```
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\frequency_meter_3\frequency_meter_3\frequency_meter_3.asm Tue Oct 28 19:54:18 2014

C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 8
\frequency
_meter_3\frequency_meter_3\frequency_meter_3.asm(35): Including file 'C:\Program Files (x86)\Atmel
\Atmel Tool
chain\AVR Assembler\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\frequency
_meter_3\frequency_meter_3\frequency_meter_3.asm(250): Including file 'C:\Users\radra_000\Box Sync
\college
sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 8\frequency_meter_3\frequency_meter_3
\lcd_dog_
asm_driver_m16A.inc'


                 * frequency_meter_3.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. This is similar to the two other program.
                 ; only the compare register is changed from the previous code

                 ;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 c01c         rjmp 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 0x0E
00000e 940c 0042    jmp isr_tc0_display

                ;*********************TIMER INITIALIZATION*************************
                ;_____________________________________________________________________
```

```
                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 e10e         ldi r16, $1E                        ;load the counter with 15625
000018 bd09         out OCR1BH, r16                     ; so that we will get 1s
000019 e804         ldi r16, $84
00001a bd08         out OCR1BL, r16                     ;
00001b 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


00001c 9508         ret
                ;____________________________________________________________________


                reset:
00001d e50f         ldi r16, low(ramend)
00001e bf0d         out spl, r16
00001f e004         ldi r16, high(ramend)
000020 bf0e         out sph, r16                        ;initialize stack pointer


000021 e000         ldi r16, $00
000022 bb01         out ddrd, r16                       ;set up the port b as inputs to read the
                                                        ; pbsw values
                                                        ; and nand gate input on pd(2)

000023 ef0f         ldi r16, 0xff                       ; set portB = output.
000024 bb07          out DDRB, r16                      ; for lcd display
000025 9ac4          sbi portB, 4                       ; set /SS of DOG LCD = 1 (Deselected)1

000026 e001         ldi r16, 1                          ; set DDRC for all in but PC0
000027 bb04         out DDRC, r16
000028 9aa8         sbi PortC, 0                        ; turn off sounder

000029 e400         ldi r16,0b01000000                  ; set up port a to
00002a bb0a         out ddra, r16                       ;read the frequency input on pa7 and output
                                                        ; pulse on pa6

00002b d08c         rcall init_lcd_dog                  ; init display, using SPI serial interface


00002c dfe3         rcall start_tc1                     ;init the timer counter 1
                        /*
                    ldi r16, 28
                    mov r9, r16
                    ldi r16, 60
                    mov r8, r16*/                       ;for testing in simulation
```

```
                main:
                   //start timer code
00002d e003        ldi r16, 0<<CS12|1<<CS11|1<<CS10    ; load 64 PRESCaLE TCCR0 value.
00002e bd0e        out TCCR1B, r16                     ; and start timer
00002f d004        rcall frequency_meter_3             ; load the timer and count the frequency
000030 94e8        clt                                 ;clear the t flag
000031 d0f4        rcall unpack                        ;when timer is done, unpack the edge counts
000032 d016        rcall message_dsp_loop              ;after its unpacked, display
000033 cff9        rjmp main
                /*************************************************************************
                    subroutine: 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:
000034 e000        ldi r16, $00
000035 2e80        mov r8, r16
000036 2e90        mov r9, r16
000037 b399        in r25, pinA                ;and positive edge counter
                   check_edge:
000038 f3fe            brts check_edge
000039 b309            in r16, pina            ;take in the current wave signal logic
00003a 1709            cp r16,r25          ;and compare to previous logic recorded,
00003b f3e1            breq check_edge         ;if it is the same then skip to tweak delay
00003c 2f90            mov r25, r16
00003d 9483            inc r8              ;and then increment the counter
00003e f7c9            brne check_edge         ;if it didnt over count then go to tweak delay
00003f 9493            inc r9              ;if so then increment the second register
000040 f7be            brtc check_edge
000041 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:
                ;codes will be added here after frequency subroutine is added.
