```
AVRASM ver. 2.1.52  C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\
ese 380 lab\lab 8\MGFM4\MGFM4\MGFM4.asm Tue Oct 28 19:26:09 2014
C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 8\
MGFM4\MGFM4\MGFM4.asm(36): Including file 'C:\Program Files (x86)\Atmel\Atmel Toolchain\AVR Assembl
er\Native\2.1.39.1005\avrassembler\Include\m16def.inc'C:\Users\radra_000\Box Sync\college sophomore
 fall 2014\fall 2014 notes and files\ese 380 lab\lab 8\MGFM4\MGFM4\MGFM4.asm(317): Including file '
 C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab
  8\MGFM4\MGFM4 \lcd_dog_asm_driver_m16A.inc'


                 * MGFM4.asm
                 *
                 ; This program will be the same as the previous lab. except the gate period will
                 ; be check by the timer and interrupt. The lcd display will now display in decimial
                 ; instead of hex. This is done using a subroutine provided by the atmel corp
                 ; The Lcd setup code is copied from asm file provided in the previous lab.
                 ;the gater period of this code will be 1/2 second, meaning both positive and
                 ; negative edges will be counted. when the pushbutton is pressed it will alternate
                 ; between counting between 1/2 sec and 1 sec. To create this code, freq 2
                 ; and freq 3 subroutines are copied from the previous 2 codes.

                 ;inputs - pa7(freq generator)
                 ;outputs - pb(J2 - connecting to the lcd)
                 ;switches are connected to pd, but since they are not used
                 ;they will not be initialized.
                 ;
                 ; register modified for the code i added:
                 ; r16 - general pupose
                 ; r8 and r9 - used as the positive edge counter
                 ; r25  - has the values of the freq, which will later be sent to y
                 ; r8, r9 - positive edge counters
                 ;
                 ; r17 - incremented for 6 times, used to empty spaces on the lcd
                 ; for 6 times. The frequency will be located at the center
                 ;
                 ; r27 - set to 6, so we can compare and stop after r17 is inc
                 ; 6 times.
                 ; r1,r2,r3,r4, r5 are used to unpack and contain the values stored
                 ; in r8 and r9, which will later be accessed to display the freq
                 ;
                 */


                .LIST


                 ;*******************************************************************
                    .cseg
                    .org 0                          ;reset/restart code entry point <<<<<<<<
000000 940c 0019    jmp reset

                    //when the counter is equal to 1s, the interrupt will be called
                    //0x0E is the adress for timer compare match B.
                    //when the interrupt is called, it will jump to isr_tc0_display
                    .org 0x002                      ;when external interrupt is pressed
000002 940c 0060    jmp switch                      ;its located on pd2

                    .org 0x00E                      ; timer intrrupt
00000e 940c 0059    jmp isr_tc0_display             ;goes here when the time is up

                ;*********************TIMER INITIALIZATION**************************
                ; THE COUNTER, FLAGS ARE INITIALIZED.
                ;_____________________________________________________________________

                start_tc1:
```

```
                //intialization for the timer
000010 e000        ldi r16, $00
000011 bd0d        out TCNT1H, r16                    ; set up counter with 0's
000012 bd0c        out TCNT1L, r16                    ;
                   ;Init Timer/counter Interrupt MaSK (TIMSK) register to enable/set
000013 e008        ldi r16, 8                         ;load with bcd 1000, this will enable the
                                                      ;"ocie1b" which is located in bit 4 of the
                                                      ;register.
                                                      ;refer to datasheet pg 115 for details
000014 bf09        out TIMSK, r16                     ;set up the timer interrupt

000015 e200        ldi r16, 1<<ICF1                   ;loading the timer interrupt flag register
000016 bf08        out TIFR, r16                      ;



000017 9478        sei                                ;enable global interrupts...


                   ;TCCR1B = FOC0 : WGM11 : COM11 : COM10 : WGM11 : CS12 : CS11 : CS10
                   ; 0 0 0 0 0 0 1 1
                   ; FOC Off; No WF Gen; COM=Nrml Port Op; Pre-scaler= 1/64


000018 9508        ret
                ;____________________________________________________________________


                reset:
000019 e50f        ldi r16, low(ramend)
00001a bf0d        out spl, r16
00001b e004        ldi r16, high(ramend)
00001c bf0e        out sph, r16                       ;initialize stack pointer


00001d e000        ldi r16, $00
00001e bb01        out ddrd, r16                      ;set up the port b as inputs to read the pbsw
                                                      ;values
                                                      ; and nand gate input on pd(2)

00001f ef0f        ldi r16, 0xff                      ; set portB = output.
000020 bb07         out DDRB, r16                     ; for lcd display
000021 9ac4         sbi portB, 4                      ; set /SS of DOG LCD = 1 (Deselected)1

000022 e001        ldi r16, 1                         ; set DDRC for all in but PC0
000023 bb04        out DDRC, r16
000024 9aa8        sbi PortC, 0                       ; turn off sounder

000025 e400        ldi r16,0b01000000                 ; set up port a to
000026 bb0a        out ddra, r16                      ;read the frequency input on pa7 and output pulse
                                                      ; on pa6

000027 d0bc        rcall init_lcd_dog                 ; init display, using SPI serial interface


000028 dfe7        rcall start_tc1                    ;init the timer counter 1
000029 e0b1        ldi r27, 1                         ;initialization of r27
                       /*
                   ldi r16, 28
                   mov r9, r16
                   ldi r16, 60
                   mov r8, r16*/                      ;for testing in simulation

                main:
                   //start timer code
00002a e003        ldi r16, 0<<CS12|1<<CS11|1<<CS10   ;load 64 PRESCaLE TCCR0 value.
