DALLAS SEMICONDUCTOR

Application Note 95
Interfacing the DS1307 with an 8051–Compatible Microcontroller

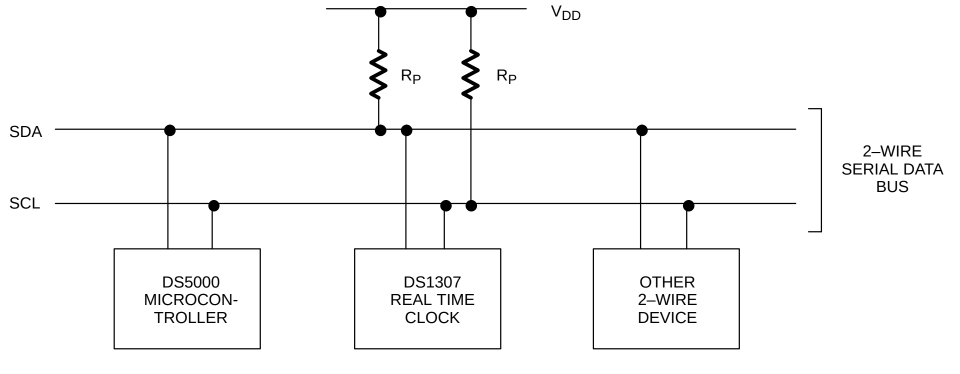
INTRODUCTION
The DS1307 Serial Real Time Clock, which incorporates a 2–wire serial interface, can be controlled using an 8051–compatible DS5000 Secure Microcontroller. The DS1307 is connected directly to two of the I/O ports on a DS5000 microcontroller and the 2–wire handshaking is handled by low–level drivers, which are discussed in this application note.

DS1307 DESCRIPTION
The DS1307 Serial Real Time Clock is a low–power, full BCD clock/calendar plus 56 bytes of nonvolatile SRAM. Address and data are transferred serially via the 2–wire bi–directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with less than 31 days, including corrections for leap year. The clock operates in either the 24–hour or 12–hour format with AM/PM indicator. The DS1307 has a built–in power sense circuit which detects power failures and automatically switches to the battery supply.

DS1307 OPERATION
The DS1307 operates as a slave device on the serial bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. The START and STOP conditions are generated using the low level drives, SEND_START and SEND_STOP found in the attached DS5000 code. Also the subroutines SEND_BYTE and READ_BYTE provide the 2–wire handshaking required for writing and reading 8–bit words to and from the DS1307.

HARDWARE CONFIGURATION
The system is configured as shown in Figure 1. The DS1307 has the 2–wire bus connected to two I/O port pins of the DS5000: SCL – P1.0, SDA – P1.1. The V_{DD} voltage is 5V, $R_P = 5K\Omega$ and the DS5000 is using a 12 MHz crystal. The other peripheral device could be any other device that recognizes the 2–wire protocol, such as the DS1621 Digital Thermometer and Thermostat. The interface with the D5000 was accomplished using the DS5000T Kit hardware and software. This development kit allows the PC to be used as a dumb terminal using the DS5000's serial ports to communicate with the keyboard and monitor.

TYPICAL 2–WIRE BUS CONFIGURATION Figure 1



The following bus protocol has been defined (see Figure 2).

- During data transfer, the data line must remain stable whenever the clock line is high. Changes in the data line while the clock line is high will be interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Start data transfer: A change in the state of the data line from high to low, while the clock line is high, defines a START condition.

Stop data transfer: A change in the state of the data line from low to high, while the clock line is high, defines the STOP condition.

Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the high period of the clock signal. The data on the line must be changed during the low period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between the START and the STOP conditions is not limited, and is determined by the master device. The information is transferred byte–wise and each receiver acknowledges with a ninth bit.

Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

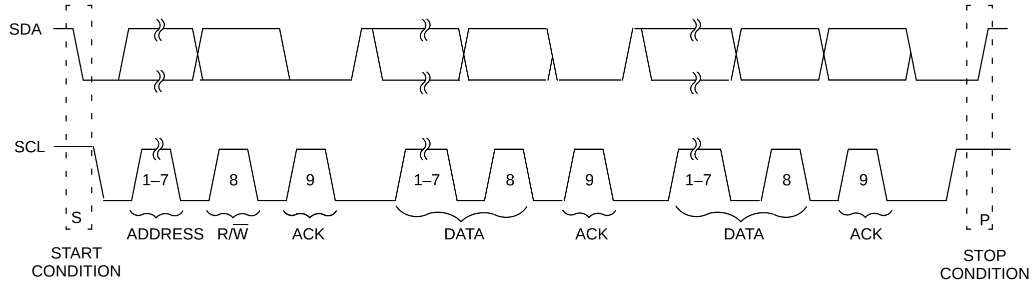
A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable low during the high period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line high to enable the master to generate the STOP condition.

Figure 2 details how data transfer is accomplished on the 2–wire bus. Depending on the state of the R/$\overline{W}$ bit, two types of data transfer are possible:

1. **Data transfer from a master transmitter to a slave receiver.** The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.

2. **Data transfer from a slave transmitter to a master receiver.** The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a not acknowledge is returned.