000042 930f            push r16
000043 9468            set                                 ;set the tflag
000044 e000            ldi r16, $00
000045 bd0d            out TCNT1H, r16                     ;
000046 bd0c            out TCNT1L, r16                     ;reset the timer counter
000047 910f            pop r16
000048 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:
000049 d0b1        rcall clr_dsp_buffs                 ; clear all three buffer lines
00004a d08b        rcall update_lcd_dog                    ;
00004b e30a        ldi r16, $3A                            ; compare weather the frequency value
00004c 1690        cp r9, r16                          ; is less than 15k
00004d f0d0        BRLO regular                            ;if less then branch off and display the
                                                       ; calculated value
```

```
                                                ;if not continue.

                overflow:

                   ;load 1st line of prompt message into dbuff1
00004e e0f0        ldi  ZH, high(line1_message<<1)     ;
00004f eaea        ldi  ZL, low(line1_message<<1)      ;
000050 d0b2        rcall load_msg                      ; load message into buffer(s).

                   /*second line will be left blank when overflows
                   ;LOAD 2ND LINE OF THE MESSAGE INTO DBUFF2
                   ldi ZH, high(line2_message<<1)      ;
                   ldi ZL, low(line2_message<<1)       ;load the table to stack
                   rcall load_msg                      ;load the frequency number into the buffer
                   */

                   ;load 3rd line of prompt message into dbuff3
000051 e0f0        ldi  ZH, high(line3_message<<1)     ;
000052 ebee        ldi  ZL, low(line3_message<<1)      ;
000053 d0af        rcall load_msg                      ; load message into buffer(s).
000054 d081        rcall update_lcd_dog

                ;---------------------------------------
                ;lines to display on the lcd
                ;---------------------------------------
                .cseg

000055 2a01
000056 2a2a
000057 6f2a
000058 6576
000059 6672
00005a 6f6c
00005b 2a77
00005c 2a2a
00005d 002a     line1_message: .db 1, "****overflow****", 0  ; test string for line #1.
00005e 0002     line2_message: .db 2,"",0
00005f 7e03
000060 7e7e
000061 7e7e
000062 313e
000063 6b35
000064 7a68
000065 7e7e
000066 7e7e
000067 007e     line3_message: .db 3, "~~~~~>15khz~~~~~", 0  ; test string for line #3.




                regular:

                   ;load 1st line of prompt message into dbuff1
000068 e0f0        ldi  ZH, high(line1_message0<<1)        ;
000069 eee4        ldi  ZL, low(line1_message0<<1)     ;
00006a d098        rcall load_msg                      ; load message into buffer(s).

                   ;LOAD 2ND LINE OF THE MESSAGE INTO DBUFF2
00006b e0f0        ldi ZH, high(line2_message0<<1)     ;
00006c efe6        ldi ZL, low(line2_message0<<1)      ;load the table to stack
00006d d095        rcall load_msg                      ;load the frequency number into the buffer

                   ;load 3rd line of prompt message into dbuff3
00006e e0f0        ldi  ZH, high(line3_message0<<1)        ;
00006f efe8        ldi  ZL, low(line3_message0<<1)     ;
000070 d092        rcall load_msg                      ; load message into buffer(s).
```

```
000071 d064         rcall update_lcd_dog

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

                .cseg
000072 2a01
000073 2a2a
000074 5246
000075 5145
000076 4555
000077 434e
000078 2a59
000079 2a2a
00007a 002a     line1_message0:    .db 1, "***FREQUENCY****", 0  ; test string for line #1.
00007b 0002     line2_message0:    .db 2,"",0
00007c 4603
00007d 334d
00007e 2a2a
00007f 482a
000080 2a5a
000081 302a
000082 352e
000083 4553
000084 0043     line3_message0:    .db 3, "FM3***HZ**0.5SEC", 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                          PRINT IN LANDSCAPE
                ;
                ;   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).
                ;*********************************************************************
000085 0000     delay_30uS:   nop     ; fine tune delay
000086 0000                   nop
000087 938f                   push  r24
000088 e08f                   ldi   r24, 0x0f  ; load delay count.
000089 958a     d30_loop:     dec   r24        ; count down to
00008a f7f1                   brne  d30_loop   ; zero.
00008b 918f                   pop   r24
00008c 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
                ;*********************************************************************
00008d dff7     v_delay:      rcall delay_30uS ; delay for ~30uS
00008e 956a                   dec   r22      ; decrement inner loop value, and
00008f f7e9                   brne  v_delay  ; loop until zero.