```

```
00002b bd0e         out TCCR1B, r16                     ;and start timer
00002c ffb0         sbrs r27, 0                         ;if the last bit is 1
00002d d006         rcall frequency_meter_2             ;load the timer and count the frequency
00002e fdb0         sbrc r27, 0
00002f d017         rcall frequency_meter_3
000030 94e8         clt                                 ;clear the t flag
000031 d120         rcall unpack                        ;when timer is done, unpack the edge counts
000032 d03e         rcall message_dsp_loop              ;after its unpacked, display
000033 cff6         rjmp main

                 /*************************************************************************
                    frequency_meter_2
                       This subroutine uses r9:r8 to store the positive edge counters
                       Every time there is a logic change from 0 to 1 or 1 to 0 it updates the
                       new value into r25 and if it is a change from 0 to 1 then the edge
                       counter is incremented. This program runs for 1 second until the t flag
                       set
                 *************************************************************************/
                 frequency_meter_2:
000034 e30c         ldi r16, $3C                        ;load the counter with 15625
000035 bd09         out OCR1BH, r16                     ; so that we will get 1s
000036 ea04         ldi r16, $A4
000037 bd08         out OCR1BL, r16                     ;
000038 e000         ldi r16, $00
000039 2e80         mov r8, r16
00003a 2e90         mov r9, r16
00003b b399         in r25, pinA                ;and positive edge counter
                    check_edge:
00003c f04e             brts finish
00003d b309             in r16, pina            ;take in the current wave signal logic
00003e 1709             cp r16,r25              ;and compare to previous logic recorded,
00003f f3e1             breq check_edge         ;if it is the same then skip to tweak delay
000040 2f90             mov r25,r16
000041 f3d0             brcs check_edge         ;if there is a carry then branch
000042 9483             inc r8                  ;and then increment the counter
000043 f7c1             brne check_edge         ;if it didnt over count then go to tweak delay
000044 9493             inc r9                  ;if so then increment the second register
000045 f7b6             brtc check_edge
                    finish:
000046 9508             ret

                 /*************************************************************************
                  frequency_meter_3
                       This subroutine uses r9:r8 to store the positive edge counters
                       Every time there is a logic change from 0 to 1 or 1 to 0 it updates the
                       new value into r25 and for every logic change the edge counter is
                       incremented. This program runs for half a second until the t flag is set
                 *************************************************************************/
                 frequency_meter_3:
000047 e10e         ldi r16, $1E                        ;load the counter with 15625
000048 bd09         out OCR1BH, r16                     ; so that we will get .5s
000049 e804         ldi r16, $84
00004a bd08         out OCR1BL, r16                     ;
00004b e000         ldi r16, $00
00004c 2e80         mov r8, r16
00004d 2e90         mov r9, r16
00004e b399         in r25, pinA                ;and positive edge counter
                    check_edge1:
00004f f3fe             brts check_edge1
000050 b309             in r16, pina            ;take in the current wave signal logic
000051 1709             cp r16,r25          ;and compare to previous logic recorded,
000052 f3e1             breq check_edge1            ;if it is the same then skip to tweak delay
000053 2f90             mov r25, r16
000054 9483             inc r8              ;and then increment the counter
000055 f7c9             brne check_edge1            ;if it didnt over count then go to tweak delay
000056 9493             inc r9              ;if so then increment the second register
```

```
000057 f7be             brtc check_edge1
000058 9508             ret




                ;-----------------------------
                ;when the interrupt is called it will jump to this subroutine
                ;set the T flag, resets the timer, update the display
                ;-----------------------------


                isr_tc0_display:
000059 930f             push r16
00005a 9468             set                             ;set the tflag
00005b e000             ldi r16, $00
00005c bd0d             out TCNT1H, r16                 ;
00005d bd0c             out TCNT1L, r16                 ;reset the timer counter
00005e 910f             pop r16
00005f 9518             reti
                /*************************************************************************
                 switch
                        This subroutine debounces the input coming in from port d and then
                         complements r27 where if it is cleared frequency meter 2 will be
                         executed and the other frequency meter is executed otherwise.
                *************************************************************************/
                switch:
000060 930f        push r16
000061 b70f        in r16, sreg
000062 930f        push r16
000063 931f        push r17
000064 b300        in r16, pind
000065 7600        andi r16,$60
000066 d120        rcall delay
000067 b310        in r17, pind
000068 7610        andi r17, $60
000069 2310        and r17,r16
00006a f009        breq finish1
00006b 95b0        com r27
                finish1:
00006c 911f        pop r17
00006d 910f        pop r16
00006e bf0f        out sreg, r16
00006f 910f        pop r16
000070 9518        reti


                ;********************LCD DISPLAY CODE*****************************
                ;-------------------------------------------------------
                ;Code to load and display each line on the lcd
                ;r25 is used to load the value of the each digit to the pointer
                ;line 2 refers to table, which containes numbers and depending
                ;on the frequncy, each number is picked and displayed
                ;-------------------------------------------------------

                message_dsp_loop:
000071 d0b5        rcall clr_dsp_buffs             ; clear all three buffer lines
000072 d08f        rcall update_lcd_dog                ;
000073 ffb0        sbrs r27, 0                     ;if the last bit is 1
000074 c002        rjmp regular1
000075 fdb0        sbrc r27, 0
000076 c01d        rjmp regular2

                regular1:
```

```
                     ;load 1st line of prompt message into dbuff1
000077 e0f1          ldi  ZH, high(line1_message<<1)     ;
000078 e0e2          ldi  ZL, low(line1_message<<1)      ;
000079 d0b5          rcall load_msg                      ; load message into buffer(s).


