```
AVRASM ver. 2.1.52  C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014
notes and files\ese 380 lab\lab 7\frequency_rajith\frequency_rajith\frequency_rajith.asm
 Tue Oct 21 20:26:51 2014

C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese
380 lab\lab 7\frequency_rajith\frequency_rajith\frequency_rajith.asm(42): Including fi
le 'C:\Program Files (x86)\Atmel\Atmel Toolchain\AVR Assembler\Native\2.1.39.1005\avra
ssembler\Include\m16def.inc'
C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese
380 lab\lab 7\frequency_rajith\frequency_rajith\frequency_rajith.asm(159): Including f
ile 'C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and file
s\ese 380 lab\lab 7\frequency_rajith\frequency_rajith\lcd_dog_asm_driver_m16A.inc'


                 * frequency_rajith.asm
                 *
                 ;For this program we will measure the freuquency of the wave,
                 ;(pa7) genereated by a wavefunction generator. The freq value
                 ; read is displayed on the lcd, attached to port b. THe program
                 ; will count the number of positive edges for a 1s period, and
                 ; stores that value on r8 and r9, we are using two registers
                 ; becuase number will be in thousands. The r18 and r19 will
                 ; be used as loop counter to count for 1s. The subroutines
                 ; clr_dsp_buff ,update_dsp and load_msg are taken directly
                 ; from the codes provided(creidts:Scott Tierno). The first
                 ;line of lcd displays "frequency", the second line will display
                 ; the frequency values unpacked and stored in r1 -r4.
                 ;
                 ;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:
                 ; r16
                 ; r21 - 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
                 ; r18, r19 - 1 s loop counter
                 ; r27 - set to 6, so we can compare and stop after r17 is inc
                 ; 6 times.
                 ; r1,r2,r3,r4 are used to unpack and contain the values stored
                 ; in r8 and r9, which will later be accessed to display the freq
                 ;


                 *  Created: 10/18/2014 9:19:14 AM
                 *   Author: radra_000
                 */


                 .list

                reset:
000000 e50f         ldi r16, low(RAMEND)              ; init stack/pointer
000001 bf0d          out SPL, r16                     ;
000002 e004          ldi r16, high(RAMEND)            ;
000003 bf0e          out SPH, r16

000004 ef0f          ldi r16, 0xff                    ; set portB = output.
000005 bb07          out DDRB, r16                    ;
000006 9ac4          sbi portB, 4                     ; set /SS of DOG LCD = 1 (Deselected)1
```

```
000007 e001         ldi r16, 1                          ; set DDRC for all in but PC0
000008 bb04         out DDRC, r16

000009 9aa8         sbi PortC, 0                        ; turn off sounder
                //  ldi r21, 0                          ;load edge counter
                // ldi r22, $A0                         ;load loop counter
                // ldi r23, $fa                         ;load r23 with 250, so whenever other
                                                        ;counter registers reach 250 we could find the
                                                        ;frequency or delay accurately

                // ldi r16, 0x59                        ;this is just a test code used for simulating
                // mov r8, r16                          ;purposes
                // ldi r16, 0x13
                // mov r9, r16
                // ldi r27, 6
00000a d06d          rcall init_lcd_dog                 ; init display, using SPI serial interface


                //main will call various subroutines. first it will measure the frequcency
                ; The unpack will take the ferquency values and store each number in r1 - r4
                ; the message_dsp_loop will take the values stored in r1-r4 and displays it
                ; and it will repeat forever.

                main:
00000b d003         rcall freq_meas_1secgate            ;count the freqency of the wave.
00000c d0d5         rcall unpack                        ;unpack the calcualted frequency value
00000d d017         rcall message_dsp_loop              ;display the values unpacked
00000e cffc         rjmp main                           ;repeat the process

                ;-------------------------------------------------
                ;code to count the number of positive edges and store
                ;the value in r8 and r9, counted for 1s.
