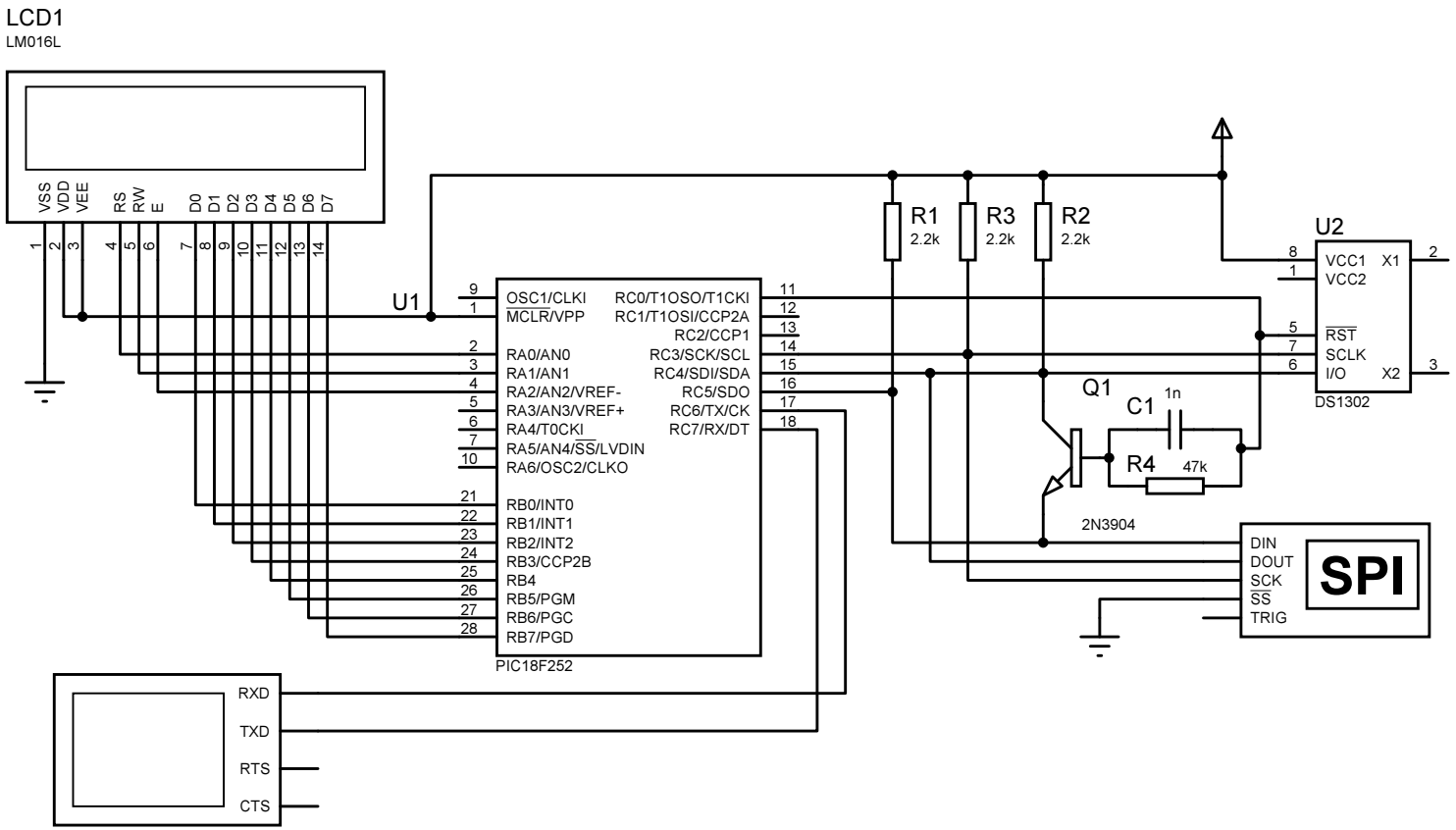
LCD1
LM016L
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
U1
OSC1/CLKI
MCLR/VPP
RA0/AN0
RA1/AN1
RA2/AN2/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/AN4/SS/LVDIN
RA6/OSC2/CLKO
RB0/INT0
RB1/INT1
RB2/INT2
RB3/CCP2B
RB4
RB5/PGM
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2A
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
PIC18F252
R1
2.2k
R3
2.2k
R2
2.2k
U2
VCC1
VCC2
RST
SCLK
I/O
X1
X2
DS1302
Q1
2N3904
C1
1n
R4
47k
DIN
DOUT
SCK
SS
TRIG
SPI
RXD
TXD
RTS
CTS

```
'****************************************************************
'*  Name    : lcd_rts_spi.BAS                                   *
'*  Author  : Boyan Petrov as Fob
*
'*  Notice  : Copyright (c) 2014 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                               *
'*  Date    : 27.03.14                                          *
'*  Version : 1.0                                               *
'*  Notes   : DS1302 with hw spi                                *
'*          :                                                   *
'****************************************************************
DEFINE OSC 20
ADCON1 = $07

;define INTHAND TmrInt

DEFINE LCD_BITS  8
DEFINE LCD_DREG  PORTB
DEFINE LCD_DBIT  0
DEFINE LCD_RSREG  PORTA
DEFINE LCD_RSBIT  0
DEFINE LCD_RWREG  PORTA
DEFINE LCD_RWBIT  1
DEFINE LCD_EREG  PORTA
DEFINE LCD_EBIT  2

DEFINE HSER_RCSTA 90h
DEFINE HSER_TXSTA 24h
DEFINE HSER_BIT 8
DEFINE HSER_BAUD 9600
DEFINE HSER_CLROERR 1
'*****************************************************************************
; Aliaces
; System
CKP         VAR SSPCON1.4   ; Clock Polarity Select