DATA TRANSFER ON 2–WIRE SERIAL BUS Figure 2

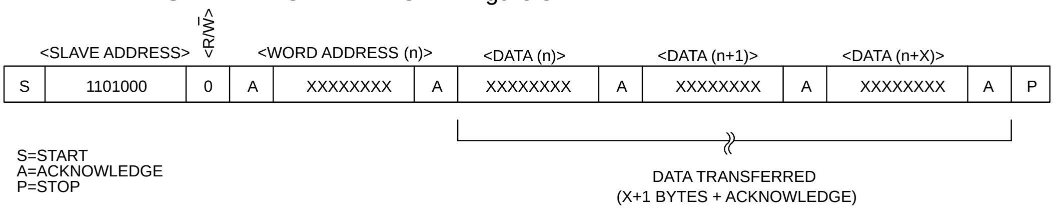


The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

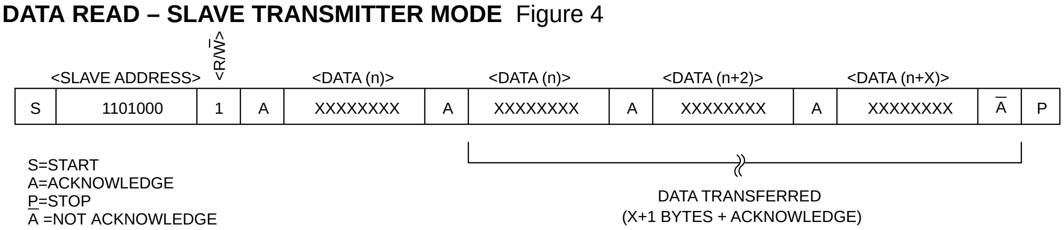
The DS1307 may operate in the following two modes:

1. **Slave receiver mode (DS1307 write mode):** Serial data and clock are received through SDA and SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit (see Figure 3). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7–bit DS1307 address, which is 1101000, followed by the direction bit (R/$\overline{W}$) which for a write is a 0. After receiving and decoding the address byte, the DS1307 outputs an acknowledge on the SDA line. After the DS1307 acknowledges the slave address + write bit, the master transmits a register address to the DS1307. This will set the register pointer on the DS1307. The master will then begin transmitting each byte of data with the DS1307 acknowledging each byte received. The master will generate a stop condition to terminate the data write.
2. **Slave transmitter mode (DS1307 read mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the DS1307 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (See Figure 4). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7–bit DS1307 address, which is 1101000, followed by the direction bit (R/$\overline{W}$) which for a read is a 1. After receiving and decoding the address byte, the DS1307 inputs an acknowledge on the SDA line. The DS1307 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode, the first address that is read is the last one stored in the register pointer. The DS1307 must be sent a Not–Acknowledge bit by the master to terminate a read.

DATA WRITE – SLAVE RECEIVER MODE Figure 3



DATA READ – SLAVE TRANSMITTER MODE Figure 4



SOFTWARE OPERATION

DS5000 INTERFACE

The software presented in Appendix 1 is written to interface the DS5000 with the DS1307 over the 2–wire interface. The DS5000 was programmed using Dallas Semiconductor's DS5000T Evaluation Kit, which allows a PC to be used as a dumb terminal. The KIT5K software environment supplied with the DS5000T Evaluation Kit provides a high–level interface for loading application software to the DS5000 or for setting its configuration parameters via the Program command. The KIT5K software includes a dumb terminal emulator to allow users to run application software in the DS5000 which communicates with the user via a PC COM port.

DS1307 SOURCE CODE

The first section of the code found in the Appendix is used to configure the DS5000 for serial communication with the PC. Also at the beginning of the code is the MASTER_CONTROLLER subroutine which is used to control the demonstration software.

The subroutines that immediately follow the MASTER_CONTROLLER subroutine are the low level drivers for controlling the 2–wire interface. They are not specific to the DS1307 but can be used with any 2–wire compatible Slave–only device. These subroutines are:

SEND_START
: This subroutine is used to generate the Start condition on the 2–wire bus.

SEND_STOP
: This subroutine is used to generate the Stop condition on the 2–wire bus.

SEND_BYTE
: This subroutine sends an 8–bit word, MSB first, over the 2–wire bus with a 9th clock pulse for the Acknowledge pulse.

READ_BYTE
: This subroutine reads an 8–bit word over the 2–wire bus. It checks for the LASTREAD flag to be cleared indicating when the last read from the slave device is to occur. If it is not the last read, the DS5000 sends an Acknowledge pulse on the 9th clock and if it is the last read from the slave device, the DS5000 sends a Not–Acknowledge.

SCL_HIGH
: This subroutine transitions the SCL line low–to–high and ensures the SCL line is high before continuing.

DELAY and DELAY_4
: These two subroutines have been included to ensure that the 2–wire bus timing is maintained.

The rest of the code included in the appendix is specifically designed to demonstrate the functions of the DS1307. The functions that are demonstrated are:

Setting Time
: The time is read in from the keyboard and stored in the DS5000 scratchpad memory. It is then transferred, over the 2–wire interface, to the DS1307.

Set RAM
: A single hex byte is read in from the keyboard and written to the entire user RAM of the DS1307.

Read Date/Time
: The date and time are read, over the 2–wire bus, and stored in the DS5000 scratchpad memory. It is then written to the screen. This continues until a key is pressed on the keyboard.

Read RAM
: The entire user RAM of the DS1307 is read into the DS5000 scratchpad memory and then written to the PC monitor.

OSC On/ OSC Off
: The DS1307 clock oscillator can be turned on or off.

SQW/OUT On/ SQW/OUT Off
: The SQW/OUT can be turned on or off. It will toggle at 1 Hz.

AC ELECTRICAL CHARACTERISTICS Table 1

PARAMETER	SYMBOL	ACTUAL	UNITS
SCL Clock Frequency	f_{SCL}	59	KHz
Bus Free Time Between a STOP and START Condition	t_{BUF}	5.7	µs
Hold Time (repeated) START Condition	$t_{HD:STA}$	6.2	µs
LOW Period of SCL Clock	t_{LOW}	10.5	µs
HIGH Period of SCL Clock	t_{HIGH}	6.5	µs
Set–up Time for a Repeated START Condition	$t_{SU:STA}$	5.3	µs
Data Hold Time	$t_{HD:DAT}$	5.5	µs
Data Set–up Time	$t_{SU:DAT}$	3.1	µs
Rise Time of Both SDA and SCL Signals	t_R		ns
Fall Time of Both SDA and SCL Signals	t_F		ns
Set–up Time for STOP Condition	$t_{SU:STO}$	5.4	µs

CONCLUSION

It has been shown that it is very straight forward to interface the DS1307 or any other 2–wire slave device to an 8051–compatible microcontroller. The only concern must be that the 2–wire timing specification is not violated by the low level drivers on the microcontroller. The delay subroutines have been inserted into the code for this purpose. The values in Table 1 are the actual timing parameters observed in the hardware setup used to develop this application note.