                ;-------------------------------------------------

                freq_meas_1secgate:
00000f ea30         ldi r19, $a0                        ;set initial values
000010 e020         ldi r18, $00                        ;for the outer loop counter
000011 b399         in r25, pinA                        ;and positive edge counter
                    pos_edge:
000012 e01a             ldi r17, 10                     ;set tweak delay to 10
000013 b309             in r16, pina                    ;take in the current wave signal logic
000014 1709             cp r16,r25                      ;and compare to previous logic recorded,
000015 f041             breq tweak                      ;if it is the same then skip to tweak delay
000016 e017             ldi r17, 7                      ;set tweak delaay for 7
000017 e090             ldi r25, $00                    ;set previous logic to 0 just in case
000018 f028             brcs tweak                      ;if there is a carry then branch
000019 e012             ldi r17, 2                      ;if not then set tweak delay to 2
00001a e890             ldi r25, $80                    ;and set previous logic to be 1
00001b 9483             inc r8                          ;and then increment the counter
00001c f409             brne tweak                      ;if it didnt over count then go to tweak delay
00001d 9493             inc r9                          ;if so then increment the second register
                        tweak:
00001e 951a                 dec r17                     ;decrement the tweak counter
00001f f7f1                 brne tweak                  ;and keep looping
                        neg_edge:
000020 952a                 dec r18                     ;decrement outer loop
000021 f781                 brne pos_edge               ;counter and if 0
000022 953a                 dec r19                     ;then decrement the second register
000023 f771                 brne pos_edge               ;if second register is not 0 then keep looping
000024 9508         ret
                ;-----------------------------------------------------------
                ;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:
000025 d095        rcall clr_dsp_buffs                 ; clear all three buffer lines
000026 d06f        rcall update_lcd_dog                    ;


                   ;load 1st line of prompt message into dbuff1
000027 e0f0        ldi  ZH, high(line1_message<<1)     ;
000028 e6e2        ldi  ZL, low(line1_message<<1)      ;
000029 d099        rcall load_msg                      ; load message into buffer(s).

                   ;LOAD 2ND LINE OF THE MESSAGE INTO DBUFF2
00002a e0f0        ldi ZH, high(line2_message<<1)      ;
00002b e7e4        ldi ZL, low(line2_message<<1)       ;load the table to stack
00002c d096        rcall load_msg                      ;load the frequency number into the buffer

                   ;load 3rd line of prompt message into dbuff3
00002d e0f0        ldi  ZH, high(line3_message<<1)     ;
00002e e7e8        ldi  ZL, low(line3_message<<1)      ;
00002f d093        rcall load_msg                      ; load message into buffer(s).
000030 d065        rcall update_lcd_dog

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

000031 2a01
000032 2a2a
000033 5246
000034 5145
000035 4555
000036 434e
000037 2a59
000038 2a2a
000039 002a     line1_message: .db 1, "***FREQUENCY****", 0  ; test string for line #1.
00003a 2002

00003b 0000     line2_message: .db 2," ",0

00003c 5c03
00003d 5c2f
00003e 5c2f
00003f 4548
000040 5452
000041 2f5a
000042 2f5c
000043 2f5c
000044 005c     line3_message: .db 3, "\/\/\HERTZ/\/\/\", 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).
                ;*****************************************************************
000045 0000     delay_30uS:   nop     ; fine tune delay
```

```
000046 0000                     nop
000047 938f                     push  r24
000048 e08f                     ldi   r24, 0x0f  ; load delay count.
000049 958a      d30_loop:      dec   r24        ; count down to
00004a f7f1                     brne  d30_loop   ; zero.
00004b 918f                     pop   r24
00004c 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
                 ;*********************************************************************
00004d dff7      v_delay:       rcall delay_30uS ; delay for ~30uS
00004e 956a                     dec   r22      ; decrement inner loop value, and
00004f f7e9                     brne  v_delay  ; loop until zero.
000050 957a                     dec   r23      ; decr outer loop count, and loop back
000051 f7d9                     brne  v_delay  ; to inner loop delay until r23 zero.
000052 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.
                 ;*********************************************************************
000053 e060      delay_40mS:    ldi   r22,0        ; load inner loop var
000054 e074                     ldi   r23,4        ; load outer loop var
000055 dff7                     rcall  v_delay     ; delay
000056 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:
```

```
000057 930f            push r16
000058 e50c            ldi r16,(1<<SPE) | (1<<MSTR) | (1<<CPOL) | (1<<CPHA)
000059 b90d            out SPCR,r16   ; Enable SPI, Master, fck/4,

                       ;kill any spurious data...
00005a b10e            in r16, SPSR   ; clear SPIF bit in SPSR
00005b b10f            in r16, SPDR   ;
00005c 910f            pop r16        ; restore r16 value...
00005d 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:
00005e 930f          push r16          ; save command, need r16.
00005f 98c3          cbi   portB, 3    ; clr PB1 = RS = 0 = command.
000060 98c4          cbi   portB, 4    ; clr PB2 = /SS = selected.
000061 b10e             in r16, SPSR   ; clear SPIF bit in SPSR.