SMP         VAR SSPSTAT.7   ; Data Sample
CKE         VAR SSPSTAT.6   ; Clock Edge Select
BF          VAR SSPSTAT.0   ' SSP buffer full
SSPEN     VAR   SSPCON1.5   ' SSP enable
WCOL       VAR  SSPCON1.7   ' SSP write collision
SSPIF     VAR   PIR1.3      ' SSP interrupt flag

;User
RTC_CS      VAR PORTC.0
SDC_CS      VAR PORTC.1     ; not used for now
'*****************************************************************************
;Variables
SSP_byte_in     VAR BYTE
SSP_byte_out    VAR BYTE

cntb            VAR BYTE       ; byte counter
temp            VAR BYTE
tempw           VAR WORD

DBA             VAR BYTE[8]    ; Data byte array - BCD date and time
DWA             VAR WORD[8]    ; Data word array - ASCII date and time
RTC_MODE        VAR BIT        ; RTC Mode - CLOCK/BURST  -> 0/1
oldsec          VAR BYTE       ; for LCD and UART update
'*****************************************************************************
; ports
TRISA = 0
```

```
TRISB = 0
TRISC = 0
TRISC.4 = 1     ; spi DATA IN

    GOTO main

; ASM interrupt here if any

'*****************************************************************************
; SPI Send/Receive
SSP_TXRX:
                                    ; Assign SSP_byte_out = $FF for read
;       SSP_byte_in = SSPBUF        ; Clear the buffer.
        SSPIF = 0                   ; Clear the interrupt flag.
        SSPBUF = SSP_byte_out       ; Send the byte.
;       If (WCOL) Then Return       ; Check for write collision if more spi.
        WHILE !SSPIF                ; Wait for send to complete.
        WEND
        SSP_byte_in = SSPBUF        ; Get the byte.