APPENDIX

DS1307.ASM

```
;       Program DS1307.ASM
;
;       This program responds to commands received over the serial
;       port to set the date/time as well as RAM data on the DS1307
;       using a DS5000 as a controller
;
$MOD51
CR              EQU             0DH
LF              EQU             0AH
MCON            EQU             0C6H
TA              EQU             0C7H
SCL             BIT             P1.0
SDA             BIT             P1.1
TRIG            BIT             P1.2
DS1307W         EQU             0D0H
DS1307R         EQU             0D1H
FLAGS           DATA            20H
LASTREAD        BIT             FLAGS.0
_12_24          BIT             FLAGS.1
PM_AM           BIT             FLAGS.2
OSC             BIT             FLAGS.3
SQW             BIT             FLAGS.4
ACK             BIT             FLAGS.5
BUS_FAULT       BIT             FLAGS.6
_2W_BUSY        BIT             FLAGS.7
BITCOUNT        DATA            21H
BYTECOUNT       DATA            22H
BYTE            DATA            23H
                CSEG            AT              0
                AJMP            START
;
                CSEG            AT              30H
;**************************************************************
;***             RESET GOES HERE TO START PROGRAM          ****
;**************************************************************
START:
                MOV             TA,             #0AAH   ;  Timed
                MOV             TA,             #55H    ;  access.
                MOV             PCON,           #0      ;  Reset watchdog timer.
                MOV             MCON,           #0F8H   ;  Turn off CE2 for memory
                                                        ;  access.
                MOV             SP,             #70H    ;  Position stack above
                                                        ;  buffer.
                MOV             IE,             #0
                MOV             TMOD,           #20H    ;  Initialize the
                MOV             TH1,            #0FAH   ;  serial port
                MOV             TL1,            #0FAH   ;  for 9600
                ORL             PCON,           #80H    ;  baud.
```

```
            MOV             SCON,           #52H
            MOV             TCON,           #40H

            MOV             R0,             #0
            MOV             R1,             #0
            DJNZ            R0,             $
            DJNZ            R1,             $–2
            SETB            SDA                     ;  ENSURE SDA HIGH
            LCALL           SCL_HIGH                ;  ENSURE SCL HIGH
            CLR                             ACK     ;  CLEAR STATUS FLAGS
            CLR                             BUS_FAULT
            CLR                             _2W_BUSY
;––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
;       THIS IS THE MASTER CONTROLLER LOOP
;––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
MASTER_CONTROLLER:
        MOV     BYTECOUNT,#20H
FORM_FEED:
        MOV     A,#LF                               ;  CLEAR SCREEN FOR MAIN MENU
        LCALL   WRITE_DATA
        DJNZ    BYTECOUNT,FORM_FEED

        MOV     DPTR,   #TEXT0                      ;  PUT MAIN MENU ON SCREEN
        LCALL   WRITE_TEXT
        MOV     DPTR,   #TEXT3
        LCALL   WRITE_TEXT
        LCALL   READ_DATA
        CLR     ACC.5                               ;  CONVERT ACC TO UPPER CASE

        CJNE    A,#’A’,NOTA                         ;  CALL SET CLOCK FUNCTION
        LCALL   SET_CLOCK
        JMP     MASTER–CONTROLLER                   ;  RETURN TO MAIN MENU
NOTA:
        CJNE    A,#’B’,NOTB                         ;  CALL SET RAM FUNCTION AND
        LCALL   SET_RAM                             ;  CALL READ RAM FUNCTION
        LCALL   READ_RAM
        JMP     MASTER–CONTROLLER                   ;  RETURN TO MAIN MENU
NOTB:
        CJNE    A,#’C’,NOTC                         ;  CALL READ CLOCK FUNCTION
        LCALL   READ_CLOCK
        JMP     MASTER–CONTROLLER                   ;  RETURN TO MAIN MENU
NOTC:
        CJNE    A,#’D’,NOTD                         ;  CALL READ RAM FUNCTION
        LCALL   READ_RAM
        JMP     MASTER–CONTROLLER                   ;  RETURN TO MAIN MENU
NOTD:
        CJNE    A,#’E’,NOTE                         ;  CALL OSC CONTROL FUNCTION
        CLR     OSC                                 ;  CLR OSC FLAG – ON
        LCALL   OSC_CONTROL
        JMP     MASTER–CONTROLLER                   ;  RETURN TO MAIN MENU
```

```
NOTE:
        CJNE    A,#'F',NOTF                         ;  CALL OSC CONTROL FUNCTION
        SETB    OSC                                 ;  SET OSC FLAG – OFF
        LCALL   OSC_CONTROL
        JMP     MASTER–CONTROLLER                   ;  RETURN TO MAIN MENU
NOTF:
        CJNE    A,#'G',NOTG                         ;  CALL SWQ CONTROL FUNCTION
        CLR     SQW                                 ;  CLR SQW FLAG – ON
        LCALL   SQW_CONTROL
        JMP     MASTER_CONTROLLER                   ;  RETURN TO MAIN MENU
NOTG:
        CJNE    A,#'H',NOTH                         ;  CALL SWQ CONTROL FUNCTION
        SETB    SQW                                 ;  SET SQW FLAG – OFF
        LCALL   SQW_CONTROL
NOTH:
        JMP     MASTER_CONTROLLER                   ;  RETURN TO MAIN MENU
;-----------------------------------------------------------------------
;       THIS SUB SENDS THE START CONDITION
;-----------------------------------------------------------------------
SEND_START:                                         ;
        SETB    _2W_BUSY                            ;  INDICATE THAT 2WIRE
                                                    :  OPERATION IN PROGRESS
        CLR     ACK                                 ;  CLEAR STATUS FLAGS
