LAB 11: Cycling chilled mirror hygrometer

Section 3

Bench 6

Syed Ahmed 108946653

Neil Vaitoonkiat 108333354

Verification Strategy Design Task 1:

1. Initialize Vector Table:

- Use directive .org reset (address $0000), in order to jump to initialization instructions of ports and timer overflow.
- Use directive .org OVF0addr (address $0012), in order to call the overflow interrupt and jump to the mux_display. The overflow interrupt calls the mux_display after overflow occurs.

2. Initialize Ports:

- Insert all zeroes except for bit 7 and output to DDRD to make all inputs except for pin 7 of port d.
- Clear flip flop by using “cbi portd, 4” and then “sbi” the same pin so the flip flop is reset.
- Place all 1’s in port B and output it to DDRB to make port B and output port.
- Insert $7F into a register and output it to DDRA to initialize the transistors used for multiplexing.

3. Initialization of Interrupt:

- First initialize the stack pointer to point to the ramend of the SRAM address space.
- Set the prescaler for the clock by outputting $84 to adcsra and outputting $07 to admux. By outputting $84 to adcsra the prescaler is set to clock divided by a factor of 16.
- Set up the internal interrupt/ timer overflow by outputting $02 to TCCR0 and then enabling the timer overflow by setting TOIE0 to 1.
- Enable the global interrupt vector by using “sei”.

4. Polling:

- Program waits for ADIF to be set or else jumps back in an infinite loop.
- Once ADIF is set, obtain the value in ADCL and mask it so that only the first four bits of the least significant nibble are show.
- Transfer that binary number to the bcd_7seg table to receive the number in hexadecimal.
- Transfer hex values into specific register to be shown by dig0, dig1, and dig2.

5. Mux_display:

- Push every register into the stack so the subroutine doesn’t change and values of the registers or the SREG upon return.
- Enable the transistor and output the value on each display while calling the var_delay to cause the fast flickering.
- Check to see if mux displays the correct value