                     ;LOAD 2ND LINE OF THE MESSAGE INTO DBUFF2
00007a e0f1          ldi ZH, high(line2_message<<1)      ;
00007b e1e4          ldi ZL, low(line2_message<<1)       ;load the table to stack
00007c d0b2          rcall load_msg                      ;load the frequency number into the buffer


                     ;load 3rd line of prompt message into dbuff3
00007d e0f1          ldi  ZH, high(line3_message<<1)     ;
00007e e1e6          ldi  ZL, low(line3_message<<1)      ;
00007f d0af          rcall load_msg                      ; load message into buffer(s).
000080 d081          rcall update_lcd_dog

                 ;-------------------------------------
                 ;lines to display on the lcd
                 ;-------------------------------------
                 .cseg
000081 2a01
000082 2a2a
000083 7246
000084 7165
000085 6575
000086 636e
000087 2a79
000088 2a2a
000089 002a      line1_message: .db 1, "***Frequency****", 0  ; test string for line #1.
00008a 0002      line2_message: .db 2,"",0
00008b 4d03
00008c 4647
00008d 2a4d
00008e 2a2a
00008f 5a48
000090 2a2a
000091 312a
000092 4553
000093 0043      line3_message: .db 3, "MGFM***HZ***1SEC", 0  ; test string for line #3.



                 regular2:

                     ;load 1st line of prompt message into dbuff1
000094 e0f1          ldi  ZH, high(line1_message0<<1)        ;
000095 e3ec          ldi  ZL, low(line1_message0<<1)     ;
000096 d098          rcall load_msg                      ; load message into buffer(s).

                     ;LOAD 2ND LINE OF THE MESSAGE INTO DBUFF2
000097 e0f1          ldi ZH, high(line2_message0<<1)     ;
000098 e4ee          ldi ZL, low(line2_message0<<1)      ;load the table to stack
000099 d095          rcall load_msg                      ;load the frequency number into the buffer

                     ;load 3rd line of prompt message into dbuff3
00009a e0f1          ldi  ZH, high(line3_message0<<1)        ;
00009b e5e0          ldi  ZL, low(line3_message0<<1)     ;
00009c d092          rcall load_msg                      ; load message into buffer(s).
00009d d064          rcall update_lcd_dog

                 ;-------------------------------------
                 ;lines to display on the lcd
                 ;-------------------------------------
```

```
                .cseg
00009e 2a01
00009f 2a2a
0000a0 5246
0000a1 5145
0000a2 4555
0000a3 434e
0000a4 2a59
0000a5 2a2a
0000a6 002a     line1_message0:    .db 1, "***FREQUENCY****", 0  ; test string for line #1.
0000a7 0002     line2_message0:    .db 2,"",0
0000a8 4d03
0000a9 4647
0000aa 2a4d
0000ab 482a
0000ac 2a5a
0000ad 312a
0000ae 322f
0000af 4553
0000b0 0043     line3_message0:    .db 3, "MGFM**HZ**1/2SEC", 0  ; test string for line #3.


                ;***************************************************************************
                ;---------------------------- SUBROUTINES ----------------------------

                ;====================================
                .include "lcd_dog_asm_driver_m16A.inc"  ; LCD DOG init/update procedures.

                ;modified 11/26/12 KLS
                ; lcd_spi_transmit_data and lcd_spi_transmit_CMD handling of SPIF flag
                ;
                ;modifued 07/21/14 FST
                ; added BLOCK comments for adjusting power_ctrl & contrast_set parameters
                ;


               ;*********************************************************************
               ;   ATMega16A  2015 Version
               ;
               ;   This AVR-asm code module is usable as an include file for assembly
               ;   language and or mixed asm/C application programs. The code is freely
               ;   usable by any University of Stonybrook undergraduate students for any
               ;   and all not-for-profit system designs and or implementations.
               ;
               ;   This code is designed to be executed on an AVR ATMega micro-computer.
               ;   And may be readily adapted for compatibility with IAR/AVR compilers.
               ;   See the IAR assembler reference guide for more information by
               ;   clicking 'Help > AVR Assembly Reference Guide" on the above menus.
               ;
               ;
               ;*********************************************************************
               ;
               ;   This module contains procedures to initialize and update
               ;   DOG text based LCD display modules, including the EA DOG163M LCD
               ;   modules configured with three (3) 16 charactors display lines.
               ;
               ;   The display module hardware interface uses a 1-direction, write only
               ;   SPI interface. (See below for more information.)
               ;
               ;   The display module software interface uses three (3) 16-byte
               ;   data (RAM) based display buffers - One for each line of the display.
               ;   (See below for more information.)