        CLR     BUS_FAULT
        JNB     SCL,FAULT                           ;  CHECK FOR BUS CLEAR
        JNB     SDA,FAULT
                                                    ;  BEGIN START CONDITION
        SETB    SDA                                 ;
        LCALL   SCL_HIGH                            ;  SDA
        CLR     SDA                                 ;
        LCALL   DELAY                               ;  SCL ^START CONDITION
        CLR     SCL                                 ;
                RET
FAULT:
        SETB    BUS_FAULT                           ;  SET FAULT STATUS
        RET                                         ;  AND RETURN
;-----------------------------------------------------------------------
;       THIS SUB SENDS THE STOP CONDITION
;-----------------------------------------------------------------------
SEND_STOP:                                          ;
        CLR     SDA                                 ; SDA
        LCALL   SCL_HIGH                            ;
        SETB    SDA                                 ; SCL ^STOP CONDITION
        CLR     _2W_BUSY
        RET                                         ;
;-----------------------------------------------------------------------
;       THIS SUB SENDS ONE BYTE OF DATA TO THE DS1307
;-----------------------------------------------------------------------
SEND_BYTE:
        MOV     BITCOUNT,#08H                       ;  SET COUNTER FOR 8 BITS
SB_LOOP:
```

```
        JNB     ACC.7,NOTONE                        ;  CHECK TO SEE IF BIT 7 OF
                                                    ;  ACC IS A 1
        SETB    SDA                                 ;  SET SDA HIGH (1)
        JMP     ONE
NOTONE:
        CLR     SDA                                 ;  CLR SDA LOW (0)
ONE:
        LCALL   SCL_HIGH                            ;  TRANSITION SCL LOW–TO–HIGH
        RL      A                                   ;  ROTATE ACC LEFT ONE BIT
        CLR     SCL                                 ;  TRANSITION SCL HIGH–TO–LOW
        DJNZ    BITCOUNT,SB_LOOP                    ;  LOOP FOR 8 BITS

        SETB    SDA                                 ;  SET SDA HIGH TO LOOK
                                                    ;  FOR ACKNOWLEDGE PULSE
        LCALL   SCL_HIGH                            ;  TRASITION SCL LOW–TO–HIGH
        CLR     ACK                                 ;  CLEAR ACKNOWLEDGE FLAG
        JNB     SDA,SB_EX                           ;  CHECK FOR ACK OR NOT ACK
        SETB    ACK                                 ;  SET ACKNOWLEDGE FLAG FOR
                                                    ;  NOT ACK
SB_EX:
        LCALL   DELAY                               ;  DELAY FOR AN OPERATION
        CLR     SCL                                 ;  TRANSITION SCL HIGH–TO–LOW
        LCALL   DELAY                               ;  DELAY FOR AN OPERATION
        RET
;----------------------------------------------------------------------
;       THIS SUB READS ONE BYTE OF DATA FROM THE DS1307
;----------------------------------------------------------------------
READ_BITS:
        MOV     BITCOUNT,#008H                      ;  SET COUNTER FOR 8 BITS OF
                                                    :  DATA
        MOV     A,#00H                              ;
        SETB    SDA                                 ;  SET SDA HIGH TO ENSURE LINE
                                                    ;  FREE
READ_BITS:
        LCALL   SCL_HIGH                            ;  TRANSITION SCL LOW–TO–HIGH
        MOV     C,SDA                               ;  MOVE DATA BIT INTO CARRY
                                                    ;  BIT
        RLC     A                                   ;  ROTATE CARRY BIT INTO ACC.0
        CLR     SCL                                 ;  TRANSITION SCL HIGH–TO–LOW
        DJNZ    BITCOUNT,READ_BITS                  ;  LOOP FOR 8 BITS

        JB      LASTREAD,ACKN                       ;  CHECK TO SEE IF THIS IS THE
                                                    :  LAST READ
        CLR     SDA                                 ;  IF NOT LAST READ SEND
                                                    :  ACKNOWLEDGE BIT
ACKN:
        LCALL   SCL_HIGH                            ;  PULSE SCL TO TRANSIMIT
                                                    ;  ACKNOWLEDGE
        CLR     SCL                                 ;  OR NOT ACKNOWLEDGE BIT
        RET
```

```
;-----------------------------------------------------------------------
;       THIS SUB SETS THE CLOCK LINE HIGH
;-----------------------------------------------------------------------
SCL_HIGH:
        SETB    SCL                                 ;  SET SCL HIGH
        JNB     SCL,$                               ;  LOOP UNTIL STRONG 1 ON SCL
        RET
;-----------------------------------------------------------------------
;       THIS SUB DELAY THE BUS
;-----------------------------------------------------------------------
DELAY:
        NOP                                         ;  DELAY FOR BUS TIMING
        RET
;-----------------------------------------------------------------------
;       THIS SUB DELAYS 4 CYCLES
;-----------------------------------------------------------------------
DELAY_4:
        NOP                                         ;  DELAY FOR BUS TIMING
        NOP
        NOP
        NOP
        RET
;-----------------------------------------------------------------------
;       THIS SUB SETS THE CLOCK
;-----------------------------------------------------------------------
SET_CLOCK:
        MOV     R1,#2EH                             ;  SET R1 TO SCRATCHPAD MEMORY
                                                    ;  FOR DATE/TIME
