* Revision History                              *
;* 1.0 12/16/01 BJV Initial Release                       *
;*                                        *
;***************************************************************************
;*                                        *
;* DS87C520 Reference:                                *
;*                                        *
;* P0.0 - 74ACQ573        P2.0 - A8                 *
;* P0.1 -  "          P2.1 - A9                *
;* P0.2 -  "          P2.2 - A10                *
;* P0.3 -  "          P2.3 - A11                *
;* P0.4 -  "          P2.4 - A12                *
;* P0.5 -  "          P2.5 - A13                *
;* P0.6 -  "          P2.6 - A14                *
;* P0.7 -  "          P2.7 - A15                *
;*                                        *
;* P3.0 -             P1.0 - SCL                *
;* P3.1 -             P1.1 - SDA                *
;* P3.2 -             P1.2 - RXD1 - TO PC SERIAL PORT      *
;* P3.3 -             P1.3 - TXD1 - TO PC SERIAL PORT      *
;* P3.4 -             P1.4 - ACK FAIL LED           *
;* P3.5 -             P1.5 - LED                *
;* P3.6 - *WR           P1.6 - LED                *
;* P3.7 - *RD           P1.7 - HEARTBEAT LED            *
;*                                        *
;* BANK 0 R0 - Used for 2-wire read and write, Do not destroy!        *
;* BANK 0 R3 - Used for binasc routine, Do not destroy!           *
;* BANK X R7 - Temp variable                                *
;*                                        *
;***************************************************************************
;* Notes:                                    *
;* 1. DS87C520 is running at 22.1184MHz                    *
;* 2. Connect serial port 1 (P1.2 and P1.3) to PC              *
;*                                        *
;***************************************************************************

;***************************************************************************
$NOMOD51                      ;disable predefined 8051 registers
$INCLUDE (REG520.INC)         ;DS87C520 definition file
;-------------------------------------------------------------------
;     SOFTWARE VERSION
;-------------------------------------------------------------------
MAJOR_VERSION  EQU   1  ;major version number
MINOR_VERSION  EQU   0  ;minor version number
;-------------------------------------------------------------------
;     CONSTANTS
;-------------------------------------------------------------------
CR    EQU   0DH               ;ASCII      CARRIAGE RETURN
LF    EQU   0AH               ;ASCII      LINE FEED
BS    EQU   08H               ;ASCII      BACK SPACE
```

```
ETX   EQU   03H               ;ASCII      END OF TEXT
BEL   EQU   07H               ;ASCII      BELL
NAK   EQU   15H               ;ASCII      NEGATIVE ACKNOWLEDGE

;------------------------------------------------------------------
;     SERIAL PORT CONFIGURATION
;------------------------------------------------------------------
SBUFIN      EQU   SBUF1       ;USE SERIAL PORT 1
SBUFON      EQU   SBUF1
RIN   BIT   SCON1.0
TIN   BIT   SCON1.1

;------------------------------------------------------------------
;     DS1077 SPECIFIC
;------------------------------------------------------------------
ACCESSDIV EQU     01H         ;Access DIV register DS1077 command
ACCESSMUX EQU     02H         ;Access MUX register DS1077 command
ACCESSBUS EQU     0DH         ;Access BUS register DS1077 command
WRITEE2  EQU      3FH         ;Write E2 DS1077 command
      DSEG AT 30H
DS1077READ: DS     1    ;control byte, A0=A1=A2=X, read
DS1077WRITE:DS     1    ;control byte, A0=A1=A2=X, write
MUXDATAMSB: DS   1            ;DS1077 DATA
MUXDATALSB: DS   1            ;DS1077 DATA
DIVDATAMSB: DS   1            ;DS1077 DATA
DIVDATALSB: DS   1            ;DS1077 DATA
BUSDATA:    DS   1            ;DS1077 DATA
;------------------------------------------------------------------
;     2-WIRE SPECIFIC
;------------------------------------------------------------------
SDA   BIT   P1.1
SCL   BIT   P1.0
ADDRESS:    DS    1           ;2-WIRE ADDRESS TO COMMUNICATE WITH

STACK       EQU   $           ;STACK IS ABOVE DATA
;*********************************************************************
;* Hardware Interrupt Vectors (Table on page 95 of DS databook)   *
;*********************************************************************
    CSEG AT 0           ;Power up and Reset
   LJMP  MAIN           ;

