

Embedded Microprocessor Systems Design I

ESE 380 - Section: L02

laboratory 10: Frequency Meter System with Auto Ranging

December 4, 2018

Instructor:	Professor Short
Roni Das	ID: 108378223
Asif Iqbal	ID: 110333685
Assigned Bench:	#2


```

1
2 AVRASM ver. 2.2.7 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm Tue Dec 04 20:30:10 2018
3
4 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(45): Including file 'C:/Program Files (x86)
  \Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avrasm\inc\m324adef.inc'
5 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(64): Including file 'C:\Users\ronid\OneDrive
  - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur
  \Direct_Period_Mesur\lcd_dog_asm_driver_m324a.inc'
6 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(685): warning: Register r13 already defined
  by the .DEF directive
7 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(686): warning: Register r14 already defined
  by the .DEF directive
8 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(687): warning: Register r15 already defined
  by the .DEF directive
9 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(689): warning: Register r18 already defined
  by the .DEF directive
10 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(690): warning: Register r19 already defined
  by the .DEF directive
11 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(694): warning: Register r18 already defined
  by the .DEF directive
12 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(695): warning: Register r19 already defined
  by the .DEF directive
13 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(696): warning: Register r20 already defined
  by the .DEF directive
14 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(697): warning: Register r21 already defined
  by the .DEF directive
15 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(704): warning: Register r17 already defined
  by the .DEF directive
16 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(45): Including file 'C:/Program Files (x86)
  \Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avrasm\inc\m324adef.inc'
17 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4
  \Direct_Period_Mesur\Direct_Period_Mesur\main.asm(64): Including file 'C:\Users\ronid\OneDrive
  - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur
  \Direct_Period_Mesur\lcd_dog_asm_driver_m324a.inc'
18
19
20 ;*****
21 ;lab_10_Dirct_period_mss_systemII.asm
22 ;
23 ;Author : Roni Das ID: 108378223
24 ; Asif Iqbal ID: 110333685
25 ;Created: 12/04/2018 8:24:08 PM

```

```

26      ;
27      ; Description: The following program displays the measurement
28      ; of the frequency of a fast periodic waveform and
29      ; displays it on a LCD. A 16 x 3 LCD is used
30      ; for this experiment. The first line illustrates
31      ; the frequency in Hz. The second
32      ; line represents the decimal count of the period.
33      ; The frequency is displayed in the unit of 1hz and
34      ; period is displayed in the unit of 1us. The signal
35      ; that is to be measure is 1khz to 65khz.
36      ;Inputs:
37      ; PORTC = PC0
38      ;
39      ;Output:
40      ; PORTB = PB0 - PB7
41      ;
42      ;Register Assignments/Purposes
43      ; r16 = general purpose
44      ; r19 = loop counter
45      ; r21 = period counter
46      ; r22 = digit 0
47      ; r23 = digit 1
48      ; r24 = digit 2
49      ; r13 = digit 0 and 1
50      ; r14 = digit 2 and 4
51      ; r25 = conversion purpose
52      ; r26 = conversion purpose
53      ; r27 = conversion purpose
54
55      ;*Courtesy of Professor Ken Short and his lecture notes.
56      ;
57      ; Section no.: L02
58      ; Experiment no.: 10
59      ; Bench no.: 02
60      ;*****
61
62      .list
63
64
65
66      .CSEG
67
68
69      ; interrupt vector table, with several 'safety' stubs
70      .org 0      ;Reset/Cold start vector
71      000000 c091      rjmp start
72      .org INT0addr
73      000002 c182      rjmp start_counter_timer
74
75
76
77
78      .list
79
80      ;*****
81      ;NAME:      clr_dsp_buffs

```

```

82      ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3
83      ;           with blanks (0x20)
84      ;ASSUMES:   Three CONTIGUOUS 16-byte dram based
85      ;           buffers named
86      ;           dsp_buff_1, dsp_buff_2, dsp_buff_3.
87      ;RETURNS:   nothing.
88      ;MODIFIES:  r25,r26, Z-ptr
89      ;CALLS:     none
90      ;CALLED BY: main application and diagnostics
91      ;*****
92
93
94      clr_dsp_buffs:
95 00007b e390      ldi R25, 48          ; load total length of
96                                     ; both buffer.
97 00007c e2a0      ldi R26, ' '        ; load blank/space into
98                                     ; R26.
99 00007d e0f1      ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a
100                                     ; pointer to 1st
101 00007e e0e0      ldi ZL, low (dsp_buff_1)  ; byte of buffer for
102                                     ; line 1.
103
104      ;set DDRAM address to 1st
105      ;position of first line.
106      store_bytes:
107 00007f 93a1      st Z+, R26          ; store ' ' into 1st/next
108                                     ; buffer byte and
109                                     ; auto inc ptr to next
110                                     ; location.
111 000080 959a      dec R25              ;
112 000081 f7e9      brne store_bytes    ; cont until r25=0, all
113                                     ; bytes written.
114 000082 9508      ret
115
116
117
118      ;*****
119      ;NAME:      load_msg
120      ;FUNCTION:  Loads a predefined string msg into
121      ;           a specified diplay
122      ;           buffer.
123      ;ASSUMES:   Z = offset of message to be loaded.
124      ;           ;Msg format is
125      ;           defined below.
126      ;RETURNS:   nothing.
127      ;MODIFIES:  r16, Y, Z
128      ;CALLS:     nothing
129      ;CALLED BY:
130      ;*****
131      ; Message structure:
132      ;   label:  .db <buff num>, <text string/message>
133      ;           , <end of string>
134      ;
135      ; Message examples (also see Messages
136      ; at the end of this file/module):
137      ;   msg_1: .db 1,"First Message ", 0

```

```

138 ; loads msg into buff 1, eom=0
139 ; msg_2: .db 1,"Another message ", 0
140 ; loads msg into buff 1, eom=0
141 ;
142 ; Notes:
143 ; a) The 1st number indicates which
144 ; buffer to load (either 1, 2, or 3).
145 ; b) The last number (zero) is an '
146 ;end of string' indicator.
147 ; c) Y = ptr to disp_buffer
148 ; Z = ptr to message
149 ; (passed to subroutine)
150 ;*****
151 load_msg:
152 000083 e0d1 ldi YH, high (dsp_buff_1) ; Load YH and YL
153 ;as a pointer to 1st
154 000084 e0c0 ldi YL, low (dsp_buff_1) ; byte of dsp_buff_1
155 ;(Note - assuming
156 ; (dsp_buff_1 for now).
157 000085 9105 lpm R16, Z+ ; get dsply buff number
158 ;(1st byte of msg).
159 000086 3001 cpi r16, 1 ; if equal to '1', ptr
160 ; already setup.
161 000087 f021 breq get_msg_byte ; jump and start message
162 ; load.
163 000088 9660 adiw YH:YL, 16 ; else set ptr to dsp
164 ;buff 2.
165 000089 3002 cpi r16, 2 ; if equal to '2', ptr
166 ;now setup.
167 00008a f009 breq get_msg_byte ; jump and start message
168 ;load.
169 00008b 9660 adiw YH:YL, 16 ; else set ptr to dsp
170 ;buff 2.
171
172 get_msg_byte:
173 00008c 9105 lpm R16, Z+ ; get next byte of msg
174 ;and see if '0'.
175 00008d 3000 cpi R16, 0 ; if equal to '0', end
176 ;of message reached.
177 00008e f011 breq msg_loaded ; jump and stop message
178 ; loading operation.
179 00008f 9309 st Y+, R16 ; else, store next byte
180 ;of msg in buffer.
181 000090 cffb rjmp get_msg_byte ; jump back and continue.
182 msg_loaded:
183 000091 9508 ret
184
185 start:
186 000092 ef0f ldi r16, low(RAMEND) ; init stack/pointer
187 000093 bf0d out SPL, r16
188 000094 e008 ldi r16, high(RAMEND)
189 000095 bf0e out SPH, r16
190
191
192 000096 ef0f ldi r16, 0xff ; set portB = output.
193 000097 b904 out DDRB, r16 ;

```

```

194 000098 9a2c          sbi portB, 4          ;set /SS of DOG LCD =
195                                     ; 1 (Deselected)
196
197 000099 9852          cbi DDRD, 2
198 00009a 985a          cbi portD, 2
199
200 00009b df9c          rcall init_lcd_dog      ; init display, using
201                                     ; SPI serial interface
202 00009c dfde          rcall clr_dsp_buffs     ; clear all three
203                                     ;lines
204
205
206                                     ;Configure port A bits 0-4 as an output
207 00009d e10c          ldi r16, 0b00011100      ;load r16 with 1s in
208                                     ; 0-4 postion
209                                     ;bit 0 and 1 position
210 00009e b901          out DDRA, r16            ;port A - bit 0 - 4 as
211                                     ;an output
212                                     ;Port A - bit 5-7 is
213                                     ; Input
214 00009f e100          ldi r16, 0b00010000      ;load r16 with all 1s a bit 4
215
216 0000a0 b902          out PORTA, r16           ;trun On LED: Color: RED
217
218
219                                     ;Configure port C bits 0 as an input pin
220 0000a1 e000          ldi r16,$00              ;load r16 with all 0s
221 0000a2 b907          out DDRC,r16             ;port C-bit 0 = input
222
223
224 0000a3 ef5f          ldi r21, $FF             ;Inititalize Display = 00
225 0000a4 e020          ldi r18, $00             ;counter +/- 2% or +/- 5%
226
227
228                                     ;power on self test
229
230
231 0000a5 e003          ldi r16, (1 << ISC01) | (1 << ISC00)
232 0000a6 9300 0069     sts EICRA, r16
233 0000a8 e001          ldi r16, 1 << INT0
234 0000a9 bb0d          out EIMSK, r16
235
236
237 0000aa e000          ldi r16,$00
238 0000ab 9300 0085     sts TCNT1H,r16
239 0000ad 9300 0084     sts TCNT1L,r16
240 0000af 9300 0080     sts TCCR1A,r16
241
242
243                                     main_loop:
244 0000b1 9478          sei
245                                     ;   sts TCCR1B,r16           ;clock stopped*/
246
247
248
249 0000b2 3535          cpi r19,$55

```

```

250 0000b3 f009      breq continue
251 0000b4 cffc      rjmp main_loop
252
253                continue:
254 0000b5 94f8      cli
255
256 0000b6 e030      ldi r19,$00
257 0000b7 9100 0084  lds r16, TCNT1L
258 0000b9 9110 0085  lds r17, TCNT1H
259 0000bb 2f21      mov r18,r17
260
261 0000bc e040      ldi r20,$00
262 0000bd 9340 0085  sts TCNT1H,r20
263 0000bf 9340 0084  sts TCNT1L,r20
264 0000c1 9340 0080  sts TCCR1A,r20
265 0000c3 9340 0081  sts TCCR1B,r20
266
267 0000c5 940e 0132  call setup_display_one
268
269                ;load_line_1 into dbuff1:
270 0000c7 e0f2      ldi ZH, high(line1_testmessage<<1)
271 0000c8 e5e2      ldi ZL, low(line1_testmessage<<1)
272
273 0000c9 940e 0056  call update_lcd_dog
274 0000cb 940e 0083  call load_msg      ;load message into buffer(s).
275
276
277 0000cd 3622      cpi r18,98                ;check lower bound 2%
278 0000ce f424      brge check_Upper_bound_2P ;branch if greater or equal
279 0000cf 352f      cpi r18,95                ;check lower bound 5%
280 0000d0 f444      brge check_Upper_bound_5P ;branch if greater or equal
281 0000d1 f0fc      brlt lower_than_5P        ;else count < 95
282 0000d2 cfde      rjmp main_loop
283
284                check_Upper_bound_2P:
285 0000d3 3627      cpi r18,103
286 0000d4 f044      brlt display_bargraph_2P; branch if lower
287 0000d5 362a      cpi r18,106
288 0000d6 f084      brlt display_bargraph_5P;branch if lower
289 0000d7 f4cc      brge higher_than_5P      ;branch if higher
290 0000d8 cfd8      rjmp main_loop
291
292                check_Upper_bound_5P:
293 0000d9 362a      cpi r18,106
294 0000da f4b4      brge higher_than_5P      ;branch if higher
295 0000db f05c      brlt display_bargraph_5P;branch if lower
296 0000dc cfd4      rjmp main_loop
297
298                display_bargraph_2P:
299                ;call display_g_LCD
300
301                ;load_line_1 into dbuff1:
302                ;ldi ZH, high(line1_testmessage<<1)
303                ;ldi ZL, low(line1_testmessage<<1)
304                ;rcall load_msg      ;load message into buffer(s).
305

```



```

306 0000dd 940e 00fb      call tol_2
307 0000df e0f2          ldi  ZH, high(line3_testmessage<<1)
308 0000e0 e6e2          ldi  ZL, low(line3_testmessage<<1)
309 0000e1 dfa1          rcall load_msg      ;load message into buffer(s).
310
311
312 0000e2 df73          rcall update_lcd_dog
313
314
315 0000e3 9a12          sbi  PORTA, 2      ;turn on green
316 0000e4 9813          cbi  PORTA, 3      ; turn off blue
317 0000e5 9814          cbi  PORTA, 4      ;turn off red
318 0000e6 cfca          rjmp main_loop
319
320      display_bargraph_5P:
321
322          ;call display_g_LCD
323
324          ;load_line_1 into dbuff1:
325          ; ldi  ZH, high(line1_testmessage<<1)
326          ; ldi  ZL, low(line1_testmessage<<1)
327          ; rcall load_msg      ;load message into buffer(s).
328
329 0000e7 940e 0108      call tol_5
330 0000e9 e0f2          ldi  ZH, high(line3_testmessage<<1)
331 0000ea e6e2          ldi  ZL, low(line3_testmessage<<1)
332 0000eb df97          rcall load_msg      ;load message into buffer(s).
333
334
335 0000ec df69          rcall update_lcd_dog
336
337
338 0000ed 9812          cbi  PORTA, 2      ;turn off green
339 0000ee 9a13          sbi  PORTA, 3      ;turn on blue
340 0000ef 9814          cbi  PORTA, 4      ;turn off red
341 0000f0 cfc0          rjmp main_loop
342
343      lower_than_5P:
344      higher_than_5P:
345
346          ;call display_g_LCD
347          ;load_line_1 into dbuff1:
348          ;ldi  ZH, high(line1_testmessage<<1)
349          ;ldi  ZL, low(line1_testmessage<<1)
350          ;rcall load_msg      ;load message into buffer(s).
351
352 0000f1 940e 0115      call tol_ORR
353 0000f3 e0f2          ldi  ZH, high(line3_testmessage<<1)
354 0000f4 e6e2          ldi  ZL, low(line3_testmessage<<1)
355 0000f5 df8d          rcall load_msg      ;load message into buffer(s).
356
357
358 0000f6 df5f          rcall update_lcd_dog
359
360 0000f7 9812          cbi  PORTA, 2      ;turn off green
361 0000f8 9813          cbi  PORTA, 3      ;turn on blue

```

```

362 0000f9 9a14          sbi PORTA, 4          ;turn on red
363
364 0000fa cfb6          rjmp main_loop
365
366
367
368 ;*****
369 ;*
370 ;* "Subroutine_name" - Tolerance
371 ;*
372 ;* Description: This subroutine does the ascii conversion
373 ;*              which is to be displayed in the LCD. This is
374 ;*              displayed in the 3rd line. Values from the
375 ;*              character set was loaded into r26 and r27
376 ;*              and then r25 was loaded with decimal value.
377 ;*              To get the ascii value , r25 was or'ed with
378 ;*              48 for the conversion (since 48 is 0 in ascii).
379 ;*              Using the Y pointer, each digit is to be
380 ;*              displayed in a specific position.
381 ;*
382 ;*
383 ;* Author: Asif Iqbal
384 ;*        Roni Das
385 ;* Version:1A
386 ;* Last updated: 11/06/2018
387 ;* Target: Perfect
388 ;* Number of words: 10
389 ;* Number of cycles: 300/307 (Min/Max)
390 ;* Low registers modified: none
391 ;* High registers modified: 3
392 ;*
393 ;* Parameters:
394 ;*
395 ;* Returns:
396 ;*
397 ;* Notes:
398 ;*
399 ;*****
400
401
402 tol_2:
403 0000fb e0d1          ldi YH,HIGH(dsp_buff_3)    ;displaying on line 3
404 0000fc e2c0          ldi YL,LOW(dsp_buff_3)
405 0000fd e092          ldi r25,$02          ;loading r25 with hex 02
406 0000fe e2a5          ldi r26, 0b00100101    ;loading r26 with 0b00100101
407 0000ff e2b0          ldi r27, $20          ;loading r27 with hex 20
408 000100 6390          ori r25, 48          ;or immediately to get the ascii
409 000101 839f          std y+7, r25          ;displaying at the 7th position
410 000102 87a8          std y+8,r26          ;displaying at the 8th position
411 000103 87b9          std y+9,r27          ;displaying at the the position
412 000104 e200          ldi r16,' '
413 000105 870a          std y+10,r16
414 000106 870c          std y+12,r16
415 000107 9508          ret
416 tol_5:
417 000108 e0d1          ldi YH,HIGH(dsp_buff_3)