        MOV     DPTR,   #YEAR                       ;  GET THE DATE/TIME
                                                    ;  INFORMATION FROM THE
        LCALL   WRITE_TEXT                          ;  USER.  WRITE THE DATE/TIME
                                                    ;  TO SCRATCHPAD
        LCALL   READ_BCD                            ;  MEMORY
        MOV     @R1,A
        DEC     R1
        MOV     DPTR,   #MONTH
        LCALL   WRITE_TEXT
        LCALL   READ_BCD
        MOV     @R1,A
        DEC     R1
        MOV     DPTR,   #DAY
        LCALL   WRITE_TEXT
        LCALL   READ_BCD
        MOV     @R1,A
        DEC     R1
        MOV     DPTR,   #DAYW
        LCALL   WRITE_TEXT
        LCALL   READ_BCD
        ANL     A,      #7
        MOV     @R1,A
        DEC     R1
```

```
        MOV     DPTR,  #HOUR
        LCALL   WRITE_TEXT
        LCALL   READ_BCD
        MOV     @R1,A
        DEC     R1
        MOV     DPTR,  #MINUTE
        LCALL   WRITE_TEXT
        LCALL   READ_BCD
        MOV     @R1,A
        DEC     R1
        MOV     DPTR,  #SECOND
        LCALL   WRITE_TEXT
        LCALL   READ_BCD
        MOV     @R1,A
        MOV     R1,#28H                                 ;  POINT TO BEGINNING OF CLOCK
                                                        ;  DATA IN SCRATCHPAD MEMORY
        LCALL   SEND_START                              ;  SEND 2WIRE START CONDITION
        MOV     A,#DS1307W                              ;  SEND DS1307 WRITE COMMAND
        LCALL   SEND_BYTE
        MOV     A,#00H                                  ;  SET DATA POINTER TO
                                                        ;  REGISTER 00H ON
        LCALL   SEND_BYTE                               ;  THE DS1307
SEND_LOOP:
        MOV     A,@R1                                   ;  MOVE THE FIRST BYTE OF DATA
                                                        ;  TO ACC
        LCALL   SEND_BYTE                               ;  SEND DATA ON 2WIRE BUT
        INC     R1
        CJNE    R1,#2FH,SEND_LOOP                       ;  LOOP UNTIL CLOCK DATA SENT
                                                        ;  TO DS1307
        LCALL   SEND_STOP                               ;  SEND 2WIRE STOP CONDITION
        RET
;-----------------------------------------------------------------------
;       THIS SUB SETS THE DS1307 USER RAM TO THE VALUE IN 'BYTE'
;-----------------------------------------------------------------------
SET_RAM:
        MOV     R1,#08H                                 ;  POINTER TO BEGINNING OF
                                                        ;  DS1307 USER RAM
        MOV     DPTR,  #TEXT5                           ;  MESSAGE TO ENTER DATA BYTE
        LCALL   WRITE_TEXT                              ;
        LCALL   READ_BCD                                ;  READ BYTE FROM KEYBOARD
        MOV     BYTE,A                                  ;  AND STORE IN 'BYTE'
        LCALL   SEND_START                              ;  SEND 2WIRE START CONDITION
        MOV     A,#DS1307W                              ;  LOAD DS1307 WRITE COMMAND
        LCALL   SEND_BYTE                               ;  SEND WRITE COMMAND
        MOV     A,#08H                                  ;  SET DS1307 DATA POINTER TO
                                                        ;  BEGINNING
        LCALL   SEND_BYTE                               ; OF USER RAM – 08H
SEND_LOOP2:
        MOV     A,BYTE                                  ;  WRITE BYTE TO ENTIRE RAM
                                                        ;  SPACE
        LCALL   SEND_BYTE                               ;  WHICH IS 08H TO 37H
```

```
        INC     R1
        CJNE    R1,#040H,SEND_LOOP2                 ;  LOOP UNTIL RAM FILLED
        LCALL   SEND_STOP                           ;  SEND 2WIRE STOP CONTION
        RET
;-----------------------------------------------------------------------
;       THIS SUB READS THE DS1307 RAM AND WRITES IT TO THE SCRATCH PAD MEMORY
;-----------------------------------------------------------------------
READ_RAM:
        MOV     DPTR,#TEXT4                         ;  SEND KEY PRESS MSG
        LCALL   WRITE_TEXT
        MOV     R1,#30H                             ;  START OF RAM REGS IN
                                                    ;  SCRATCH PAD
        MOV     BYTECOUNT,#00H                      ;  COUNTER FOR 56 RAM BYTES
        CLR     LASTREAD                            ;  FLAG TO CHECK FOR LAST READ
        LCALL   SEND_START                          ;  SEND 2WIRE START CONDITION
        MOV     A,#DS1307W                          ;  SEND DS1307 WRITE COMMAND
        LCALL   SEND_BYTE
        MOV     A,#08H                              ;  SET POINTER TO REG 08H ON
                                                    :  DS1307
        LCALL   SEND_BYTE
        LCALL   SEND_STOP                           ;  SEND STOP CONDITION
        LCALL   SEND_START                          ;  SEND START CONDITION
        MOV     A,#DS1307R                          ;  SEND DS1307 READ COMMAND
        LCALL   SEND_BYTE
READ_LOOP2:
        MOV     A,BYTECOUNT                         ;  CHECK TO SEE OF DOING LAST
                                                    ;  READ
        CJNE    A,#37H,NOT_LAST2
        SETB    LASTREAD                            ;  IF LAST READ SET LASTREAD
                                                    ;  FLAG
NOT_LAST2:
        LCALL   READ_BYTE                           ;  READ A BYTE OF DATA
        MOV     @R1,A                               ;  MOVE DATA INTO SCRATCHPAD
                                                    ;  MEMORY
        INC     R1                                  ;  INC POINTERS
        INC     BYTECOUNT
        MOV     A,BYTECOUNT