    CSEG AT 002Bh    ;Timer 2 Interrupt
   LJMP  TMR2_INT    ;Service Heartbeat LED

;*********************************************************************
;* Main Program                             *
;*********************************************************************
      CSEG AT 0080h
MAIN: CLR   EA                ;DISABLE INTERRUPTS
      MOV   ADDRESS,0         ;SET TO ADDRESS A0=A1=A2=0
      MOV   DS1077READ,#0B1H  ;SET TO ADDRESS A0=A1=A2=0
      MOV   DS1077WRITE,#0B0H ;SET TO ADDRESS A0=A1=A2=0
      MOV   SP,#STACK         ;INITIALIZE STACK
      LCALL INIT_520          ;INITIALIZE DS87C520
      LCALL INIT_LCD          ;INITIALIZE LCD
      LCALL PTBANNER          ;PRINT BANNER TO SCREEN
      LCALL INIT2WIRE         ;INITIALIZE 2-WIRE PORT
      LCALL READ1077          ;INITIALIZE DS1077'S
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 DATA TO LCD
      SETB  EA                ;ENABLE INTERRUPTS
;*********************************************************************
```

```
MAINLOOP:
      LCALL CRLF              ;NEWLINE
      MOV   DPTR, #MENU       ;POINT TO MENU STRING
      LCALL PTXT              ;DISPLAY MENU
MAINLOOP2:
      JNB   RIN,NOCHAR        ;CHECK FOR USER INPUT FROM UART
      LCALL ECHO              ;ECHO CHAR TO SCREEN
      LCALL CRLF              ;NEWLINE
      CLR   RIN               ;CLEAR RECEIVE FLAG
      CJNE  A,#ETX,PRECHAR1   ;JUMP IF NOT ETX (CTRL-C)
      CALL  PTBANNER          ;REPRINT THE BANNER TO THE SCREEN
      LJMP  MAINLOOP2         ;LOOP FOREVER
NOCHAR:CPL  P1.6              ;TOGGLE LED - FOR DEBUG ONLY
   LJMP     MAINLOOP2         ;LOOP FOREVER
PRECHAR1:
;*****************************************************
;***  MENU COMMANDS                   ***
;*****************************************************
CHAR1:CJNE  A,#'1',CHAR2      ;<1> READ DS1077 AND DISPLAY REGISTERS
      LCALL READ1077          ;READ DS1077
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 DATA TO LCD
      LJMP  MAINLOOP
;*****************************************************
CHAR2:CJNE  A,#'2',CHAR3      ;<2> EDIT MUX WORD
      MOV   DPTR, #MUXPROMPT  ;POINT TO MESSAGE TO BE DISPLAYED
      LCALL PTXT              ;DISPLAY MESSAGE
      LCALL INHEXD            ;GET FIRST INPUT CHAR FROM KEYBOARD
      SWAP  A                 ;SWAP INPUT TO UPPER NIBBLE
      ANL   A,#0F0H           ;CLEAR LOWER NIBBLE
      MOV   MUXDATAMSB,A      ;STORE WHILE GETTING SECOND CHAR
      LCALL INHEXD            ;GET SECOND INPUT CHAR FROM KEYBOARD
      ANL   A,#00Fh           ;CLEAR UPPER NIBBLE
      ORL   A,MUXDATAMSB      ;OR FIRST AND SECOND CHARS TOGETHER
      MOV   MUXDATAMSB,A      ;STORE MSB - FIRST TWO DIGITS
      LCALL INHEXD            ;GET THIRD INPUT CHAR FROM KEYBOARD
      SWAP  A                 ;SWAP INPUT TO UPPER NIBBLE
      ANL   A,#0F0H           ;CLEAR LOWER NIBBLE
      MOV   MUXDATALSB,A      ;STORE WHILE GETTING FOURTH CHAR
      LCALL INHEXD            ;GET FOURTH INPUT CHAR FROM KEYBOARD
      ANL   A,#00Fh           ;CLEAR UPPER NIBBLE
      ORL   A,MUXDATALSB      ;OR THIRD AND FOURTH CHARS TOGETHER
      MOV   MUXDATALSB,A      ;STORE LSB - THIRD AND FOURTH DIGITS
      LCALL CRLF              ;NEWLINE
      LCALL WRITE1077MUX      ;WRITE MUX REGISTER TO DS1077 (2 BYTES)
      LCALL READ1077          ;READ DS1077
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 REGISTERS TO LCD
      LJMP  MAINLOOP
;*****************************************************
CHAR3:CJNE  A,#'3',CHAR4      ;<3> EDIT DIV WORD
      MOV   DPTR, #DIVPROMPT  ;POINT TO MESSAGE TO BE DISPLAYED
      LCALL PTXT              ;DISPLAY MESSAGE
      LCALL INHEXD            ;GET FIRST INPUT CHAR FROM KEYBOARD
      SWAP  A                 ;SWAP INPUT TO UPPER NIBBLE
      ANL   A,#0F0H           ;CLEAR LOWER NIBBLE