```

```

418 000109 e2c0      ldi YL,LOW(dsp_buff_3)
419 00010a e095      ldi r25,$05
420 00010b e2a5      ldi r26, 0b00100101
421 00010c e2b0      ldi r27, $20
422 00010d 6390      ori r25, 48
423 00010e 839f      std y+7, r25
424 00010f 87a8      std y+8,r26
425 000110 87b9      std y+9,r27
426 000111 e200      ldi r16,' '
427 000112 870a      std y+10,r16
428 000113 870c      std y+12,r16
429
430 000114 9508      ret
431
432 tol_ORR:
433 000115 e0d1      ldi YH,HIGH(dsp_buff_3)
434 000116 e2c0      ldi YL,LOW(dsp_buff_3)
435 000117 e49f      ldi r25,$4F
436 000118 e5a2      ldi r26,$52
437 000119 839f      std y+7, r25
438 00011a 8798      std y+8, r25
439 00011b 87a9      std y+9,r26
440
441 00011c e200      ldi r16,' '
442 00011d 870a      std y+10,r16
443 00011e 870c      std y+12,r16
444
445 00011f 9508      ret
446
447 ;converting binary to ascii
448 000120 e0d1      ldi YH,HIGH(dsp_buff_2) ;displaying at line 2
449 000121 e1c0      ldi YL,LOW(dsp_buff_2)
450 000122 6360      ori r22, 48      ;ori toconvert register content to ascii
451 000123 6370      ori r23, 48
452 000124 6380      ori r24, 48
453 000125 838f      std y+7, r24;displaying it to the 7th position in lcd
454 000126 8778      std y+8, r23
455 000127 8769      std y+9, r22
456 000128 9508      ret
457
458
459 ;*****
460 ;***** ALL MESSAGES: Fixed format, flash stored/loaded *
461 ;*****
462
463
464 000129 4601
465 00012a 7172
466 00012b 3d20
467 00012c 0020      line1_testmessage: .db 1, "Frq = ", 0; message for line #1.
468 00012d 5002
469 00012e 6472
470 00012f 3d20
471 000130 0020      line2_testmessage: .db 2, "Prd = ", 0; message for line #2.
472 000131 0003      line3_testmessage: .db 3, "", 0; message for line #3.
473

```

```

474
475
476
477             setup_display_one:
478             ;clr r17
479 000132 940e 0168    call bin2BCD16             ;output r13,r14,15
480 000134 2d6d        mov r22, r13             ;new
481 000135 2d7d        mov r23, r13             ;new
482 000136 706f        andi r22, $0F             ;digit 0 value
483 000137 7f70        andi r23, $F0
484 000138 9572        swap r23                 ;digit 1 value
485 000139 2d8e        mov r24, r14
486 00013a 2d9e        mov r25, r14
487 00013b 708f        andi r24, $0F             ;digit 2 value
488 00013c 7f90        andi r25, $F0
489 00013d 9592        swap r25
490 00013e 2daf        mov r26, r15
491 00013f 70af        andi r26, $0F
492
493
494
495 000140 2f2a        mov r18, r26
496 000141 e604        ldi r16,100
497 000142 9f20        mul r18,r16
498 000143 2d20        mov r18,r0
499
500
501 000144 2f49        mov r20, r25
502 000145 e00a        ldi r16,10
503 000146 9f40        mul r20,r16
504
505 000147 1d20        adc r18,r0
506 000148 2f48        mov r20,r24
507 000149 1f24        adc r18,r20
508
509
510             ;mov r25, r13
511             ;andi r25, $0F
512
513             ;converting binary to ascii
514             convert_ascii:
515             ;ldi r19, 9 ; looping for 8 bits
516 00014a e0d1        ldi YH,HIGH(dsp_buff_1)
517 00014b e0c0        ldi YL,LOW(dsp_buff_1)
518             ;adiw YH:YL, 7
519 00014c 6360        ori r22, 48
520 00014d 6370        ori r23, 48
521 00014e 6380        ori r24, 48
522 00014f 6390        ori r25, 48
523 000150 63a0        ori r26, 48
524
525
526 000151 940e 0159    call msg
527
528 000153 83af        std y+7, r26
529 000154 8798        std y+8, r25

```

```

530 000155 878a          std y+10, r24
531 000156 877b          std y+11, r23
532 000157 876c          std y+12, r22
533 000158 9508          ret
534
535
536                      msg:
537
538 000159 e500          ldi r16,'P'
539 00015a 8308          std y+0,r16
540 00015b e502          ldi r16,'R'
541 00015c 8309          std y+1,r16
542 00015d e404          ldi r16,'D'
543 00015e 830a          std y+2,r16
544 00015f e30d          ldi r16,'='
545 000160 830c          std y+4,r16
546 000161 e20e          ldi r16,'.'
547 000162 8709          std y+9,r16
548 000163 e60d          ldi r16,'m'
549 000164 870e          std y+14,r16
550 000165 e703          ldi r16,'s'
551 000166 870f          std y+15,r16
552 000167 9508          ret
553                      ;***** END OF FILE *****
554
555                      ;*****
556                      ;*
557                      ;* "bin2BCD16" - 16-bit Binary to BCD conversion
558                      ;*
559                      ;* This subroutine converts a 16-bit number (fbinH:fbinL) to
560                      ;* a 5-digit
561                      ;* packed BCD number represented by 3 bytes (tBCD2:tBCD1:tBCD0).
562                      ;* MSD of the 5-digit number is placed in the lowermost
563                      ;* nibble of tBCD2.
564                      ;*
565                      ;* Number of words :25
566                      ;* Number of cycles:751/768 (Min/Max)
567                      ;* Low registers used :3 (tBCD0,tBCD1,tBCD2)
568                      ;* High registers used :4(fbinL,fbinH,cnt16a,tmp16a)
569                      ;* Pointers used :Z
570                      ;*
571                      ;*****
572
573                      ;***** Subroutine Register Variables
574
575                      .equ   AtBCD0   =13      ;address of tBCD0
576                      .equ   AtBCD2   =15      ;address of tBCD1
577
578                      .def    tBCD0    =r13     ;BCD value digits 1 and 0
579                      .def    tBCD1    =r14     ;BCD value digits 3 and 2
580                      .def    tBCD2    =r15     ;BCD value digit 4
581                      .def    fbinL    =r16     ;binary value Low byte
582                      .def    fbinH    =r17     ;binary value High byte
583                      .def    cnt16a   =r18     ;loop counter
584                      .def    tmp16a   =r19     ;temporary value
585

```

```

586                                     ;***** Code
587
588 bin2BCD16:
589     000168 930f        push r16
590     000169 931f        push r17
591     00016a e120        ldi cnt16a,16    ;Init loop counter
592     00016b 24ff        clr tBCD2      ;clear result (3 bytes)
593     00016c 24ee        clr tBCD1
594     00016d 24dd        clr tBCD0
595     00016e 27ff        clr ZH          ;clear ZH (not needed for AT90Sxx0x)
596     00016f 0f00        bBCDx_1:lslfbInL    ;shift input value
597     000170 1f11        rol fbinH         ;through all bytes
598     000171 1cdd        rol tBCD0         ;
599     000172 1cee        rol tBCD1
600     000173 1cff        rol tBCD2
601     000174 952a        dec cnt16a       ;decrement loop counter
602     000175 f419        brnebBCDx_2      ;if counter not zero
603     000176 911f        pop r17
604     000177 910f        pop r16
605     000178 9508        ret              ; return
606
607     000179 e1e0        bBCDx_2:ldir30,AtBCD2+1;Z points to result MSB + 1
608
609     00017a 9132        ld tmp16a,-Z     ;get (Z) with pre-decrement
610
611                                     ;-----
612                                     ;For AT90Sxx0x, substitute the above line with:
613                                     ;
614                                     ; dec ZL
615                                     ; ld tmp16a,Z
616                                     ;
617                                     ;-----
618     00017b 5f3d        subitmp16a,-$03 ;add 0x03
619     00017c fd33        sbrctmp16a,3;if bit 3 not clear
620     00017d 8330        st Z,tmp16a; store back
621     00017e 8130        ld tmp16a,Z;get (Z)
622     00017f 5d30        subitmp16a,-$30 ;add 0x30
623     000180 fd37        sbrctmp16a,7;if bit 7 not clear
624     000181 8330        st Z,tmp16a; store back
625     000182 30ed        cpi ZL,AtBCD0    ;done all three?
626     000183 f7b1        brnebBCDx_3      ;loop again if not
627     000184 cfea        rjmp bBCDx_1
628
629
630
631
632 start_counter_timer:
633
634     000185 e000        ldi r16,$00
635     000186 9300 0085    sts TCNT1H,r16
636     000188 9300 0084    sts TCNT1L,r16
637     00018a 9300 0080    sts TCCR1A,r16
638     00018c e001        ldi r16,$01
639
640
641     00018d e020        ldi r18, $00;counter +/- 2% or +/- 5

```

```

642                                     lb_0:
643 00018e 994a                         sbic PIND,2      ;Check for a clear bit @ PINC0
644 00018f cffe                         rjmp lb_0
645
646                                     lb_1:
647 000190 9b4a                         sbis PIND,2      ;Check for a set bit @ PINC0
648 000191 cffe                         rjmp lb_1
649                                     lb_2:
650
651 000192 9300 0081                     sts TCCR1B,r16   ;start counting
652
653                                     ;call var_delay
654                                     ;inc r18
655 000194 994a                         sbic PIND,2      ;Check for a clear bit @ PINC0
656 000195 cffc                         rjmp lb_2
657                                     lb_3:
658                                     ;call var_delay
659                                     ;inc r18
660 000196 9b4a                         sbis PIND,2      ;Check for a set bit @ PINC0
661 000197 cffe                         rjmp lb_3
662
663 000198 e000                         ldi r16,$00
664 000199 9300 0081                     sts TCCR1B,r16   ;clock stopped
665
666 00019b 9100 0084                     lds r16, TCNT1L
667 00019d 9110 0085                     lds r17, TCNT1H
668 00019f 2f21                         mov r18,r17
669 0001a0 e535                         ldi r19,$55
670
671 0001a1 9518                         reti
672
673                                     DIVISION:
674
675
676                                     ;*****
677                                     ;*
678                                     ;* "div32u" - 32/32 Bit Unsigned Division
679                                     ;*
680                                     ;* Ken Short
681                                     ;*
682                                     ;* This subroutine divides the two 32-bit numbers
683                                     ;* "dd32u3:dd32u2:dd32u1:dd32u0" (dividend) and "
684                                     ;* dv32u3:dv32u2:dv32u3:dv32u2"
685                                     ;* (divisor).
686                                     ;* The result is placed in "dres32u3:dres32u2
687                                     ;* dres32u2" and the
688                                     ;* remainder in "drem32u3:drem32u2:drem32u3:drem3
689                                     ;*
690                                     ;* Number of words :
691                                     ;* Number of cycles:655/751 (Min/Max) ATmega16
692                                     ;* #Low registers used :2 (drem16uL,drem16uH)
693                                     ;* #High registers used :5
694                                     ;* (dres16uL/dd16uL,dres16uH/dd16uH,dv16uL,dv16uH,
695                                     ;*          dcnt16u)
696                                     ;* A $0000 divisor returns $FFFF
697                                     ;*

```

```

698 ;*****
699
700 ;***** Subroutine Register Variables
701
702 .def drem32u0=r12 ;remainder
703 .def drem32u1=r13
704 .def drem32u2=r14
705 .def drem32u3=r15
706
707 .def dres32u0=r18 ;result (quotient)
708 .def dres32u1=r19
709 .def dres32u2=r20
710 .def dres32u3=r21
711
712 .def dd32u0 =r18 ;dividend
713 .def dd32u1 =r19
714 .def dd32u2 =r20
715 .def dd32u3 =r21
716
717 .def dv32u0 =r22 ;divisor
718 .def dv32u1 =r23
719 .def dv32u2 =r24
720 .def dv32u3 =r25
721
722 .def dcnt32u =r17
723
724 ;***** Code
725
726 div32u:
727 0001a2 24cc clr drem32u0;clear remainder Low byte
728 0001a3 24dd clr drem32u1
729 0001a4 24ee clr drem32u2
730 0001a5 18ff sub drem32u3,drem32u3;clear remainder High byte
731 0001a6 e211 ldi dcnt32u,33 ;init loop counter
732 d32u_1:
733 0001a7 1f22 rol dd32u0 ;shift left dividend
734 0001a8 1f33 rol dd32u1
735 0001a9 1f44 rol dd32u2
736 0001aa 1f55 rol dd32u3
737 0001ab 951a dec dcnt32u ;decrement counter
738 0001ac f409 brned32u_2 ;if done
739 0001ad 9508 ret ; return
740 d32u_2:
741 0001ae 1ccc rol drem32u0;shift dividend into remainder
742 0001af 1cdd roldrem32u1
743 0001b0 1cee roldrem32u2
744 0001b1 1cff rol drem32u3
745
746 0001b2 1ac6 sub drem32u0,dv32u0 ;remainder = remainder - divisor
747 0001b3 0ad7 sbcdrem32u1,dv32u1
748 0001b4 0ae8 sbcdrem32u2,dv32u2
749 0001b5 0af9 sbc drem32u3,dv32u3 ;
750 0001b6 f430 brccd32u_3 ; branch if reult is pos or zero
751
752 0001b7 0ec6 add drem32u0,dv32u0 ;if result negative
753 0001b8 1ed7 adc drem32u1,dv32u1

```



```

754 0001b9 1ee8          adc drem32u2,dv32u2
755 0001ba 1ef9          adc drem32u3,dv32u3
756 0001bb 9488          clc          ; clear carry to be shifted into result
757 0001bc cfea          rjmpd32u_1      ;else
758 0001bd 9408          d32u_3:sec          ; set carry to be shifted into result
759 0001be cfe8          rjmpd32u_1

```

760

761

762

763 RESOURCE USE INFORMATION

764 -----

765

766 Notice:

767 The register and instruction counts are symbol table hit counts,

768 and hence implicitly used resources are not counted, eg, the

769 'lpm' instruction without operands implicitly uses r0 and z,

770 none of which are counted.

771

772 x,y,z are separate entities in the symbol table and are

773 counted separately from r26..r31 here.