6. Dew point measurement:

- Set TEC, thermo-electric cooler is turned on and is first heated for 5 seconds
- Clear TEC and thermo-electric cooler is cooled turn on LED
- Poll DPS until it’s set, LED turns off
- Go back to main program

```
AVRASM ver. 2.1.42  C:\Users\Syed\Desktop\380 LAB 6\lab10\lab10.asm Wed Dec 04 19:01:46 20
13

C:\Users\Syed\Desktop\380 LAB 6\lab10\lab10.asm(40): Including file 'C:\Program Files (x86
)\Atmel\AVR Tools\AvrAssembler2\Appnotes\m16def.inc'
C:\Users\Syed\Desktop\380 LAB 6\lab10\lab10.asm(121): warning: Register r23 already define
d by the .DEF directive


                 ;*
                 ;* "Dew_measurement" - Measuring dew
                 ;*
                 ;* Description: Program measures the ambient
                 ;* temperature first for a few seconds,
                 ;* the program than starts cooling the TEC
                 ;* mirror until there's condensation on it,
                 ;* the program keeps polling PD6 ( DPS)
                 ;* until DPS is high, once DPS is set the
                 ;* LED is turned off than dew point meas is
                 ;* complete.
                 ;*
                 ;*
                 ;* Author: Syed Ahmed and Neil Vaitoonkait
                 ;* Version: 0.0

                 ;*
                 ;* Parameters: Push buttons and associated ADC
                 ;* push button 7 used to select ADC 7
                 ;* push button 6 used to select ADC 6
                 ;* push button 5 used to select ADC 5
                 ;*
                 ;*
                 ;* Subroutine hex_2_7seg called
                 ;* Subroutine mux_diplay called
                 ;* Subroutine var delay called
                 ;*
                 ;* Notes: 0s turn on digits and 0s turn on segments
                 ;* The segments are a through g at PB6 through
                 ;* PB0 respectively. The digit drivers are
                 ;* PA2 through PA0 for digits dig2 through dig0.
                 ;* The values inside each register gets changed
                 ;* permanantely.
                 ;* registers used: r16, r17, r24
                 ;* global registers:
                 ;*********************************************************************
***

                 .list

                 reset:
                    .org RESET
000000 c012         rjmp start
                    .org int1addr
000004 c04e         rjmp keypress_isr
                    .org $12
000012 c067         rjmp mux_display
                 start:
                    //make pin4 output, else input
000013 eb00         ldi r16, 0b10110000
000014 bb01         out ddrd, r16 ;make portd 10110000
000015 ef0f         ldi r16, $ff
000016 bb07         out ddrb, r16 ;make portb output
                    //activation of transistors
000017 e007         ldi r16, $07  ;output for transistors, bit 7 for ADC
000018 bb0a         out ddra, r16 ;make pa0-pa2 output
000019 eb00         ldi r16, 0b10110000 ;
00001a bb04         out ddrc, r16
```

```
                        //stack
00001b e50f             ldi r16, LOW(RAMEND) ;low byte
00001c bf0d             out SPL, r16
00001d e004             ldi r16, HIGH(RAMEND);high byte
00001e bf0e             out SPH, r16
                        ;read positive trigger
00001f e00c             ldi r16, (1<<ISC11)|(1<<ISC10) ;sense rising edge
000020 bf05             out MCUCR, r16
000021 e800             ldi r16, 1<<INT1  ;enable interrupt 1
000022 bf0b             out GICR, r16
                    ;****************Enable ADC*****************
000023 e814             ldi r17, $84     ;enable adc with internal reference volt
000024 b916             out adcsra, r17
000025 ec17             ldi r17, $C7    ;use internal reference voltage
000026 b917             out ADMUX, r17

                    ;* registers used: r17, r25
                    display_post:
000027 ef18             ldi r17, $f8  ;make all leds display 8
000028 bb1b             out PORTA, r17
000029 e090             ldi r25, $00
00002a bb98             out portb, r25
00002b 940e 009a        call var_delay1 ;1 second delay
00002d ef1f             ldi r17, $ff ;turn off all leds
00002e bb1b             out PORTA, r17

                    ;****************Timer Overflow*************