      MOV   DIVDATAMSB,A      ;STORE WHILE GETTING SECOND CHAR
      LCALL INHEXD            ;GET SECOND INPUT CHAR FROM KEYBOARD
      ANL   A,#00Fh           ;CLEAR UPPER NIBBLE
      ORL   A,DIVDATAMSB      ;OR FIRST AND SECOND CHARS TOGETHER
      MOV   DIVDATAMSB,A      ;STORE MSB - FIRST TWO DIGITS
      LCALL INHEXD            ;GET THIRD INPUT CHAR FROM KEYBOARD
      SWAP  A                 ;SWAP INPUT TO UPPER NIBBLE
      ANL   A,#0F0H           ;CLEAR LOWER NIBBLE
      MOV   DIVDATALSB,A      ;STORE WHILE GETTING FOURTH CHAR
      LCALL INHEXD            ;GET FOURTH INPUT CHAR FROM KEYBOARD
      ANL   A,#00Fh           ;CLEAR UPPER NIBBLE
      ORL   A,DIVDATALSB      ;OR THIRD AND FOURTH CHARS TOGETHER
      MOV   DIVDATALSB,A      ;STORE LSB - THIRD AND FOURTH DIGITS
      LCALL CRLF              ;NEWLINE
      LCALL WRITE1077DIV      ;WRITE DIV REGISTER TO DS1077 (2 BYTES)
      LCALL READ1077          ;READ DS1077
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 REGISTERS TO LCD
      LJMP  MAINLOOP
;*******************************************************
CHAR4:CJNE  A,#'4',CHAR5      ;<4> EDIT BUS WORD
      MOV   DPTR, #BUSPROMPT  ;POINT TO MESSAGE TO BE DISPLAYED
      LCALL PTXT              ;DISPLAY MESSAGE
      LCALL INHEXD            ;GET FIRST INPUT CHAR FROM KEYBOARD
      SWAP  A                 ;SWAP INPUT TO UPPER NIBBLE
      ANL   A,#0F0H           ;CLEAR LOWER NIBBLE
      MOV   BUSDATA,A         ;STORE WHILE GETTING SECOND CHAR
      LCALL INHEXD            ;GET SECOND INPUT CHAR FROM KEYBOARD
      ANL   A,#00Fh           ;CLEAR UPPER NIBBLE
      ORL   A,BUSDATA         ;OR FIRST AND SECOND CHARS TOGETHER
      MOV   BUSDATA,A         ;STORE - ONE BYTE REGISTER
      LCALL CRLF              ;NEWLINE
      LCALL WRITE1077BUS      ;WRITE BUS REGISTER TO DS1077 (1 BYTE)
      LCALL READ1077          ;READ DS1077
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 REGISTERS TO LCD
      LJMP  MAINLOOP
;*******************************************************
CHAR5:CJNE  A,#'5',CHAR6      ;<5> FIRST DS1077 FAST
      MOV   ADDRESS,#00H      ;ADDRESS OF FIRST DS1077
      MOV   DS1077WRITE,#0B0H ;SET WHICH DS1077 ON BUS TO WRITE
      MOV   DS1077READ,#0B1H  ;SET WHICH DS1077 ON BUS TO READ
      MOV   DIVDATAMSB,#00H   ;SET DIV REGISTER (2 BYTES)
      MOV   DIVDATALSB,#40H   ;SET DIV REGISTER (2 BYTES)
      LCALL WRITE1077DIV      ;WRITE TO DIV REGISTER
      LCALL READ1077          ;READ DS1077
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 REGISTERS TO LCD
      LJMP  MAINLOOP
;*******************************************************
CHAR6:CJNE  A,#'6',CHAR7      ;<6> FIRST DS1077 SLOW
      MOV   ADDRESS,#00H      ;ADDRESS OF FIRST DS1077
      MOV   DS1077WRITE,#0B0H ;SET WHICH DS1077 ON BUS TO WRITE
      MOV   DS1077READ,#0B1H  ;SET WHICH DS1077 ON BUS TO READ
      MOV   DIVDATAMSB,#10H   ;SET DIV REGISTER (2 BYTES)
      MOV   DIVDATALSB,#00H   ;SET DIV REGISTER (2 BYTES)
      LCALL WRITE1077DIV      ;WRITE TO DIV REGISTER
      LCALL READ1077          ;READ DS1077
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 REGISTERS TO LCD
      LJMP  MAINLOOP
;*******************************************************
CHAR7:CJNE  A,#'7',CHAR8      ;<7> SECOND DS1077 FAST
      MOV   ADDRESS,#01H      ;ADDRESS OF SECOND DS1077
      MOV   DS1077WRITE,#0B2H ;SET WHICH DS1077 ON BUS TO WRITE
      MOV   DS1077READ,#0B3H  ;SET WHICH DS1077 ON BUS TO READ
      MOV   DIVDATAMSB,#01H   ;SET DIV REGISTER (2 BYTES)
      MOV   DIVDATALSB,#00H   ;SET DIV REGISTER (2 BYTES)
      LCALL WRITE1077DIV      ;WRITE TO DIV REGISTER
      LCALL READ1077          ;READ DS1077
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 REGISTERS TO LCD
      LJMP  MAINLOOP
```