774

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

776

777 "ATmega324A" register use summary:

778 x : 0 y : 31 z : 10 r0 : 2 r1 : 0 r2 : 0 r3 : 0 r4 : 0

779 r5 : 0 r6 : 0 r7 : 0 r8 : 0 r9 : 0 r10: 0 r11: 0 r12: 4

780 r13: 8 r14: 8 r15: 8 r16: 10 r17: 10 r18: 17 r19: 12 r20: 18

781 r21: 2 r22: 10 r23: 11 r24: 13 r25: 19 r26: 13 r27: 4 r28: 8

782 r29: 8 r30: 10 r31: 9

783 Registers used: 23 out of 35 (65.7%)

784

785 "ATmega324A" instruction use summary:

786 .lds : 0 .sts : 0 adc : 5 add : 1 adiw : 2 and : 0

787 andi : 5 asr : 0 bclr : 0 bld : 0 brbc : 0 brbs : 0

788 brcc : 1 brcs : 0 break : 0 breq : 4 brge : 4 brhc : 0

789 brhs : 0 brid : 0 brie : 0 brlo : 0 brlt : 4 brmi : 0

790 brne : 10 brpl : 0 brsh : 0 brtc : 0 brts : 0 brvc : 0

791 brvs : 0 bset : 0 bst : 0 call : 8 cbi : 11 cbr : 0

792 clc : 1 clh : 0 cli : 1 cln : 0 clr : 7 cls : 0

793 clt : 0 clv : 0 clz : 0 com : 0 cp : 0 cpc : 0

794 cpi : 10 cpse : 0 dec : 9 eor : 0 fmul : 0 fmuls : 0

795 fmulsu: 0 icall : 0 ijmp : 0 in : 9 inc : 0 jmp : 0

796 ld : 5 ldd : 0 ldi : 88 lds : 4 lpm : 2 lsl : 1

797 lsr : 0 mov : 11 movw : 0 mul : 2 muls : 0 mulsu : 0

798 neg : 0 nop : 2 or : 0 ori : 10 out : 10 pop : 8

799 push : 8 rcall : 43 ret : 19 reti : 1 rjmp : 19 rol : 12

800 ror : 0 sbc : 3 sbci : 0 sbi : 10 sbic : 2 sbis : 2

801 sbiw : 0 sbr : 0 sbrc : 2 sbrr : 2 sec : 1 seh : 0

802 sei : 1 sen : 0 ser : 0 ses : 0 set : 0 sev : 0

803 sez : 0 sleep : 0 spm : 0 st : 4 std : 30 sts : 13

804 sub : 2 subi : 2 swap : 2 tst : 0 wdr : 0

805 Instructions used: 48 out of 113 (42.5%)

806

807 "ATmega324A" memory use summary [bytes]:

808 Segment Begin End Code Data Used Size Use%

809 -----

810	[.cseg]	0x000000	0x000380	876	18	894	32768	2.7%
811	[.dseg]	0x000100	0x000130	0	48	48	2048	2.3%
812	[.eseg]	0x000000	0x000000	0	0	0	1024	0.0%
813								
814	Assembly complete, 0 errors, 10 warnings							
815								

```

1
2 AVRASM ver. 2.2.7 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm Tue Dec 04 20:38:20 2018
3
4 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(44): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avram\inc\m324adef.inc'
5 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(63): Including file 'C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\lcd_dog_asm_driver_m324a.inc'
6 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(684): warning: Register r13 already defined by the .DEF directive
7 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(685): warning: Register r14 already defined by the .DEF directive
8 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(686): warning: Register r15 already defined by the .DEF directive
9 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(688): warning: Register r18 already defined by the .DEF directive
10 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(689): warning: Register r19 already defined by the .DEF directive
11 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(693): warning: Register r18 already defined by the .DEF directive
12 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(694): warning: Register r19 already defined by the .DEF directive
13 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(695): warning: Register r20 already defined by the .DEF directive
14 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(696): warning: Register r21 already defined by the .DEF directive
15 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(703): warning: Register r17 already defined by the .DEF directive
16 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(44): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avram\inc\m324adef.inc'
17 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(63): Including file 'C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\lcd_dog_asm_driver_m324a.inc'
18
19
20 ;*****
21 ;lab_10_sample_interval.asm
22 ;
23 ;Author : Roni Das ID: 108378223
24 ; Asif Iqbal ID: 110333685
25 ;Created: 12/04/2018 8:24:08 PM

```

```

26      ;
27      ; Description: The following program displays the measurement
28      ; of the direct frequency of a periodic waveform and
29      ; displays it on a LCD. The longer the period is
30      ; the more precise the measurements are. A 16 x 3 LCD is used
31      ; for this experiment. The first line illustrates
32      ; the frequency in Hz. The second line represents the decimal
33      ; count of the period. The program below generates a period
34      ; of approximately 1s;
35      ;Inputs:
36      ; PORTC = PC0
37      ;
38      ;Output:
39      ; PORTB = PB0 - PB7
40      ;
41      ;Register Assignments/Purposes
42      ; r16 = general purpose
43      ; r19 = loop counter
44      ; r21 = period counter
45      ; r22 = digit 0
46      ; r23 = digit 1
47      ; r24 = digit 2
48      ; r13 = digit 0 and 1
49      ; r14 = digit 2 and 4
50      ; r25 = conversion purpose
51      ; r26 = conversion purpose
52      ; r27 = conversion purpose
53
54      ;*Courtesy of Professor Ken Short and his lecture notes.
55      ;
56      ; Section no.: L02
57      ; Experiment no.: 10
58      ; Bench no.: 02
59      ;*****
60
61      .list
62
63
64
65      .CSEG
66
67
68      ; interrupt vector table, with several 'safety' stubs
69      .org 0      ;Reset/Cold start vector
70      000000 c091      rjmp start
71      .org INT0addr
72      000002 c182      rjmp start_counter_timer
73
74
75
76
77      .list
78
79      ;*****
80      ;NAME:      clr_dsp_buffs
81      ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3

```

```

82 ; with blanks (0x20)
83 ;ASSUMES: Three CONTIGUOUS 16-byte dram based
84 ; buffers named
85 ; dsp_buff_1, dsp_buff_2, dsp_buff_3.
86 ;RETURNS: nothing.
87 ;MODIFIES: r25,r26, Z-ptr
88 ;CALLS: none
89 ;CALLED BY: main application and diagnostics
90 ;*****
91
92
93 clr_dsp_buffs:
94 00007b e390 ldi R25, 48 ; load total length of
95 ;both buffer.
96 00007c e2a0 ldi R26, ' ' ; load blank/space into
97 ;R26.
98 00007d e0f1 ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a
99 ;pointer to 1st
100 00007e e0e0 ldi ZL, low (dsp_buff_1) ; byte of buffer for
101 ;line 1.
102
103 ;set DDRAM address to 1st
104 ;position of first line.
105 store_bytes:
106 00007f 93a1 st Z+, R26 ; store ' ' into 1st/next
107 ; buffer byte and
108 ; auto inc ptr to next
109 ;location.
110 000080 959a dec R25 ;
111 000081 f7e9 brne store_bytes ; cont until r25=0, all
112 ;bytes written.
113 000082 9508 ret
114
115
116
117 ;*****
118 ;NAME: load_msg
119 ;FUNCTION: Loads a predefined string msg into
120 ; a specified diplay
121 ; buffer.
122 ;ASSUMES: Z = offset of message to be loaded.
123 ;Msg format is
124 ; defined below.
125 ;RETURNS: nothing.
126 ;MODIFIES: r16, Y, Z
127 ;CALLS: nothing
128 ;CALLED BY:
129 ;*****
130 ; Message structure:
131 ; label: .db <buff num>, <text string/message>
132 ; , <end of string>
133 ;
134 ; Message examples (also see Messages
135 ;at the end of this file/module):
136 ; msg_1: .db 1,"First Message ", 0
137 ; loads msg into buff 1, eom=0

```

```

138 ; msg_2: .db 1,"Another message ", 0
139 ; loads msg into buff 1, eom=0
140 ;
141 ; Notes:
142 ; a) The 1st number indicates which
143 ; buffer to load (either 1, 2, or 3).
144 ; b) The last number (zero) is an '
145 ;end of string' indicator.
146 ; c) Y = ptr to disp_buffer
147 ; Z = ptr to message
148 ; (passed to subroutine)
149 ;*****
150 load_msg:
151 000083 e0d1 ldi YH, high (dsp_buff_1) ; Load YH and YL
152 ;as a pointer to 1st
153 000084 e0c0 ldi YL, low (dsp_buff_1) ; byte of dsp_buff_1
154 ;(Note - assuming
155 ; (dsp_buff_1 for now).
156 000085 9105 lpm R16, Z+ ; get dsply buff number
157 ;(1st byte of msg).
158 000086 3001 cpi r16, 1 ; if equal to '1', ptr
159 ; already setup.
160 000087 f021 breq get_msg_byte ; jump and start message
161 ; load.
162 000088 9660 adiw YH:YL, 16 ; else set ptr to dsp
163 ;buff 2.
164 000089 3002 cpi r16, 2 ; if equal to '2', ptr
165 ;now setup.
166 00008a f009 breq get_msg_byte ; jump and start message
167 ;load.
168 00008b 9660 adiw YH:YL, 16 ; else set ptr to dsp
169 ;buff 2.
170
171 get_msg_byte:
172 00008c 9105 lpm R16, Z+ ; get next byte of msg
173 ;and see if '0'.
174 00008d 3000 cpi R16, 0 ; if equal to '0', end
175 ;of message reached.
176 00008e f011 breq msg_loaded ; jump and stop message
177 ; loading operation.
178 00008f 9309 st Y+, R16 ; else, store next byte
179 ;of msg in buffer.
180 000090 cffb rjmp get_msg_byte ; jump back and continue.
181 msg_loaded:
182 000091 9508 ret
183
184 start:
185 000092 ef0f ldi r16, low(RAMEND) ; init stack/pointer
186 000093 bf0d out SPL, r16
187 000094 e008 ldi r16, high(RAMEND)
188 000095 bf0e out SPH, r16
189
190
191 000096 ef0f ldi r16, 0xff ; set portB = output.
192 000097 b904 out DDRB, r16 ;
193 000098 9a2c sbi portB, 4 ;set /SS of DOG LCD =

```

```

194                                     ; 1 (Deselected)
195
196 000099 9852          cbi DDRD, 2
197 00009a 985a          cbi portD, 2
198
199 00009b df9c          rcall init_lcd_dog          ; init display, using
200                                     ; SPI serial interface
201 00009c dfde          rcall clr_dsp_buffs         ; clear all three
202                                     ;lines
203
204
205                                     ;Configure port A bits 0-4 as an output
206 00009d e10c          ldi r16, 0b00011100        ;load r16 with 1s in
207                                     ; 0-4 postion
208                                     ;bit 0 and 1 position
209 00009e b901          out DDRA, r16               ;port A - bit 0 - 4 as
210                                     ;an output
211                                     ;Port A - bit 5-7 is
212                                     ; Input
213 00009f e100          ldi r16, 0b00010000        ;load r16 with all 1s a
214
215 0000a0 b902          out PORTA, r16              ;trun On LED: Color: RED
216
217
218                                     ;Configure port C bits 0 as an input pin
219 0000a1 e000          ldi r16,$00                 ;load r16 with all 0s
220 0000a2 b907          out DDRC,r16                ;port C-bit 0 = input
221
222
223 0000a3 ef5f          ldi r21, $FF                ;Initilalize Display = 00
224 0000a4 e020          ldi r18, $00                ;counter +/- 2% or +/- 5%
225
226
227                                     ;power on self test
228
229
230 0000a5 e003          ldi r16, (1 << ISC01) | (1 << ISC00)
231 0000a6 9300 0069     sts EICRA, r16
232 0000a8 e001          ldi r16, 1 << INT0
233 0000a9 bb0d          out EIMSK, r16
234
235
236 0000aa e000          ldi r16,$00
237 0000ab 9300 0085     sts TCNT1H,r16
238 0000ad 9300 0084     sts TCNT1L,r16
239 0000af 9300 0080     sts TCCR1A,r16
240
241
242                                     main_loop:
243 0000b1 9478          sei
244                                     ;   sts TCCR1B,r16          ;clock stopped*/
245
246
247
248 0000b2 3535          cpi r19,$55
249 0000b3 f009          breq continue

```

```

250 0000b4 cffc          rjmp main_loop
251
252          continue:
253 0000b5 94f8          cli
254
255 0000b6 e030          ldi r19,$00
256 0000b7 9100 0084     lds r16, TCNT1L
257 0000b9 9110 0085     lds r17, TCNT1H
258 0000bb 2f21          mov r18,r17
259
260 0000bc e040          ldi r20,$00
261 0000bd 9340 0085     sts TCNT1H,r20
262 0000bf 9340 0084     sts TCNT1L,r20
263 0000c1 9340 0080     sts TCCR1A,r20
264 0000c3 9340 0081     sts TCCR1B,r20
265
266 0000c5 940e 0132     call setup_display_one
267
268          ;load_line_1 into dbuff1:
269 0000c7 e0f2          ldi ZH, high(line1_testmessage<<1)
270 0000c8 e5e2          ldi ZL, low(line1_testmessage<<1)
271
272 0000c9 940e 0056     call update_lcd_dog
273 0000cb 940e 0083     call load_msg      ;load message into buffer(s).
274
275
276 0000cd 3622          cpi r18,98                ;check lower bound 2%
277 0000ce f424          brge check_Upper_bound_2P      ;branch if greater or equal
278 0000cf 352f          cpi r18,95                ;check lower bound 5%
279 0000d0 f444          brge check_Upper_bound_5P      ;branch if greater or equal
280 0000d1 f0fc          brlt lower_than_5P          ;else count < 95
281 0000d2 cfde          rjmp main_loop
282
283          check_Upper_bound_2P:
284 0000d3 3627          cpi r18,103
285 0000d4 f044          brlt display_bargraph_2P; branch if lower
286 0000d5 362a          cpi r18,106
287 0000d6 f084          brlt display_bargraph_5P;branch if lower
288 0000d7 f4cc          brge higher_than_5P          ;branch if higher
289 0000d8 cfd8          rjmp main_loop
290
291          check_Upper_bound_5P:
292 0000d9 362a          cpi r18,106
293 0000da f4b4          brge higher_than_5P          ;branch if higher
294 0000db f05c          brlt display_bargraph_5P;branch if lower
295 0000dc cfd4          rjmp main_loop
296
297          display_bargraph_2P:
298          ;call display_g_LCD
299
300          ;load_line_1 into dbuff1:
301          ;ldi ZH, high(line1_testmessage<<1)
302          ;ldi ZL, low(line1_testmessage<<1)
303          ;rcall load_msg      ;load message into buffer(s).
304
305 0000dd 940e 00fb     call tol_2

```



```

306 0000df e0f2      ldi  ZH, high(line3_testmessage<<1)
307 0000e0 e6e2      ldi  ZL, low(line3_testmessage<<1)
308 0000e1 dfa1      rcall load_msg      ;load message into buffer(s).
309
310
311 0000e2 df73      rcall update_lcd_dog
312
313
314 0000e3 9a12      sbi  PORTA, 2      ;turn on green
315 0000e4 9813      cbi  PORTA, 3      ; turn off blue
316 0000e5 9814      cbi  PORTA, 4      ;turn off red
317 0000e6 cfca      rjmp main_loop
318
319      display_bargraph_5P:
320
321      ;call display_g_LCD
322
323      ;load_line_1 into dbuff1:
324      ; ldi  ZH, high(line1_testmessage<<1)
325      ; ldi  ZL, low(line1_testmessage<<1)
326      ; rcall load_msg      ;load message into buffer(s).
327
328 0000e7 940e 0108      call tol_5
329 0000e9 e0f2      ldi  ZH, high(line3_testmessage<<1)
330 0000ea e6e2      ldi  ZL, low(line3_testmessage<<1)
331 0000eb df97      rcall load_msg      ;load message into buffer(s).
332
333
334 0000ec df69      rcall update_lcd_dog
335
336
337 0000ed 9812      cbi  PORTA, 2      ;turn off green
338 0000ee 9a13      sbi  PORTA, 3      ;turn on blue
339 0000ef 9814      cbi  PORTA, 4      ;turn off red
340 0000f0 cfc0      rjmp main_loop
341
342      lower_than_5P:
343      higher_than_5P:
344
345      ;call display_g_LCD
346      ;load_line_1 into dbuff1:
347      ;ldi  ZH, high(line1_testmessage<<1)
348      ;ldi  ZL, low(line1_testmessage<<1)
349      ;rcall load_msg      ;load message into buffer(s).
350
351 0000f1 940e 0115      call tol_ORR
352 0000f3 e0f2      ldi  ZH, high(line3_testmessage<<1)
353 0000f4 e6e2      ldi  ZL, low(line3_testmessage<<1)
354 0000f5 df8d      rcall load_msg      ;load message into buffer(s).
355
356
357 0000f6 df5f      rcall update_lcd_dog
358
359 0000f7 9812      cbi  PORTA, 2      ;turn off green
360 0000f8 9813      cbi  PORTA, 3      ;turn on blue
361 0000f9 9a14      sbi  PORTA, 4      ;turn on red

```

```

362
363 0000fa cfb6          rjmp main_loop
364
365
366
367 ;*****
368 ;*
369 ;* "Subroutine_name" - Tolerance
370 ;*
371 ;* Description: This subroutine does the ascii conversion
372 ;*              which is to be displayed in the LCD. This is
373 ;*              displayed in the 3rd line. Values from the
374 ;*              character set was loaded into r26 and r27
375 ;*              and then r25 was loaded with decimal value.
376 ;*              To get the ascii value , r25 was or'ed with
377 ;*              48 for the conversion (since 48 is 0 in ascii).
378 ;*              Using the Y pointer, each digit is to be
379 ;*              displayed in a specific position.
380 ;*
381 ;*
382 ;* Author: Asif Iqbal
383 ;*        Roni Das
384 ;* Version:1A
385 ;* Last updated: 11/06/2018
386 ;* Target: Perfect
387 ;* Number of words: 10
388 ;* Number of cycles: 300/307 (Min/Max)
389 ;* Low registers modified: none
390 ;* High registers modified: 3
391 ;*
392 ;* Parameters:
393 ;*
394 ;* Returns:
395 ;*
396 ;* Notes:
397 ;*
398 ;*****
399
400
401 tol_2:
402 0000fb e0d1          ldi YH,HIGH(dsp_buff_3)      ;displaying on line 3
403 0000fc e2c0          ldi YL,LOW(dsp_buff_3)
404 0000fd e092          ldi r25,$02              ;loading r25 with hex 02
405 0000fe e2a5          ldi r26, 0b00100101      ;loading r26 with 0b00100101
406 0000ff e2b0          ldi r27, $20              ;loading r27 with hex 20
407 000100 6390          ori r25, 48              ;or immediately to get the ascii
408 000101 839f          std y+7, r25              ;displaying at the 7th position
409 000102 87a8          std y+8,r26              ;displaying at the 8th position
410 000103 87b9          std y+9,r27              ;displaying at the the position
411 000104 e200          ldi r16, ' '
412 000105 870a          std y+10,r16
413 000106 870c          std y+12,r16
414 000107 9508          ret
415 tol_5:
416 000108 e0d1          ldi YH,HIGH(dsp_buff_3)
417 000109 e2c0          ldi YL,LOW(dsp_buff_3)