        CJNE    A,#38H,READ_LOOP2                   ;  LOOP FOR ENTIRE DS1307 RAM

        LCALL   SEND_STOP                           ;  SEND 2WIRE STOP CONDITION
        LCALL   DISP_RAM                            ;  DISPLAY DATA IN SCRATCHPAD
                                                    ;  MEMORY
        JNB     RI,$                                ;  WAIT UNTIL A KEY IS PRESSED
        CLR     RI
RET
;-----------------------------------------------------------------------
;       THIS SUB DISPLAYS THE RAM DATA SAVED IN SCRATCHPAD MEMORY
;-----------------------------------------------------------------------
DISP_RAM:
        MOV     R1,#30H                             ;  START OF RAM IN SCRATCHPAD
                                                    ;  MEMORY
```

```
        MOV     BITCOUNT,#00H
        MOV     DPTR,#TEXT6                         ;  DISPLAY TABLE HEADING
        LCALL   WRITE_TEXT
DISP_ADDR:
        LCALL   DISP_LOC                            ;  DISPLAY VALUE OF CURRENT
                                                    ;  RAM LOCATION
DIS_LOOP:
        MOV     A,@R1                               ;  DISPLAY RAM DATA SAVED IN
                                                    ;  SCRATCHPAD
        LCALL   WRITE_BCD                           ;  CONVERT TO BCD FORMAT AND
                                                    ;  DISPLAY
        INC     R1
        INC     BITCOUNT
        MOV     A,#20H                              ;  SPACE BETWEEN DATA BYTES
        LCALL   WRITE_DATA
        MOV     A,BITCOUNT
        CJNE    A,#08H,DIS_LOOP                     ;  LINE FEED AFTER 8 BYTES OF
                                                    ;  DATA
        MOV     BITCOUNT,#00H
        MOV     DPTR,#TEXT3                         ;  'CR,LF'
        LCALL   WRITE_TEXT
        CJNE    R1,#68H,DISP_ADDR                   ;  DISPLAY DATA FOR 56 BYTES
                                                    ;  OF RAM
        RET
;-----------------------------------------------------------------------
;       THIS SUB WRITES THE RAM LOCATION OF THE DATA
;-----------------------------------------------------------------------
DISP_LOC:
        MOV     A,R1                                ;  DISPLAY THE HEX VALUE FOR
                                                    ;  THE DATA
        ADD     A,#-28H                             ;  IN THE DS1307 RAM SPACE
        LCALL   WRITE_BCD                           ;  CONVERTS SCRATCHPAD ADDRESS
        MOV     A,#20H                              ;  INTO DS1307 RAM ADDRESS
        LCALL   WRITE_DATA
        MOV     A,#20H
        LCALL   WRITE_DATA
        MOV     A,#20H
        LCALL   WRITE_DATA
        RET
;-----------------------------------------------------------------------
;       THIS SUB READS THE CLOCK AND WRITES IT TO THE SCRATCH PAD MEMORY
;-----------------------------------------------------------------------
READ_CLOCK:
        MOV     DPTR,#TEXT4                         ;  KEY PRESS MSG
        LCALL   WRITE_TEXT
READ_AGAIN:
        MOV     R1,#28H                             ;  START OF CLOCK REG IN
                                                    ;  SCRATCHPAD
        MOV     BYTECOUNT,#00H                      ;  COUNTER UP TO 8 BYTES FOR
                                                    ;  CLOCK
        CLR     LASTREAD                            ;  FLAG TO CHECK FOR LAST READ
```

```
        LCALL   SEND_START                        ;  SEND START CONDITION
        MOV     A,#DS1307W                        ;  SET POINTER TO REG 00H ON
                                                  ;  DS1307
        LCALL   SEND_BYTE
        MOV     A,#00H
        LCALL   SEND_BYTE
        LCALL   SEND_STOP                         ;  SEND STOP CONDITION
        LCALL   SEND_START                        ;  SEND START CONDITION
        MOV     A,#DS1307R                        ;  SEND READ COMMAND TO DS1307
        LCALL   SEND_BYTE
READ_LOOP:
        MOV     A,BYTECOUNT                       ;  CHECK TO SEE OF DOING LAST
                                                  ;  READ
        CJNE    A,#07H,NOT_LAST
        SETB    LASTREAD                          ;  SET LASTREAD FLAG
NOT_LAST:
        LCALL   READ_BYTE                         ;  READ A BYTE OF DATA
        MOV     @R1,A                             ;  MOVE DATA IN SCRATCHPAD
                                                  ;  MEMORY
        MOV     A,BYTECOUNT                       ;  CHECK TO SEE IF READING
                                                  ;  SECONDS REG
        CJNE    A,#00H,NOT_FIRST
        CLR     OSC                               ;  CLR OSC FLAG
        MOV     A,@R1                             ;  MOVE SECONDS REG INTO ACC
        JNB     ACC.7,NO_OSC                      ;  JUMP IF BIT 7 OF IS A 0
        SETB    OSC                               ;  SET OSC FLAG, BIT 7 IS A 1
        CLR     ACC.7                             ;  CLEAR BIT 7 FOR DISPLAY
                                                  ;  PURPOSES
        MOV     @R1,A                             ; MOVE DATA BACK TO SCRATCHPAD
NO_OSC:
NOT_FIRST:
        INC     R1                                ;  INC COUNTERS
        INC     BYTECOUNT
        MOV     A,BYTECOUNT
        CJNE    A,#08H,READ_LOOP                  ;  LOOP FOR ENTIRE CLOCK
                                                  ;  REGISTERS
        LCALL   SEND_STOP                         ;  SEND 2WIRE STOP CONDITION
        LCALL   DISP_CLOCK                        ;  DISPLAY DATE/TIME FROM
                                                  ;  SCRATCHPAD
        JNB     RI,READ_AGAIN                     ;  READ AND DISPLAY UNTIL A
                                                  ;  KEY IS PRESSED
        CLR     RI
        RET