000062 b10f             in r16, SPDR   ;
000063 910f          pop r16           ; restore command
000064 b90f          out SPDR,r16      ; write data to SPI port.


                     ;Wait for transmission complete
                wait_transmit:
000065 b10e          in r16, SPSR     ; read status reg
000066 ff07          sbrs r16, SPIF   ; if bit 7  = 0 wait
000067 cffd          rjmp wait_transmit
000068 b10f         in r16, SPDR       ;added by Ken to clear SPIF
000069 9ac4          sbi   portB, 4   ; set PB2 = /SS = deselected
00006a 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:
00006b 930f          push r16          ; save command, need r16.
00006c 9ac3          sbi   portB, 3    ; clr PB1 = RS = 1 = data.
00006d 98c4          cbi   portB, 4    ; clr PB2 = /SS = selected.
00006e b10e             in r16, SPSR   ; clear SPIF bit in SPSR.
00006f b10f             in r16, SPDR   ;
000070 910f          pop r16           ; restore command.
000071 b90f          out SPDR,r16      ; write data to SPI port.


                     ;Wait for transmission complete
                wait_transmit1:
```

```
000072 b10e           in r16, SPSR     ; read status reg
000073 ff07           sbrs r16, SPIF   ; if bit 7  = 0 wait
000074 cffd           rjmp wait_transmit1
000075 b10f          in r16, SPDR       ;clear SPIF (because it follows in r16,SPSR)
000076 9ac4           sbi   portB, 4   ; set PB2 = /SS = deselected
000077 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:
000078 dfde             rcall init_spi_lcd   ; init SPI port for DOG LCD.


                start_dly_40ms:
000079 dfd9             rcall  delay_40mS    ; startup delay.


                func_set1:
00007a e309             ldi    r16,0x39      ; send fuction set #1
00007b dfe2             rcall  lcd_spi_transmit_CMD   ;
00007c dfc8             rcall  delay_30uS    ; delay for command to be processed


                func_set2:
00007d e309             ldi    r16,0x39      ; send fuction set #2
00007e dfdf             rcall  lcd_spi_transmit_CMD
00007f dfc5             rcall  delay_30uS    ; delay for command to be processed


                bias_set:
000080 e10e             ldi    r16,0x1E      ; set bias value.
000081 dfdc             rcall  lcd_spi_transmit_CMD
000082 dfc2             rcall  delay_30uS    ;


                                             ; ========================================
                                             ; === CALIBRATION PARAMETER - USER ADJUSTABLE
                power_ctrl:                  ; === (CAUTION... VERY DELICATE ADJUSTMENT)
000083 e500             ldi    r16,0x50      ; === 5V ~= 0x50 nominal;      Adjust by 1 ONLY
000084 dfd9             rcall  lcd_spi_transmit_CMD ; === 3.3V ~= 0x55 nominal     and think hex!
000085 dfbf             rcall  delay_30uS    ;   Hex = 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f
                                             ; ========================================


                follower_ctrl:
000086 e60c             ldi    r16,0x6C      ; follower mode on...
000087 dfd6             rcall  lcd_spi_transmit_CMD
000088 dfca             rcall  delay_40mS    ;
                                             ; ========================================
                                             ; === CALIBRATION PARAMETER - USER ADJUSTABLE
                contrast_set:                ; === LCD CONTRAST SETTING ADJUSTMENT
000089 e707             ldi    r16,0x77      ; ===
00008a dfd3             rcall  lcd_spi_transmit_CMD ; === Delicate: increases for 3.3V vs 5V
00008b dfb9             rcall  delay_30uS    ; ========================================


                display_on:
00008c e00c             ldi    r16,0x0c      ; display on, cursor off, blink off
```

```
00008d dfd0             rcall  lcd_spi_transmit_CMD
00008e dfb6             rcall  delay_30uS    ;


                clr_display:
00008f e001             ldi    r16,0x01      ; clear display, cursor home
000090 dfcd             rcall  lcd_spi_transmit_CMD


000091 dfb3             rcall  delay_30uS    ;


                entry_mode:
000092 e006             ldi    r16,0x06      ; clear display, cursor home
000093 dfca             rcall  lcd_spi_transmit_CMD;
000094 dfb0             rcall  delay_30uS    ;
000095 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:
000096 dfc0             rcall init_spi_lcd    ; init SPI port for LCD.
000097 e140             ldi   r20,16          ; init 'chars per line' counter.
000098 934f             push  r20             ; save for later used.