000090 957a                   dec   r23      ; decr outer loop count, and loop back
000091 f7d9                   brne  v_delay  ; to inner loop delay until r23 zero.
000092 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.
                ;*********************************************************************
000093 e060     delay_40mS:   ldi   r22,0        ; load inner loop var
000094 e074                   ldi   r23,4        ; load outer loop var
000095 dff7                   rcall  v_delay     ; delay
000096 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
                ;DESCRIPTION: init SPI port for command and data writes to LCD via SPI
                ;*********************************************************************
                init_spi_lcd:
000097 930f           push r16
000098 e50c           ldi r16,(1<<SPE) | (1<<MSTR) | (1<<CPOL) | (1<<CPHA)
000099 b90d           out SPCR,r16   ; Enable SPI, Master, fck/4,

                      ;kill any spurious data...
00009a b10e           in r16, SPSR   ; clear SPIF bit in SPSR
00009b b10f           in r16, SPDR   ;
00009c 910f           pop r16        ; restore r16 value...
00009d 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
                ;DESCRIPTION: outputs a byte passed in r16 via SPI port. Waits for data
                ;            to be written by spi port before continuing.
                ;*********************************************************************
                lcd_spi_transmit_CMD:
```

```
00009e 930f           push r16          ; save command, need r16.
00009f 98c3           cbi   portB, 3    ; clr PB1 = RS = 0 = command.
0000a0 98c4           cbi   portB, 4    ; clr PB2 = /SS = selected.
0000a1 b10e              in r16, SPSR   ; clear SPIF bit in SPSR.
0000a2 b10f              in r16, SPDR   ;
0000a3 910f           pop r16           ; restore command
0000a4 b90f           out SPDR,r16      ; write data to SPI port.


                      ;Wait for transmission complete
                    wait_transmit:
0000a5 b10e           in r16, SPSR     ; read status reg
0000a6 ff07           sbrs r16, SPIF   ; if bit 7  = 0 wait
0000a7 cffd           rjmp wait_transmit
0000a8 b10f          in r16, SPDR       ;added by Ken to clear SPIF
0000a9 9ac4           sbi   portB, 4   ; set PB2 = /SS = deselected
0000aa 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:
0000ab 930f           push r16          ; save command, need r16.
0000ac 9ac3           sbi   portB, 3    ; clr PB1 = RS = 1 = data.
0000ad 98c4           cbi   portB, 4    ; clr PB2 = /SS = selected.
0000ae b10e              in r16, SPSR   ; clear SPIF bit in SPSR.
0000af b10f              in r16, SPDR   ;
0000b0 910f           pop r16           ; restore command.
0000b1 b90f           out SPDR,r16      ; write data to SPI port.


                      ;Wait for transmission complete
                    wait_transmit1:
0000b2 b10e           in r16, SPSR     ; read status reg
0000b3 ff07           sbrs r16, SPIF   ; if bit 7  = 0 wait
0000b4 cffd           rjmp wait_transmit1
0000b5 b10f          in r16, SPDR       ;clear SPIF (because it follows in r16,SPSR)
0000b6 9ac4           sbi   portB, 4   ; set PB2 = /SS = deselected
0000b7 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:
0000b8 dfde              rcall init_spi_lcd   ; init SPI port for DOG LCD.
```

```
                start_dly_40ms:
0000b9 dfd9             rcall  delay_40mS    ; startup delay.


                func_set1:
0000ba e309             ldi    r16,0x39      ; send fuction set #1
0000bb dfe2             rcall  lcd_spi_transmit_CMD   ;
0000bc dfc8             rcall  delay_30uS    ; delay for command to be processed


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


                bias_set:
0000c0 e10e             ldi    r16,0x1E      ; set bias value.
0000c1 dfdc             rcall  lcd_spi_transmit_CMD
0000c2 dfc2             rcall  delay_30uS    ;


                                               ; ==========================================
                                               ; === CALIBRATION PARAMETER - USER ADJUSTABLE
                power_ctrl:                    ; === (CAUTION... VERY DELICATE ADJUSTMENT)
0000c3 e500             ldi    r16,0x50        ; === 5V ~= 0x50 nominal;      Adjust by 1 ONLY
0000c4 dfd9             rcall  lcd_spi_transmit_CMD ; === 3.3V ~= 0x55 nominal     and think hex!