;-----------------------------------------------------------------------
;       THIS SUB DISPLAYS THE DATE AND TIME SAVED IN SCRATCHPAD MEMORY
;-----------------------------------------------------------------------
DISP_CLOCK:
        MOV     DPTR,#TEXT1                       ;  DATE:
        LCALL   WRITE_TEXT
        MOV     R1,#2DH                           ;  MONTH
        MOV     A,@R1
```

```
        LCALL   WRITE_BCD
        MOV     A,#'/'
        LCALL   WRITE_DATA
        MOV     R1,#2CH                                 ;  DATE
        MOV     A,@R1
        LCALL   WRITE_BCD
        MOV     A,#'/'
        LCALL   WRITE_DATA
        MOV     R1,#2EH                                 ;  YEAR
        MOV     A,@R1
        LCALL   WRITE_BCD
        MOV     A,#09H                                  ;  TAB
        LCALL   WRITE_DATA
        MOV     DPTR,#TEXT2                             ;  TIME:
        LCALL   WRITE_TEXT
        MOV     R1,#2AH                                 ;  HOURS
        MOV     A,@R1
        LCALL   WRITE_BCD
        MOV     A,#3AH                                  ;  COLON
        LCALL   WRITE_DATA
        MOV     R1,#29H                                 ;  MINUTES
        MOV     A,@R1
        LCALL   WRITE_BCD
        MOV     A,#3AH                                  ;  COLON
        LCALL   WRITE_DATA
        MOV     R1,#28H                                 ;  SECONDS
        MOV     A,@R1
        LCALL   WRITE_BCD
        RET
;------------------------------------------------------------------------
;       THIS SUB SETS THE OSCILLATOR ACCORDING TO THE OSC BIT
;------------------------------------------------------------------------
OSC_CONTROL:
        LCALL   SEND_START                              ;  SEND START CONDITION
        MOV     A,#DS1307W                              ;  SET POINTER TO REG 00H ON
                                                        ;  DS1307
        LCALL   SEND_BYTE
        MOV     A,#00H
        LCALL   SEND_BYTE
        SETB    LASTREAD                                ;  SET LAST READ FOR SINGLE
                                                        ;  READ
        LCALL   SEND_STOP                               ;  SEND STOP CONDITION
        LCALL   SEND_START                              ;  SEND START CONDITION
        MOV     A,#DS1307R                              ;  SEND READ COMMAND TO DS1307
        LCALL   SEND_BYTE
        LCALL   READ_BYTE                               ;  READ SECONDS REGISTER
        CLR     ACC.7                                   ;  TURN OSC ON
        JNB     OSC,OSC_SET
        SETB     ACC.7                                  ;  TURN OSC OFF IF OSC BIT IS
                                                        ;  SET IN
OSC_SET:                                                ;  SECONDS REGISTER
```

```
        PUSH    ACC                             ;  SAVE SECONDS DATA ON STACK
        LCALL   SEND_STOP                       ;  SEND STOP CONDITION
        LCALL   SEND_START                      ;  SEND START CONDITION
        MOV     A,#DS1307W                      ;  SET POINTER TO REG 00H ON
                                                ;  DS1307
        LCALL   SEND_BYTE
        MOV     A,#00H
        LCALL   SEND_BYTE
        POP     ACC                             ;  SEND SECONDS REGISTER TO
                                                ;  CONTROL
        LCALL   SEND_BYTE                       ;  OSCILLATOR ON DS1307
        LCALL   SEND_STOP
        RET
;-----------------------------------------------------------------------
;       THIS SUB CONTROLS THE SQW OUTPUT
;-----------------------------------------------------------------------
SQW_CONTROL:
        LCALL   SEND_START                      ;  SEND START CONDITION
        MOV     A,#DS1307W                      ;  SET POINTER TO REG 07H ON
                                                ;  DS1307
        LCALL   SEND_BYTE
        MOV     A,#07H
        LCALL   SEND_BYTE

        MOV     A,#90H                          ;  SQW/OUT ON AT 1HZ
        JNB     SQW,SQW_SET                     ;  JUMP IS SQW BIT IS ACTIVE
        MOV     A,#80H                          ;  TURN SQW/OUT OFF – OFF HIGH
SQW_SET:
        LCALL   SEND_BYTE
        LCALL   SEND_STOP
        RET
;-----------------------------------------------------------------------
;       THIS SUB IS A SCOPE TRIGGER BIT
;-----------------------------------------------------------------------
TRIGGER:
        CLR     TRIG
        SETB    TRIG
        LCALL   DELAY_4
        CLR     TRIG
RET
;-----------------------------------------------------------------------
;       THIS SUB READS DATA FROM THE SCREEN AND CONVERTS IT TO BCD FORM
;       DATA SHOULD BE HEX DIGITS: 1,2,3...9,A,B,C,D,E,F
;-----------------------------------------------------------------------
READ_BCD:
        MOV     R0,#0                           ;  CLEAR R0
BCD_LOOP:
        LCALL   READ_DATA                       ;  READ BYTE FROM KEYBOARD
        LCALL   WRITE_DATA                      ;  WRITE BYTE BACK TO SCREEN
        CJNE    A,#0DH,BCD                      ;  CHECK FOR CR
        MOV     A,R0                            ;  MOVE R0 TO ACC AND RETURN
```

```
        RET
BCD:
        ADD     A,#-30H                                 ;  BEGIN TO CONVERT TO ACTUAL
                                                        ;  VALUE
        JNB     ACC.4,NUMBER                            ;  JUMP IF NOT A–F
        ADD     A,#-07H                                 ;  IF A–F SUBTRACT 7
NUMBER:
        ANL     A,#0FH                                  ;  ENSURE BITS 4–7 ARE CLEARED
        ANL     0,#0FH                                  ;  ENSURE BITS 4–7 ARE CLEARED
        XCH     A,R0                                    ;  EXCHANGE R0 AND ACC
