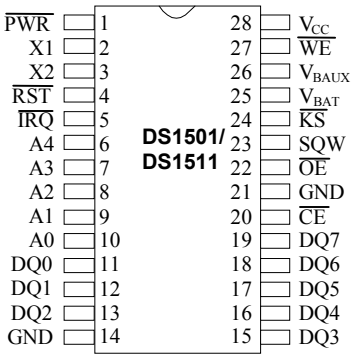
Application Note 508
Using the Power-Control Functions of the DS1501/DS1511 Real-Time Clock

www.maxim-ic.com

PIN ASSIGNMENT (Top View)



28-Pin DIP, 28-Pin SO

DESCRIPTION

The DS1501/DS1511 is a watchdog real-time clock (RTC) that provides additional functions for system power control. These features include a kickstart input, power-on output pin, and time of day/date alarm power enable.

In this example, the DS1511 is used to wake the system once per minute, output the time and date, wait one second and power-down. An auxiliary battery is required to use the power control features.

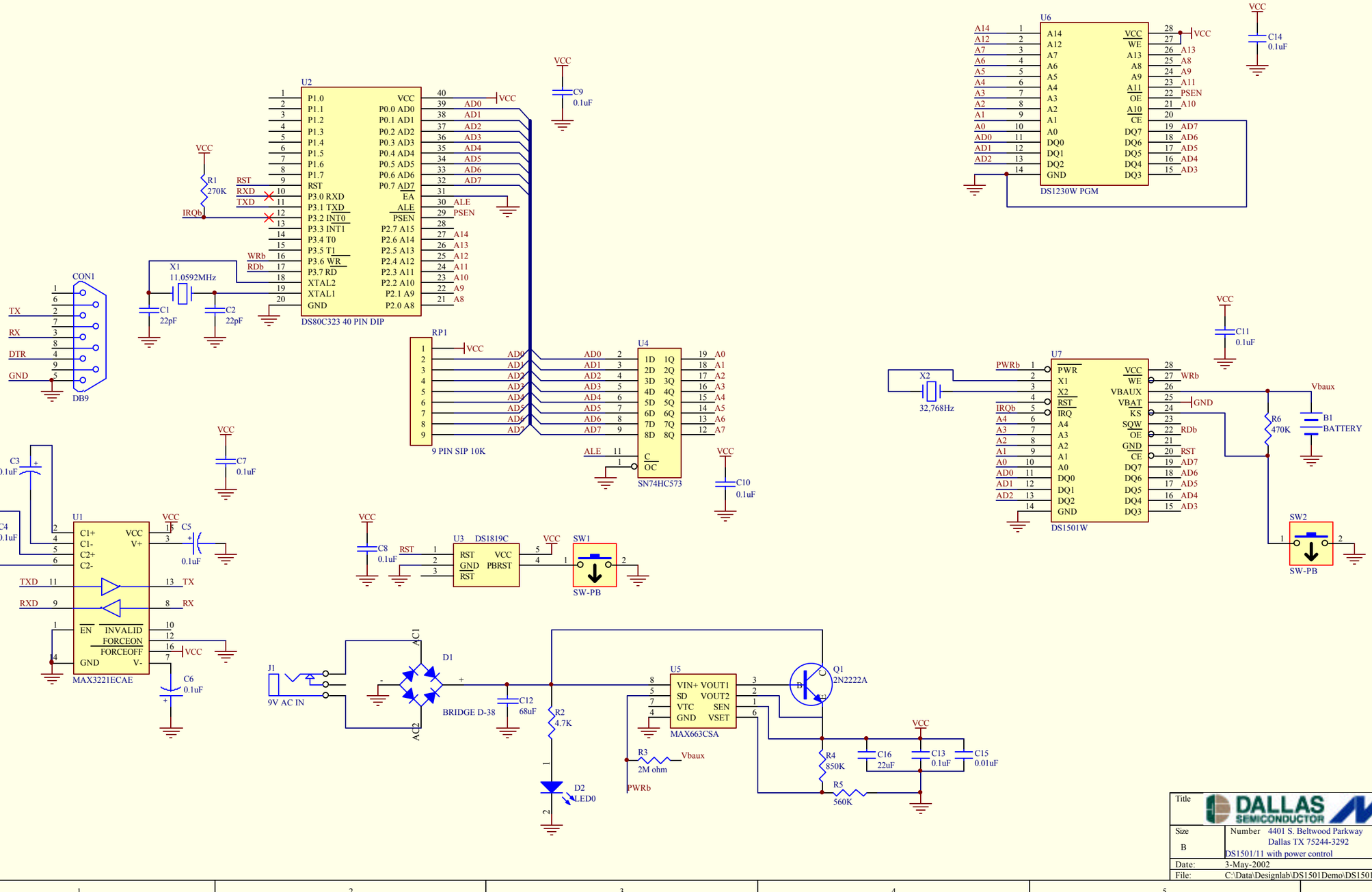
The circuit uses a DS80C323 3V 80C32-compatible microcontroller. A DS1230W is used for program memory. The DS1501/DS1511 is mapped into data memory. The example assembly code does not take advantage of the power management features of the DS80C323, which could reduce the total power consumption of the circuit.