0000c5 dfbf             rcall  delay_30uS      ;   Hex = 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f
                                               ; ==========================================

                follower_ctrl:
0000c6 e60c             ldi    r16,0x6C      ; follower mode on...
0000c7 dfd6             rcall  lcd_spi_transmit_CMD
0000c8 dfca             rcall  delay_40mS    ;
                                               ; ==========================================
                                               ; === CALIBRATION PARAMETER - USER ADJUSTABLE
                contrast_set:                  ; === LCD CONTRAST SETTING ADJUSTMENT
0000c9 e707             ldi    r16,0x77        ; ===
0000ca dfd3             rcall  lcd_spi_transmit_CMD ; === Delicate: increases for 3.3V vs 5V
0000cb dfb9             rcall  delay_30uS      ; ==========================================

                display_on:
0000cc e00c             ldi    r16,0x0c      ; display on, cursor off, blink off
0000cd dfd0             rcall  lcd_spi_transmit_CMD
0000ce dfb6             rcall  delay_30uS    ;


                clr_display:
0000cf e001             ldi    r16,0x01      ; clear display, cursor home
0000d0 dfcd             rcall  lcd_spi_transmit_CMD


0000d1 dfb3             rcall  delay_30uS    ;


                entry_mode:
0000d2 e006             ldi    r16,0x06      ; clear display, cursor home
0000d3 dfca             rcall  lcd_spi_transmit_CMD;
0000d4 dfb0             rcall  delay_30uS    ;
0000d5 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:
0000d6 dfc0             rcall init_spi_lcd    ; init SPI port for LCD.
0000d7 e140             ldi   r20,16          ; init 'chars per line' counter.
0000d8 934f             push  r20             ; save for later used.


                        ;send line 1 to the LCD module.
                wr_line1:
0000d9 e0f0             ldi   ZH, high (dsp_buff_1)  ; init ptr to line 1 display buffer.
0000da e6e0             ldi   ZL, low (dsp_buff_1)   ;
                snd_ddram_addr:
0000db e800             ldi   r16,0x80             ; init DDRAM addr-ctr
0000dc dfc1             rcall lcd_spi_transmit_CMD ;
0000dd dfa7             rcall delay_30uS
                snd_buff_1:
0000de 9101             ld    r16, Z+
0000df dfcb             rcall lcd_spi_transmit_DATA
0000e0 dfa4             rcall delay_30uS
0000e1 954a             dec   r20
0000e2 f7d9             brne  snd_buff_1


                        ;send line 2 to the LCD module.
                init_for_buff_2:
0000e3 914f             pop   r20     ; reload r20 = chars per line counter
0000e4 934f             push  r20     ; save for line 3
                wr_line2:
0000e5 e0f0             ldi   ZH, high (dsp_buff_2)  ; init ptr to line 2 display buffer.
0000e6 e7e0             ldi   ZL, low (dsp_buff_2)
                snd_ddram_addr2:
0000e7 e900             ldi   r16,0x90               ; init DDRAM addr-ctr
0000e8 dfb5             rcall lcd_spi_transmit_CMD          ;
0000e9 df9b             rcall delay_30uS
                snd_buff_2:
0000ea 9101             ld    r16, Z+
0000eb dfbf             rcall lcd_spi_transmit_DATA
0000ec df98             rcall delay_30uS
0000ed 954a             dec   r20
0000ee f7d9             brne  snd_buff_2


                        ;send line 3 to the LCD module.
                init_for_buff_3:
0000ef 914f             pop   r20     ; reload r20 = chars per line counter
                wr_line3:
0000f0 e0f0             ldi   ZH, high (dsp_buff_3)  ; init ptr to line 2 display buffer.
0000f1 e8e0             ldi   ZL, low (dsp_buff_3)
                snd_ddram_addr3:
0000f2 ea00             ldi   r16,0xA0               ; init DDRAM addr-ctr
0000f3 dfaa             rcall lcd_spi_transmit_CMD          ;
0000f4 df90             rcall delay_30uS


                snd_buff_3:
0000f5 9101             ld    r16, Z+
0000f6 dfb4             rcall lcd_spi_transmit_DATA
0000f7 df8d             rcall delay_30uS
0000f8 954a             dec   r20
```

```
0000f9 f7d9             brne  snd_buff_3
0000fa 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:
0000fb e390          ldi R25, 48                    ; load total length of both buffer.