```
;*****************************************************
CHAR8:CJNE  A,#'8',CHAR9      ;<8> SECOND DS1077 SLOW
      MOV   ADDRESS,#01H      ;ADDRESS OF SECOND DS1077
      MOV   DS1077WRITE,#0B2H ;SET WHICH DS1077 ON BUS TO WRITE
      MOV   DS1077READ,#0B3H  ;SET WHICH DS1077 ON BUS TO READ
      MOV   DIVDATAMSB,#20H   ;SET DIV REGISTER (2 BYTES)
      MOV   DIVDATALSB,#00H   ;SET DIV REGISTER (2 BYTES)
      LCALL WRITE1077DIV      ;WRITE TO DIV REGISTER
      LCALL READ1077          ;READ DS1077
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 REGISTERS TO LCD
      LJMP  MAINLOOP
;*****************************************************
CHAR9:CJNE  A,#'9',CHAR0      ;<9> CHANGE 2-WIRE COMMUNICATION ADDRESS
      MOV   DPTR, #ADDRESSPROMPT; POINT TO MESSAGE TO BE DISPLAYED
      LCALL PTXT              ;DISPLAY MESSAGE
      LCALL INHEXD            ;GET INPUT CHAR FROM KEYBOARD (0-7)
      ANL   A,#07H            ;ONLY XXXXX210
      MOV   ADDRESS,A         ;SAVE UNFORMATTED ADDRESS
      RL    A                 ;CONVERT TO BITS 3-1
      ANL   A,#0EH            ;STRIP OTHER BITS
      ORL   A,#0B0H           ;OR IN DEVICE IDENTIFIER
      MOV   DS1077WRITE,A     ;SAVE FOR WRITE COMMANDS R/W=0
      ORL   A,#01H            ;SET R/W BIT
      MOV   DS1077READ,A      ;SAVE FOR READ COMMAND R/W=1
      LCALL CRLF              ;NEWLINE
      LCALL READ1077          ;READ DS1077 REGISTERS
      LCALL LCDSHOWSTAT       ;DISPLAY DS1077 REGISTERS
      LJMP  MAINLOOP
;*****************************************************
CHAR0:CJNE  A,#'0',LAST       ;<0> DO NOTHING
      LJMP  MAINLOOP
;*****************************************************
LAST: LJMP  MAINLOOP          ;

;***************************************************************************
;****  TIMER 2 ISR - LED Heartbeat                     ****
;****                                  ****
;***************************************************************************
TMR2_INT:
      CLR   EA                ; DISABLE INTERRUPTS
      PUSH  ACC               ; SAVE ACC
      ANL   T2CON,#07FH       ; ACKNOWLEDGE INTERRUPT
      CPL   P1.7              ; TOGGLE LED
NOHEART:POP ACC               ; RESTORE ACC
      SETB  EA                ; ENABLE INTERRUPTS
    RETI

;***************************************************************************
;****  Spurious Interrupt - Used as a trap for unknown/unwanted ints. ****
;****                                  ****
;***************************************************************************
NOISR:MOV   DPTR, #DB_NOISR   ; POINT TO MESSAGE TO BE DISPLAYED
      LCALL PTXT              ; DISPLAY MESSAGE
      RETI



;***************************************************************************
;****  Initialize DS87C520 -                       ****
;****  INITIALIZE SERIAL PORT 1 FOR 19200 BAUD (22.1184MHZ)      ****
;***************************************************************************
```