A MAX663 voltage regulator is used to power the circuit. This regulator features a shutdown (SHDN) input to control the supply output. A pullup resistor connects SHDN to V_{BAUX}. The DS1511 $\overline{PWR}$ pin, when active, pulls SHDN low, turning on the supply. The drawback to this circuit is that current is drawn from V_{BAUX} while V_{CC} is active. When V_{CC} is off, the pullup resistor must be capable of holding SHDN above the minimum V_{IH} level. The combined maximum leakage for the $\overline{PWR}$ and SHDN pins is approximately 1µA, limiting the pullup to a maximum of about 2MΩ.

Another method of powering the SHDN input would use the unregulated V_{IN}. However, the V_{IH} maximum of the DS1511 must not be exceeded.

The kickstart input is functional as long as V_{BAUX} is present. The time of day/date power-enable bit (TPE) must be set to a 1 to allow a wake-up on alarm. In this example, the alarm registers are set to alarm when the seconds match.

A schematic of the circuit is shown in on page 2. Software follows on page 3.


U2
DS80C323 40 PIN DIP
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
RST
P3.0 RXD
P3.1 TXD
P3.2 INT0
P3.3 INT1
P3.4 T0
P3.5 T1
P3.6 WR
P3.7 RD
XTAL2
XTAL1
GND
VCC
P0.0 AD0
P0.1 AD1
P0.2 AD2
P0.3 AD3
P0.4 AD4
P0.5 AD5
P0.6 AD6
P0.7 AD7
EA
ALE
PSEN
P2.7 A15
P2.6 A14
P2.5 A13
P2.4 A12
P2.3 A11
P2.2 A10
P2.1 A9
P2.0 A8
VCC
C9
0.1uF
R1
270K
RST
RXD
TXD
IRQb
X1
11.0592MHz
C1
22pF
C2
22pF
WRb
RDb
CON1
TX
RX
DTR
GND
DB9
RP1
9 PIN SIP 10K
U4
SN74HC573
ALE
C10
0.1uF
C7
0.1uF
C3
0.1uF
C4
0.1uF
U1
MAX3221ECAE
C1+
C1-
C2+
C2-
VCC
V+
V-
TXD
RXD
TX
RX
EN
INVALID
FORCEON
FORCEOFF
C5
0.1uF
C6
0.1uF
U3 DS1819C
RST
GND
PBRST
C8
0.1uF
SW1
SW-PB
J1
9V AC IN
BRIDGE D-38
D1
C12
68uF
R2
4.7K
D2
LED0
U5
MAX663CSA
VIN+
VOUT1
SD
VOUT2
VTC
SEN
GND
VSET
Q1
2N2222A
R3
2M ohm
Vbaux
PWRb
R4
850K
R5
560K
C16
22uF
C13
0.1uF
C15
0.01uF
U6
DS1230W PGM
WE
OE
CE
C14
0.1uF
U7
DS1501W
PWR
X1
X2
RST
IRQ
VBAUX
VBAT
KS
SQW
R6
470K
B1
BATTERY
SW2
SW-PB
X2
32,768Hz
C11
0.1uF
Title
DALLAS SEMICONDUCTOR MAXIM
Size
B
Number
4401 S. Beltwood Parkway
Dallas TX 75244-3292
DS1501/11 with power control
Revision
B
Date:
3-May-2002
Sheet of 1
File:
C:\Data\Designlab\DS1501Demo\DS1501Demo.dsn
Drawn By:

CIRCUIT SOFTWARE

```
; Program DS1501.SRC
;
; This program demonstrates various features of the DS1501
;
CR EQU 0DH
LF EQU 0AH
MCON EQU 0C6H
TA EQU 0C7H
CSEG AT 0
AJMP START
CSEG AT 3             ; external event 0 vector
LJMP RD_RTC           ; print time & date and clear alarm
CSEG AT 02BH
RETI
;
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
; Initialize the serial port for 9600 baud
MOV TMOD,#20H  ; baud rate = timer overflow rate / 32
MOV TH1,#0FAH  ; 256-(11,059,200/192/9600)=256-6=FA
MOV TL1,#0FAH  ; 256-(14,745,600/192/19200)=256-4=FC
ORL PCON,#80H  ; PCON.7 = 1 for 192, 0 for 384 for baud calculation
MOV SCON,#52H
MOV TCON,#40H
; initialize external 0 interrupt
SETB   EX0
SETB   EA
;-------------------------------------------------------------------------
; THIS IS THE MASTER CONTROLLER LOOP
;-------------------------------------------------------------------------
; this code need only execute once, initializes registers
; wait for trec - on wakeup, this code must be executed
MOV    R0,#21  ; set up to loop 21 times
TREC:
MOV    TH0,#0  ; set up timer 0
MOV    TL0,#0  ; preset counters
CLR    TF0            ; clear timer overflow indicator
MOV    A,TMOD  ; get TMOD
ANL    A,#0F0H; gate=0,timer,mode 0, timer 1 not changed
MOV    TMOD,A  ; put back
SETB   TR0            ; start timer 0 running
TIM_LP:
JNB    TF0,TIM_LP     ; loop until timer overflow
DJNZ   R0, TREC       ; timer overflow ~9mS at 11.0MHz
;
; The following code assumes that the RTC has not been initialized.
; The RTC is set up so the code can execute to the main menu.
;
; disable alarms, enable transfers, oscillator
MOV    DPTR,#0005H    ; point to month register
MOVX   A,@DPTR; read the contents
CLR    ACC.7          ; enable the oscillator without disturbing the other bits
MOVX   @DPTR, A       ; write data to register
;
MOV    DPTR,#000FH    ; point to control B register
MOVX   A, @DPTR       ; get current settings
SETB   ACC.7          ; enable transfers
MOVX   @DPTR, A       ; write data to register
;
LCALL  ALMASK_INIT    ; initialize alarm registers and masks
;
MASTER_CONTROLLER:    ; this is where we return after each user entry
;
MOV DPTR, #TEXT0      ; put main menu on screen
```

```
LCALL WRITE_TEXT
;
LCALL READ_CLK ; print time and date on next clock increment
;
; loop read and print alarm and control register contents
;
MOV     DPTR,#0008H     ; point to alarm seconds
MOV     R0,#06H
READ_LOOP:
MOV     A,#' '
LCALL   WRITE_DATA
MOVX    A, @DPTR
LCALL WRITE_BCD        ;write data in acc to serial port
INC     DPTR
DJNZ    R0, READ_LOOP
;
MOV     A,#' '
LCALL   WRITE_DATA
MOVX    A, @DPTR       ; now read control A register (clears alarm flags when read)
MOV     B,A            ; save it
LCALL WRITE_BCD        ;write data in acc to serial port
;
MOV     A,#' '
LCALL   WRITE_DATA
INC     DPTR
MOVX    A, @DPTR       ; read control B register
LCALL WRITE_BCD        ;write data in acc to serial port
JNB     ACC.4, NOT_TDF ; and if TPE (Time of Day/Date Alarm Power Enable Bit) is true
JNB     B.3, NOT_TDF   ; if TDF (if Time of Day/Date Alarm Flag is true)
; debug
MOV     DPTR,#000EH     ; point to alarm seconds
MOVX    A, @DPTR       ; read control A register
LCALL WRITE_BCD        ;write data in acc to serial port
INC     DPTR
MOVX    A, @DPTR       ; read control A register
LCALL WRITE_BCD        ;write data in acc to serial port
;
; because A6 and earlier re-triggers if you write to the part within 1 sec of the alarm
; we need to wait
MOV     R0,#120 ; set up to loop 120 times
ONESEC:
MOV     TH0,#0  ; set up timer 0
MOV     TL0,#0  ; preset counters
CLR     TF0             ; clear timer overflow indicator
MOV     A,TMOD  ; get TMOD
ANL     A,#0F0H ; gate=0,timer,mode 0, timer 1 not changed
MOV     TMOD,A  ; put back
SETB    TR0             ; start timer 0 running
TIM_LP1:
JNB     TF0,TIM_LP1     ; loop until timer overflow
DJNZ    R0, ONESEC      ; timer overflow ~9mS at 11.0MHz
;