00002f e081             ldi r24, 1<<TOV0 ;enable timeroverflow
000030 bf88             out TIFR, r24
000031 e082             ldi r24, $02
000032 bf83             out TCCR0, r24
000033 e081             ldi r24, 1<<TOIE0 ;set timer overflow flag
000034 bf89             out TIMSK, r24
000035 9478             sei

                    ;* registers used: r16,
                    ;* global: r0,r1,r2
                    ADC_loop:
000036 9478             sei     ;enable interrupt calls
000037 9a36             sbi ADCSRA, ADSC ;start conversion
                    ADC_polling:
000038 9b34             sbis ADCSRA, ADIF ;wait for conversion end
000039 cffe             rjmp ADC_polling ;keep polling
00003a 9a34             sbi ADCSRA, ADIF ;stop conversion
                        //division
00003b b1a4             in r26, ADCL  ;load ADCL
00003c b1b5             in r27, ADCH  ;load ADCH
00003d e090             ldi r25, $00 ;0 for decimal

00003e 95b6             lsr r27     ;shift most significant to right
00003f 95a7             ror r26     ;shift least significant to right of decimal
000040 9597             ror r25     ;

000041 95b6             lsr r27    ;shift again for *4
000042 95a7             ror r26     ;shit again
000043 9597             ror r25     ;shift again

                        ;***** Subroutine Register Variables

                    .def   fbin    =r23        ;8-bit binary value
                    .def   tBCDL   =r23        ;BCD result MSD
                    .def   tBCDH   =r17        ;BCD result LSD

000044 2f7a             mov fbin, r26           ;moves r26 into fbin for BCD

000045 d05f             rcall bin2bcd8          ;calls BCD subroutine
000046 2f01             mov r16, r17            ;copy bcd value
000047 700f             andi r16, $0f           ;mask unneeded bits
```

```
000048 d020         rcall bcd_7seg          ;get segmentted value
                    //value for r0
000049 2e20         mov r2, r16             ;send value to r2

00004a 2f07         mov r16, r23            ;copy bcd value
                    ;swap r16               ;make unneeded bits
00004b 700f         andi r16, $0f           ;make unneeded bits
00004c d01c         rcall bcd_7seg          ;get segmentted value
00004d 2e10         mov r1, r16             ;send value to r1

00004e b105         in r16, ADCH            ;read high byte
00004f 7003         andi r16, $03           ;send
000050 d018         rcall bcd_7seg          ;get segmentted value
                    //value for r2
000051 2e00         mov r0, r16             ;send to mux_display
000052 cfe3         rjmp ADC_loop           ;jump back



                    ;*****************************************************
                    ;*
                    ;* "keypress_isr"- Interrupt for keypress
                    ;* ATMega16
                    ;*
                    ;* Description: Detects a keypress from encoder
                    ;* connected to EO of encoder when key press detected
                    ;* the program jumps to ISR for measurement, the
                    ;* program compares the value entered and goes to the
                    ;* correct measurand program
                    ;* Author: Syed Ahmed and Neil Vaitoonkait
                    ;* Version: 0.0
                    ;*
                    ;*
                    ;* Parameters:
                    ;* Usage of the SREG I register
                    ;* r19 to load PIND value
                    ;* reti
                    ;*****************************************************
                    keypress_isr:
000053 932f            push r18             ;copy register
000054 b72f            in r18, SREG         ;copy sreg
000055 932f            push r18             ;copy register

000056 b330            in r19, PIND         ;read pin
000057 7037            andi r19, $07        ;mask bits

000058 9a94            sbi Portd, 4         ;turn on TEC DPH

000059 3030            cpi r19, $00         ;read push button
00005a f021            breq dew_point       ;branch to dew_point

                    return:
00005b 912f            pop r18              ;restore register
00005c bf2f            out SREG, r18        ;restore register