0000fc e2a0          ldi R26, ' '                   ; load blank/space into R26.
0000fd e0f0          ldi ZH, high (dsp_buff_1)      ; Load ZH and ZL as a pointer to 1st
0000fe e6e0          ldi ZL, low (dsp_buff_1)       ; byte of buffer for line 1.

                    ;set DDRAM address to 1st position of first line.
                store_bytes:
0000ff 93a1          st  Z+, R26                    ; store ' ' into 1st/next buffer byte and
                                                    ; auto inc ptr to next location.
000100 959a          dec  R25                       ;
000101 f7e9          brne store_bytes               ; cont until r25=0, all bytes written.
000102 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:
000103 e0d0          ldi YH, high (dsp_buff_1)      ; Load YH and YL as a pointer to 1st
000104 e6c0          ldi YL, low (dsp_buff_1)       ; byte of dsp_buff_1 (Note - assuming
                                                    ; (dsp_buff_1 for now).
000105 9105          lpm R16, Z+                    ; get dsply buff number (1st byte of msg).
000106 3001          cpi r16, 1                        ; if equal to '1', ptr already setup.
000107 f031          breq get_msg_byte              ; jump and start message load.
000108 9660          adiw YH:YL, 16                    ; else set ptr to dsp buff 2.
000109 e010         ldi r17, $00
00010a e0b6         ldi r27, 6
```

```
00010b 3002          cpi r16, 2                      ; if equal to '2', ptr now setup.
00010c f031          breq digit_load               ; jump and start message load.
00010d 9660          adiw YH:YL, 16                  ; else set ptr to dsp buff 3.

                get_msg_byte:
00010e 9105          lpm R16, Z+                   ; get next byte of msg and see if '0'.
00010f 3000          cpi R16, 0                      ; if equal to '0', end of message reached.
000110 f0a1          breq msg_loaded               ; jump and stop message loading operation.
000111 9309          st Y+, R16                      ; else, store next byte of msg in buffer.
000112 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:
000113 9513        inc r17
000114 e290        ldi r25, $20                 ;load empty spaces for 6 places
000115 d00d        rcall get_dis_freq           ;display
000116 131b        cpse r17, r27                ;check if 6 places typed
000117 cffb        rjmp digit_load              ;repeat until 6 places
000118 2d95        mov r25, r5                  ;load the first number in freq
000119 d009        rcall get_dis_freq           ;display
00011a 2d94        mov r25, r4                  ;load the first number in freq
00011b d007        rcall get_dis_freq           ;display
00011c 2d93        mov r25, r3                  ;load the second number in freq
00011d d005        rcall get_dis_freq           ;display
00011e 2d92        mov r25, r2                  ;load the third number in freq
00011f d003        rcall get_dis_freq           ;display
000120 2d91        mov r25, r1                  ;load the fourth number in freq
000121 d001        rcall get_dis_freq           ;display
000122 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
000123 9399        st Y+, r25                   ;display the selected frquency
000124 9508        ret


                msg_loaded:
000125 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:
000126 930f        push r16                     ;store the value currently in r16
000127 d017        rcall bin2BCD16              ;convert the values from binary to bcd
                   //sub r13, r9                    ;to fix slight error in frequency conversion
000128 2c2d        mov r2, r13                  ;make a copy of r13 in r2
000129 2c4e        mov r4, r14                  ;make a copy of r14 in r4
00012a 2c6f        mov r6, r15                  ;make a copy of r15 in r6
```

```
00012b e00f         ldi r16, $0f                    ;use and function to
00012c 22d0         and r13, r16                    ;mask the upper nibble of r8
00012d 2c1d         mov r1, r13                     ;move lower nibble to r1
00012e 22e0         and r14, r16                    ;mask upper nibble of r9
00012f 2c3e         mov r3, r14                     ;move lower nible to r3
000130 22f0         and r15, r16                    ;mask the upper nibble of r8