; power down if we woke up from an alarm event, but not from kickstart...
MOV     DPTR,#000EH     ; point to control A register
MOV     A,#10H  ; shut off power
MOVX    @DPTR, A
;
; somewhere in the following code, we'll stop if power has been shut off
;
NOT_TDF:
; The following code takes a keystroke and checks for match with
; a routine.
LCALL   READ_DATA       ; wait for keystroke
CLR ACC.5               ; convert character to upper case
;
CJNE A,#'E',NOTE
LCALL EWAKE             ; set up RTC to wake on alarm event
NOTE:
;
CJNE A,#'F',NOTF
LCALL FIL_RAM  ; fill extended RAM with data
NOTF:
;
CJNE A,#'I',NOTI
LCALL INT_RD   ; reading time & date on interrupt
NOTI:
```

```
;
CJNE A,#'L',NOTL
LP_RD:
LCALL READ_CLK ; loop reading the time & date
; wait until key entry to exit routine
JNB RI,LP_RD   ; LOOP WHILE RI BIT IS LOW
CLR RI
MOV A,SBUF     ; GET DATA BYTE FROM SERIAL BUFFER
MOV A, #0;     ; make sure keystroke doesn't match following checks
;
NOTL:
;
CJNE A,#'P',NOTP
LCALL PWR_OFF ; turn off power
NOTP:
;
CJNE A,#'R',NOTR
LCALL RD_RAM ; read extended RAM
NOTR:
;
CJNE A,#'S',NOTS
LCALL SET_CLOCKM ; call set clock function (manual set)
NOTS:
;
CJNE A,#'W',NOTW
LCALL WRT_REG ; write data to register
NOTW:
;
JMP MASTER_CONTROLLER
;---------------------------------------------------------------------
; This subroutine waits until the seconds register increments, then
; prints the time and date.
;---------------------------------------------------------------------
; Writes to the DS1501/11 in revision A7 and earlier can cause incorrect
; data to be transferred from the internal registers and the user copy.
; The work-around is to wait at least 488uS from the last write to the
; device.  This allows time for a "good" transfer to occur.  The TE bit
; should not be used; keep enabled.  The time and date registers must be
; read twice and compared.  If the time and date incremented while reading,
; the data will not match, and the registers should be read and compared
; again.
;---------------------------------------------------------------------
READ_CLK:
; enable transfers
MOV     DPTR,#000FH    ; point to control register b
MOVX    A, @DPTR       ; get present register values
SETB    ACC.7          ; set TE bit
MOVX    @DPTR, A
;
MOV     TH0,#1EH       ; set up timer 0
MOV     TL0,#0  ; preset counters
CLR     TF0            ; clear timer overflow indicator
MOV     A,TMOD         ; get TMOD
ANL     A,#0F0H ; gate=0,timer,mode 0, timer 1 not changed
MOV     TMOD,A         ; put back
SETB    TR0            ; start timer 0 running
TIM_LP4:
JNB     TF0,TIM_LP4    ; loop until timer overflow: ~ 500uS
;---------------------------------------------------------------------
; in this routine, we read the seconds register until we do not
; get a match.  Then we read all of the time & date registers.
; An interrupt after we break from the loop could cause a problem
; if it pushed the reads into the next update; ~ 1 second.
;---------------------------------------------------------------------
RD_CK_LP:
MOV     DPTR,#0000H    ; point to seconds
MOVX    A, @DPTR       ; read register
XRL     A, R0          ; if current secs match previous secs, A=0
JZ      RD_CK_LP       ;  so loop until secs change, else continue
;
MOV     A,#CR          ; print a carriage return
LCALL   WRITE_DATA
;
MOV     A,#LF          ; print a line feed
LCALL   WRITE_DATA
;
```

```
MOV     DPTR,#0007H     ; point to century
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     DPTR,#0006H     ; point to year
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#' '          ; print a space
LCALL   WRITE_DATA
;
MOV     DPTR,#0005H     ; point to month
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#'/'          ; print a slash
LCALL   WRITE_DATA
;
MOV     DPTR,#0004H     ; point to date
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#' '          ; print a space
LCALL   WRITE_DATA
;
MOV     DPTR,#0002H     ; point to hour