00005d 912f            pop r18              ;restore register
00005e 9518            reti                 ;return to where interrupt

                    dew_point:
00005f 9895            cbi portd, 5         ;heat mirror
000060 d039            rcall var_delay1     ;delay of 1 second
000061 d038            rcall var_delay1     ;delay of 1 second
000062 9a95            sbi portd, 5         ;start cooling the mirror
                    cooling:
000063 9b86            sbis pind, 6         ;wait for DPS signal to be high
000064 cffe            rjmp cooling         ;keep polling
000065 98ae            cbi portc, 6         ;turn on LED
000066 ecd7            ldi r29, $c7         ;turn on adc7
000067 b9d7            out admux, r29       ;turn on adc7
```

```
000068 cff2         rjmp return         ;return back
                ;*****************************************************
                ;*
                ;* "bcd_7seg" - Subroutine converts value into segment
                ;* ATMega16
                ;*
                ;* Description: obtains value from adc conversion turns
                ;* them into segment value used for displaying the LED
                ;* the values are loaded onto a table
                ;* Author: Syed Ahmed and Neil Vaitoonkait
                ;* Version: 0.0
                ;*
                ;*
                ;* Parameters:
                ;* r16 - value to be converted
                ;* r30 low z pointer
                ;* r31 high z pointer
                ;*
                ;*****************************************************

                bcd_7seg:
000069 931f         push r17        ;copy register
00006a e0f0         ldi ZH, high(hextable * 2)  ;load high byte
00006b eee4         ldi ZL, low (hextable * 2)  ;load high byte
00006c e010         ldi r17, $00        ;load 0's
00006d 1ff1         adc ZH, r17         ;load 0's to zh
00006e 0fe0         add ZL, r16         ;read the value coming in
00006f 9104         lpm r16, Z          ;load hex value from table
000070 911f         pop r17             ;restore
000071 9508         ret                 ;return back
000072 4f01
000073 0612
000074 244c
000075 0f60
000076 0c00
000077 8008
000078 81b1
000079 b8b0         hextable: .db $01, $4f, $12, $06, $4c, $24, $60, $0f, $00, $0c, $08, $
80, $b1, $81, $b0, $b8

                ;*****************************************************
                ;*
                ;* "mux_display" - subroutine for display
                ;*
                ;* Description: subroutine outputs value to portb
                ;* value is multiplexed so digits are visible each
                ;* digit is turned on with its respective bcd value
                ;* delay fixes the flicker frequency
                ;*
                ;* Author: Syed Ahmed Neil Vaitoonkait
                ;*
                ;* Parameters:
                ;* r0 - digit 0
                ;* r1 - digit 1
                ;* r2 - digit 2
                ;* r20 - counter
                ;*
                ;*****************************************************
                mux_display:
00007a 93cf         push r28        ;copy register
00007b b7cf         in r28, SREG    ;copy sreg
00007c 93cf         push r28        ;copy register
00007d 934f         push r20        ;copy register

00007e ef4e         ldi r20, $FE        ;turn on porta transistor
00007f bb4b         out PORTA, r20      ;turn on porta
000080 ba08         out PORTB, r0       ;output value for r0
                    //dig0
```

```
000081 d00d         rcall var_delay     ;delay for mux display
000082 ef4d         ldi r20, $fd        ;turn on porta transistor
000083 bb4b         out porta, r20      ;turn on porta transistor
000084 ba18         out portb, r1       ;output value for r1
                    //dig1
000085 d009         rcall var_delay     ;delay for mux display
000086 ef4b         ldi r20, $fb        ;turn on porta transistor
000087 bb4b         out porta, r20      ;turn on porta transistor
000088 ba28         out portb, r2       ;output value for r2
                    //dig2
000089 d005         rcall var_delay     ;delay for mux display
00008a 914f         pop r20             ;restore register
00008b 91cf         pop r28             ;restore register