```
INIT_520:
      MOV   WDCON,#0A0H        ; SMOD=1(UP TO 115200 BPS) AND TURN ON PFI
      MOV   SCON1,#50H         ; MODE1, ASYNC, 10BITS, TIMER1
      MOV   TMOD,#21H          ; TIMER1-8BIT AUTO RELOAD,TIMER0-16BIT
      MOV   TCON,#50H          ; TIMER0 AND 1 ENABLED,/INT0 AND 1 NOT USED
      MOV   TH1,#0FAH          ; TIMER1 RESET VALUE FOR 19200 BAUD
      MOV   P1,#0FFH           ; TURN OFF PORT1 LEDS
      MOV   T2CON,#04H         ; TURN ON TIMER2
      MOV   IE,#020H           ; ENABLE TIMER2 INTERRUPT
      CLR   RS0                ; SELECT REGISTER BANK 0
      CLR   RS1
      RET

;****************************************************************************
;****  Read the DS1077 registers                    ****
;****                                       ****
;****************************************************************************
READ1077:
                               ; READ MUX
      LCALL START2WIRE         ; 2-WIRE START
      MOV   A,DS1077WRITE      ; DEVICE IDENTIFIER, SLAVE ADDRESS, READ
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,#ACCESSMUX       ; READ MUXREG COMMAND
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL START2WIRE         ; REPEATED 2-WIRE START

      MOV   A,DS1077READ       ; DEVICE IDENTIFIER, SLAVE ADDRESS, READ
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL READBITS           ; READ DATA FROM DS1077 A
      MOV   MUXDATAMSB,A       ; SAVE MUX MSB
      LCALL ACKSLAVEREAD       ; ACK SLAVE SO WE CAN READ NEXT BYTE

      LCALL READBITS           ; READ DATA FROM DS1077 A
      MOV   MUXDATALSB,A       ; SAVE MUX LSB
      ;LCALL      ACKSLAVEREAD; NO ACK - NO MORE TO READ

      LCALL STOP2WIRE          ; 2-WIRE STOP
                               ; READ DIV
      LCALL START2WIRE         ; 2-WIRE START

      MOV   A,DS1077WRITE      ; DEVICE IDENTIFIER, SLAVE ADDRESS, READ
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,#ACCESSDIV       ; READ DIVREG COMMAND
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL START2WIRE         ; REPEATED 2-WIRE START

      MOV   A,DS1077READ       ; DEVICE IDENTIFIER, SLAVE ADDRESS, READ
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL READBITS           ; READ DATA FROM DS1077 A
      MOV   DIVDATAMSB,A       ; SAVE MUX MSB
      LCALL ACKSLAVEREAD       ; ACK SLAVE SO WE CAN READ NEXT BYTE

      LCALL READBITS           ; READ DATA FROM DS1077 A
      MOV   DIVDATALSB,A       ; SAVE MUX LSB
      ;LCALL      ACKSLAVEREAD; NO ACK - NO MORE TO READ

      LCALL STOP2WIRE          ; 2-WIRE STOP

                               ; READ BUS BYTE
      LCALL START2WIRE         ; 2-WIRE START

      MOV   A,DS1077WRITE      ; DEVICE IDENTIFIER, SLAVE ADDRESS, READ
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,#ACCESSBUS       ; READ BUSREG COMMAND
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL START2WIRE         ; REPEATED 2-WIRE START

      MOV   A,DS1077READ       ; DEVICE IDENTIFIER, SLAVE ADDRESS, READ
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL READBITS           ; READ DATA FROM DS1077 A
      MOV   BUSDATA,A          ; SAVE MUX MSB
      ;LCALL      ACKSLAVEREAD; NO ACK - NO MORE TO READ

      LCALL STOP2WIRE          ; 2-WIRE STOP
      RET

;*****************************************************************************
;****  Write the DS1077 MUX register                  ****
;****                                  ****
;*****************************************************************************
WRITE1077MUX:
                               ; WRITE MUX
      LCALL START2WIRE         ; 2-WIRE START

      MOV   A,DS1077WRITE      ; DEVICE IDENTIFIER, SLAVE ADDRESS, WRITE
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,#ACCESSMUX       ; MUXREG COMMAND
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,MUXDATAMSB
      LCALL WRITEBITS          ; SEND MSB
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,MUXDATALSB
      LCALL WRITEBITS          ; SEND LSB
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL STOP2WIRE          ; 2-WIRE STOP

      RET
```

```
;***************************************************************************
;****  Write the DS1077 DIV register                     ****
;****                                 ****
;***************************************************************************
WRITE1077DIV:
      LCALL START2WIRE         ; 2-WIRE START

      MOV   A,DS1077WRITE      ; DEVICE IDENTIFIER, SLAVE ADDRESS, WRITE
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,#ACCESSDIV       ; DIVREG COMMAND
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,DIVDATAMSB
      LCALL WRITEBITS          ; SEND MSB
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,DIVDATALSB
      LCALL WRITEBITS          ; SEND LSB
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL STOP2WIRE          ; 2-WIRE STOP