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#':'          ; print a slash
LCALL   WRITE_DATA
;
MOV     DPTR,#0001H     ; point to minutes
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#':'          ; print a ':'
LCALL   WRITE_DATA
;
MOV     DPTR,#0000H     ; point to seconds
MOVX    A, @DPTR        ; read register
MOV     R0,A            ; store new seconds value for next loop
LCALL WRITE_BCD         ; write BCD to serial port
;
RETI
;---------------------------------------------------------------------
; This subroutine reads the time and date upon an interrupt.  We should
; really wait 488uS on entry to make sure we had time for a good time
; and date transfer since the last write.
;---------------------------------------------------------------------
RD_RTC:
; save registers
PUSH    ACC
PUSH    PSW
PUSH    DPH
PUSH    DPL
PUSH    B
;
MOV     A,#CR           ; print a carriage return
LCALL   WRITE_DATA
;
MOV     A,#LF           ; print a line feed
LCALL   WRITE_DATA
;
MOV     DPTR,#0007H     ; point to century
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     DPTR,#0006H     ; point to year
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#' '          ; print a space
LCALL   WRITE_DATA
;
MOV     DPTR,#0005H     ; point to month
```

```
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#'/'          ; print a slash
LCALL   WRITE_DATA
;
MOV     DPTR,#0004H     ; point to date
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#' '          ; print a space
LCALL   WRITE_DATA
;
MOV     DPTR,#0002H     ; point to hour
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#':'          ; print a slash
LCALL   WRITE_DATA
;
MOV     DPTR,#0001H     ; point to minutes
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#':'          ; print a ':'
LCALL   WRITE_DATA
;
MOV     DPTR,#0000H     ; point to seconds
MOVX    A, @DPTR        ; read register
MOV     R0,A            ; store new seconds value for next loop
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#'-'          ; print a '-'
LCALL   WRITE_DATA
; clear alarm interrupt in RTC
MOV     DPTR,#000EH     ; point to control register
MOVX    A, @DPTR        ; read register
;
; restore registers
POP     B
POP     DPL
POP     DPH
POP     PSW
POP     ACC
RETI
;-------------------------------------------------------------------------
; This subroutine sets up reading the time and date on an once-per-second
; alarm.  Waits for a keystroke to exit.  On exit, disables alarm.
;-------------------------------------------------------------------------
INT_RD:
; set up part to once per second alarms
MOV     DPTR,#0008H     ; point to alarm register
MOV     A, #80H; set mask
MOVX    @DPTR, A
MOV     DPTR,#0009H     ; point to alarm register
MOVX    @DPTR, A
MOV     DPTR,#000AH     ; point to alarm register
MOVX    @DPTR, A
MOV     DPTR,#000BH     ; point to alarm register
MOVX    @DPTR, A
; now enable the time of day alarm
MOV     DPTR,#000FH     ; point to control register b
MOVX    A, @DPTR        ; get present register values
SETB    ACC.3           ; set the TIE (alarm interrupt enable) bit
MOVX    @DPTR, A
;
; enable transfers
MOV     DPTR,#000FH     ; point to control register b
MOVX    A, @DPTR        ; get present register values
SETB    ACC.7           ; set TE bit
MOVX    @DPTR, A
; wait until key entry to exit routine
IRDLP:
JNB RI,IRDLP ; LOOP WHILE RI BIT IS LOW
CLR RI
MOV A,SBUF ; GET DATA BYTE FROM SERIAL BUFFER
```

```
;
MOV     DPTR,#000FH     ; point to control register b
MOVX    A, @DPTR        ; get present register values
CLR     ACC.3           ; clear the TIE (alarm interrupt enable) bit
MOVX    @DPTR, A
;
RET
;---------------------------------------------------------------------
; THIS SUB READS DATA FROM THE SCREEN AND CONVERTS IT TO BCD FORMAT
; 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,DIGIT ; JUMP IF NOT A-F
ADD A,#-07H ; IF A-F SUBTRACT 7
DIGIT:
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 IN HEX FORMAT