00008c bfcf         out SREG, r28       ;restore sreg
00008d 91cf         pop r28             ;restore sreg
00008e 9518         reti                ;return for isr
                ;*****************************************************
                ;*
                ;* "var_delay" - subroutine for delay
                ;*
                ;* Description: subroutine creates a delay so there's
                ;* no flicker frequency and displays are lit with no
                ;* ghosting the delay is created through occupying
                ;* clock cycles
                ;*
                ;* Author: Syed Ahmed Neil Vaitoonkait
                ;*
                ;* Parameters:
                ;* r21 - outter loop
                ;* r27 - innter loop
                ;*
                ;*****************************************************
                var_delay:
00008f 935f         push r21        ;copy register
000090 93bf         push r27        ;copy register
000091 e25d         ldi r21, 45     ;45 decrements
                outter_loop:
000092 e2bd         ldi r27, 45     ;decrement until 0
                inner_loop:
000093 95ba         dec r27         ;decrement until 0
000094 f7f1         brne inner_loop ;branch out if 0
000095 955a         dec r21         ;decrement until 0
000096 f7d9         brne outter_loop ;branch out if 0
000097 91bf         pop r27         ;restore register
000098 915f         pop r21         ;restore register
000099 9508         ret             ;return to where subroutine called

                var_delay1:
00009a 931f             push r17    ;copy register
00009b 930f             push r16    ;copy register
00009c ef16             ldi r17, 246 ;246 decrements
                    outer_loop1:
00009d ef06             ldi r16, 246 ;246 decrements
                    inner_loop1:
00009e 950a             dec r16     ;keep decrementing
00009f f7f1             brne inner_loop1 ;branch out when 0
0000a0 951a             dec r17     ;keep decrementing
0000a1 f7d9             brne outer_loop1    ;branch out when 0
0000a2 910f             pop r16     ;restore register
0000a3 911f             pop r17     ;restore register
0000a4 9508             ret         ;return subroutine
                ;*********************************************************************
***
                ;* Author: ATMEL
                ;* Modified/Used by: Syed Ahmed & Neil Vaitoonkait
                ;* "mpy8u" - 8x8 Bit Unsigned Multiplication
                ;*
                ;* This subroutine multiplies the two register variables mp8u and mc8u.
```

```
                ;* The result is placed in registers m8uH, m8uL
                ;*
                ;* Number of words :9 + return
                ;* Number of cycles    :58 + return
                ;* Low registers used  :None
                ;* High registers used  :4 (mp8u,mc8u/m8uL,m8uH,mcnt8u)
                ;*
                ;* Note: Result Low byte and the multiplier share the same register.
                ;* This causes the multiplier to be overwritten by the result.
                ;*
                ;*********************************************************************
***

                bin2bcd8:
0000a5 2711        clr tBCDH       ;clear result MSD
0000a6 507a      bBCD8_1:subi   fbin,10     ;input = input - 10
0000a7 f010        brcs    bBCD8_2     ;abort if carry set
0000a8 9513        inc tBCDH       ;inc MSD
                ;---------------------------------------------------------------------
---
                ;              ;Replace the above line with this one
                ;              ;for packed BCD output
                ;  subi    tBCDH,-$10  ;tBCDH = tBCDH + 10
                ;---------------------------------------------------------------------
---
0000a9 cffc        rjmp    bBCD8_1     ;loop again
0000aa 5f76      bBCD8_2:subi   fbin,-10    ;compensate extra subtraction
                ;---------------------------------------------------------------------
---
                ;              ;Add this line for packed BCD output
                ;  add fbin,tBCDH
                ;---------------------------------------------------------------------
---
0000ab 9508        ret
```