               ;
               ;*********************************************************************
```

```
                ;
                ;   *** Port B Interface Definitions:
                ;
                ;  Port B             PB7   PB6   PB5   PB4   PB3   PB2   PB1   PB0
                ;  Port B alt names   SCK   MISO  MOSI  /SS   /RS    -     -     -
                ;  LCD Mod Signal     D6     -     D7   /CSB   -     -     -     -
                ;  LCD Mod Pin #      29     -     28    38    -     -     -     -
                ;
                ;  Notes: RS ==>  0 = command regs, 1 = data regs
                ;         /SS = active low SPI select signal
                ;
                ;******************


                ;***  DATA Segment  ************************************************
                .DSEG
000060          dsp_buff_1:   .byte 16
000070          dsp_buff_2:   .byte 16
000080          dsp_buff_3:   .byte 16



                ;***  CODE Segment Subroutines  ***********************************
                .CSEG



                ;************************
                ;NAME:        delay_30uS
                ;ASSUMES:     nothing
                ;RETURNS:     nothing
                ;MODIFIES:    R24, SREG
                ;CALLED BY:   init_dsp
                ;DESCRIPTION: This procedure will generate a fixed delay of just over
                ;             30 uS (assuming a 1 MHz clock).
                ;*****************************************************************
0000b1 0000     delay_30uS:   nop     ; fine tune delay
0000b2 0000                   nop
0000b3 938f                   push  r24
0000b4 e08f                   ldi   r24, 0x0f  ; load delay count.
0000b5 958a     d30_loop:     dec   r24        ; count down to
0000b6 f7f1                   brne  d30_loop   ; zero.
0000b7 918f                   pop   r24
0000b8 9508                   ret



                ;*********************
                ;NAME:        v_delay
                ;ASSUMES:     R22, R23 = initial count values defining how many
                ;             30uS delays will be called. This procedure can generate
                ;             short delays (r23 = small #) or much longer delays (where
                ;             R23 value is large).
                ;RETURNS:     nothing
                ;MODIFIES:    R22, R23, SREG
                ;CALLED BY:   init_dsp, plus...
                ;DESCRIPTION: This procedure will generate a variable delay for a fixed
                ;             period of time based the values pasted in R24 and R25.
                ;
                ;Sample Delays:
                ;
                ;             R22  R23  DelayTime
                ;             ---  ---  ---------
                ;              1    1   ~65.5 uS
                ;              0    1   ~14.2 mS
```

```
                ;              0    9   ~130 mS
                ;*********************************************************************
0000b9 dff7     v_delay:      rcall delay_30uS ; delay for ~30uS
0000ba 956a                   dec   r22      ; decrement inner loop value, and
0000bb f7e9                   brne  v_delay  ; loop until zero.
0000bc 957a                   dec   r23      ; decr outer loop count, and loop back
0000bd f7d9                   brne  v_delay  ; to inner loop delay until r23 zero.
0000be 9508                   ret



                ;************************
                ;NAME:        delay_40mS
                ;ASSUMES:     nothing
                ;RETURNS:     nothing
                ;MODIFIES:    R22,R23, SREG
                ;CALLED BY:   init_dsp, ????
                ;DESCRIPTION: This procedure will generate a fixed delay of just over
                ;             40 mS.
                ;*********************************************************************
0000bf e060     delay_40mS:   ldi   r22,0        ; load inner loop var
0000c0 e074                   ldi   r23,4        ; load outer loop var
0000c1 dff7                   rcall  v_delay     ; delay
0000c2 9508                   ret



                ;************************
                ;NAME:       init_spi_lcd
                ;ASSUMES:    IMPORTANT: PortB set as output (during program init)
                ;RETURNS:    nothing
                ;MODIFIES:   DDRB, SPCR
                ;CALLED BY:  init_dsp, update
                ;DESCRITION: init SPI port for command and data writes to LCD via SPI
                ;*********************************************************************
                init_spi_lcd:
0000c3 930f           push r16
0000c4 e50c           ldi r16,(1<<SPE) | (1<<MSTR) | (1<<CPOL) | (1<<CPHA)
0000c5 b90d           out SPCR,r16   ; Enable SPI, Master, fck/4,

                      ;kill any spurious data...
0000c6 b10e           in r16, SPSR   ; clear SPIF bit in SPSR
0000c7 b10f           in r16, SPDR   ;
0000c8 910f           pop r16        ; restore r16 value...
0000c9 9508           ret



                ;*******************************
                ;NAME:       lcd_spi_transmit_CMD
                ;ASSUMES:    r16 = byte for LCD.
                ;            SPI port is configured.
                ;RETURNS:    nothing
                ;MODIFIES:   R16, PortB, SPCR
                ;CALLED BY:  init_dsp, update
                ;DESCRITION: outputs a byte passed in r16 via SPI port. Waits for data
                ;            to be written by spi port before continuing.
                ;*********************************************************************
                lcd_spi_transmit_CMD:
0000ca 930f          push r16          ; save command, need r16.
0000cb 98c3          cbi   portB, 3    ; clr PB1 = RS = 0 = command.
0000cc 98c4          cbi   portB, 4    ; clr PB2 = /SS = selected.
0000cd b10e             in r16, SPSR   ; clear SPIF bit in SPSR.
0000ce b10f             in r16, SPDR   ;
```

```
0000cf 910f           pop r16           ; restore command
0000d0 b90f           out SPDR,r16      ; write data to SPI port.


                      ;Wait for transmission complete
                    wait_transmit:
0000d1 b10e           in r16, SPSR     ; read status reg
0000d2 ff07           sbrs r16, SPIF   ; if bit 7  = 0 wait
0000d3 cffd           rjmp wait_transmit
0000d4 b10f          in r16, SPDR       ;added by Ken to clear SPIF
0000d5 9ac4           sbi   portB, 4   ; set PB2 = /SS = deselected
0000d6 9508           ret




                    ;*********************************
                    ;NAME:       lcd_spi_transmit_DATA
                    ;ASSUMES:    r16 = byte to transmit to LCD.