                        ;send line 1 to the LCD module.
                wr_line1:
000099 e0f0             ldi   ZH, high (dsp_buff_1)  ; init ptr to line 1 display buffer.
00009a e6e0             ldi   ZL, low (dsp_buff_1)   ;
                snd_ddram_addr:
00009b e800             ldi   r16,0x80             ; init DDRAM addr-ctr
00009c dfc1             rcall lcd_spi_transmit_CMD ;
00009d dfa7             rcall delay_30uS
                snd_buff_1:
00009e 9101             ld    r16, Z+
00009f dfcb             rcall lcd_spi_transmit_DATA
0000a0 dfa4             rcall delay_30uS
0000a1 954a             dec   r20
0000a2 f7d9             brne  snd_buff_1


                        ;send line 2 to the LCD module.
                init_for_buff_2:
0000a3 914f             pop   r20     ; reload r20 = chars per line counter
0000a4 934f             push  r20     ; save for line 3
                wr_line2:
0000a5 e0f0             ldi   ZH, high (dsp_buff_2)  ; init ptr to line 2 display buffer.
0000a6 e7e0             ldi   ZL, low (dsp_buff_2)
                snd_ddram_addr2:
0000a7 e900             ldi   r16,0x90               ; init DDRAM addr-ctr
0000a8 dfb5             rcall lcd_spi_transmit_CMD          ;
0000a9 df9b             rcall delay_30uS
                snd_buff_2:
0000aa 9101             ld    r16, Z+
```

```
0000ab dfbf             rcall lcd_spi_transmit_DATA
0000ac df98             rcall delay_30uS
0000ad 954a             dec   r20
0000ae f7d9             brne  snd_buff_2


                        ;send line 3 to the LCD module.
                init_for_buff_3:
0000af 914f             pop   r20     ; reload r20 = chars per line counter
                wr_line3:
0000b0 e0f0             ldi   ZH, high (dsp_buff_3)  ; init ptr to line 2 display buffer.
0000b1 e8e0             ldi   ZL, low (dsp_buff_3)
                snd_ddram_addr3:
0000b2 ea00             ldi   r16,0xA0               ; init DDRAM addr-ctr
0000b3 dfaa             rcall lcd_spi_transmit_CMD           ;
0000b4 df90             rcall delay_30uS


                snd_buff_3:
0000b5 9101             ld    r16, Z+
0000b6 dfb4             rcall lcd_spi_transmit_DATA
0000b7 df8d             rcall delay_30uS
0000b8 954a             dec   r20
0000b9 f7d9             brne  snd_buff_3
0000ba 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:
0000bb e390          ldi R25, 48                     ; load total length of both buffer.
0000bc e2a0          ldi R26, ' '                    ; load blank/space into R26.
0000bd e0f0          ldi ZH, high (dsp_buff_1)       ; Load ZH and ZL as a pointer to 1st
0000be e6e0          ldi ZL, low (dsp_buff_1)        ; byte of buffer for line 1.

                    ;set DDRAM address to 1st position of first line.
                store_bytes:
0000bf 93a1          st  Z+, R26                     ; store ' ' into 1st/next buffer byte and
                                                     ; auto inc ptr to next location.
0000c0 959a          dec  R25                        ;
0000c1 f7e9          brne store_bytes                ; cont until r25=0, all bytes written.
0000c2 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:
0000c3 e0d0          ldi YH, high (dsp_buff_1)         ; Load YH and YL as a pointer to 1st
0000c4 e6c0          ldi YL, low (dsp_buff_1)          ; byte of dsp_buff_1 (Note - assuming
                                                       ; (dsp_buff_1 for now).
0000c5 9105          lpm R16, Z+                       ; get dsply buff number (1st byte of msg).
0000c6 3001          cpi r16, 1                            ; if equal to '1', ptr already setup.
0000c7 f021          breq get_msg_byte                 ; jump and start message load.
0000c8 9660          adiw YH:YL, 16                        ; else set ptr to dsp buff 2.
0000c9 3002          cpi r16, 2                            ; if equal to '2', ptr now setup.
0000ca f031          breq digit_load                   ; jump and start message load.
0000cb 9660          adiw YH:YL, 16                        ; else set ptr to dsp buff 3.

                get_msg_byte:
0000cc 9105          lpm R16, Z+                       ; get next byte of msg and see if '0'.
0000cd 3000          cpi R16, 0                            ; if equal to '0', end of message reached.
0000ce f091          breq msg_loaded                   ; jump and stop message loading operation.
0000cf 9309          st Y+, R16                            ; else, store next byte of msg in buffer.