RESOURCE USE INFORMATION

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

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

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

```
ATmega16 register use summary:
r0 :   2 r1 :   2 r2 :   2 r3 :   0 r4 :   0 r5 :   0 r6 :   0 r7 :   0
r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   0 r13:   0 r14:   0 r15:   0
r16:  31 r17:  19 r18:   6 r19:   3 r20:   8 r21:   4 r22:   0 r23:   4
r24:   6 r25:   5 r26:   4 r27:   7 r28:   6 r29:   2 r30:   2 r31:   2
x  :   0 y  :   0 z  :   1
Registers used: 19 out of 35 (54.3%)

ATmega16 instruction use summary:
.lds  :   0 .sts  :   0 adc   :   1 add   :   1 adiw  :   0 and   :   0
andi  :   4 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   1 break :   0 breq  :   1 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :   4 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   1 cbi   :   2 cbr   :   0
clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   1 cls   :   0
clt   :   0 clv   :   0 clz   :   0 com   :   0 cp    :   0 cpc   :   0
cpi   :   1 cpse  :   0 dec   :   4 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :   6 inc   :   1 jmp   :   0
ld    :   0 ldd   :   0 ldi   :  28 lds   :   0 lpm   :   2 lsl   :   0
lsr   :   2 mov   :   6 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   0 or    :   0 ori   :   0 out   :  25 pop   :  10
push  :  10 rcall :   9 ret   :   4 reti  :   2 rjmp  :   8 rol   :   0
ror   :   4 sbc   :   0 sbci  :   0 sbi   :   4 sbic  :   0 sbis  :   2
sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   0 sec   :   0 seh   :   0
sei   :   2 sen   :   0 ser   :   0 ses   :   0 set   :   0 sev   :   0
sez   :   0 sleep :   0 spm   :   0 st    :   0 std   :   0 sts   :   0
sub   :   0 subi  :   2 swap  :   0 tst   :   0 wdr   :   0
Instructions used: 29 out of 113 (25.7%)

ATmega16 memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x000158    296     16    312   16384   1.9%
[.dseg] 0x000060 0x000060      0      0      0    1024   0.0%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors, 1 warnings
```

```
AVRASM ver. 2.1.42  C:\Users\Syed\Desktop\380 LAB 6\lab10\lab10.asm Wed Dec 04 19:31:34 20
13

C:\Users\Syed\Desktop\380 LAB 6\lab10\lab10.asm(40): Including file 'C:\Program Files (x86
)\Atmel\AVR Tools\AvrAssembler2\Appnotes\m16def.inc'


                 ;*
                 ;* "DewPoint_cycling" - Measuring dew
                 ;*
                 ;* Description: Program measures the ambient
                 ;* temperature first for a few seconds,
                 ;* the program than starts cooling the TEC
                 ;* mirror until there's condensation on it,
                 ;* the program keeps polling PD6 ( DPS)
                 ;* until DPS is high, once DPS is set the
                 ;* LED is turned off than dew point meas is
                 ;* complete.
                 ;*
                 ;*
                 ;* Author: Syed Ahmed and Neil Vaitoonkait
                 ;* Version: 0.0

                 ;*
                 ;* Parameters: Push buttons and associated ADC
                 ;* push button 7 used to select ADC 7
                 ;* push button 6 used to select ADC 6
                 ;* push button 5 used to select ADC 5
                 ;*
                 ;*
                 ;* Subroutine hex_2_7seg called
                 ;* Subroutine mux_diplay called
                 ;* Subroutine var delay called
                 ;*
                 ;* Notes: 0s turn on digits and 0s turn on segments
                 ;* The segments are a through g at PB6 through
                 ;* PB0 respectively. The digit drivers are
                 ;* PA2 through PA0 for digits dig2 through dig0.
                 ;* The values inside each register gets changed
                 ;* permanantely.
                 ;* registers used: r16, r17, r24
                 ;* global registers:
                 ;*********************************************************************
***

                 .list


                 reset:
                    .org RESET
000000 c00a         rjmp start
                    .org int1addr
000004 c049         rjmp keypress_isr
                    .org 0x0A
00000a c05b         rjmp slow_decrease
                 start:
                    //make pin4 output, else input
00000b eb00         ldi r16, 0b10110000
00000c bb01         out ddrd, r16 ;make portd 10110000
00000d ef0f         ldi r16, $ff
00000e bb07         out ddrb, r16 ;make portb output
                    //activation of transistors
00000f e017         ldi r17, $07  ;output for transistors, bit 7 for ADC
000010 bb0a         out ddra, r16 ;make pa0-pa2 output
000011 eb10         ldi r17, 0b10110000 ;
                    ;out ddrc, r17
                    //stack
```

```
                        ldi r17, LOW(RAMEND) ;low byte
                        out SPL, r17
                        ldi r16, HIGH(RAMEND);high byte
                        out SPH, r17
                        ;read positive trigger
                        ldi r16, (1<<ISC11)|(1<<ISC10) ;sense rising edge
                        out MCUCR, r16
                        ldi r16, 1<<INT1  ;enable interrupt 1
                        out GICR, r16
                    ;****************Enable ADC*****************
