# ECED 3204 Microprocessor

# Assignment #7 Reference Solution

# http://www.jasongu.org/3204/assignments.html

Assignment #7 contains the following problems:

**E14.1** Configure the MEGA SPI module with the specified setting:

```
        ldi     r20,0x79        ; master mode, SCK idle low, shift data on falling edge
        out     SPCR,r20        ; disable SPI interrupts, shift data lsb first, baud rate = 1 MHz
        clr     r20
        out     SPSR,r20        ; clear SPI2X flag
        in      r20,SPDR        ; clear all SPI flags
        ldi     r20,0x07        ; configure SS/PB0, SCK/PB1,MOSI/PB2, MISO/PB3
        out     DDRB,r20        ; pin direction
```

**E14.4** Interface the MC14489 with the MEGA1280 using the SPI interface.

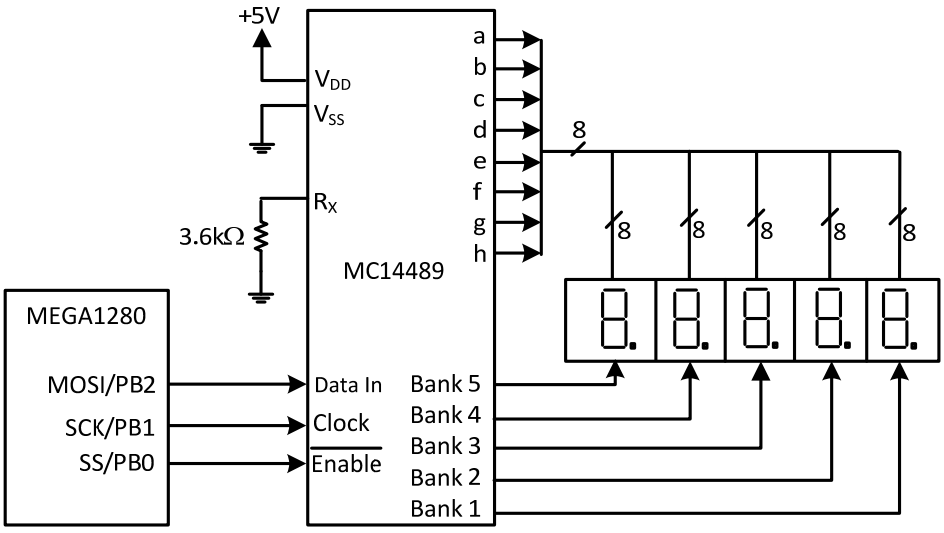


Figure E14.4 MEGA1280 driving five 7-segment displays using an MC14489

The assembly program to display 13579 on the seven-segment displays driven by the MC14489:

```
        .include <m1280def.inc>
        .cseg
        .org    0x00
        rjmp    start
        .org    0xF6
start:  ldi     r20,low(RAMEND)
        out     SPL,r20
        ldi     r20,high(RAMEND)
        out     SPH,r20
        rcall   initSPI
```

```
                ldi     r16,0x01            ; configures MC14489 to normal, hex decode mode
                rcall   initMC14489         ;      "
                ldi     ZL,low(dispDat<<1)
                ldi     ZH,high(dispDat<<1)
                ldi     r21,3
                cbi     PORTB,0             ; enable SPI transfer to MC14489
tloop:          lpm     r16,Z+
                rcall   putcSPI_master
                dec     r21
                brne    tloop
                sbi     PORTB,0             ; update seven-segment displays
again:          nop
                rjmp    again

dispDat:        .db     0x81,0x35,0x79      ; data to display 13579 in normal brightness on MC14489
                .include "spiUtil_mega.asm"
; ---------------------------------------------------------------------------------------------------------------------
-
; The following subroutine enable SPI to master mode, shift data msb first at 4 MHz on the
; rising edge of the SCK clock with SCK idle low.
; ---------------------------------------------------------------------------------------------------------------------
-
initSPI:        ldi     r20,0x50            ; enable SPI to master mode, disable interrupt,
                out     SPCR,r20            ; set shift rate to 4 MHz, shift data on rising edge, msb first
                ldi     r20,SPSR            ; clear all SPI flags
                lds     r20,SPDR            ;      "
                clr     r20                 ; clear SPI2X bit
                out     SPSR,r20            ;      "
                ldi     r20,0x07            ; configures MISO/PB3, MOSI/PB2, SCK/PB1, SS/PB0 pin
                out     DDRB,r20            ; direction
                ret
; ---------------------------------------------------------------------------------------------------------------------
; This subroutine configures the MC14489's C5~C1 to hex decode mode with normal brightness.
; ---------------------------------------------------------------------------------------------------------------------
initMC14489:
                cbi     PORTB,PB0           ; enable SPI transfer to MC14489
                call    putcSPI_master      ; output the configuration command to MC14489
                sbi     PORTB,PB0           ; execute configuration command
                ret
```