      RET

;***************************************************************************
;****  Write the DS1077 BUS register                     ****
;****                                 ****
;***************************************************************************
WRITE1077BUS:
      LCALL START2WIRE         ; 2-WIRE START

      MOV   A,DS1077WRITE      ; DEVICE IDENTIFIER, SLAVE ADDRESS, WRITE
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,#ACCESSBUS       ; BUSREG COMMAND
      LCALL WRITEBITS          ; SEND COMMAND BYTE
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      MOV   A,BUSDATA
      LCALL WRITEBITS          ; SEND MSB
      LCALL ACKSLAVEWRITE      ; CHECK FOR SLAVE ACKNOWLEDGE

      LCALL STOP2WIRE          ; 2-WIRE STOP

      RET




;*********************************************************************
;****  2-Wire Start Condition Generator Routine          ****
;*********************************************************************
;*  requires wait2us routine                    *
;*  uses no registers                       *
;*********************************************************************
START2WIRE:
      SETB  SDA
      NOP
      NOP
      NOP
      NOP
      SETB  SCL
      NOP
      NOP
      NOP
      NOP
      CLR   SDA                ; PULL SDA LOW
      LCALL WAIT2US            ; WAIT 2US
      CLR   SCL                ; PULL SCL LOW
      LCALL WAIT2US            ; WAIT 2US
      RET

;*********************************************************************
;****  2-Wire Stop Condition                   ****
;****  Used to send a stop condition              ****
;*********************************************************************
;*  requires wait2us routine                    *
;*  uses no registers                       *
;*********************************************************************
STOP2WIRE:
   CLR      SDA                ; PULL SDA LOW
                               ; SDA MUST BE LOW SO IT CAN GO HIGH
                               ; WHILE THE CLOCK IS HIGH TO GENERATE
                               ; THE STOP CONDITION
   NOP                  ; WASTE 180NS/NOP, STOP SETUP TIME
      NOP
      NOP
      NOP
   SETB     SCL                ; PULL SCL HIGH
      NOP
      NOP
      NOP
      NOP
   SETB     SDA                ; PULL SDA HIGH
      LCALL WAIT2US            ; WAIT 2US
      RET

;*********************************************************************
;****  2-Wire Initialization Routine               ****
;****  Inits SCL and SDA to Set Condition            ****
;*********************************************************************
;*  requires no routines                       *
;*  Uses no Registers                        *
;*********************************************************************
INIT2WIRE:
      SETB  SCL                ; START PROGRAM WITH SCL HIGH
   SETB     SDA                ; START PROGRAM WITH SDA HIGH
   RET

;*********************************************************************
;****  Write Bits Routine                      ****
;****   Serializes and Transmits the data in the Accumulator at ****
;****   the time the routine is called            ****
;*********************************************************************
;*  requires no other routines                      *
;*  Destroys Window 0 R0 register and ACC              *
;*********************************************************************
```

```
WRITEBITS:
      MOV   R0, #8      ; sets up for transfer of 8 bits
NEXTWRITEBIT:
      RLC   A                 ; move the MSB of the ACC into C
      MOV   SDA, C            ; write C onto SDA line
      SETB  SCL               ; set SCL
      NOP
      NOP                     ; clock high time, 180ns/nop
      NOP
      NOP
      CLR   SCL               ; clear SCL
      NOP
      NOP                     ; clock low time, 180ns/nop + other
      NOP                     ; instructions between last nop and
      NOP                     ; next setb SCL
      DJNZ  R0, NEXTWRITEBIT  ; if the 8th data bit not sent yet
                              ;  then keep sending data
      RET

;**********************************************************************
;****  2-Wire Readbits Routine                  ****
;****   Reads 8-bits of data from the slave device, and stores ****
;****   the received data in the Accumulator          ****
;**********************************************************************
;*  requires no other routines                   *
;*  Destroys Window0 R0 register and ACC              *
;**********************************************************************
READBITS:
      SETB  SDA               ; SDA must be set for an open
                              ;  collector read
      MOV   R0, #8            ; sets up for transfer of 8 bits
NEXTREADBIT:
      SETB  SCL               ; set SCL
      NOP                     ; clock high time, 180ns/nop + other
      NOP                     ;  instructions before clr SCL
      MOV   C, SDA            ; Place Data on SDA into C
      RLC   A                 ; move the C into LSB of A
      CLR   SCL               ; clear SCL
      NOP
      NOP                     ; clock low time, 180ns/nop + other
      NOP                     ; instructions before next setb SCL
      NOP
      NOP
      DJNZ  R0, NEXTREADBIT   ; if the 8th data bit not sent yet
                              ;  keep sending data
      RET