000012 e814             ldi r17, $84     ;enable adc with internal reference volt
000013 b916             out adcsra, r17
000014 ec17             ldi r17, $C7    ;use internal reference voltage
000015 b917             out ADMUX, r17

                    ;* registers used: r17, r25
                    display_post:
000016 ef18             ldi r17, $f8  ;make all leds display 8
000017 bb1b             out PORTA, r17
000018 e090             ldi r25, $00
000019 bb98             out portb, r25
00001a 940e 009d        call var_delay1 ;1 second delay
00001c ef1f             ldi r17, $ff ;turn off all leds
00001d bb1b             out PORTA, r17

                    ;****************Timer0 Overflow*************
00001e e081             ldi r24, 1<<TOV0 ;enable timeroverflow
00001f bf88             out TIFR, r24
000020 e082             ldi r24, $02
000021 bf83             out TCCR0, r24
000022 e081             ldi r24, 1<<TOIE0 ;set timer overflow flag
000023 bf89             out TIMSK, r24
000024 9478             sei

                    ;****************Timer1 Overflow*************
000025 ef0f             ldi r16, 0xFF
000026 bd0b             out OCR1AH, r16
000027 ef0f             ldi r16, 0xFF
000028 bd0a             out OCR1AL, r16 ; sets 0x0FFF as TOP value

000029 ef0f             ldi r16, 0xFF
00002a bd09             out OCR1BH, r16
00002b ef0f             ldi r16, 0xFF
00002c bd08             out OCR1BL, r16 ; duty cycle begins at ~100%

00002d e203             ldi r16, 0x23 ; 00100011
00002e bd0f             out TCCR1A, r16 ; non inverting & mode 15
00002f e50a             ldi r16, 0x5A ; 01011010
000030 bd0e             out TCCR1B, r16 ; rising edge input capture, mode 14 & clk/8

                    ;* registers used: r16,
                    ;* global: r0,r1,r2
                    ADC_loop:
000031 9478             sei     ;enable interrupt calls
000032 9a36             sbi ADCSRA, ADSC ;start conversion
                    ADC_polling:
000033 9b34             sbis ADCSRA, ADIF ;wait for conversion end
000034 cffe             rjmp ADC_polling ;keep polling
000035 9a34             sbi ADCSRA, ADIF ;stop conversion
                        //division
000036 b1a4             in r26, ADCL  ;load ADCL
000037 b1b5             in r27, ADCH  ;load ADCH
000038 e090             ldi r25, $00 ;0 for decimal

000039 95b6             lsr r27     ;shift most significant to right
00003a 95a7             ror r26     ;shift least significant to right of decimal
00003b 9597             ror r25     ;
```

```
00003c 95b6         lsr r27    ;shift again for *4
00003d 95a7         ror r26     ;shit again
00003e 9597         ror r25     ;shift again

                    ;***** Subroutine Register Variables

                .def   fbin    =r23        ;8-bit binary value
                .def   tBCDL   =r21        ;BCD result MSD
                .def   tBCDH   =r17        ;BCD result LSD

00003f 2f7a         mov fbin, r26          ;moves r26 into fbin for BCD

000040 d067         rcall bin2bcd8         ;calls BCD subroutine
000041 2f01         mov r16, r17           ;copy bcd value
000042 700f         andi r16, $0f          ;mask unneeded bits
000043 d028         rcall bcd_7seg         ;get segmentted value
                    //value for r0
000044 2e20         mov r2, r16            ;send value to r2

000045 2f07         mov r16, r23           ;copy bcd value
                    ;swap r16              ;make unneeded bits
000046 700f         andi r16, $0f          ;make unneeded bits
000047 d024         rcall bcd_7seg         ;get segmentted value
000048 2e10         mov r1, r16            ;send value to r1

000049 b105         in r16, ADCH           ;read high byte
00004a 7003         andi r16, $03          ;send
00004b d020         rcall bcd_7seg         ;get segmentted value
                    //value for r2
00004c 2e00         mov r0, r16            ;send to mux_display
00004d cfe3         rjmp ADC_loop          ;jump back



                ;*****************************************************
                ;*
                ;* "keypress_isr"- Interrupt for keypress
                ;* ATMega16
                ;*
                ;* Description: Detects a keypress from encoder
                ;* connected to EO of encoder when key press detected
                ;* the program jumps to ISR for measurement, the
                ;* program compares the value entered and goes to the
                ;* correct measurand program
                ;* Author: Syed Ahmed and Neil Vaitoonkait
                ;* Version: 0.0
                ;*
                ;*
                ;* Parameters:
                ;* Usage of the SREG I register
                ;* r19 to load PIND value
                ;* reti
                ;*****************************************************
                keypress_isr:
00004e 932f         push r18            ;copy register
00004f b72f         in r18, SREG        ;copy sreg
000050 932f         push r18            ;copy register

000051 b330         in r19, PIND        ;read pin
000052 7037         andi r19, $07       ;mask bits

000053 9a94         sbi Portd, 4        ;turn on TEC DPH

000054 3030         cpi r19, $00        ;read push button
000055 f031         breq dew_point      ;branch to dew_point

000056 3031         cpi r19, $01
000057 f071         breq slow_decrease
```