```

```

418 00010a e095      ldi r25,$05
419 00010b e2a5      ldi r26, 0b00100101
420 00010c e2b0      ldi r27, $20
421 00010d 6390      ori r25, 48
422 00010e 839f      std y+7, r25
423 00010f 87a8      std y+8,r26
424 000110 87b9      std y+9,r27
425 000111 e200      ldi r16,' '
426 000112 870a      std y+10,r16
427 000113 870c      std y+12,r16
428
429 000114 9508      ret
430
431 tol_ORR:
432 000115 e0d1      ldi YH,HIGH(dsp_buff_3)
433 000116 e2c0      ldi YL,LOW(dsp_buff_3)
434 000117 e49f      ldi r25,$4F
435 000118 e5a2      ldi r26,$52
436 000119 839f      std y+7, r25
437 00011a 8798      std y+8, r25
438 00011b 87a9      std y+9,r26
439
440 00011c e200      ldi r16,' '
441 00011d 870a      std y+10,r16
442 00011e 870c      std y+12,r16
443
444 00011f 9508      ret
445 ;converting binary to ascii
446 convert_ascii_2:
447 000120 e0d1      ldi YH,HIGH(dsp_buff_2) ;displaying at line 2
448 000121 e1c0      ldi YL,LOW(dsp_buff_2)
449 000122 6360      ori r22, 48 ;ori toconvert register content to ascii
450 000123 6370      ori r23, 48
451 000124 6380      ori r24, 48
452 000125 838f      std y+7, r24;displaying it to the 7th position in lcd
453 000126 8778      std y+8, r23
454 000127 8769      std y+9, r22
455 000128 9508      ret
456
457
458 ;*****
459 ;***** ALL MESSAGES: Fixed format, flash stored/loaded *
460 ;*****
461
462
463 000129 4601
464 00012a 7172
465 00012b 3d20
466 00012c 0020      line1_testmessage: .db 1, "Frq = ", 0; message for line #1.
467 00012d 5002
468 00012e 6472
469 00012f 3d20
470 000130 0020      line2_testmessage: .db 2, "Prd = ", 0; message for line #2.
471 000131 0003      line3_testmessage: .db 3, "", 0; message for line #3.
472
473

```

```

474
475
476             setup_display_one:
477             ;clr r17
478 000132 940e 0168    call bin2BCD16           ;output r13,r14,15
479 000134 2d6d        mov r22, r13           ;new
480 000135 2d7d        mov r23, r13           ;new
481 000136 706f        andi r22, $0F          ;digit 0 value
482 000137 7f70        andi r23, $F0
483 000138 9572        swap r23              ;digit 1 value
484 000139 2d8e        mov r24, r14
485 00013a 2d9e        mov r25, r14
486 00013b 708f        andi r24, $0F          ;digit 2 value
487 00013c 7f90        andi r25, $F0
488 00013d 9592        swap r25
489 00013e 2daf        mov r26, r15
490 00013f 70af        andi r26, $0F
491
492
493
494 000140 2f2a        mov r18, r26
495 000141 e604        ldi r16,100
496 000142 9f20        mul r18,r16
497 000143 2d20        mov r18,r0
498
499
500 000144 2f49        mov r20, r25
501 000145 e00a        ldi r16,10
502 000146 9f40        mul r20,r16
503
504 000147 1d20        adc r18,r0
505 000148 2f48        mov r20,r24
506 000149 1f24        adc r18,r20
507
508
509             ;mov r25, r13
510             ;andi r25, $0F
511
512             ;converting binary to ascii
513             convert_ascii:
514             ;ldi r19, 9 ; looping for 8 bits
515 00014a e0d1        ldi YH,HIGH(dsp_buff_1)
516 00014b e0c0        ldi YL,LOW(dsp_buff_1)
517             ;adiw YH:YL, 7
518 00014c 6360        ori r22, 48
519 00014d 6370        ori r23, 48
520 00014e 6380        ori r24, 48
521 00014f 6390        ori r25, 48
522 000150 63a0        ori r26, 48
523
524
525 000151 940e 0159    call msg
526
527 000153 83af        std y+7, r26
528 000154 8798        std y+8, r25
529 000155 878a        std y+10, r24

```

```

530 000156 877b      std y+11, r23
531 000157 876c      std y+12, r22
532 000158 9508      ret
533
534
535      msg:
536
537 000159 e500      ldi r16,'P'
538 00015a 8308      std y+0,r16
539 00015b e502      ldi r16,'R'
540 00015c 8309      std y+1,r16
541 00015d e404      ldi r16,'D'
542 00015e 830a      std y+2,r16
543 00015f e30d      ldi r16,'='
544 000160 830c      std y+4,r16
545 000161 e20e      ldi r16,'.'
546 000162 8709      std y+9,r16
547 000163 e60d      ldi r16,'m'
548 000164 870e      std y+14,r16
549 000165 e703      ldi r16,'s'
550 000166 870f      std y+15,r16
551 000167 9508      ret
552      ;***** END OF FILE *****
553
554      ;*****
555      ;*
556      ;* "bin2BCD16" - 16-bit Binary to BCD conversion
557      ;*
558      ;* This subroutine converts a 16-bit number (fbinH:fbinL) to
559      ;* a 5-digit
560      ;* packed BCD number represented by 3 bytes (tBCD2:tBCD1:tBCD0).
561      ;* MSD of the 5-digit number is placed in the lowermost
562      ;* nibble of tBCD2.
563      ;*
564      ;* Number of words :25
565      ;* Number of cycles:751/768 (Min/Max)
566      ;* Low registers used :3 (tBCD0,tBCD1,tBCD2)
567      ;* High registers used :4(fbinL,fbinH,cnt16a,tmp16a)
568      ;* Pointers used :Z
569      ;*
570      ;*****
571
572      ;***** Subroutine Register Variables
573
574      .equ    AtBCD0    =13      ;address of tBCD0
575      .equ    AtBCD2    =15      ;address of tBCD1
576
577      .def     tBCD0     =r13      ;BCD value digits 1 and 0
578      .def     tBCD1     =r14      ;BCD value digits 3 and 2
579      .def     tBCD2     =r15      ;BCD value digit 4
580      .def     fbinL     =r16      ;binary value Low byte
581      .def     fbinH     =r17      ;binary value High byte
582      .def     cnt16a    =r18      ;loop counter
583      .def     tmp16a    =r19      ;temporary value
584
585      ;***** Code

```

```

586
587 bin2BCD16:
588 000168 930f    push r16
589 000169 931f    push r17
590 00016a e120    ldi cnt16a,16    ;Init loop counter
591 00016b 24ff    clr tBCD2        ;clear result (3 bytes)
592 00016c 24ee    clr tBCD1
593 00016d 24dd    clr tBCD0
594 00016e 27ff    clr ZH          ;clear ZH (not needed for AT90Sxx0x)
595 00016f 0f00    bBCDx_1:lsfbinL    ;shift input value
596 000170 1f11    rol fbinH        ;through all bytes
597 000171 1cdd    rol tBCD0        ;
598 000172 1cee    rol tBCD1
599 000173 1cff    rol tBCD2
600 000174 952a    dec cnt16a        ;decrement loop counter
601 000175 f419    brnebBCDx_2      ;if counter not zero
602 000176 911f    pop r17
603 000177 910f    pop r16
604 000178 9508    ret              ; return
605
606 000179 e1e0    bBCDx_2:ldir30,AtBCD2+1;Z points to result MSB + 1
607 bBCDx_3:
608 00017a 9132    ld tmp16a,-Z     ;get (Z) with pre-decrement
609
610 ;-----
611 ;For AT90Sxx0x, substitute the above line with:
612 ;
613 ; dec ZL
614 ; ld tmp16a,Z
615 ;
616 ;-----
616 00017b 5f3d    subitmp16a,-$03 ;add 0x03
617 00017c fd33    sbrctmp16a,3;if bit 3 not clear
618 00017d 8330    st Z,tmp16a; store back
619 00017e 8130    ld tmp16a,Z;get (Z)
620 00017f 5d30    subitmp16a,-$30 ;add 0x30
621 000180 fd37    sbrctmp16a,7;if bit 7 not clear
622 000181 8330    st Z,tmp16a; store back
623 000182 30ed    cpi ZL,AtBCD0    ;done all three?
624 000183 f7b1    brnebBCDx_3      ;loop again if not
625 000184 cfea    rjmp bBCDx_1
626
627
628
629
630
631 start_counter_timer:
632
633 000185 e000    ldi r16,$00
634 000186 9300 0085    sts TCNT1H,r16
635 000188 9300 0084    sts TCNT1L,r16
636 00018a 9300 0080    sts TCCR1A,r16
637 00018c e001    ldi r16,$01
638
639
640 00018d e020    ldi r18, $00;counter +/- 2% or +/- 5
641 lb_0:

```

```

642 00018e 994a          sbic PIND,2          ;Check for a clear bit @ PINC0
643 00018f cffe          rjmp lb_0
644
645          lb_1:
646 000190 9b4a          sbis PIND,2          ;Check for a set bit @ PINC0
647 000191 cffe          rjmp lb_1
648          lb_2:
649
650 000192 9300 0081      sts TCCR1B,r16      ;start counting
651
652          ;call var_delay
653          ;inc r18
654 000194 994a          sbic PIND,2          ;Check for a clear bit @ PINC0
655 000195 cffc          rjmp lb_2
656          lb_3:
657          ;call var_delay
658          ;inc r18
659 000196 9b4a          sbis PIND,2          ;Check for a set bit @ PINC0
660 000197 cffe          rjmp lb_3
661
662 000198 e000          ldi r16,$00
663 000199 9300 0081      sts TCCR1B,r16          ;clock stopped
664
665 00019b 9100 0084      lds r16, TCNT1L
666 00019d 9110 0085      lds r17, TCNT1H
667 00019f 2f21          mov r18,r17
668 0001a0 e535          ldi r19,$55
669
670 0001a1 9518          reti
671
672          DIVISION:
673
674
675          ;*****
676          ;*
677          ;* "div32u" - 32/32 Bit Unsigned Division
678          ;*
679          ;* Ken Short
680          ;*
681          ;* This subroutine divides the two 32-bit numbers
682          ;* "dd32u3:dd32u2:dd32u1:dd32u0" (dividend) and "
683          ;* dv32u3:dv32u2:dv32u1:dv32u0"
684          ;* (divisor).
685          ;* The result is placed in "dres32u3:dres32u2
686          ;* dres32u1:dres32u0" and the
687          ;* remainder in "drem32u3:drem32u2:drem32u1:drem32u0"
688          ;*
689          ;* Number of words :
690          ;* Number of cycles:655/751 (Min/Max) ATmega16
691          ;* #Low registers used :2 (drem16uL,drem16uH)
692          ;* #High registers used :5
693          ;* (dres16uL/dd16uL,dres16uH/dd16uH,dv16uL,dv16uH,
694          ;*          dcnt16u)
695          ;* A $0000 divisor returns $FFFF
696          ;*
697          ;*****

```

```

698
699             ;***** Subroutine Register Variables
700
701             .def    drem32u0=r12      ;remainder
702             .def    drem32u1=r13
703             .def    drem32u2=r14
704             .def    drem32u3=r15
705
706             .def    dres32u0=r18      ;result (quotient)
707             .def    dres32u1=r19
708             .def    dres32u2=r20
709             .def    dres32u3=r21
710
711             .def    dd32u0  =r18      ;dividend
712             .def    dd32u1  =r19
713             .def    dd32u2  =r20
714             .def    dd32u3  =r21
715
716             .def    dv32u0  =r22      ;divisor
717             .def    dv32u1  =r23
718             .def    dv32u2  =r24
719             .def    dv32u3  =r25
720
721             .def    dcnt32u =r17
722
723             ;***** Code
724
725             div32u:
726             0001a2 24cc      clr drem32u0;clear remainder Low byte
727             0001a3 24dd      clr drem32u1
728             0001a4 24ee      clr drem32u2
729             0001a5 18ff      sub drem32u3,drem32u3;clear remainder High byte
730             0001a6 e211      ldi dcnt32u,33 ;init loop counter
731             d32u_1:
732             0001a7 1f22      rol dd32u0      ;shift left dividend
733             0001a8 1f33      rol dd32u1
734             0001a9 1f44      rol dd32u2
735             0001aa 1f55      rol dd32u3
736             0001ab 951a      dec dcnt32u      ;decrement counter
737             0001ac f409      brned32u_2      ;if done
738             0001ad 9508      ret              ; return
739             d32u_2:
740             0001ae 1ccc      rol drem32u0;shift dividend into remainder
741             0001af 1cdd      roldrem32u1
742             0001b0 1cee      roldrem32u2
743             0001b1 1cff      rol drem32u3
744
745             0001b2 1ac6      sub drem32u0,dv32u0 ;remainder = remainder - divisor
746             0001b3 0ad7      sbcdrem32u1,dv32u1
747             0001b4 0ae8      sbcdrem32u2,dv32u2
748             0001b5 0af9      sbc drem32u3,dv32u3 ;
749             0001b6 f430      brccd32u_3      ; branch if reult is pos or zero
750
751             0001b7 0ec6      add drem32u0,dv32u0 ;if result negative
752             0001b8 1ed7      adc drem32u1,dv32u1
753             0001b9 1ee8      adc drem32u2,dv32u2

```



```

754 0001ba 1ef9          adc drem32u3,dv32u3
755 0001bb 9488          clc          ; clear carry to be shifted into result
756 0001bc cfea          rjmpd32u_1      ;else
757 0001bd 9408          d32u_3:sec      ; set carry to be shifted into result
758 0001be cfe8          rjmpd32u_1

```

759

760

761

762 RESOURCE USE INFORMATION

763 -----

764

765 Notice:

766 The register and instruction counts are symbol table hit counts,

767 and hence implicitly used resources are not counted, eg, the

768 'lpm' instruction without operands implicitly uses r0 and z,

769 none of which are counted.

770

771 x,y,z are separate entities in the symbol table and are

772 counted separately from r26..r31 here.

773

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

775

776 "ATmega324A" register use summary:

777 x : 0 y : 31 z : 10 r0 : 2 r1 : 0 r2 : 0 r3 : 0 r4 : 0

778 r5 : 0 r6 : 0 r7 : 0 r8 : 0 r9 : 0 r10: 0 r11: 0 r12: 4

779 r13: 8 r14: 8 r15: 8 r16: 10 r17: 10 r18: 17 r19: 12 r20: 18

780 r21: 2 r22: 10 r23: 11 r24: 13 r25: 19 r26: 13 r27: 4 r28: 8

781 r29: 8 r30: 10 r31: 9

782 Registers used: 23 out of 35 (65.7%)

783

784 "ATmega324A" instruction use summary:

785 .lds : 0 .sts : 0 adc : 5 add : 1 adiw : 2 and : 0

786 andi : 5 asr : 0 bclr : 0 bld : 0 brbc : 0 brbs : 0

787 brcc : 1 brcs : 0 break : 0 breq : 4 brge : 4 brhc : 0

788 brhs : 0 brid : 0 brie : 0 brlo : 0 brlt : 4 brmi : 0

789 brne : 10 brpl : 0 brsh : 0 brtc : 0 brts : 0 brvc : 0

790 brvs : 0 bset : 0 bst : 0 call : 8 cbi : 11 cbr : 0

791 clc : 1 clh : 0 cli : 1 cln : 0 clr : 7 cls : 0

792 clt : 0 clv : 0 clz : 0 com : 0 cp : 0 cpc : 0

793 cpi : 10 cpse : 0 dec : 9 eor : 0 fmul : 0 fmulu : 0

794 fmulu: 0 icall : 0 ijmp : 0 in : 9 inc : 0 jmp : 0

795 ld : 5 ldd : 0 ldi : 88 lds : 4 lpm : 2 lsl : 1

796 lsr : 0 mov : 11 movw : 0 mul : 2 muls : 0 mulu : 0

797 neg : 0 nop : 2 or : 0 ori : 10 out : 10 pop : 8

798 push : 8 rcall : 43 ret : 19 reti : 1 rjmp : 19 rol : 12

799 ror : 0 sbc : 3 sbci : 0 sbi : 10 sbic : 2 sbis : 2

800 sbiw : 0 sbr : 0 sbrc : 2 sbrs : 2 sec : 1 seh : 0

801 sei : 1 sen : 0 ser : 0 ses : 0 set : 0 sev : 0

802 sez : 0 sleep : 0 spm : 0 st : 4 std : 30 sts : 13

803 sub : 2 subi : 2 swap : 2 tst : 0 wdr : 0

804 Instructions used: 48 out of 113 (42.5%)

805

806 "ATmega324A" memory use summary [bytes]:

807 Segment Begin End Code Data Used Size Use%

808 -----

809 [.cseg] 0x000000 0x000380 876 18 894 32768 2.7%

810 [.dseg] 0x000100 0x000130 0 48 48 2048 2.3%

811 [.eseg] 0x000000 0x000000 0 0 0 1024 0.0%

812

813 Assembly complete, 0 errors, 10 warnings

814

```

1
2 AVRASM ver. 2.2.7 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm Tue Dec 04 20:45:21 2018
3
4 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(49): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avrasm\inc\m324adef.inc'
5 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(68): Including file 'C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\lcd_dog_asm_driver_m324a.inc'
6 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(689): warning: Register r13 already defined by the .DEF directive
7 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(690): warning: Register r14 already defined by the .DEF directive
8 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(691): warning: Register r15 already defined by the .DEF directive
9 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(693): warning: Register r18 already defined by the .DEF directive
10 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(694): warning: Register r19 already defined by the .DEF directive
11 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(698): warning: Register r18 already defined by the .DEF directive
12 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(699): warning: Register r19 already defined by the .DEF directive
13 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(700): warning: Register r20 already defined by the .DEF directive
14 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(701): warning: Register r21 already defined by the .DEF directive
15 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(708): warning: Register r17 already defined by the .DEF directive
16 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(49): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avrasm\inc\m324adef.inc'
17 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(68): Including file 'C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\lcd_dog_asm_driver_m324a.inc'
18
19
20 ;*****
21 ;lab_10_direct_freq_meas.asm
22 ;
23 ;Author : Roni Das ID: 108378223
24 ; Asif Iqbal ID: 110333685
25 ;Created: 12/04/2018 8:24:08 PM

```

```

26      ;
27      ; Description: The following program displays the measurement
28      ; of the direct frequency of a periodic waveform and
29      ; displays it on a LCD. The longer the period is
30      ; the more precise the measurements are. A 16 x 3 LCD is used
31      ; for this experiment. The first line illustrates
32      ; the frequency in Hz. The second line represents the decimal
33      ; count of the period. The program below generates a period
34      ; of approximately 1s. The frequency is displayed in the unit
35      ; of 1hz and period is displayed in the unit of 1us. The signal
36      ; that is to be measure is 1khz to 20khz. The period of the
37      ; signal rages from 1,000 us to 50,000 us. The timer/counter
38      ; prescalar was set to 1;
39      ;
40      ;Inputs:
41      ; PORTC = PC0
42      ;
43      ;Output:
44      ; PORTB = PB0 - PB7
45      ;
46      ;Register Assignments/Purposes
47      ; r16 = general purpose
48      ; r19 = loop counter
49      ; r21 = period counter
50      ; r22 = digit 0
51      ; r23 = digit 1
52      ; r24 = digit 2
53      ; r13 = digit 0 and 1
54      ; r14 = digit 2 and 4
55      ; r25 = conversion purpose
56      ; r26 = conversion purpose
57      ; r27 = conversion purpose
58
59      ;*Courtesy of Professor Ken Short and his lecture notes.
60      ;
61      ; Section no.: L02
62      ; Experiment no.: 10
63      ; Bench no.: 02
64      ;*****
65
66      .list
67
68
69
70      .CSEG
71
72
73      ; interrupt vector table, with several 'safety' stubs
74      .org 0      ;Reset/Cold start vector
75      000000 c091      rjmp start
76      .org INT0addr
77      000002 c182      rjmp start_counter_timer
78
79
80
81

```

```

82      .list
83
84      ;*****
85      ;NAME:      clr_dsp_buffs
86      ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3
87      ;           with blanks (0x20)
88      ;ASSUMES:   Three CONTIGUOUS 16-byte dram based
89      ;           buffers named
90      ;           dsp_buff_1, dsp_buff_2, dsp_buff_3.
91      ;RETURNS:   nothing.
92      ;MODIFIES:  r25,r26, Z-ptr
93      ;CALLS:     none
94      ;CALLED BY: main application and diagnostics
95      ;*****
96
97
98      clr_dsp_buffs:
99      00007b e390      ldi R25, 48          ; load total length of
100                                     ;both buffer.
101      00007c e2a0      ldi R26, ' '          ; load blank/space into
102                                     ;R26.
103      00007d e0f1      ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a
104                                     ;pointer to 1st
105      00007e e0e0      ldi ZL, low (dsp_buff_1)  ; byte of buffer for
106                                     ;line 1.
107
108      ;set DDRAM address to 1st
109      ;position of first line.
110      store_bytes:
111      00007f 93a1      st Z+, R26          ; store ' ' into 1st/next
112                                     ; buffer byte and
113                                     ; auto inc ptr to next
114                                     ;location.
115      000080 959a      dec R25              ;
116      000081 f7e9      brne store_bytes    ; cont until r25=0, all
117                                     ;bytes written.
118      000082 9508      ret
119
120
121
122      ;*****
123      ;NAME:      load_msg
124      ;FUNCTION:  Loads a predefined string msg into
125      ;           a specified diplay
126      ;           buffer.
127      ;ASSUMES:   Z = offset of message to be loaded.
128      ;           ;Msg format is
129      ;           defined below.
130      ;RETURNS:   nothing.
131      ;MODIFIES:  r16, Y, Z
132      ;CALLS:     nothing
133      ;CALLED BY:
134      ;*****
135      ; Message structure:
136      ;   label:  .db <buff num>, <text string/message>
137      ;           , <end of string>

```

```

138 ;
139 ; Message examples (also see Messages
140 ; at the end of this file/module):
141 ; msg_1: .db 1,"First Message ", 0
142 ; loads msg into buff 1, eom=0
143 ; msg_2: .db 1,"Another message ", 0
144 ; loads msg into buff 1, eom=0
145 ;
146 ; Notes:
147 ; a) The 1st number indicates which
148 ; buffer to load (either 1, 2, or 3).
149 ; b) The last number (zero) is an '
150 ; end of string' indicator.
151 ; c) Y = ptr to disp_buffer
152 ; Z = ptr to message
153 ; (passed to subroutine)
154 ;*****
155 load_msg:
156 000083 e0d1 ldi YH, high (dsp_buff_1) ; Load YH and YL
157 ; as a pointer to 1st
158 000084 e0c0 ldi YL, low (dsp_buff_1) ; byte of dsp_buff_1
159 ; (Note - assuming
160 ; (dsp_buff_1 for now).
161 000085 9105 lpm R16, Z+ ; get dsply buff number
162 ; (1st byte of msg).
163 000086 3001 cpi r16, 1 ; if equal to '1', ptr
164 ; already setup.
165 000087 f021 breq get_msg_byte ; jump and start message
166 ; load.
167 000088 9660 adiw YH:YL, 16 ; else set ptr to dsp
168 ; buff 2.
169 000089 3002 cpi r16, 2 ; if equal to '2', ptr
170 ; now setup.
171 00008a f009 breq get_msg_byte ; jump and start message
172 ; load.
173 00008b 9660 adiw YH:YL, 16 ; else set ptr to dsp
174 ; buff 2.
175
176 get_msg_byte:
177 00008c 9105 lpm R16, Z+ ; get next byte of msg
178 ; and see if '0'.
179 00008d 3000 cpi R16, 0 ; if equal to '0', end
180 ; of message reached.
181 00008e f011 breq msg_loaded ; jump and stop message
182 ; loading operation.
183 00008f 9309 st Y+, R16 ; else, store next byte
184 ; of msg in buffer.
185 000090 cffb rjmp get_msg_byte ; jump back and continue.
186 msg_loaded:
187 000091 9508 ret
188
189 start:
190 000092 ef0f ldi r16, low(RAMEND) ; init stack/pointer
191 000093 bf0d out SPL, r16
192 000094 e008 ldi r16, high(RAMEND)
193 000095 bf0e out SPH, r16

```

```

194
195
196 000096 ef0f          ldi r16, 0xff          ; set portB = output.
197 000097 b904          out DDRB, r16          ;
198 000098 9a2c          sbi portB, 4          ;set /SS of DOG LCD =
199                                ; 1 (Deselected)
200
201 000099 9852          cbi DDRD, 2
202 00009a 985a          cbi portD, 2
203
204 00009b df9c          rcall init_lcd_dog      ; init display, using
205                                ; SPI serial interface
206 00009c dfde          rcall clr_dsp_buffs    ; clear all three
207                                ;lines
208
209
210                                ;Configure port A bits 0-4 as an output
211 00009d e10c          ldi r16, 0b00011100    ;load r16 with 1s in
212                                ; 0-4 position
213                                ;bit 0 and 1 position
214 00009e b901          out DDRA, r16      ;port A - bit 0 - 4 as
215                                ;an output
216                                ;Port A - bit 5-7 is
217                                ; Input
218 00009f e100          ldi r16, 0b00010000    ;load r16 with all 1s a
219
220 0000a0 b902          out PORTA, r16      ;trun On LED: Color: RED
221
222
223                                ;Configure port C bits 0 as an input pin
224 0000a1 e000          ldi r16,$00          ;load r16 with all 0s
225 0000a2 b907          out DDRC,r16      ;port C-bit 0 = input
226
227
228 0000a3 ef5f          ldi r21, $FF          ;Initilalize Display = 00
229 0000a4 e020          ldi r18, $00          ;counter +/- 2% or +/- 5%
230
231
232                                ;power on self test
233
234
235 0000a5 e003          ldi r16, (1 << ISC01) | (1 << ISC00)
236 0000a6 9300 0069     sts EICRA, r16
237 0000a8 e001          ldi r16, 1 << INT0
238 0000a9 bb0d          out EIMSK, r16
239
240
241 0000aa e000          ldi r16,$00
242 0000ab 9300 0085     sts TCNT1H,r16
243 0000ad 9300 0084     sts TCNT1L,r16
244 0000af 9300 0080     sts TCCR1A,r16
245
246
247                                main_loop:
248 0000b1 9478          sei
249                                ; sts TCCR1B,r16          ;clock stopped*/

```

```

250
251
252
253 0000b2 3535          cpi r19,$55
254 0000b3 f009          breq continue
255 0000b4 cffc          rjmp main_loop
256
257          continue:
258 0000b5 94f8          cli
259
260 0000b6 e030          ldi r19,$00
261 0000b7 9100 0084      lds r16, TCNT1L
262 0000b9 9110 0085      lds r17, TCNT1H
263 0000bb 2f21          mov r18,r17
264
265 0000bc e040          ldi r20,$00
266 0000bd 9340 0085      sts TCNT1H,r20
267 0000bf 9340 0084      sts TCNT1L,r20
268 0000c1 9340 0080      sts TCCR1A,r20
269 0000c3 9340 0081      sts TCCR1B,r20
270
271 0000c5 940e 0132      call setup_display_one
272
273          ;load_line_1 into dbuff1:
274 0000c7 e0f2          ldi ZH, high(line1_testmessage<<1)
275 0000c8 e5e2          ldi ZL, low(line1_testmessage<<1)
276
277 0000c9 940e 0056      call update_lcd_dog
278 0000cb 940e 0083      call load_msg      ;load message into buffer(s).
279
280
281 0000cd 3622          cpi r18,98          ;check lower bound 2%
282 0000ce f424          brge check_Upper_bound_2P      ;branch if greater or equal
283 0000cf 352f          cpi r18,95          ;check lower bound 5%
284 0000d0 f444          brge check_Upper_bound_5P      ;branch if greater or equal
285 0000d1 f0fc          brlt lower_than_5P      ;else count < 95
286 0000d2 cfde          rjmp main_loop
287
288          check_Upper_bound_2P:
289 0000d3 3627          cpi r18,103
290 0000d4 f044          brlt display_bargraph_2P; branch if lower
291 0000d5 362a          cpi r18,106
292 0000d6 f084          brlt display_bargraph_5P;branch if lower
293 0000d7 f4cc          brge higher_than_5P      ;branch if higher
294 0000d8 cfd8          rjmp main_loop
295
296          check_Upper_bound_5P:
297 0000d9 362a          cpi r18,106
298 0000da f4b4          brge higher_than_5P      ;branch if higher
299 0000db f05c          brlt display_bargraph_5P;branch if lower
300 0000dc cfd4          rjmp main_loop
301
302          display_bargraph_2P:
303          ;call display_g_LCD
304
305          ;load_line_1 into dbuff1:

```



```

306 ;ldi ZH, high(line1_testmessage<<1)
307 ;ldi ZL, low(line1_testmessage<<1)
308 ;rcall load_msg ;load message into buffer(s).
309
310 0000dd 940e 00fb call tol_2
311 0000df e0f2 ldi ZH, high(line3_testmessage<<1)
312 0000e0 e6e2 ldi ZL, low(line3_testmessage<<1)
313 0000e1 dfa1 rcall load_msg ;load message into buffer(s).
314
315
316 0000e2 df73 rcall update_lcd_dog
317
318
319 0000e3 9a12 sbi PORTA, 2 ;turn on green
320 0000e4 9813 cbi PORTA, 3 ; turn off blue
321 0000e5 9814 cbi PORTA, 4 ;turn off red
322 0000e6 cfca rjmp main_loop
323
324 display_bargraph_5P:
325
326 ;call display_g_LCD
327
328 ;load_line_1 into dbuff1:
329 ; ldi ZH, high(line1_testmessage<<1)
330 ; ldi ZL, low(line1_testmessage<<1)
331 ; rcall load_msg ;load message into buffer(s).
332
333 0000e7 940e 0108 call tol_5
334 0000e9 e0f2 ldi ZH, high(line3_testmessage<<1)
335 0000ea e6e2 ldi ZL, low(line3_testmessage<<1)
336 0000eb df97 rcall load_msg ;load message into buffer(s).
337
338
339 0000ec df69 rcall update_lcd_dog
340
341
342 0000ed 9812 cbi PORTA, 2 ;turn off green
343 0000ee 9a13 sbi PORTA, 3 ;turn on blue
344 0000ef 9814 cbi PORTA, 4 ;turn off red
345 0000f0 cfc0 rjmp main_loop
346
347 lower_than_5P:
348 higher_than_5P:
349
350 ;call display_g_LCD
351 ;load_line_1 into dbuff1:
352 ;ldi ZH, high(line1_testmessage<<1)
353 ;ldi ZL, low(line1_testmessage<<1)
354 ;rcall load_msg ;load message into buffer(s).
355
356 0000f1 940e 0115 call tol_ORR
357 0000f3 e0f2 ldi ZH, high(line3_testmessage<<1)
358 0000f4 e6e2 ldi ZL, low(line3_testmessage<<1)
359 0000f5 df8d rcall load_msg ;load message into buffer(s).
360
361

```

```

362 0000f6 df5f          rcall update_lcd_dog
363
364 0000f7 9812          cbi PORTA, 2          ;turn off green
365 0000f8 9813          cbi PORTA, 3          ;turn on blue
366 0000f9 9a14          sbi PORTA, 4          ;turn on red
367
368 0000fa cfb6          rjmp main_loop
369
370
371
372                      ;*****
373                      ;*
374                      ;* "Subroutine_name" - Tolerance
375                      ;*
376                      ;* Description: This subroutine does the ascii conversion
377                      ;*              which is to be displayed in the LCD. This is
378                      ;*              displayed in the 3rd line. Values from the
379                      ;*              character set was loaded into r26 and r27
380                      ;*              and then r25 was loaded with decimal value.
381                      ;*              To get the ascii value , r25 was or'ed with
382                      ;*              48 for the conversion (since 48 is 0 in ascii).
383                      ;*              Using the Y pointer, each digit is to be
384                      ;*              displayed in a specific position.
385                      ;*
386                      ;*
387                      ;* Author: Asif Iqbal
388                      ;*          Roni Das
389                      ;* Version:1A
390                      ;* Last updated: 11/06/2018
391                      ;* Target: Perfect
392                      ;* Number of words: 10
393                      ;* Number of cycles: 300/307 (Min/Max)
394                      ;* Low registers modified: none
395                      ;* High registers modified: 3
396                      ;*
397                      ;* Parameters:
398                      ;*
399                      ;* Returns:
400                      ;*
401                      ;* Notes:
402                      ;*
403                      ;*****
404
405
406 tol_2:
407 0000fb e0d1          ldi YH,HIGH(dsp_buff_3)      ;displaying on line 3
408 0000fc e2c0          ldi YL,LOW(dsp_buff_3)
409 0000fd e092          ldi r25,$02                ;loading r25 with hex 02
410 0000fe e2a5          ldi r26, 0b00100101         ;loading r26 with 0b00100101
411 0000ff e2b0          ldi r27, $20                ;loading r27 with hex 20
412 000100 6390          ori r25, 48                 ;or immediately to get the ascii
413 000101 839f          std y+7, r25                 ;displaying at the 7th position
414 000102 87a8          std y+8,r26                 ;displaying at the 8th position
415 000103 87b9          std y+9,r27                 ;displaying at the the position
416 000104 e200          ldi r16,' '
417 000105 870a          std y+10,r16