                    ;            SPI port is configured.
                    ;RETURNS:    nothing
                    ;MODIFIES:   R16, SPCR
                    ;CALLED BY:  init_dsp, update
                    ;DESCRITION: outputs a byte passed in r16 via SPI port. Waits for
                    ;            data to be written by spi port before continuing.
                    ;*****************************************************************
                    lcd_spi_transmit_DATA:
0000d7 930f           push r16          ; save command, need r16.
0000d8 9ac3           sbi   portB, 3    ; clr PB1 = RS = 1 = data.
0000d9 98c4           cbi   portB, 4    ; clr PB2 = /SS = selected.
0000da b10e              in r16, SPSR   ; clear SPIF bit in SPSR.
0000db b10f              in r16, SPDR   ;
0000dc 910f           pop r16           ; restore command.
0000dd b90f           out SPDR,r16      ; write data to SPI port.


                      ;Wait for transmission complete
                    wait_transmit1:
0000de b10e           in r16, SPSR     ; read status reg
0000df ff07           sbrs r16, SPIF   ; if bit 7  = 0 wait
0000e0 cffd           rjmp wait_transmit1
0000e1 b10f          in r16, SPDR       ;clear SPIF (because it follows in r16,SPSR)
0000e2 9ac4           sbi   portB, 4   ; set PB2 = /SS = deselected
0000e3 9508           ret



                    ;************************
                    ;NAME:       init_lcd_dog
                    ;ASSUMES:    nothing
                    ;RETURNS:    nothing
                    ;MODIFIES:   R16, R17
                    ;CALLED BY:  main application
                    ;DESCRITION: inits DOG module LCD display for SPI (serial) operation.
                    ;NOTE:  Can be used as is with MCU clock speeds of 4MHz or less.
                    ;*****************************************************************
                    ; public __version_1 void init_dsp(void)
                    init_lcd_dog:
0000e4 dfde              rcall init_spi_lcd   ; init SPI port for DOG LCD.


                    start_dly_40ms:
0000e5 dfd9              rcall  delay_40mS    ; startup delay.


                    func_set1:
```

```
0000e6 e309             ldi    r16,0x39      ; send fuction set #1
0000e7 dfe2             rcall  lcd_spi_transmit_CMD   ;
0000e8 dfc8             rcall  delay_30uS    ; delay for command to be processed


                func_set2:
0000e9 e309             ldi    r16,0x39      ; send fuction set #2
0000ea dfdf             rcall  lcd_spi_transmit_CMD
0000eb dfc5             rcall  delay_30uS    ; delay for command to be processed


                bias_set:
0000ec e10e             ldi    r16,0x1E      ; set bias value.
0000ed dfdc             rcall  lcd_spi_transmit_CMD
0000ee dfc2             rcall  delay_30uS    ;


                                              ; ==========================================
                                              ; === CALIBRATION PARAMETER - USER ADJUSTABLE
                power_ctrl:                   ; === (CAUTION... VERY DELICATE ADJUSTMENT)
0000ef e500             ldi    r16,0x50       ; === 5V ~= 0x50 nominal;      Adjust by 1 ONLY
0000f0 dfd9             rcall  lcd_spi_transmit_CMD ; === 3.3V ~= 0x55 nominal     and think hex!
0000f1 dfbf             rcall  delay_30uS     ;   Hex = 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f
                                              ; ==========================================

                follower_ctrl:
0000f2 e60c             ldi    r16,0x6C      ; follower mode on...
0000f3 dfd6             rcall  lcd_spi_transmit_CMD
0000f4 dfca             rcall  delay_40mS    ;
                                              ; ==========================================
                                              ; === CALIBRATION PARAMETER - USER ADJUSTABLE
                contrast_set:                 ; === LCD CONTRAST SETTING ADJUSTMENT
0000f5 e707             ldi    r16,0x77       ; ===
0000f6 dfd3             rcall  lcd_spi_transmit_CMD ; === Delicate: increases for 3.3V vs 5V
0000f7 dfb9             rcall  delay_30uS     ; ==========================================

                display_on:
0000f8 e00c             ldi    r16,0x0c      ; display on, cursor off, blink off
0000f9 dfd0             rcall  lcd_spi_transmit_CMD
0000fa dfb6             rcall  delay_30uS    ;


                clr_display:
0000fb e001             ldi    r16,0x01      ; clear display, cursor home
0000fc dfcd             rcall  lcd_spi_transmit_CMD


0000fd dfb3             rcall  delay_30uS    ;


                entry_mode:
0000fe e006             ldi    r16,0x06      ; clear display, cursor home
0000ff dfca             rcall  lcd_spi_transmit_CMD;
000100 dfb0             rcall  delay_30uS    ;
000101 9508             ret




                ;*************************
                ;NAME:       update_lcd_dog
                ;ASSUMES:    display buffers loaded with display data
                ;RETURNS:    nothing
                ;MODIFIES:   R16,R20,R30,R31,SREG
                ;
                ;DESCRITION: Updates the LCD display lines 1, 2, and 3, using the
```

```
                ;  contents of dsp_buff_1, dsp_buff_2, and dsp_buff_3, respectively.
                ;*****************************************************************
                ; public __version_1 void update_dsp_dog (void)
                update_lcd_dog:
000102 dfc0             rcall init_spi_lcd    ; init SPI port for LCD.
000103 e140             ldi   r20,16          ; init 'chars per line' counter.
000104 934f             push  r20             ; save for later used.