        SWAP    A                                       ;  NIBBLE SWAP ACC
        ORL     A,R0                                    ;  INSERT BITS 0–3 OF R0 INTO
                                                        ;  ACC
        MOV     R0,A                                    ;  MOVE ACC INTO R0
        SJMP    BCD_LOOP                                ;  LOOP UNTIL CR ENCOUNTERED
;------------------------------------------------------------------------
;       THIS SUB WRITES THE BYTE TO THE SCREEN
;------------------------------------------------------------------------
WRITE_BCD:
        PUSH    ACC                                     ;  SAVE ACC ON STACK
        SWAP    A                                       ;  NIBBLE SWAP ACC
        ANL     A,#0FH                                  ;  CLEAR BITS 4–7 OF ACC
        ADD     A,#07H                                  ;  ADD 7 TO ACC TO CONVERT TO
                                                        ;  ASCII HEX
        JNB     ACC.4,LESSNINE                          ;  CHECK TO SEE IF LESS THAN
                                                        ;  NINE 0–8
        CJNE    A,#10H,NOTNINE                          ;  JUMP IS GREATER THAN NINE
                                                        ;  A–F
LESSNINE:
        ADD     A,#-07H                                 ;  SUBTRACT 7 FOR 0–9
NOTNINE:
        ADD     A,#30H                                  ;  ADD 30 TO CONVERT TO ASCII
                                                        ;  EQUIVALENT
        LCALL   WRITE_DATA                              ;  WRITE BYTE TO SCREEN
        POP     ACC                                     ;  RECALL ACC FROM STACK
        ANL     A,#0FH                                  ;  PERFORM CONVERSION ON OTHER
                                                        ;  HALF OF BYTE
        ADD     A,#07H
        JNB     ACC.4,NINE2
        CJNE    A,#10H,NOTNINE2
NINE2:
        ADD     A,#-07H
NOTNINE2:
        ADD     A,#30H
        LCALL   WRITE_DATA
        RET
;------------------------------------------------------------------------
;------------------------------------------------------------------------
READ_DATA:
        JNB     RI,READ_DATA                            ;  LOOP WHILE RI BIT IS LOW
        CLR     RI                                      ;
```

```
        MOV     A,SBUF                              ;  GET DATA BYTE FROM SERIAL
                                                    ;  BUFFER
        RET
;----------------------------------------------------------------------
;----------------------------------------------------------------------
WRITE_DATA:
        JNB     TI,WRITE_DATA                       ;  LOOP WHILE TI BIT IS LOW
        CLR     TI                                  ;
        MOV     SBUF,A                              ;  SEND DATA BYTE TO SERIAL
                                                    ;  BUFFER
        RET
;----------------------------------------------------------------------
;----------------------------------------------------------------------
WRITE_TEXT:
        PUSH    ACC                                 ;  SAVE ACC BYTE ON STACK WT1:
        CLR      A                                  ;  CLEAR ACC
        MOVC    A,@A+DPTR                           ;  MOVE FIRST BYTE OF STRING
                                                    ;  TO ACC
        INC     DPTR                                ;  INC DATA POINTER
        CJNE    A,#0,WT2                            ;  CHECK FOR STRING
                                                    ;  TERMINATOR – 0
        POP     ACC                                 ;  RESTORE ACC
        RET                                         ;  RETURN WHEN STRING IS SENT
WT2:
        LCALL   WRITE_DATA                          ;  SEND BYTE OF STRING OVER
                                                    ;  SERIAL PORT
        SJMP    WT1
;----------------------------------------------------------------------
;       TEXT STRINGS USED FOR USER INTERFACE OVER SERIAL PORT
;----------------------------------------------------------------------
YEAR:
        DB      CR,LF,'YEAR (0 – 99)  :      ',0
MONTH:
        DB       CR,LF,'MONTH (1 – 12):      ',0
DAY:
        DB      CR,LF,'DAY OF MONTH   :      ',0
DAYW:
        DB      CR,LF,'DAY OF WEEK    :      ',0
HOUR:
        DB      CR,LF,'HOUR (0 – 23)  :      ',0
MINUTE:
        DB      CR,LF,'MINUTE (0 – 59):      ',0
SECOND:
        DB      CR,LF,'SECOND (0 – 59):      ',0
TRIER:
        DB      CR,LF,'PRESS ANY KEY TO SET THIS TIME',CR,LF,0
TEXT0:
        DB      CR,LF,'******* DALLAS SEMICONDUCTOR *******  '
        DB      CR,LF,'   DS1307 DEMONSTRATION PROGRAM       ',CR,LF
        DB      CR,LF,'PLEASE CHOOSE AN OPTION TO CONTINUE   '
        DB      CR,LF,'------------------------------------  '
```

```
        DB      CR,LF,'A. SET TIME              B. SET RAM     '
        DB      CR,LF,'C. READ DATE/TIME        D. READ RAM    '
        DB      CR,LF,'E. OSC ON                F. OSC OFF     '
        DB      CR,LF,'G. SQW/OUT ON–1HZ        H. SQW/OUT OFF '
        DB      CR,LF,'ESC. TO QUIT                            ',0
TEXT1:
        DB      CR,'DATE:      ',0
TEXT2:
        DB      'TIME:  ',0
TEXT3:
        DB      CR,LF,0
TEXT4:
        DB      CR,LF,'PRESS ANY KEY TO RETURN'
        DB      CR,LF,0
TEXT5:
        DB      CR,LF,'ENTER THE BYTE VALUE WHICH WILL FILL THE RAM'
        DB      CR,LF,0
TEXT6:
        DB      CR,LF,'RAM            RAM'
        DB      CR,LF,'ADDR           DATA'
        DB      CR,LF,'-----------------------------'
        DB      CR,LF,0
;*******************************************
;**** END OF PROGRAM **********************
;*******************************************
END
```