```

```

418 000106 870c          std y+12,r16
419 000107 9508          ret
420                      tol_5:
421 000108 e0d1          ldi YH,HIGH(dsp_buff_3)
422 000109 e2c0          ldi YL,LOW(dsp_buff_3)
423 00010a e095          ldi r25,$05
424 00010b e2a5          ldi r26, 0b00100101
425 00010c e2b0          ldi r27, $20
426 00010d 6390          ori r25, 48
427 00010e 839f          std y+7, r25
428 00010f 87a8          std y+8,r26
429 000110 87b9          std y+9,r27
430 000111 e200          ldi r16,' '
431 000112 870a          std y+10,r16
432 000113 870c          std y+12,r16
433
434 000114 9508          ret
435
436                      tol_ORR:
437 000115 e0d1          ldi YH,HIGH(dsp_buff_3)
438 000116 e2c0          ldi YL,LOW(dsp_buff_3)
439 000117 e49f          ldi r25,$4F
440 000118 e5a2          ldi r26,$52
441 000119 839f          std y+7, r25
442 00011a 8798          std y+8, r25
443 00011b 87a9          std y+9,r26
444
445                      ldi r16,' '
446 00011d 870a          std y+10,r16
447 00011e 870c          std y+12,r16
448
449 00011f 9508          ret
450                      ;converting binary to ascii
451                      convert_ascii_2:
452 000120 e0d1          ldi YH,HIGH(dsp_buff_2) ;displaying at line 2
453 000121 e1c0          ldi YL,LOW(dsp_buff_2)
454 000122 6360          ori r22, 48          ;ori toconvert register content to ascii
455 000123 6370          ori r23, 48
456 000124 6380          ori r24, 48
457 000125 838f          std y+7, r24;displaying it to the 7th position in lcd
458 000126 8778          std y+8, r23
459 000127 8769          std y+9, r22
460 000128 9508          ret
461
462
463                      ;*****
464                      ;***** ALL MESSAGES: Fixed format, flash stored/loaded      *
465                      ;*****
466
467
468 000129 4601
469 00012a 7172
470 00012b 3d20
471 00012c 0020          line1_testmessage: .db 1, "Frq = ", 0; message for line #1.
472 00012d 5002
473 00012e 6472

```

```

474 00012f 3d20
475 000130 0020 line2_testmessage: .db 2, "Prd = ", 0; message for line #2.
476 000131 0003 line3_testmessage: .db 3, "", 0; message for line #3.
477
478
479
480
481 setup_display_one:
482 ;clr r17
483 000132 940e 0168 call bin2BCD16 ;output r13,r14,15
484 000134 2d6d mov r22, r13 ;new
485 000135 2d7d mov r23, r13 ;new
486 000136 706f andi r22, $0F ;digit 0 value
487 000137 7f70 andi r23, $F0
488 000138 9572 swap r23 ;digit 1 value
489 000139 2d8e mov r24, r14
490 00013a 2d9e mov r25, r14
491 00013b 708f andi r24, $0F ;digit 2 value
492 00013c 7f90 andi r25, $F0
493 00013d 9592 swap r25
494 00013e 2daf mov r26, r15
495 00013f 70af andi r26, $0F
496
497
498
499 000140 2f2a mov r18, r26
500 000141 e604 ldi r16,100
501 000142 9f20 mul r18,r16
502 000143 2d20 mov r18,r0
503
504
505 000144 2f49 mov r20, r25
506 000145 e00a ldi r16,10
507 000146 9f40 mul r20,r16
508
509 000147 1d20 adc r18,r0
510 000148 2f48 mov r20,r24
511 000149 1f24 adc r18,r20
512
513
514 ;mov r25, r13
515 ;andi r25, $0F
516
517 ;converting binary to ascii
518 convert_ascii:
519 ;ldi r19, 9 ; looping for 8 bits
520 00014a e0d1 ldi YH,HIGH(dsp_buff_1)
521 00014b e0c0 ldi YL,LOW(dsp_buff_1)
522 ;adiw YH:YL, 7
523 00014c 6360 ori r22, 48
524 00014d 6370 ori r23, 48
525 00014e 6380 ori r24, 48
526 00014f 6390 ori r25, 48
527 000150 63a0 ori r26, 48
528
529

```

```

530 000151 940e 0159      call msg
531
532 000153 83af          std y+7, r26
533 000154 8798          std y+8, r25
534 000155 878a          std y+10, r24
535 000156 877b          std y+11, r23
536 000157 876c          std y+12, r22
537 000158 9508          ret
538
539
540      msg:
541
542 000159 e500          ldi r16,'P'
543 00015a 8308          std y+0,r16
544 00015b e502          ldi r16,'R'
545 00015c 8309          std y+1,r16
546 00015d e404          ldi r16,'D'
547 00015e 830a          std y+2,r16
548 00015f e30d          ldi r16,'='
549 000160 830c          std y+4,r16
550 000161 e20e          ldi r16,'.'
551 000162 8709          std y+9,r16
552 000163 e60d          ldi r16,'m'
553 000164 870e          std y+14,r16
554 000165 e703          ldi r16,'s'
555 000166 870f          std y+15,r16
556 000167 9508          ret
557      ;***** END OF FILE *****
558
559      ;*****
560      ;*
561      ;* "bin2BCD16" - 16-bit Binary to BCD conversion
562      ;*
563      ;* This subroutine converts a 16-bit number (fbinH:fbinL) to
564      ;* a 5-digit
565      ;* packed BCD number represented by 3 bytes (tBCD2:tBCD1:tBCD0).
566      ;* MSD of the 5-digit number is placed in the lowermost
567      ;* nibble of tBCD2.
568      ;*
569      ;* Number of words :25
570      ;* Number of cycles:751/768 (Min/Max)
571      ;* Low registers used :3 (tBCD0,tBCD1,tBCD2)
572      ;* High registers used :4(fbinL,fbinH,cnt16a,tmp16a)
573      ;* Pointers used :Z
574      ;*
575      ;*****
576
577      ;***** Subroutine Register Variables
578
579      .equ    AtBCD0    =13      ;address of tBCD0
580      .equ    AtBCD2    =15      ;address of tBCD1
581
582      .def    tBCD0     =r13      ;BCD value digits 1 and 0
583      .def    tBCD1     =r14      ;BCD value digits 3 and 2
584      .def    tBCD2     =r15      ;BCD value digit 4
585      .def    fbinL     =r16      ;binary value Low byte

```

```

586 .def fbinH =r17 ;binary value High byte
587 .def cnt16a =r18 ;loop counter
588 .def tmp16a =r19 ;temporary value
589
590 ;***** Code
591
592 bin2BCD16:
593 000168 930f push r16
594 000169 931f push r17
595 00016a e120 ldi cnt16a,16 ;Init loop counter
596 00016b 24ff clr tBCD2 ;clear result (3 bytes)
597 00016c 24ee clr tBCD1
598 00016d 24dd clr tBCD0
599 00016e 27ff clr ZH ;clear ZH (not needed for AT90Sxx0x)
600 00016f 0f00 bBCDx_1:lsr fbinL ;shift input value
601 000170 1f11 rol fbinH ;through all bytes
602 000171 1cdd rol tBCD0 ;
603 000172 1cee rol tBCD1
604 000173 1cff rol tBCD2
605 000174 952a dec cnt16a ;decrement loop counter
606 000175 f419 brnebBCDx_2 ;if counter not zero
607 000176 911f pop r17
608 000177 910f pop r16
609 000178 9508 ret ; return
610
611 000179 e1e0 bBCDx_2:ldir30,AtBCD2+1;Z points to result MSB + 1
612 bBCDx_3:
613 00017a 9132 ld tmp16a,-Z ;get (Z) with pre-decrement
614 ;-----
615 ;For AT90Sxx0x, substitute the above line with:
616 ;
617 ; dec ZL
618 ; ld tmp16a,Z
619 ;
620 ;-----
621 00017b 5f3d subitmp16a,-$03 ;add 0x03
622 00017c fd33 sbrctmp16a,3;if bit 3 not clear
623 00017d 8330 st Z,tmp16a; store back
624 00017e 8130 ld tmp16a,Z;get (Z)
625 00017f 5d30 subitmp16a,-$30 ;add 0x30
626 000180 fd37 sbrctmp16a,7;if bit 7 not clear
627 000181 8330 st Z,tmp16a; store back
628 000182 30ed cpi ZL,AtBCD0 ;done all three?
629 000183 f7b1 brnebBCDx_3 ;loop again if not
630 000184 cfea rjmp bBCDx_1
631
632
633
634
635
636 start_counter_timer:
637
638 000185 e000 ldi r16,$00
639 000186 9300 0085 sts TCNT1H,r16
640 000188 9300 0084 sts TCNT1L,r16
641 00018a 9300 0080 sts TCCR1A,r16

```

```

642 00018c e001          ldi r16,$01
643
644
645 00018d e020          ldi r18, $00;counter +/- 2% or +/- 5
646          lb_0:
647 00018e 994a          sbic PIND,2      ;Check for a clear bit @ PINC0
648 00018f cffe          rjmp lb_0
649
650          lb_1:
651 000190 9b4a          sbis PIND,2      ;Check for a set bit @ PINC0
652 000191 cffe          rjmp lb_1
653          lb_2:
654
655 000192 9300 0081      sts TCCR1B,r16 ;start counting
656
657          ;call var_delay
658          ;inc r18
659 000194 994a          sbic PIND,2      ;Check for a clear bit @ PINC0
660 000195 cffc          rjmp lb_2
661          lb_3:
662          ;call var_delay
663          ;inc r18
664 000196 9b4a          sbis PIND,2      ;Check for a set bit @ PINC0
665 000197 cffe          rjmp lb_3
666
667 000198 e000          ldi r16,$00
668 000199 9300 0081      sts TCCR1B,r16      ;clock stopped
669
670 00019b 9100 0084      lds r16, TCNT1L
671 00019d 9110 0085      lds r17, TCNT1H
672 00019f 2f21          mov r18,r17
673 0001a0 e535          ldi r19,$55
674
675 0001a1 9518          reti
676
677          DIVISION:
678
679
680          ;*****
681          ;*
682          ;* "div32u" - 32/32 Bit Unsigned Division
683          ;*
684          ;* Ken Short
685          ;*
686          ;* This subroutine divides the two 32-bit numbers
687          ;* "dd32u3:dd32u2:dd32u1:dd32u0" (dividend) and "
688          ;* dv32u3:dv32u2:dv32u3:dv32u2"
689          ;* (divisor).
690          ;* The result is placed in "dres32u3:dres32u2
691          ;* dres32u2" and the
692          ;* remainder in "drem32u3:drem32u2:drem32u3:drem3
693          ;*
694          ;* Number of words :
695          ;* Number of cycles:655/751 (Min/Max) ATmega16
696          ;* #Low registers used :2 (drem16uL,drem16uH)
697          ;* #High registers used :5

```

```

698 ; (dres16uL/dd16uL,dres16uH/dd16uH,dv16uL,dv16uH,
699 ;*          dcnt16u)
700 ;* A $0000 divisor returns $FFFF
701 ;*
702 ;*****
703
704 ;***** Subroutine Register Variables
705
706 .def  drem32u0=r12    ;remainder
707 .def  drem32u1=r13
708 .def  drem32u2=r14
709 .def  drem32u3=r15
710
711 .def  dres32u0=r18    ;result (quotient)
712 .def  dres32u1=r19
713 .def  dres32u2=r20
714 .def  dres32u3=r21
715
716 .def  dd32u0  =r18    ;dividend
717 .def  dd32u1  =r19
718 .def  dd32u2  =r20
719 .def  dd32u3  =r21
720
721 .def  dv32u0  =r22    ;divisor
722 .def  dv32u1  =r23
723 .def  dv32u2  =r24
724 .def  dv32u3  =r25
725
726 .def  dcnt32u =r17
727
728 ;***** Code
729
730 div32u:
731 0001a2 24cc      clr drem32u0;clear remainder Low byte
732 0001a3 24dd      clr drem32u1
733 0001a4 24ee      clr drem32u2
734 0001a5 18ff      sub drem32u3,drem32u3;clear remainder High byte
735 0001a6 e211      ldi dcnt32u,33 ;init loop counter
736
737 0001a7 1f22      rol dd32u0      ;shift left dividend
738 0001a8 1f33      rol dd32u1
739 0001a9 1f44      rol dd32u2
740 0001aa 1f55      rol dd32u3
741 0001ab 951a      dec dcnt32u      ;decrement counter
742 0001ac f409      brned32u_2      ;if done
743 0001ad 9508      ret              ; return
744
745 0001ae 1ccc      rol drem32u0;shift dividend into remainder
746 0001af 1cdd      roldrem32u1
747 0001b0 1cee      roldrem32u2
748 0001b1 1cff      rol drem32u3
749
750 0001b2 1ac6      sub drem32u0,dv32u0 ;remainder = remainder - divisor
751 0001b3 0ad7      sbcdrem32u1,dv32u1
752 0001b4 0ae8      sbcdrem32u2,dv32u2
753 0001b5 0af9      sbc drem32u3,dv32u3 ;

```



```

754 0001b6 f430          brccd32u_3      ; branch if result is pos or zero
755
756 0001b7 0ec6          add drem32u0,dv32u0 ;if result negative
757 0001b8 1ed7          adc drem32u1,dv32u1
758 0001b9 1ee8          adc drem32u2,dv32u2
759 0001ba 1ef9          adc drem32u3,dv32u3
760 0001bb 9488          clc              ; clear carry to be shifted into result
761 0001bc cfea          rjmpd32u_1        ;else
762 0001bd 9408          d32u_3:sec          ; set carry to be shifted into result
763 0001be cfe8          rjmpd32u_1
764
765
766

```

767 RESOURCE USE INFORMATION

```

768 -----
769

```

770 Notice:

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

775

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

778

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

780

781 "ATmega324A" register use summary:

```

782 x : 0 y : 31 z : 10 r0 : 2 r1 : 0 r2 : 0 r3 : 0 r4 : 0
783 r5 : 0 r6 : 0 r7 : 0 r8 : 0 r9 : 0 r10: 0 r11: 0 r12: 4
784 r13: 8 r14: 8 r15: 8 r16: 10 r17: 10 r18: 17 r19: 12 r20: 18
785 r21: 2 r22: 10 r23: 11 r24: 13 r25: 19 r26: 13 r27: 4 r28: 8
786 r29: 8 r30: 10 r31: 9

```

787 Registers used: 23 out of 35 (65.7%)

788

789 "ATmega324A" instruction use summary:

```

790 .lds : 0 .sts : 0 adc : 5 add : 1 adiw : 2 and : 0
791 andi : 5 asr : 0 bclr : 0 bld : 0 brbc : 0 brbs : 0
792 brcc : 1 brcs : 0 break : 0 breq : 4 brge : 4 brhc : 0
793 brhs : 0 brid : 0 brie : 0 brlo : 0 brlt : 4 brmi : 0
794 brne : 10 brpl : 0 brsh : 0 brtc : 0 brts : 0 brvc : 0
795 brvs : 0 bset : 0 bst : 0 call : 8 cbi : 11 cbr : 0
796 clc : 1 clh : 0 cli : 1 cln : 0 clr : 7 cls : 0
797 clt : 0 clv : 0 clz : 0 com : 0 cp : 0 cpc : 0
798 cpi : 10 cpse : 0 dec : 9 eor : 0 fmul : 0 fmuls : 0
799 fmulsu: 0 icall : 0 ijmp : 0 in : 9 inc : 0 jmp : 0
800 ld : 5 ldd : 0 ldi : 88 lds : 4 lpm : 2 lsl : 1
801 lsr : 0 mov : 11 movw : 0 mul : 2 muls : 0 mulsu : 0
802 neg : 0 nop : 2 or : 0 ori : 10 out : 10 pop : 8
803 push : 8 rcall : 43 ret : 19 reti : 1 rjmp : 19 rol : 12
804 ror : 0 sbc : 3 sbci : 0 sbi : 10 sbic : 2 sbis : 2
805 sbiw : 0 sbr : 0 sbrc : 2 sbrs : 2 sec : 1 seh : 0
806 sei : 1 sen : 0 ser : 0 ses : 0 set : 0 sev : 0
807 sez : 0 sleep : 0 spm : 0 st : 4 std : 30 sts : 13
808 sub : 2 subi : 2 swap : 2 tst : 0 wdr : 0

```

809 Instructions used: 48 out of 113 (42.5%)

810

811 "ATmega324A" memory use summary [bytes]:

812 Segment Begin End Code Data Used Size Use%

813 -----

814 [.cseg] 0x000000 0x000380 876 18 894 32768 2.7%

815 [.dseg] 0x000100 0x000130 0 48 48 2048 2.3%

816 [.eseg] 0x000000 0x000000 0 0 0 1024 0.0%

817

818 Assembly complete, 0 errors, 10 warnings

819

```

1
2 AVRASM ver. 2.2.7 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm Tue Dec 04 20:42:03 2018
3
4 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(52): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avram\inc\m324adef.inc'
5 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(71): Including file 'C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\lcd_dog_asm_driver_m324a.inc'
6 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(692): warning: Register r13 already defined by the .DEF directive
7 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(693): warning: Register r14 already defined by the .DEF directive
8 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(694): warning: Register r15 already defined by the .DEF directive
9 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(696): warning: Register r18 already defined by the .DEF directive
10 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(697): warning: Register r19 already defined by the .DEF directive
11 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(701): warning: Register r18 already defined by the .DEF directive
12 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(702): warning: Register r19 already defined by the .DEF directive
13 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(703): warning: Register r20 already defined by the .DEF directive
14 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(704): warning: Register r21 already defined by the .DEF directive
15 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(711): warning: Register r17 already defined by the .DEF directive
16 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(52): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avram\inc\m324adef.inc'
17 C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\main.asm(71): Including file 'C:\Users\ronid\OneDrive - Stony Brook University\School\Stony 2018 Fall\ESE 380\Labs\lab4\Direct_Period_Mesur\Direct_Period_Mesur\lcd_dog_asm_driver_m324a.inc'
18
19
20 ;*****
21 ;lab_10_auto_range_meas.asm
22 ;
23 ;Author : Roni Das ID: 108378223
24 ; Asif Iqbal ID: 110333685
25 ;Created: 12/04/2018 8:24:08 PM

```

```

26      ;
27      ; Description: The following program displays the measurement
28      ; of the direct frequency of a periodic waveform and
29      ; displays it on a LCD. The longer the period is
30      ; the more precise the measurements are. A 16 x 3 LCD is used
31      ; for this experiment. The first line illustrates
32      ; the frequency in Hz. The second line represents the decimal
33      ; count of the period. The program below generates a period
34      ; of approximately 1s. The frequency is displayed in the unit
35      ; of 1hz and period is displayed in the unit of 1us. The signal
36      ; that is to be measure is 20khz to 65khz. When the frequency is
37
38      ; greater than 1khz, the system measures the frequency directly
39      ; and computes the period. The direct measured frequency has
40      ; an asterisk next to it. The program in general determines
41      ; the range of frequency of the input signal and then based on
42      ; measures the signal appropriately.
43      ;
44      ; Inputs:
45      ;   PORTC = PC0
46      ;
47      ; Output:
48      ;   PORTB = PB0 - PB7
49      ;
50      ; Register Assignments/Purposes
51      ;   r16 = general purpose
52      ;   r19 = loop counter
53      ;   r21 = period counter
54      ;   r22 = digit 0
55      ;   r23 = digit 1
56      ;   r24 = digit 2
57      ;   r13 = digit 0 and 1
58      ;   r14 = digit 2 and 4
59      ;   r25 = conversion purpose
60      ;   r26 = conversion purpose
61      ;   r27 = conversion purpose
62
63      ; *Courtesy of Professor Ken Short and his lecture notes.
64      ;
65      ; Section no.: L02
66      ; Experiment no.: 10
67      ; Bench no.: 02
68      ; *****
69
70      .list
71
72
73      .CSEG
74
75
76      ; interrupt vector table, with several 'safety' stubs
77      .org 0      ; Reset/Cold start vector
78      rjmp start
79      .org INT0addr
80      rjmp start_counter_timer

```

```

81
82
83
84
85         .list
86
87         ;*****
88         ;NAME:      clr_dsp_buffs
89         ;FUNCTION:  Initializes dsp_buffers 1, 2, and 3
90         ;           with blanks (0x20)
91         ;ASSUMES:   Three CONTIGUOUS 16-byte dram based
92         ;           buffers named
93         ;           dsp_buff_1, dsp_buff_2, dsp_buff_3.
94         ;RETURNS:   nothing.
95         ;MODIFIES:  r25,r26, Z-ptr
96         ;CALLS:     none
97         ;CALLED BY: main application and diagnostics
98         ;*****
99
100
101         clr_dsp_buffs:
102 00007b e390          ldi R25, 48          ; load total length of
103                                     ;both buffer.
104 00007c e2a0          ldi R26, ' '          ; load blank/space into
105                                     ;R26.
106 00007d e0f1          ldi ZH, high (dsp_buff_1) ; Load ZH and ZL as a
107                                     ;pointer to 1st
108 00007e e0e0          ldi ZL, low (dsp_buff_1)  ; byte of buffer for
109                                     ;line 1.
110
111         ;set DDRAM address to 1st
112         ;position of first line.
113         store_bytes:
114 00007f 93a1          st Z+, R26          ; store ' ' into 1st/next
115                                     ; buffer byte and
116                                     ; auto inc ptr to next
117                                     ;location.
118 000080 959a          dec R25              ;
119 000081 f7e9          brne store_bytes    ; cont until r25=0, all
120                                     ;bytes written.
121 000082 9508          ret
122
123
124
125         ;*****
126         ;NAME:      load_msg
127         ;FUNCTION:  Loads a predefined string msg into
128         ;           a specified diplay
129         ;           buffer.
130         ;ASSUMES:   Z = offset of message to be loaded.
131         ;           ;Msg format is
132         ;           defined below.
133         ;RETURNS:   nothing.
134         ;MODIFIES:  r16, Y, Z
135         ;CALLS:     nothing
136         ;CALLED BY:

```

```

137 ;*****
138 ; Message structure:
139 ;   label:  .db <buff num>, <text string/message>
140 ;           , <end of string>
141 ;
142 ; Message examples (also see Messages
143 ;at the end of this file/module):
144 ;   msg_1: .db 1,"First Message ", 0
145 ; loads msg into buff 1, eom=0
146 ;   msg_2: .db 1,"Another message ", 0
147 ; loads msg into buff 1, eom=0
148 ;
149 ; Notes:
150 ;   a) The 1st number indicates which
151 ; buffer to load (either 1, 2, or 3).
152 ;   b) The last number (zero) is an '
153 ;end of string' indicator.
154 ;   c) Y = ptr to disp_buffer
155 ;       Z = ptr to message
156 ; (passed to subroutine)
157 ;*****
158 load_msg:
159 000083 e0d1      ldi YH, high (dsp_buff_1) ; Load YH and YL
160                                     ;as a pointer to 1st
161 000084 e0c0      ldi YL, low (dsp_buff_1) ; byte of dsp_buff_1
162                                     ;(Note - assuming
163                                     ; (dsp_buff_1 for now).
164 000085 9105      lpm R16, Z+          ; get dsply buff number
165                                     ;(1st byte of msg).
166 000086 3001      cpi r16, 1          ; if equal to '1', ptr
167                                     ; already setup.
168 000087 f021      breq get_msg_byte   ; jump and start message
169                                     ; load.
170 000088 9660      adiw YH:YL, 16      ; else set ptr to dsp
171                                     ;buff 2.
172 000089 3002      cpi r16, 2          ; if equal to '2', ptr
173                                     ;now setup.
174 00008a f009      breq get_msg_byte   ; jump and start message
175                                     ;load.
176 00008b 9660      adiw YH:YL, 16      ; else set ptr to dsp
177                                     ;buff 2.
178
179 get_msg_byte:
180 00008c 9105      lpm R16, Z+          ; get next byte of msg
181                                     ;and see if '0'.
182 00008d 3000      cpi R16, 0          ; if equal to '0', end
183                                     ;of message reached.
184 00008e f011      breq msg_loaded     ; jump and stop message
185                                     ; loading operation.
186 00008f 9309      st Y+, R16          ; else, store next byte
187                                     ;of msg in buffer.
188 000090 cffb      rjmp get_msg_byte   ; jump back and continue.
189 msg_loaded:
190 000091 9508      ret
191
192 start:

```

```

193 000092 ef0f      ldi r16, low(RAMEND)      ; init stack/pointer
194 000093 bf0d      out SPL, r16
195 000094 e008      ldi r16, high(RAMEND)
196 000095 bf0e      out SPH, r16
197
198
199 000096 ef0f      ldi r16, 0xff            ; set portB = output.
200 000097 b904      out DDRB, r16           ;
201 000098 9a2c      sbi portB, 4             ;set /SS of DOG LCD =
202                                     ; 1 (Deselected)
203
204 000099 9852      cbi DDRD, 2
205 00009a 985a      cbi portD, 2
206
207 00009b df9c      rcall init_lcd_dog      ; init display, using
208                                     ; SPI serial interface
209 00009c dfde      rcall clr_dsp_buffs   ; clear all three
210                                     ; lines
211
212
213                                     ;Configure port A bits 0-4 as an output
214 00009d e10c      ldi r16, 0b00011100     ;load r16 with 1s in
215                                     ; 0-4 position
216                                     ;bit 0 and 1 position
217 00009e b901      out DDRA, r16      ;port A - bit 0 - 4 as
218                                     ;an output
219                                     ;Port A - bit 5-7 is
220                                     ; Input
221 00009f e100      ldi r16, 0b00010000     ;load r16 with all 1s a
222
223 0000a0 b902      out PORTA, r16      ;trun On LED: Color: RED
224
225
226                                     ;Configure port C bits 0 as an input pin
227 0000a1 e000      ldi r16,$00            ;load r16 with all 0s
228 0000a2 b907      out DDRC,r16      ;port C-bit 0 = input
229
230
231 0000a3 ef5f      ldi r21, $FF            ;Initilalize Display = 00
232 0000a4 e020      ldi r18, $00            ;counter +/- 2% or +/- 5%
233
234
235                                     ;power on self test
236
237
238 0000a5 e003      ldi r16, (1 << ISC01) | (1 << ISC00)
239 0000a6 9300 0069  sts EICRA, r16
240 0000a8 e001      ldi r16, 1 << INT0
241 0000a9 bb0d      out EIMSK, r16
242
243
244 0000aa e000      ldi r16,$00
245 0000ab 9300 0085  sts TCNT1H,r16
246 0000ad 9300 0084  sts TCNT1L,r16
247 0000af 9300 0080  sts TCCR1A,r16
248

```

```

249
250             main_loop:
251 0000b1 9478             sei
252             ; sts TCCR1B,r16             ;clock stopped*/
253
254
255
256 0000b2 3535             cpi r19,$55
257 0000b3 f009             breq continue
258 0000b4 cffc             rjmp main_loop
259
260             continue:
261 0000b5 94f8             cli
262
263 0000b6 e030             ldi r19,$00
264 0000b7 9100 0084         lds r16, TCNT1L
265 0000b9 9110 0085         lds r17, TCNT1H
266 0000bb 2f21             mov r18,r17
267
268 0000bc e040             ldi r20,$00
269 0000bd 9340 0085         sts TCNT1H,r20
270 0000bf 9340 0084         sts TCNT1L,r20
271 0000c1 9340 0080         sts TCCR1A,r20
272 0000c3 9340 0081         sts TCCR1B,r20
273
274 0000c5 940e 0132         call setup_display_one
275
276             ;load_line_1 into dbuff1:
277 0000c7 e0f2             ldi ZH, high(line1_testmessage<<1)
278 0000c8 e5e2             ldi ZL, low(line1_testmessage<<1)
279
280 0000c9 940e 0056         call update_lcd_dog
281 0000cb 940e 0083         call load_msg ;load message into buffer(s).
282
283
284 0000cd 3622             cpi r18,98             ;check lower bound 2%
285 0000ce f424             brge check_Upper_bound_2P ;branch if greater or equal
286 0000cf 352f             cpi r18,95             ;check lower bound 5%
287 0000d0 f444             brge check_Upper_bound_5P ;branch if greater or equal
288 0000d1 f0fc             brlt lower_than_5P ;else count < 95
289 0000d2 cfde             rjmp main_loop
290
291             check_Upper_bound_2P:
292 0000d3 3627             cpi r18,103
293 0000d4 f044             brlt display_bargraph_2P; branch if lower
294 0000d5 362a             cpi r18,106
295 0000d6 f084             brlt display_bargraph_5P;branch if lower
296 0000d7 f4cc             brge higher_than_5P ;branch if higher
297 0000d8 cfd8             rjmp main_loop
298
299             check_Upper_bound_5P:
300 0000d9 362a             cpi r18,106
301 0000da f4b4             brge higher_than_5P ;branch if higher
302 0000db f05c             brlt display_bargraph_5P;branch if lower
303 0000dc cfd4             rjmp main_loop
304

```



```

305                                     display_bargraph_2P:
306                                     ;call display_g_LCD
307
308                                     ;load_line_1 into dbuff1:
309                                     ;ldi ZH, high(line1_testmessage<<1)
310                                     ;ldi ZL, low(line1_testmessage<<1)
311                                     ;rcall load_msg      ;load message into buffer(s).
312
313 0000dd 940e 00fb                     call tol_2
314 0000df e0f2                         ldi ZH, high(line3_testmessage<<1)
315 0000e0 e6e2                         ldi ZL, low(line3_testmessage<<1)
316 0000e1 dfa1                         rcall load_msg      ;load message into buffer(s).
317
318
319 0000e2 df73                         rcall update_lcd_dog
320
321
322 0000e3 9a12                         sbi PORTA, 2          ;turn on green
323 0000e4 9813                         cbi PORTA, 3          ; turn off blue
324 0000e5 9814                         cbi PORTA, 4          ;turn off red
325 0000e6 cfca                         rjmp main_loop
326
327                                     display_bargraph_5P:
328
329                                     ;call display_g_LCD
330
331                                     ;load_line_1 into dbuff1:
332                                     ; ldi ZH, high(line1_testmessage<<1)
333                                     ; ldi ZL, low(line1_testmessage<<1)
334                                     ; rcall load_msg      ;load message into buffer(s).
335
336 0000e7 940e 0108                     call tol_5
337 0000e9 e0f2                         ldi ZH, high(line3_testmessage<<1)
338 0000ea e6e2                         ldi ZL, low(line3_testmessage<<1)
339 0000eb df97                         rcall load_msg      ;load message into buffer(s).
340
341
342 0000ec df69                         rcall update_lcd_dog
343
344
345 0000ed 9812                         cbi PORTA, 2          ;turn off green
346 0000ee 9a13                         sbi PORTA, 3          ;turn on blue
347 0000ef 9814                         cbi PORTA, 4          ;turn off red
348 0000f0 cfc0                         rjmp main_loop
349
350                                     lower_than_5P:
351                                     higher_than_5P:
352
353                                     ;call display_g_LCD
354                                     ;load_line_1 into dbuff1:
355                                     ;ldi ZH, high(line1_testmessage<<1)
356                                     ;ldi ZL, low(line1_testmessage<<1)
357                                     ;rcall load_msg      ;load message into buffer(s).
358
359 0000f1 940e 0115                     call tol_ORR
360 0000f3 e0f2                         ldi ZH, high(line3_testmessage<<1)

```

```

361 0000f4 e6e2      ldi ZL, low(line3_testmessage<<1)
362 0000f5 df8d      rcall load_msg      ;load message into buffer(s).
363
364
365 0000f6 df5f      rcall update_lcd_dog
366
367 0000f7 9812      cbi PORTA, 2        ;turn off green
368 0000f8 9813      cbi PORTA, 3        ;turn on blue
369 0000f9 9a14      sbi PORTA, 4        ;turn on red
370
371 0000fa cfb6      rjmp main_loop
372
373
374
375 ;*****
376 ;*
377 ;* "Subroutine_name" - Tolerance
378 ;*
379 ;* Description: This subroutine does the ascii conversion
380 ;*              which is to be displayed in the LCD. This is
381 ;*              displayed in the 3rd line. Values from the
382 ;*              character set was loaded into r26 and r27
383 ;*              and then r25 was loaded with decimal value.
384 ;*              To get the ascii value , r25 was or'ed with
385 ;*              48 for the conversion (since 48 is 0 in ascii).
386 ;*              Using the Y pointer, each digit is to be
387 ;*              displayed in a specific position.
388 ;*
389 ;*
390 ;* Author: Asif Iqbal
391 ;*         Roni Das
392 ;* Version:1A
393 ;* Last updated: 11/06/2018
394 ;* Target: Perfect
395 ;* Number of words: 10
396 ;* Number of cycles: 300/307 (Min/Max)
397 ;* Low registers modified: none
398 ;* High registers modified: 3
399 ;*
400 ;* Parameters:
401 ;*
402 ;* Returns:
403 ;*
404 ;* Notes:
405 ;*
406 ;*****
407
408
409 tol_2:
410 0000fb e0d1      ldi YH,HIGH(dsp_buff_3)      ;displaying on line 3
411 0000fc e2c0      ldi YL,LOW(dsp_buff_3)
412 0000fd e092      ldi r25,$02                ;loading r25 with hex 02
413 0000fe e2a5      ldi r26, 0b00100101      ;loading r26 with 0b00100101
414 0000ff e2b0      ldi r27, $20                ;loading r27 with hex 20
415 000100 6390      ori r25, 48                ;or immediately to get the ascii
416 000101 839f      std y+7, r25                ;displaying at the 7th position