                        ;send line 1 to the LCD module.
                wr_line1:
000105 e0f0             ldi   ZH, high (dsp_buff_1)  ; init ptr to line 1 display buffer.
000106 e6e0             ldi   ZL, low (dsp_buff_1)   ;
                snd_ddram_addr:
000107 e800             ldi   r16,0x80             ; init DDRAM addr-ctr
000108 dfc1             rcall lcd_spi_transmit_CMD ;
000109 dfa7             rcall delay_30uS
                snd_buff_1:
00010a 9101             ld    r16, Z+
00010b dfcb             rcall lcd_spi_transmit_DATA
00010c dfa4             rcall delay_30uS
00010d 954a             dec   r20
00010e f7d9             brne  snd_buff_1


                        ;send line 2 to the LCD module.
                init_for_buff_2:
00010f 914f             pop   r20     ; reload r20 = chars per line counter
000110 934f             push  r20     ; save for line 3
                wr_line2:
000111 e0f0             ldi   ZH, high (dsp_buff_2)  ; init ptr to line 2 display buffer.
000112 e7e0             ldi   ZL, low (dsp_buff_2)
                snd_ddram_addr2:
000113 e900             ldi   r16,0x90               ; init DDRAM addr-ctr
000114 dfb5             rcall lcd_spi_transmit_CMD           ;
000115 df9b             rcall delay_30uS
                snd_buff_2:
000116 9101             ld    r16, Z+
000117 dfbf             rcall lcd_spi_transmit_DATA
000118 df98             rcall delay_30uS
000119 954a             dec   r20
00011a f7d9             brne  snd_buff_2


                        ;send line 3 to the LCD module.
                init_for_buff_3:
00011b 914f             pop   r20     ; reload r20 = chars per line counter
                wr_line3:
00011c e0f0             ldi   ZH, high (dsp_buff_3)  ; init ptr to line 2 display buffer.
00011d e8e0             ldi   ZL, low (dsp_buff_3)
                snd_ddram_addr3:
00011e ea00             ldi   r16,0xA0               ; init DDRAM addr-ctr
00011f dfaa             rcall lcd_spi_transmit_CMD           ;
000120 df90             rcall delay_30uS


                snd_buff_3:
000121 9101             ld    r16, Z+
000122 dfb4             rcall lcd_spi_transmit_DATA
000123 df8d             rcall delay_30uS
000124 954a             dec   r20
000125 f7d9             brne  snd_buff_3
000126 9508             ret


                ;****** End Of LCD DOG Include Module *********************************
```

```
                ;====================================

                ;************************
                ;NAME:      clr_dsp_buffs
                ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3 with blanks (0x20)
                ;ASSUMES:   Three CONTIGUOUS 16-byte dram based buffers named
                ;           dsp_buff_1, dsp_buff_2, dsp_buff_3.
                ;RETURNS:   nothing.
                ;MODIFIES:  r25,r26, Z-ptr
                ;CALLS:     none
                ;CALLED BY: main application and diagnostics
                ;*********************************************************************
                clr_dsp_buffs:
000127 e390          ldi R25, 48                    ; load total length of both buffer.
000128 e2a0          ldi R26, ' '                   ; load blank/space into R26.
000129 e0f0          ldi ZH, high (dsp_buff_1)      ; Load ZH and ZL as a pointer to 1st
00012a e6e0          ldi ZL, low (dsp_buff_1)       ; byte of buffer for line 1.

                    ;set DDRAM address to 1st position of first line.
                store_bytes:
00012b 93a1          st  Z+, R26                    ; store ' ' into 1st/next buffer byte and
                                                    ; auto inc ptr to next location.
00012c 959a          dec  R25                       ;
00012d f7e9          brne store_bytes               ; cont until r25=0, all bytes written.
00012e 9508          ret


                ;******************
                ;NAME:      load_msg
                ;FUNCTION:  Loads a predefined string msg into a specified diplay
                ;           buffer.
                ;ASSUMES:   Z = offset of message to be loaded. Msg format is
                ;           defined below.
                ;RETURNS:   nothing.
                ;MODIFIES:  r16, Y, Z
                ;CALLS:     nothing
                ;CALLED BY:
                ;*********************************************************************
                ; Message structure:
                ;   label:  .db <buff num>, <text string/message>, <end of string>
                ;
                ; Message examples (also see Messages at the end of this file/module):
                ;   msg_1: .db 1,"First Message ", 0   ; loads msg into buff 1, eom=0
                ;   msg_2: .db 1,"Another message ", 0 ; loads msg into buff 1, eom=0
                ;
                ; Notes:
                ;   a) The 1st number indicates which buffer to load (either 1, 2, or 3).
                ;   b) The last number (zero) is an 'end of string' indicator.
                ;   c) Y = ptr to disp_buffer
                ;      Z = ptr to message (passed to subroutine)
                ;*********************************************************************
                load_msg:
00012f e0d0          ldi YH, high (dsp_buff_1)      ; Load YH and YL as a pointer to 1st
000130 e6c0          ldi YL, low (dsp_buff_1)       ; byte of dsp_buff_1 (Note - assuming
                                                    ; (dsp_buff_1 for now).
000131 9105          lpm R16, Z+                    ; get dsply buff number (1st byte of msg).
000132 3001          cpi r16, 1                         ; if equal to '1', ptr already setup.
000133 f031          breq get_msg_byte              ; jump and start message load.
000134 9660          adiw YH:YL, 16                     ; else set ptr to dsp buff 2.