;**********************************************************************
;****  2-Wire Acknowledge Slave Routine for WRITES        ****
;****  Used to acknowledge slave devices DURING WRITES     ****
;**********************************************************************
;*  requires outstr routines                   *
;*  uses DPTR register                      *
;**********************************************************************
ACKSLAVEWRITE:
      SETB  SDA               ; set  SDA
      NOP                     ; wait 180ns/nop
      NOP
      SETB  SCL               ; set SCL
   NOP
      NOP                     ; wait 180ns/nop + other instructions
      NOP                     ; with clock high
      JB    SDA, ACK_FAIL     ; if SDA high (acknowledge fails),
                              ;  then jump to error routine
      CLR   SCL               ;  else ack passes, set SCL and
      NOP                     ; wait 180ns/nop + other instructions
      NOP                     ; for clock to go high
      SETB  P1.4              ; turn off LED
      RET                     ; return
ACK_FAIL:
      CLR   P1.4              ; turn on LED
      CLR   SCL               ; clr SCL
      CLR   SDA               ; clr SDA
      NOP
      NOP                     ; clock time low, 180ns/nop + clr
      NOP                     ;  SDA instruction
      NOP
      NOP
      NOP
      SETB  SCL               ; set SCL
      NOP
      NOP                     ; clock time high, 180ns/nop
      NOP
      NOP
      NOP
      NOP
      NOP
      SETB  SDA               ; create stop condition
      RET

;*********************************************************************
;****  2-Wire Acknowledge Slave Routine for READS        ****
;****  Used to acknowledge slave devices DURING READS      ****
;*********************************************************************
;*  requires no other routines                    *
;*  uses no registers                        *
;*********************************************************************
ACKSLAVEREAD:
      CLR  SDA                ; clear SDA
      NOP                     ; wait 180ns/nop
      NOP
      SETB  SCL               ; set SCL
   NOP
      NOP                     ; wait 180ns/nop
      NOP
      NOP
      CLR   SCL         ; clear SCL
      RET

;***************************************************************************
;****  DISPLAY DS1050 DATA TO LCD                    ****
;***************************************************************************
LCDSHOWSTAT:
      MOV   A,#0C0h           ; SET TO SECOND LINE OF LCD DISPLAY
      MOV   DPTR,#8000h       ; ADDRESS OF LCD
      MOVX  @DPTR,A           ; WRITE ADDRESS TO LCD
      LCALL DELAY40U          ; WAIT FOR LCD
      LCALL DELAY40U          ; WAIT FOR LCD
      MOV   DPTR,#MUXREG      ; TEXT MESSAGE TO DISPLAY
      LCALL LCDSTR            ; PRINT TEXT TO LCD
      MOV   A,MUXDATAMSB      ; RECALL DATA
      LCALL BINTOASCII        ; ACC=FIRST DIGIT,B=SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (FIRST DIGIT)
      MOV   A,B               ; GET SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (SECOND DIGIT)

      MOV   A,MUXDATALSB      ; RECALL DATA
      LCALL BINTOASCII        ; ACC=FIRST DIGIT,B=SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY PWM1 DATA ON LCD (FIRST DIGIT)
      MOV   A,B               ; GET SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY PWM1 DATA ON LCD (SECOND DIGIT)

      MOV   A,#094h           ; SET TO THIRD LINE OF LCD DISPLAY
      MOV   DPTR,#8000h       ; ADDRESS OF LCD
      MOVX  @DPTR,A           ; WRITE ADDRESS TO LCD
      LCALL DELAY40U          ; WAIT FOR LCD
      LCALL DELAY40U          ; WAIT FOR LCD
      MOV   DPTR,#DIVREG      ; TEXT MESSAGE TO DISPLAY
      LCALL LCDSTR            ; PRINT STRING TO LCD
      MOV   A,DIVDATAMSB      ; RECALL DATA
      LCALL BINTOASCII        ; ACC=FIRST DIGIT,B=SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (FIRST DIGIT)
      MOV   A,B               ; GET SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (SECOND DIGIT)
      MOV   A,DIVDATALSB      ; RECALL DATA
      LCALL BINTOASCII        ; ACC=FIRST DIGIT,B=SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (FIRST DIGIT)
      MOV   A,B               ; GET SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (SECOND DIGIT)