; input: Acc  Exits: All registers unchanged
;---------------------------------------------------------------------
WRITE_BCD:
PUSH    ACC     ; save for 2nd nibble conversion
PUSH    ACC     ; save for return
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
POP     ACC     ; restore for return
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
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
;---------------------------------------------------------------------
; THIS SUB SETS THE CLOCK (MANUAL)
;---------------------------------------------------------------------
SET_CLOCKM:
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    DPTR,#0000H    ; point to minutes
MOV R1,#28H ; point to beginning of data in scratchpad
;
SEND_LOOP:
MOV A,@R1              ; move data out of scratchpad to Acc
MOVX   @DPTR, A        ; write data to register
INC R1
INC DPL
CJNE R1,#2FH,SEND_LOOP ; loop until all time/date data sent
MOV A, #20H
MOVX   @DPTR, A              ; send century data (fixed)
RET
;---------------------------------------------------------------------
; Set alarm registers and set masks to 'seconds match' operation
;---------------------------------------------------------------------
ALMASK_INIT:
MOV    DPTR,#0008H    ; point to alarm seconds
MOV    A,#10H         ; alarm when seconds match
MOVX   @DPTR, A       ; write data to register
;
MOV    DPTR,#0009H    ; point to alarm minutes
MOV    A,#80H         ; mask
MOVX   @DPTR, A       ; write data to register
;
```

```
MOV     DPTR,#000AH     ; point to alarm seconds
MOV     A,#80H          ; mask
MOVX    @DPTR, A        ; write data to register
;
MOV     DPTR,#000BH     ; point to alarm day/date
MOV     A,#80H          ; mask
MOVX    @DPTR, A        ; write data to register
;
MOV     DPTR,#000CH     ; point to Watchdog msec
MOV     A,#0H           ; mask
MOVX    @DPTR, A        ; write data to register
;
MOV     DPTR,#000DH     ; point to Watchdog msec
MOV     A,#0H           ; mask
MOVX    @DPTR, A        ; write data to register
RET
;-------------------------------------------------------------------------
; Set up RTC to wake up on seconds match if system is powered down
;-------------------------------------------------------------------------
EWAKE:
;
LCALL   ALMASK_INIT     ; set alarms to activate on seconds match
;
MOV     DPTR,#000FH     ; point to control B register
MOVX    A, @DPTR        ; get present register values
SETB    ACC.4           ; enable TPE (Time of Day/Date Alarm Power Enable Bit)
MOVX    @DPTR, A
;
MOV     DPTR,#000EH     ; point to control A register
MOVX    A, @DPTR        ; clear alarm flags
;
RET
;-------------------------------------------------------------------------
; power down system
;-------------------------------------------------------------------------
PWR_OFF:
MOV     DPTR,#000EH     ; point to control A register
MOV     A,#10H          ; shut off power
MOVX    @DPTR, A
RET
;-------------------------------------------------------------------------
; write data to register
;-------------------------------------------------------------------------
WRT_REG:
MOV     DPTR, #ADDRLOC
LCALL   WRITE_TEXT
LCALL   READ_BCD
MOV     R1,A
MOV     DPTR, #DATAVAL
LCALL   WRITE_TEXT
LCALL   READ_BCD
;
MOV     DPH,#0  ; clear upper 8 bits of DPTR
MOV     DPL,R1  ; set data pointer to address
MOVX    @DPTR, A        ; write data to register
RET
;
;-------------------------------------------------------------------------
; fill extended RAM with user defined data
;-------------------------------------------------------------------------
FIL_RAM:
MOV     DPTR, #DATAVAL
LCALL   WRITE_TEXT
LCALL   READ_BCD        ; get data value to write
MOV     R1,A
;
MOV     R0, #0  ; set address value
FRLP:
MOV     DPTR,#0010H     ; point to extended RAM address register
MOV     A, R0           ; put in acc
MOVX    @DPTR, A        ; write address to register
;
MOV     DPTR,#0013H     ; point to extended RAM data register