000135 e010         ldi r17, $00
000136 e0b6         ldi r27, 6
000137 3002          cpi r16, 2                         ; if equal to '2', ptr now setup.
000138 f031          breq digit_load                ; jump and start message load.
000139 9660          adiw YH:YL, 16                     ; else set ptr to dsp buff 3.

                get_msg_byte:
```

```
00013a 9105          lpm R16, Z+                    ; get next byte of msg and see if '0'.
00013b 3000          cpi R16, 0                         ; if equal to '0', end of message reached.
00013c f0a1          breq msg_loaded                ; jump and stop message loading operation.
00013d 9309          st Y+, R16                         ; else, store next byte of msg in buffer.
00013e cffb          rjmp get_msg_byte              ; jump back and continue...

                ;_____________________________________________
                ; digital_load will only be accessed when displaying line 2,
                ; since the frequency to be displayed in line 2 is constantly
                ; changing for different waveform, the line 2 has to be adjusted
                ; according.
                ; r17, will inc until 6, to display 6 empty spaces
                ; r4 will contain the first digit of the frequency
                ; r3 will contain the second digit of the frequency
                ; r2 will contain the third digit of the frequency
                ; r1 will contian the fouth digit of the frequency
                ;get_dis_freq subroutine will just transfer each value stored in
                ; r25 to y pointer
                ;_____________________________________________
                digit_load:
00013f 9513        inc r17
000140 e290        ldi r25, $20                ;load empty spaces for 6 places
000141 d00d        rcall get_dis_freq          ;display
000142 131b        cpse r17, r27               ;check if 6 places typed
000143 cffb        rjmp digit_load             ;repeat until 6 places
000144 2d95        mov r25, r5                 ;load the first number in freq
000145 d009        rcall get_dis_freq          ;display
000146 2d94        mov r25, r4                 ;load the first number in freq
000147 d007        rcall get_dis_freq          ;display
000148 2d93        mov r25, r3                 ;load the second number in freq
000149 d005        rcall get_dis_freq          ;display
00014a 2d92        mov r25, r2                 ;load the third number in freq
00014b d003        rcall get_dis_freq          ;display
00014c 2d91        mov r25, r1                 ;load the fourth number in freq
00014d d001        rcall get_dis_freq          ;display
00014e c002        rjmp msg_loaded             ;go to the next line of the lcd

                get_dis_freq:
                // ldi r16, $00                ;clear for later use
                // add ZL, r25                 ;add low byte
                // adc ZH, r16                 ;add in the carry
                // lpm r25, Z+                 ;load bid pattern from table into r25
00014f 9399        st Y+, r25                  ;display the selected frquency
000150 9508        ret


                msg_loaded:
000151 9508          ret

                ;-----------------------------------------------
                ;unpacks the values store in r8 and r9 to r1- r4
                ; r4 containe the left most number ie the thousanth
                ;digit and r1 the right most number
                ;-----------------------------------------------
                unpack:
000152 930f        push r16                    ;store the value currently in r16
000153 d017        rcall bin2BCD16             ;convert the values from binary to bcd
                   //sub r13, r9                   ;to fix slight error in frequency conversion
000154 2c2d        mov r2, r13                 ;make a copy of r13 in r2
000155 2c4e        mov r4, r14                 ;make a copy of r14 in r4
000156 2c6f        mov r6, r15                 ;make a copy of r15 in r6
000157 e00f        ldi r16, $0f                ;use and function to
000158 22d0        and r13, r16                ;mask the upper nibble of r8
000159 2c1d        mov r1, r13                 ;move lower nibble to r1
00015a 22e0        and r14, r16                ;mask upper nibble of r9
00015b 2c3e        mov r3, r14                 ;move lower nible to r3
```

```
00015c 22f0         and r15, r16                    ;mask the upper nibble of r8
00015d 2c5f         mov r5, r15                     ;move lower nibble to r1
00015e 9500         com r16                         ;load with f0 to mask lower nibble
00015f 2220         and r2, r16                     ;mask lower nibble of r8
000160 9422         swap r2                         ;switch upper and lower nibble
000161 2240         and r4, r16                     ;mask lower nibble of r9
000162 9442         swap r4                         ;switch upper and lower nibble
                    //and r6, r16                   ;mask lower nibble of r9
                    //swap r6                       ;switch upper and lower nibble
000163 e300         ldi r16, $30
000164 0e10         add r1, r16
000165 0e20         add r2, r16
000166 0e30         add r3, r16
000167 0e40         add r4, r16
000168 0e50         add r5, r16                     ;converting the bcd's to ascii
000169 910f         pop r16                         ;retrive the value previosly stored
00016a 9508         ret
```

```
                    ;***************************************************************************
                    ;*
                    ;* "bin2BCD16" - 16-bit Binary to BCD conversion
                    ;*
                    ;* This subroutine converts a 16-bit number (fbinH:fbinL) to a 5-digit
                    ;* packed BCD number represented by 3 bytes (tBCD2:tBCD1:tBCD0).
                    ;* MSD of the 5-digit number is placed in the lowermost nibble of tBCD2.