      MOV   A,#0D4h           ; SET TO THIRD LINE OF LCD DISPLAY
      MOV   DPTR,#8000h       ; ADDRESS OF LCD
      MOVX  @DPTR,A           ; WRITE ADDRESS TO LCD
      LCALL DELAY40U          ; WAIT FOR LCD
      LCALL DELAY40U          ; WAIT FOR LCD
      MOV   DPTR,#BUSREG      ; TEXT MESSAGE TO DISPLAY
      LCALL LCDSTR            ; PRINT STRING TO LCD
      MOV   A,BUSDATA         ; RECALL DATA
      LCALL BINTOASCII        ; ACC=FIRST DIGIT,B=SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (FIRST DIGIT)
      MOV   A,B               ; GET SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (SECOND DIGIT)

      MOV   DPTR,#ADDRESSREG  ; TEXT MESSAGE TO DISPLAY
      LCALL LCDSTR            ; PRINT STRING TO LCD
      MOV   A,ADDRESS         ; RECALL 2-WIRE COMMUNICATION ADDRESS
      LCALL BINTOASCII        ; ACC=FIRST DIGIT,B=SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (FIRST DIGIT)
      MOV   A,B               ; GET SECOND DIGIT
      LCALL LCDCHAR           ; DISPLAY DATA ON LCD (SECOND DIGIT)
      RET

;**********************************************************************
;****  Wait 2us Function                         ****
;****  Wastes 1.6us of processor time with call, nop and return ****
;**********************************************************************
;*  Requires no other routines or registers             *
;**********************************************************************
```

```
WAIT2US:
      NOP                     ; 1 nops @4cc each + lcall @16cc + ret @16cc
                              ; produces approximately 1.6us of delay with a
                              ; 22.22MHz clock
      RET

;**************************************************************************
;****  40us Delay                          ****
;**************************************************************************
DELAY40U:MOV      A,#000Fh    ;180ns*2cycles
LOOP40U: NOP                  ;180ns
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP                   ;
        DEC A           ;
        JNZ LOOP40U     ;14 cycles in loop
     RET                ;4 cycles

;**************************************************************************
;****  Delay 5ms                          ****
;**************************************************************************
DELAY5MS:MOV      A,#079h
LOOP5MS: PUSH     ACC         ; WASTE TIME
        LCALL     DELAY40U    ; WASTE TIME
        POP ACC         ; WASTE TIME
        DEC A           ; LOOP COUNTER
        JNZ LOOP5MS     ; CONTINUE WASTING TIME
     RET

;**************************************************************************
;****  STRINGS AND ERROR MESSAGES                          ****
;**************************************************************************
BANNER:      DB    CR,LF
      DB  'DS1077(L) APPNOTE FIRMWARE  REV ', ETX
BANNER2:     DB    CR,LF
      DB  'DS1077(L) ', CR, LF, ETX
PERIOD:      DB    '.', ETX
PROMPT:      DB    CR, '> ', ETX
MENU:        DB    'MENU COMMANDS', CR,LF
             DB    '1 -READ DS1077 AND DISPLAY TO LCD',CR,LF
             DB    '2 -EDIT MUX REGISTER OF DS1077', CR, LF
             DB    '3 -EDIT DIV REGISTER OF DS1077', CR, LF
             DB    '4 -EDIT BUS REGISTER OF DS1077', CR, LF
             DB    '5 -FIRST DS1077 FAST', CR, LF
             DB    '6 -FIRST DS1077 SLOW', CR, LF
             DB    '7 -SECOND DS1077 FAST', CR, LF
             DB    '8 -SECOND DS1077 SLOW', CR, LF
             DB    '9 -CHANGE COMMUNICATION ADDRESS', CR, LF
             DB    'ENTER YOUR SELECTION: ',ETX
LCDBANNER:   DB    'DS1077(L) APPNOTE',00h
MUXPROMPT:   DB    'Enter 4 hex digits for MUX register: ', ETX
DIVPROMPT:   DB    'Enter 4 hex digits for DIV register: ', ETX
BUSPROMPT:   DB    'Enter 2 hex digits for BUS register: ', ETX
ADDRESSPROMPT:DB  'Enter 1 hex digit for 2-wire address: ', ETX
MUXREG:      DB    'MUX = ',00h
```

```
DIVREG:     DB    'DIV = ',00h
BUSREG:     DB    'BUS = ',00h
ADDRESSREG: DB    ' ADDRESS=',00h
ACKF:       DB    'ACK FAILED', CR, LF, ETX
DB_POWER:   DB    'LOW POWER INT', CR, LF, ETX
DB_NOISR:   DB    'NOISR INT', CR, LF, ETX
BUETX:      DB    ETX          ;Just in case
;****************************************************************************
$INCLUDE (DSLIB.A51)     ;SERIAL PORT DEBUG ROUTINES
    END
;*******************************************************************************
```