MOV     A, R1           ; get data value to write
MOVX    @DPTR, A        ; write data to register
INC     R0
```

```
CJNE    R0, #0, FRLP; read all 256 locations
RET
;
;------------------------------------------------------------------------
; read extended -- from 0 to 256
;------------------------------------------------------------------------
RD_RAM:
MOV     A,#CR           ; print a carriage return
LCALL   WRITE_DATA
;
MOV     A,#LF           ; print a line feed
LCALL   WRITE_DATA
;
MOV     R1, #0  ; set address value
RLP:
MOV     DPTR,#0010H     ; point to extended RAM address register
MOV     A, R1           ; put extended RAM address in acc
MOVX    @DPTR, A        ; write address value to register
;
MOV     DPTR,#0013H     ; point to extended RAM data register
MOVX    A, @DPTR        ; read data from register
LCALL   WRITE_BCD       ; print to screen
MOV     A, #' '
LCALL   WRITE_DATA
INC     R1
CJNE    R1, #0, RLP     ; read all 256 locations
RET
;
;------------------------------------------------------------------------
; read clock, display data once per second
;------------------------------------------------------------------------
CLK_RD:
;
MOV     DPTR,#000FH     ; point to control register b
MOVX    A, @DPTR        ; get current settings
SETB    ACC.7           ; enable transfers
MOVX    @DPTR, A
;
MOV     TH0,#0          ; set up timer 0
MOV     TL0,#0          ; preset counters
CLR     TF0             ; clear timer overflow indicator
MOV     A,TMOD          ; get TMOD
ANL     A,#0F0H ; gate=0,timer,mode 0, timer 1 not changed
MOV     TMOD,A          ; put back
SETB    TR0             ; start timer 0 running
TIM_LP3:
JNB     TF0,TIM_LP3     ; loop until timer overflow
;
MOV     DPTR,#000FH     ; point to control register b
MOV     A,#0            ; disable transfers
MOVX    @DPTR, A
;
MOV     DPTR,#0000H     ; point to seconds
MOVX    A, @DPTR        ; read register
;
MOV     DPTR,#0007H     ; point to century
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     DPTR,#0006H     ; point to year
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#' '          ; print a space
LCALL   WRITE_DATA
;
MOV     DPTR,#0005H     ; point to month
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#'/'          ; print a slash
LCALL   WRITE_DATA
;
MOV     DPTR,#0004H     ; point to date
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
```

```
;
MOV     A,#' '          ; print a space
LCALL   WRITE_DATA
;
MOV     DPTR,#0002H     ; point to hour
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#':'          ; print a slash
LCALL   WRITE_DATA
;
MOV     DPTR,#0001H     ; point to minutes
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#':'          ; print a ':'
LCALL   WRITE_DATA
;
MOV     DPTR,#0000H     ; point to seconds
MOVX    A, @DPTR        ; read register
LCALL WRITE_BCD         ; write BCD to serial port
;
MOV     A,#' '          ; print a space
LCALL   WRITE_DATA
;
MOV     DPTR,#000FH     ; point to control register b
MOV     A,#80H          ; enable transfers
MOVX    @DPTR, A
;
MOV     DPTR,#000BH     ; point to extended RAM address register
MOV     A,#0            ;
MOVX    @DPTR, A        ; write data to register
;
MOV     DPTR,#000BH     ; point to extended RAM data register
MOV     A,#55H          ;
MOVX    @DPTR, A        ; write data to register
;
JMP CLK_RD
;-------------------------------------------------------------------------
; TEXT STRINGS USED FOR USER INTERFACE OVER SERIAL PORT
;-------------------------------------------------------------------------
ADDRLOC:
DB CR,LF,'Address (HEX): ',0
DATAVAL:
DB 'DATA (HEX): ',0
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
TEXT0:
DB CR,LF,'Dallas Semiconductor Maxim DS1501 build 119'
DB CR,LF,'F)ill Ex RAM  R)ead Ex RAM I)nt Read'
DB CR,LF,'L)oop Read    P)ower off   S)et time'
DB CR,LF,'W)rite reg    E)nable Wake on alarm',CR,LF,0
;**** END OF PROGRAM ************
END
```