0000d0 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:
0000d1 9513        inc r17
0000d2 e290        ldi r25, $20                  ;load empty spaces for 6 places
0000d3 d00b        rcall get_dis_freq            ;display
0000d4 131b        cpse r17, r27                 ;check if 6 places typed
0000d5 cffb        rjmp digit_load               ;repeat until 6 places
0000d6 2d94        mov r25, r4                   ;load the first number in freq
0000d7 d007        rcall get_dis_freq            ;display
0000d8 2d93        mov r25, r3                   ;load the second number in freq
0000d9 d005        rcall get_dis_freq            ;display
0000da 2d92        mov r25, r2                   ;load the third number in freq
0000db d003        rcall get_dis_freq            ;display
0000dc 2d91        mov r25, r1                   ;load the fourth number in freq
0000dd d001        rcall get_dis_freq            ;display
0000de 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
```

```
0000df 9399         st Y+, r25                      ;display the selected frquency
0000e0 9508         ret


                msg_loaded:
0000e1 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:
0000e2 930f         push r16                        ;store the value currently in r16

0000e3 2c28         mov r2, r8                      ;make a copy of r8 in r2
0000e4 2c49         mov r4, r9                      ;make a copy of r9 in r4
0000e5 e00f         ldi r16, $0f                    ;use and function to
0000e6 2280         and r8, r16                     ;mask the upper nibble of r8
0000e7 2c18         mov r1, r8                      ;move lower nibble to r1
0000e8 2290         and r9, r16                     ;mask upper nibble of r9
0000e9 2c39         mov r3, r9                      ;move lower nible to r3
0000ea 9500         com r16                         ;load with f0 to mask lower nibble
0000eb 2220         and r2, r16                     ;mask lower nibble of r8
0000ec 9422         swap r2                         ;switch upper and lower nibble
0000ed 2240         and r4, r16                     ;mask lower nibble of r9
0000ee 9442         swap r4                         ;switch upper and lower nibble
0000ef e300         ldi r16, $30                    ;add thirty to every number
0000f0 0e10         add r1, r16                     ;to convert from hex to ascii
0000f1 0e20         add r2, r16
0000f2 0e30         add r3, r16
0000f3 0e40         add r4, r16
0000f4 910f         pop r16                         ;retrive the value previosly stored

0000f5 9508         ret

                ;-----------------------------------------------
                ;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:
0000f6 e644         ldi r20,100
                    outer:
0000f7 e251             ldi r21, 33
                        inner:
0000f8 955a                 dec r21
0000f9 f7f1                 brne inner
0000fa 954a                 dec r20
0000fb f7d9                 brne outer
0000fc 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 :   0 r6 :   0 r7 :   0
r8 :   4 r9 :   4 r10:   0 r11:   0 r12:   0 r13:   0 r14:   0 r15:   0
r16:  66 r17:   6 r18:   2 r19:   2 r20:  10 r21:   2 r22:   2 r23:   2
r24:   4 r25:  12 r26:   2 r27:   1 r28:   3 r29:   3 r30:   7 r31:   7
x  :   0 y  :   2 z  :   6
Registers used: 24 out of 35 (68.6%)

"ATmega16" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   0 add   :   4 adiw  :   2 and   :   4
andi  :   0 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   1 break :   0 breq  :   4 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :  13 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   3 cbr   :   0
clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   0 cls   :   0
clt   :   0 clv   :   0 clz   :   0 com   :   1 cp    :   1 cpc   :   0
cpi   :   3 cpse  :   1 dec   :  12 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :  12 inc   :   3 jmp   :   0
ld    :   3 ldd   :   0 ldi   :  51 lds   :   0 lpm   :   2 lsl   :   0
lsr   :   0 mov   :   8 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   2 or    :   0 ori   :   0 out   :   7 pop   :   7
push  :   7 rcall :  50 ret   :  14 reti  :   0 rjmp  :   6 rol   :   0
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   5 sbic  :   0 sbis  :   0
sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   2 sec   :   0 seh   :   0
sei   :   0 sen   :   0 ser   :   0 ses   :   0 set   :   0 sev   :   0
sez   :   0 sleep :   0 spm   :   0 st    :   3 std   :   0 sts   :   0
sub   :   0 subi  :   0 swap  :   2 tst   :   0 wdr   :   0
Instructions used: 29 out of 113 (25.7%)

"ATmega16" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x0001fa    466     40    506   16384   3.1%
[.dseg] 0x000060 0x000090      0     48     48    1024   4.7%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors, 0 warnings
```