**E14.15** Write a program to display the character string **AbCdEFyH35** on the seven-segment displays driven by two MC14489s connected to the SPI module of the MEGA2560.
**Solution**: The configuration data to be send to these MC14489 is 0xD8,0x00,0x00,0x01; whereas display data is 0x8F,0xC2,0x35,0x8A,0xBC,0xDE.
The program is as follows:

```
                .include <m2560def.inc>
                .cseg
                .org    0x00
                rjmp    start
                .org    0xF6
start:          ldi     r20,low(RAMEND)     ; set up stack pointer
```

```
                out     SPL,r20             ;     "
                ldi     r20,high(RAMEND)    ;     "
                out     SPH,r20             ;     "
                ldi     r16,3               ; configure USART1 to MSPI mode and shift data
                ldi     r17,0               ; at 2 MHz
                call    initMegaMSPI        ;     "
; send out configuration data to all three MC14489s
                ldi     r28,4
                cbi     PORTD,PD0           ; enable SPI transfer to MC14489
                ldi     ZL,low(confDat<<1) ; set up configuration data pointer
                ldi     ZH,high(confDat<<1);      "
loop1:          lpm     r16,Z+
                call    putch_USART_MSPI
                dec     r28
                brne    loop1
                sbi     PORTD,PD0
; send out display data to 2 MC14489s
                ldi     r28,6               ; set up loop count
                ldi     ZL,low(dispDat<<1)  ; set up table pointer
                ldi     ZH,high(dispDat<<1);      "
                cbi     PORTD,PD0           ; enable SPI transfer to MC14489
loop2:          lpm     r16,Z+              ; transfer display data to MC14489
                call    putch_USART_MSPI;     "
                dec     r28                 ;     "
                brne    loop2               ;     "
                sbi     PORTD,PD0           ; update seven-segment displays
forever:        nop
                rjmp    forever
confDat:        .db     0xD8,0x00,0x00,0x01
dispDat:        .db     0x8F,0xC2,0x35,0x8A,0xBC,0xDE
; ---------------------------------------------------------------------------------------------------------------
; This subroutine initializes the USART1 of the MEGA AVR to MSPI mode, and also enables
; reception and transmission. The setting of baud rate is passed in r17:r16.
; ---------------------------------------------------------------------------------------------------------------
initMegaMSPI:
                clr     r20
                sts     UBRR0H,r20
                sts     UBRR0L,r20
                in      r20,DDRD
                andi    r20,0xD4            ; clear bit 5,3,1,0
                ori     r20,0x29            ; set bit 5, 3, and 0
                out     DDRD,r20            ; configure PD5/XCK1, PD3/TXD1 for output
                ldi     r20,(1<<RXEN1)|(1<<TXEN1) ; enable receiver and transmitter
                sts     UCSR1B,r20
                ldi     r20,(1<<UMSEL11)|(1<<UMSEL10)|(0<<UCPHA1)|(0<<UCPOL1)
                sts     UCSR1C,r20          ; select MSPI mode
                sts     UBRR1H,r17          ; set up baud rate
                sts     UBRR1L,r16          ;     "
                ret
; ---------------------------------------------------------------------------------------------------------------
; The following subroutine outputs a character in r16 to USART1 in MSPI mode.
; ---------------------------------------------------------------------------------------------------------------
putch_USART_MSPI:
                lds     r20,UCSR1A          ; wait until USART data register is empty
                sbrs    r20,UDRE1           ;     "
                rjmp    putch_USART_MSPI ;     "
```

```
            sts     UDR1,r16        ; start data transmission
waitTXC1:   lds     r20,UCSR1A      ; wait until the character has been shifted out
            sbrs    r20,RXC1        ;     "
            rjmp    waitTXC1        ;     "
            lds     r22,UDR1        ; clear RXC1 flag
            ret
```

**E15.6** Use the polling method to write a n assembly subroutine to set up the TH or TL value to the DS1631A shown in Figure 15.26. The high byte and low byte of TH or TL are passed in r16 and r17 respectively

```
; ---------------------------------------------------------------------------------------------------------------------
; This subroutine sets up TH or TL. The value is passed in r16 and r17. When r18 holds 1, the
; subroutine sets up TH. Otherwise, it sets up TL.
; ---------------------------------------------------------------------------------------------------------------------
setupTHoTL:
            call    startTWI        ; generate a START condition
            push    r16
            ldi     r16,DS1631      ; send out control byte + /W bit
            call    sendByteTWI     ;     "
            cpi     r18,1           ; check what command to send
            brne    setTL           ;     "
            ldi     r16,accessTH    ; send AccessTH command
            call    sendByteTWI     ;     '
            rjmp    sndVal
setTL:      ldi     r16,accessTL    ; send AccessTL command
            call    sendByteTWI     ;     "
sndVal:     pop     r16             ; restore high byte of value
            call    sendByteTWI     ; send out the high byte
            mov     r16,r17         ; send out the low byte
            call    sendByteTWI     ;     "
            call    stopTWI
            ret
```