000131 2c5f         mov r5, r15                     ;move lower nibble to r1
000132 9500         com r16                         ;load with f0 to mask lower nibble
000133 2220         and r2, r16                     ;mask lower nibble of r8
000134 9422         swap r2                         ;switch upper and lower nibble
000135 2240         and r4, r16                     ;mask lower nibble of r9
000136 9442         swap r4                         ;switch upper and lower nibble
                    //and r6, r16                   ;mask lower nibble of r9
                    //swap r6                       ;switch upper and lower nibble
000137 e300         ldi r16, $30
000138 0e10         add r1, r16
000139 0e20         add r2, r16
00013a 0e30         add r3, r16
00013b 0e40         add r4, r16
00013c 0e50         add r5, r16                     ;converting the bcd's to ascii
00013d 910f         pop r16                         ;retrive the value previosly stored
00013e 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:
00013f 1889            sub r8, r9
000140 2d19            mov fbinH, r9            ;copy the values of edge counter to fbin
000141 2d08            mov fbinL, r8
```

```
000142 e120         ldi cnt16a,16           ;Init loop counter
000143 24ff         clr tBCD2               ;clear result (3 bytes)
000144 24ee         clr tBCD1
000145 24dd         clr tBCD0
000146 27ff         clr ZH                  ;clear ZH (not needed for AT90Sxx0x)
                bBCDx_1:
000147 0f00         lsl fbinL               ;shift input value
000148 1f11         rol fbinH               ;through all bytes
000149 1cdd         rol tBCD0               ;
00014a 1cee         rol tBCD1
00014b 1cff         rol tBCD2
00014c 952a         dec cnt16a              ;decrement loop counter
00014d f409         brne bBCDx_2            ;if counter not zero
00014e 9508         ret                     ;   return

                bBCDx_2:
00014f e1e0         ldi r30,AtBCD2+1        ;Z points to result MSB + 1
                bBCDx_3:
000150 9132         ld  tmp16a,-Z           ;get (Z) with pre-decrement
000151 5f3d         subi tmp16a,-$03        ;add 0x03
000152 fd33         sbrc tmp16a,3           ;if bit 3 not clear
000153 8330         st Z,tmp16a             ;store back
000154 8130         ld  tmp16a,Z            ;get (Z)
000155 5d30         subi tmp16a,-$30        ;add 0x30
000156 fd37         sbrc tmp16a,7           ;if bit 7 not clear
000157 8330         st  Z,tmp16a            ;store back
000158 30ed         cpi ZL,AtBCD0           ;done all three?
000159 f7b1         brne bBCDx_3            ;loop again if not
00015a cfec         rjmp bBCDx_1



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 :   4 r9 :   5 r10:   0 r11:   0 r12:   0 r13:   5 r14:   5 r15:   5
r16:  98 r17:   5 r18:   2 r19:   8 r20:   8 r21:   0 r22:   2 r23:   2
r24:   4 r25:  12 r26:   2 r27:   2 r28:   3 r29:   3 r30:  11 r31:  10
x  :   0 y  :   2 z  :  10
Registers used: 28 out of 35 (80.0%)

"ATmega16" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   0 add   :   5 adiw  :   2 and   :   5
andi  :   0 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   0 break :   0 breq  :   4 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   1 brlt  :   0 brmi  :   0
brne  :  10 brpl  :   0 brsh  :   0 brtc  :   1 brts  :   1 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   :   1 cp    :   2 cpc   :   0
cpi   :   4 cpse  :   1 dec   :   8 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :  12 inc   :   3 jmp   :   1
```

```
ld    :   5 ldd   :   0 ldi   :  61 lds   :   0 lpm   :   2 lsl   :   1
lsr   :   0 mov   :  16 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   2 or    :   0 ori   :   0 out   :  18 pop   :   8
push  :   8 rcall :  56 ret   :  15 reti  :   1 rjmp  :   8 rol   :   4
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   5 sbic  :   0 sbis  :   0
sbiw  :   0 sbr   :   0 sbrc  :   2 sbrs  :   2 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: 42 out of 113 (37.2%)

"ATmega16" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x0002b6    592     76    668   16384   4.1%
[.dseg] 0x000060 0x000090      0     48     48    1024   4.7%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors, 0 warnings
```