```

```

417 000102 87a8          std y+8,r26          ;displaying at the 8th position
418 000103 87b9          std y+9,r27          ;displaying at the the position
419 000104 e200          ldi r16,' '
420 000105 870a          std y+10,r16
421 000106 870c          std y+12,r16
422 000107 9508          ret
423                      tol_5:
424 000108 e0d1          ldi YH,HIGH(dsp_buff_3)
425 000109 e2c0          ldi YL,LOW(dsp_buff_3)
426 00010a e095          ldi r25,$05
427 00010b e2a5          ldi r26, 0b00100101
428 00010c e2b0          ldi r27, $20
429 00010d 6390          ori r25, 48
430 00010e 839f          std y+7, r25
431 00010f 87a8          std y+8,r26
432 000110 87b9          std y+9,r27
433 000111 e200          ldi r16,' '
434 000112 870a          std y+10,r16
435 000113 870c          std y+12,r16
436
437 000114 9508          ret
438
439                      tol_ORR:
440 000115 e0d1          ldi YH,HIGH(dsp_buff_3)
441 000116 e2c0          ldi YL,LOW(dsp_buff_3)
442 000117 e49f          ldi r25,$4F
443 000118 e5a2          ldi r26,$52
444 000119 839f          std y+7, r25
445 00011a 8798          std y+8, r25
446 00011b 87a9          std y+9,r26
447
448 00011c e200          ldi r16,' '
449 00011d 870a          std y+10,r16
450 00011e 870c          std y+12,r16
451
452 00011f 9508          ret
453                      ;converting binary to ascii
454                      convert_ascii_2:
455 000120 e0d1          ldi YH,HIGH(dsp_buff_2) ;displaying at line 2
456 000121 e1c0          ldi YL,LOW(dsp_buff_2)
457 000122 6360          ori r22, 48          ;ori toconvert register content to ascii
458 000123 6370          ori r23, 48
459 000124 6380          ori r24, 48
460 000125 838f          std y+7, r24;displaying it to the 7th position in lcd
461 000126 8778          std y+8, r23
462 000127 8769          std y+9, r22
463 000128 9508          ret
464
465
466                      ;*****
467                      ;***** ALL MESSAGES: Fixed format, flash stored/loaded *
468                      ;*****
469
470
471 000129 4601
472 00012a 7172

```

```

473 00012b 3d20
474 00012c 0020 line1_testmessage: .db 1, "Frq = ", 0; message for line #1.
475 00012d 5002
476 00012e 6472
477 00012f 3d20
478 000130 0020 line2_testmessage: .db 2, "Prd = ", 0; message for line #2.
479 000131 0003 line3_testmessage: .db 3, "", 0; message for line #3.
480
481
482
483
484 setup_display_one:
485 ;clr r17
486 000132 940e 0168 call bin2BCD16 ;output r13,r14,15
487 000134 2d6d mov r22, r13 ;new
488 000135 2d7d mov r23, r13 ;new
489 000136 706f andi r22, $0F ;digit 0 value
490 000137 7f70 andi r23, $F0
491 000138 9572 swap r23 ;digit 1 value
492 000139 2d8e mov r24, r14
493 00013a 2d9e mov r25, r14
494 00013b 708f andi r24, $0F ;digit 2 value
495 00013c 7f90 andi r25, $F0
496 00013d 9592 swap r25
497 00013e 2daf mov r26, r15
498 00013f 70af andi r26, $0F
499
500
501
502 000140 2f2a mov r18, r26
503 000141 e604 ldi r16,100
504 000142 9f20 mul r18,r16
505 000143 2d20 mov r18,r0
506
507
508 000144 2f49 mov r20, r25
509 000145 e00a ldi r16,10
510 000146 9f40 mul r20,r16
511
512 000147 1d20 adc r18,r0
513 000148 2f48 mov r20,r24
514 000149 1f24 adc r18,r20
515
516
517 ;mov r25, r13
518 ;andi r25, $0F
519
520 ;converting binary to ascii
521 convert_ascii:
522 ;ldi r19, 9 ; looping for 8 bits
523 00014a e0d1 ldi YH,HIGH(dsp_buff_1)
524 00014b e0c0 ldi YL,LOW(dsp_buff_1)
525 ;adiw YH:YL, 7
526 00014c 6360 ori r22, 48
527 00014d 6370 ori r23, 48
528 00014e 6380 ori r24, 48

```

```

529 00014f 6390          ori r25, 48
530 000150 63a0          ori r26, 48
531
532
533 000151 940e 0159      call msg
534
535 000153 83af          std y+7, r26
536 000154 8798          std y+8, r25
537 000155 878a          std y+10, r24
538 000156 877b          std y+11, r23
539 000157 876c          std y+12, r22
540 000158 9508          ret
541
542
543          msg:
544
545 000159 e500          ldi r16, 'P'
546 00015a 8308          std y+0, r16
547 00015b e502          ldi r16, 'R'
548 00015c 8309          std y+1, r16
549 00015d e404          ldi r16, 'D'
550 00015e 830a          std y+2, r16
551 00015f e30d          ldi r16, '='
552 000160 830c          std y+4, r16
553 000161 e20e          ldi r16, '.'
554 000162 8709          std y+9, r16
555 000163 e60d          ldi r16, 'm'
556 000164 870e          std y+14, r16
557 000165 e703          ldi r16, 's'
558 000166 870f          std y+15, r16
559 000167 9508          ret
560          ;***** END OF FILE *****
561
562          ;*****
563          ;*
564          ;* "bin2BCD16" - 16-bit Binary to BCD conversion
565          ;*
566          ;* This subroutine converts a 16-bit number (fbinH:fbinL) to
567          ;* a 5-digit
568          ;* packed BCD number represented by 3 bytes (tBCD2:tBCD1:tBCD0).
569          ;* MSD of the 5-digit number is placed in the lowermost
570          ;* nibble of tBCD2.
571          ;*
572          ;* Number of words :25
573          ;* Number of cycles:751/768 (Min/Max)
574          ;* Low registers used :3 (tBCD0,tBCD1,tBCD2)
575          ;* High registers used :4(fbinL,fbinH,cnt16a,tmp16a)
576          ;* Pointers used :Z
577          ;*
578          ;*****
579
580          ;***** Subroutine Register Variables
581
582          .equ AtBCD0 =13          ;address of tBCD0
583          .equ AtBCD2 =15          ;address of tBCD1
584

```

```

585      .def   tBCD0    =r13    ;BCD value digits 1 and 0
586      .def   tBCD1    =r14    ;BCD value digits 3 and 2
587      .def   tBCD2    =r15    ;BCD value digit 4
588      .def   fbinL    =r16    ;binary value Low byte
589      .def   fbinH    =r17    ;binary value High byte
590      .def   cnt16a    =r18    ;loop counter
591      .def   tmp16a    =r19    ;temporary value
592
593      ;***** Code
594
595      bin2BCD16:
596 000168 930f      push r16
597 000169 931f      push r17
598 00016a e120      ldi cnt16a,16    ;Init loop counter
599 00016b 24ff      clr tBCD2        ;clear result (3 bytes)
600 00016c 24ee      clr tBCD1
601 00016d 24dd      clr tBCD0
602 00016e 27ff      clr ZH        ;clear ZH (not needed for AT90Sxx0x)
603 00016f 0f00      bBCDx_1:ls1fbinL    ;shift input value
604 000170 1f11      rol fbinH        ;through all bytes
605 000171 1cdd      rol tBCD0        ;
606 000172 1cee      rol tBCD1
607 000173 1cff      rol tBCD2
608 000174 952a      dec cnt16a        ;decrement loop counter
609 000175 f419      brnebBCDx_2      ;if counter not zero
610 000176 911f      pop r17
611 000177 910f      pop r16
612 000178 9508      ret            ; return
613
614 000179 e1e0      bBCDx_2:ldir30,AtBCD2+1;Z points to result MSB + 1
615      bBCDx_3:
616 00017a 9132      ld tmp16a,-Z    ;get (Z) with pre-decrement
617      ;-----
618      ;For AT90Sxx0x, substitute the above line with:
619      ;
620      ; dec ZL
621      ; ld tmp16a,Z
622      ;
623      ;-----
624 00017b 5f3d      subitmp16a,-$03 ;add 0x03
625 00017c fd33      sbrctmp16a,3;if bit 3 not clear
626 00017d 8330      st Z,tmp16a; store back
627 00017e 8130      ld tmp16a,Z;get (Z)
628 00017f 5d30      subitmp16a,-$30 ;add 0x30
629 000180 fd37      sbrctmp16a,7;if bit 7 not clear
630 000181 8330      st Z,tmp16a; store back
631 000182 30ed      cpi ZL,AtBCD0    ;done all three?
632 000183 f7b1      brnebBCDx_3    ;loop again if not
633 000184 cfea      rjmp bBCDx_1
634
635
636
637
638
639      start_counter_timer:
640

```

```

641 000185 e000          ldi r16,$00
642 000186 9300 0085      sts TCNT1H,r16
643 000188 9300 0084      sts TCNT1L,r16
644 00018a 9300 0080      sts TCCR1A,r16
645 00018c e001          ldi r16,$01
646
647
648 00018d e020          ldi r18, $00;counter +/- 2% or +/- 5
649 lb_0:
650 00018e 994a          sbic PIND,2      ;Check for a clear bit @ PINC0
651 00018f cffe          rjmp lb_0
652
653 lb_1:
654 000190 9b4a          sbis PIND,2      ;Check for a set bit @ PINC0
655 000191 cffe          rjmp lb_1
656 lb_2:
657
658 000192 9300 0081      sts TCCR1B,r16 ;start counting
659
660          ;call var_delay
661          ;inc r18
662 000194 994a          sbic PIND,2      ;Check for a clear bit @ PINC0
663 000195 cffc          rjmp lb_2
664 lb_3:
665          ;call var_delay
666          ;inc r18
667 000196 9b4a          sbis PIND,2      ;Check for a set bit @ PINC0
668 000197 cffe          rjmp lb_3
669
670 000198 e000          ldi r16,$00
671 000199 9300 0081      sts TCCR1B,r16          ;clock stopped
672
673 00019b 9100 0084      lds r16, TCNT1L
674 00019d 9110 0085      lds r17, TCNT1H
675 00019f 2f21          mov r18,r17
676 0001a0 e535          ldi r19,$55
677
678 0001a1 9518          reti
679
680 DIVISION:
681
682
683 ;*****
684 ;*
685 ;* "div32u" - 32/32 Bit Unsigned Division
686 ;*
687 ;* Ken Short
688 ;*
689 ;* This subroutine divides the two 32-bit numbers
690 ;* "dd32u3:dd32u2:dd32u1:dd32u0" (dividend) and "
691 ;* dv32u3:dv32u2:dv32u3:dv32u2"
692 ;* (divisor).
693 ;* The result is placed in "dres32u3:dres32u2
694 ;* dres32u2" and the
695 ;* remainder in "drem32u3:drem32u2:drem32u3:drem3
696 ;*

```

```

697      ;* Number of words :
698      ;* Number of cycles:655/751 (Min/Max) ATmega16
699      ;* #Low registers used :2 (drem16uL,drem16uH)
700      ;* #High registers used :5
701      ; (dres16uL/dd16uL,dres16uH/dd16uH,dv16uL,dv16uH,
702      ;*          dcnt16u)
703      ;* A $0000 divisor returns $FFFF
704      ;*
705      ;*****
706
707      ;***** Subroutine Register Variables
708
709      .def    drem32u0=r12      ;remainder
710      .def    drem32u1=r13
711      .def    drem32u2=r14
712      .def    drem32u3=r15
713
714      .def    dres32u0=r18      ;result (quotient)
715      .def    dres32u1=r19
716      .def    dres32u2=r20
717      .def    dres32u3=r21
718
719      .def    dd32u0  =r18      ;dividend
720      .def    dd32u1  =r19
721      .def    dd32u2  =r20
722      .def    dd32u3  =r21
723
724      .def    dv32u0  =r22      ;divisor
725      .def    dv32u1  =r23
726      .def    dv32u2  =r24
727      .def    dv32u3  =r25
728
729      .def    dcnt32u =r17
730
731      ;***** Code
732
733      div32u:
734      0001a2 24cc      clr drem32u0;clear remainder Low byte
735      0001a3 24dd      clr drem32u1
736      0001a4 24ee      clr drem32u2
737      0001a5 18ff      sub drem32u3,drem32u3;clear remainder High byte
738      0001a6 e211      ldi dcnt32u,33 ;init loop counter
739
740      d32u_1:
741      0001a7 1f22      rol dd32u0      ;shift left dividend
742      0001a8 1f33      rol dd32u1
743      0001a9 1f44      rol dd32u2
744      0001aa 1f55      rol dd32u3
745      0001ab 951a      dec dcnt32u      ;decrement counter
746      0001ac f409      brned32u_2      ;if done
747      0001ad 9508      ret              ; return
748
749      d32u_2:
750      0001ae 1ccc      rol drem32u0;shift dividend into remainder
751      0001af 1cdd      roldrem32u1
752      0001b0 1cee      roldrem32u2
753      0001b1 1cff      rol drem32u3

```



```

753 0001b2 1ac6      sub drem32u0,dv32u0 ;remainder = remainder - divisor
754 0001b3 0ad7      sbcdrem32u1,dv32u1
755 0001b4 0ae8      sbcdrem32u2,dv32u2
756 0001b5 0af9      sbc drem32u3,dv32u3 ;
757 0001b6 f430      brccd32u_3      ; branch if result is pos or zero
758
759 0001b7 0ec6      add drem32u0,dv32u0 ;if result negative
760 0001b8 1ed7      adc drem32u1,dv32u1
761 0001b9 1ee8      adc drem32u2,dv32u2
762 0001ba 1ef9      adc drem32u3,dv32u3
763 0001bb 9488      clc          ; clear carry to be shifted into result
764 0001bc cfea      rjmpd32u_1      ;else
765 0001bd 9408      d32u_3:sec      ; set carry to be shifted into result
766 0001be cfe8      rjmpd32u_1
767
768
769

```

770 RESOURCE USE INFORMATION

```

771 -----

```

```

772

```

```

773 Notice:

```

```

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

```

```

778

```

```

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

```

```

781

```

```

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

```

```

783

```

```

784 "ATmega324A" register use summary:

```

```

785 x : 0 y : 31 z : 10 r0 : 2 r1 : 0 r2 : 0 r3 : 0 r4 : 0
786 r5 : 0 r6 : 0 r7 : 0 r8 : 0 r9 : 0 r10: 0 r11: 0 r12: 4
787 r13: 8 r14: 8 r15: 8 r16: 10 r17: 10 r18: 17 r19: 12 r20: 18
788 r21: 2 r22: 10 r23: 11 r24: 13 r25: 19 r26: 13 r27: 4 r28: 8
789 r29: 8 r30: 10 r31: 9

```

```

790 Registers used: 23 out of 35 (65.7%)

```

```

791

```

```

792 "ATmega324A" instruction use summary:

```

```

793 .lds : 0 .sts : 0 adc : 5 add : 1 adiw : 2 and : 0
794 andi : 5 asr : 0 bclr : 0 bld : 0 brbc : 0 brbs : 0
795 brcc : 1 brcs : 0 break : 0 breq : 4 brge : 4 brhc : 0
796 brhs : 0 brid : 0 brie : 0 brlo : 0 brlt : 4 brmi : 0
797 brne : 10 brpl : 0 brsh : 0 brtc : 0 brts : 0 brvc : 0
798 brvs : 0 bset : 0 bst : 0 call : 8 cbi : 11 cbr : 0
799 clc : 1 clh : 0 cli : 1 cln : 0 clr : 7 cls : 0
800 clt : 0 clv : 0 clz : 0 com : 0 cp : 0 cpc : 0
801 cpi : 10 cpse : 0 dec : 9 eor : 0 fmul : 0 fmuls : 0
802 fmulsu: 0 icall : 0 ijmp : 0 in : 9 inc : 0 jmp : 0
803 ld : 5 ldd : 0 ldi : 88 lds : 4 lpm : 2 lsl : 1
804 lsr : 0 mov : 11 movw : 0 mul : 2 muls : 0 mulsu : 0
805 neg : 0 nop : 2 or : 0 ori : 10 out : 10 pop : 8
806 push : 8 rcall : 43 ret : 19 reti : 1 rjmp : 19 rol : 12
807 ror : 0 sbc : 3 sbci : 0 sbi : 10 sbic : 2 sbis : 2
808 sbiw : 0 sbr : 0 sbrc : 2 sbrs : 2 sec : 1 seh : 0

```

809 sei : 1 sen : 0 ser : 0 ses : 0 set : 0 sev : 0
 810 sez : 0 sleep : 0 spm : 0 st : 4 std : 30 sts : 13
 811 sub : 2 subi : 2 swap : 2 tst : 0 wdr : 0

812 Instructions used: 48 out of 113 (42.5%)

813

814 "ATmega324A" memory use summary [bytes]:

815	Segment	Begin	End	Code	Data	Used	Size	Use%
816	-----							
817	[.cseg]	0x000000	0x000380	876	18	894	32768	2.7%
818	[.dseg]	0x000100	0x000130	0	48	48	2048	2.3%
819	[.eseg]	0x000000	0x000000	0	0	0	1024	0.0%

820

821 Assembly complete, 0 errors, 10 warnings

822