                    ;*
                    ;* Number of words :25
                    ;* Number of cycles    :751/768 (Min/Max)
                    ;* Low registers used  :3 (tBCD0,tBCD1,tBCD2)
                    ;* High registers used  :4(fbinL,fbinH,cnt16a,tmp16a)
                    ;* Pointers used   :Z
                    ;*
                    ;***************************************************************************
                    //.include "..\8515def.inc"
                    ;***** Subroutine Register Variables


                    .equ   AtBCD0  =13          ;address of tBCD0
                    .equ   AtBCD2  =15          ;address of tBCD1

                    .def   tBCD0   =r13         ;BCD value digits 1 and 0
                    .def   tBCD1   =r14         ;BCD value digits 3 and 2
                    .def   tBCD2   =r15         ;BCD value digit 4
                    .def   fbinL   =r16         ;binary value Low byte
                    .def   fbinH   =r17         ;binary value High byte
                    .def   cnt16a  =r18         ;loop counter
                    .def   tmp16a  =r19         ;temporary value

                    ;***** Code

                    bin2BCD16:
00016b 1889            sub r8, r9
00016c 2d19            mov fbinH, r9            ;copy the values of edge counter to fbin
00016d 2d08            mov fbinL, r8
00016e e120            ldi cnt16a,16            ;Init loop counter
00016f 24ff            clr tBCD2                ;clear result (3 bytes)
000170 24ee            clr tBCD1
000171 24dd            clr tBCD0
000172 27ff            clr ZH                   ;clear ZH (not needed for AT90Sxx0x)
```

```
                bBCDx_1:
000173 0f00         lsl fbinL               ;shift input value
000174 1f11         rol fbinH               ;through all bytes
000175 1cdd         rol tBCD0               ;
000176 1cee         rol tBCD1
000177 1cff         rol tBCD2
000178 952a         dec cnt16a              ;decrement loop counter
000179 f409         brne bBCDx_2            ;if counter not zero
00017a 9508         ret                     ;   return

                bBCDx_2:
00017b e1e0         ldi r30,AtBCD2+1        ;Z points to result MSB + 1
                bBCDx_3:
00017c 9132         ld  tmp16a,-Z           ;get (Z) with pre-decrement
00017d 5f3d         subi tmp16a,-$03        ;add 0x03
00017e fd33         sbrc tmp16a,3           ;if bit 3 not clear
00017f 8330         st Z,tmp16a             ;store back
000180 8130         ld  tmp16a,Z            ;get (Z)
000181 5d30         subi tmp16a,-$30        ;add 0x30
000182 fd37         sbrc tmp16a,7           ;if bit 7 not clear
000183 8330         st  Z,tmp16a            ;store back
000184 30ed         cpi ZL,AtBCD0           ;done all three?
000185 f7b1         brne bBCDx_3            ;loop again if not
000186 cfec         rjmp bBCDx_1

                ;-----------------------------------------------
                ;This subroutine is placed here, if it was required during
                ; the lab. Its not called anywhere in the code.
                ;delays for 10ms
                ;r20 set to 100
                ;r21 set to 33
                ; combined delay will yield 9999 clock cycles
                ;-----------------------------------------------
                delay:
000187 e644         ldi r20,100
                    outer:
000188 e251             ldi r21, 33
                        inner:
000189 955a                 dec r21
00018a f7f1                 brne inner
00018b 954a                 dec r20
00018c f7d9                 brne outer
00018d 9508         ret


RESOURCE USE INFORMATION
------------------------

Notice:
The register and instruction counts are symbol table hit counts,
and hence implicitly used resources are not counted, eg, the
'lpm' instruction without operands implicitly uses r0 and z,
none of which are counted.

x,y,z are separate entities in the symbol table and are
counted separately from r26..r31 here.

.dseg memory usage only counts static data declared with .byte

"ATmega16" register use summary:
r0 :   0 r1 :   3 r2 :   5 r3 :   3 r4 :   5 r5 :   3 r6 :   1 r7 :   0
r8 :   6 r9 :   6 r10:   0 r11:   0 r12:   0 r13:   5 r14:   5 r15:   5
r16: 115 r17:  10 r18:   2 r19:   8 r20:  10 r21:   2 r22:   2 r23:   2
r24:   4 r25:  15 r26:   2 r27:   8 r28:   3 r29:   3 r30:  12 r31:  11
x  :   0 y  :   2 z  :  10
Registers used: 29 out of 35 (82.9%)
```

```
"ATmega16" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   0 add   :   5 adiw  :   2 and   :   6
andi  :   2 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   1 break :   0 breq  :   6 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :  13 brpl  :   0 brsh  :   0 brtc  :   2 brts  :   2 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   3 cbr   :   0
clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   4 cls   :   0
clt   :   1 clv   :   0 clz   :   0 com   :   2 cp    :   2 cpc   :   0
cpi   :   4 cpse  :   1 dec   :  10 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :  17 inc   :   5 jmp   :   3
ld    :   5 ldd   :   0 ldi   :  68 lds   :   0 lpm   :   2 lsl   :   1
lsr   :   0 mov   :  19 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   2 or    :   0 ori   :   0 out   :  21 pop   :  11
push  :  11 rcall :  59 ret   :  17 reti  :   2 rjmp  :   9 rol   :   4
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   5 sbic  :   0 sbis  :   0
sbiw  :   0 sbr   :   0 sbrc  :   4 sbrs  :   4 sec   :   0 seh   :   0
sei   :   1 sen   :   0 ser   :   0 ses   :   0 set   :   1 sev   :   0
sez   :   0 sleep :   0 spm   :   0 st    :   5 std   :   0 sts   :   0
sub   :   1 subi  :   2 swap  :   2 tst   :   0 wdr   :   0
Instructions used: 43 out of 113 (38.1%)

"ATmega16" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x00031c    700     76    776   16384   4.7%
[.dseg] 0x000060 0x000090      0     48     48    1024   4.7%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors, 0 warnings
```