RETURN

;*****************************************************************************
; Subroutine to write in clock mode
; Input:
; address byte -> tempw.highbyte
; byte to send -> tempw.lowbyte
; Output: none
SET_RTC_CM:
        RTC_CS = 1
        PAUSEUS 10
        SSP_byte_out  = tempw.HIGHBYTE REV 8
        GOSUB ssp_txrx
        SSP_byte_out  = tempw.LOWBYTE  REV 8
        GOSUB ssp_txrx
        RTC_CS = 0
        PAUSEUS 10
RETURN
; *****************************************************************************
; Subroutine to read in clock mode
; Input : address byte -> temp
; Output : read byte -> temp
GET_RTC_CM:

        RTC_CS = 1
        PAUSEUS 10
        SSP_byte_out  = temp REV 8
        GOSUB ssp_txrx
        SSP_byte_out  = $FF
        GOSUB ssp_txrx
        temp = SSP_byte_in  REV 8
        RTC_CS = 0
        PAUSEUS 10

RETURN

 ;
*****************************************************************************
; Subroutine to read in burst mode
; Input : none
; Output : Byte Array DBA[cntb]
```

```
GET_RTC_BM:

        RTC_CS = 1
        PAUSEUS 10
        SSP_byte_out  = $FD ; reordered $BF
        GOSUB ssp_txrx

    FOR cntb = 0 TO 7
        SSP_byte_out  = $FF
        GOSUB ssp_txrx
        DBA[cntb] = SSP_byte_in  REV 8

    NEXT cntb

        PAUSEUS 10
        RTC_CS = 0
        PAUSEUS 10

    RETURN
;
*****************************************************************************
; Subroutine to convert BCD_TO_ASCII
; Input : temp -> data in BCD
; Output : tempw -> 2 bytes ASCII
BCD_TO_ASCII:
    tempw.HIGHBYTE = temp
    tempw.HIGHBYTE = tempw.HIGHBYTE & $F0
    tempw.HIGHBYTE = tempw.HIGHBYTE >> 4
    tempw.HIGHBYTE = tempw.HIGHBYTE  + $30
    tempw.LOWBYTE = temp
    tempw.LOWBYTE = tempw.LOWBYTE & $0F
    tempw.LOWBYTE = tempw.LOWBYTE + $30
RETURN
;*****************************************************************************

 main:

    RTC_CS = 0

 ; Init LCD
     LCDOUT $FE, 1
     PAUSE 200
     LCDOUT $FE, $0C
     PAUSE 200
     LCDOUT $FE, $06
     PAUSE 200

; init SSP
    SSPCON1 = %00010010
    SSPSTAT = %00000000     ; slow
    SMP = 0                 ; middle
    CKP = 1                 ; iddle high
    cke = 0                 ; rising edge
    SSPEN = 1               ; enable
    RTC_MODE = 1            ; 0 - burst
                            ; 1 - clock

; init RTC:
; check halt flag in seconds register
; and set it if clear
    temp = $81
```

```
    GOSUB get_rtc_cm
    IF temp.7 = 1 THEN
        temp.7= 0
        tempw.HIGHBYTE = $80
        tempw.LOWBYTE = temp
        GOSUB  Set_rtc_cm
    ENDIF

;*****************************************************************************

 LOOP:

    IF RTC_MODE THEN
        ; Read all byte by byte
        FOR cntb = 0 TO 7
            temp = $81 + 2*cntb
            GOSUB get_rtc_cm
            DBA[cntb] = temp
        NEXT
    ELSE
        ; Read all at once
        GOSUB get_rtc_bm
    ENDIF

    PAUSE 100

    ;Check if change to update
    IF oldsec = DBA[0] THEN GOTO LOOP
    oldsec = DBA[0]

     ; BCD UART here
    HSEROUT [ HEX2 DBA[2],":", HEX2 DBA[1],":", HEX2 DBA[0], 10, 13  ]
    HSEROUT [ HEX2 DBA[4],":", HEX2 DBA[5],":", HEX2 DBA[6], 10, 13  ]

   ; BCD LCD here
    LCDOUT $FE, $80,"Time ", HEX2 DBA[2],":", HEX2 DBA[1],":", HEX2 DBA[0]
    PAUSE 100
    LCDOUT $FE, $C0,"Date ", HEX2 DBA[4],":", HEX2 DBA[5],":", HEX2 DBA[6]


 GOTO LOOP

 END
```