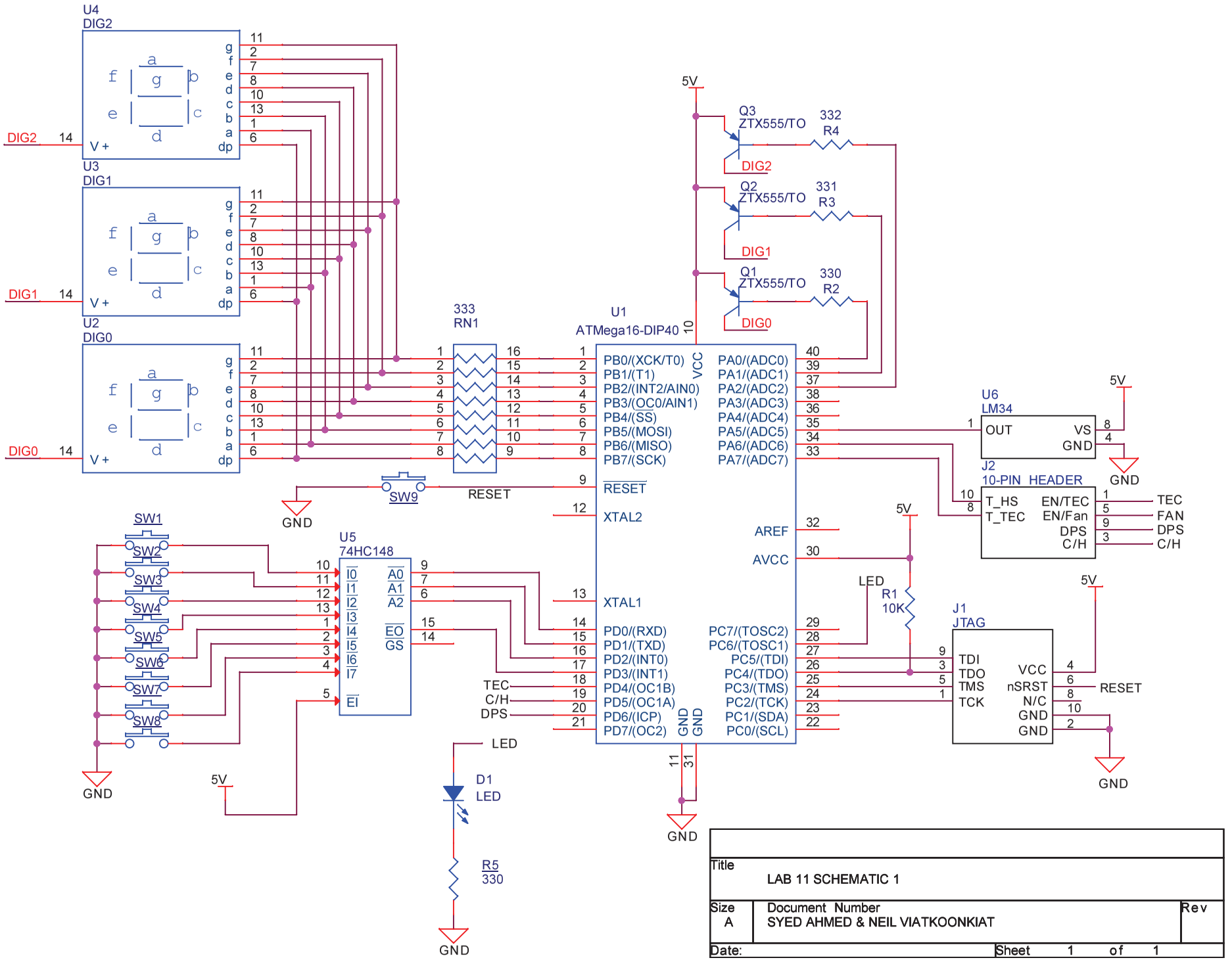
```
                return:
000058 912f        pop r18             ;restore register
000059 bf2f        out SREG, r18       ;restore register
00005a 912f        pop r18             ;restore register
00005b 9518        reti                ;return to where interrupt

                dew_point:
00005c 9895        cbi portd, 5        ;heat mirror
00005d d03f        rcall var_delay1    ;delay of 1 second
00005e d03e        rcall var_delay1    ;delay of 1 second
00005f 9a95        sbi portd, 5        ;start cooling the mirror
                cooling:
000060 9b86        sbis pind, 6        ;wait for DPS signal to be high
000061 cffe        rjmp cooling        ;keep polling
000062 98ae        cbi portc, 6        ;turn on LED
000063 ecd7        ldi r29, $c7        ;turn on adc7
000064 b9d7        out admux, r29      ;turn on adc7
000065 cff2        rjmp return         ;return back

                slow_decrease:
000066 930f        push r16

                   //****** when single stepping in simulation, OCR1BH & OCR1BL are swapp
ed!
000067 ed03        ldi r16, 0xD3
000068 bd09        out OCR1BH, r16
000069 eb00        ldi r16, 0xB0
00006a bd08        out OCR1BL, r16 ; changes duty cycle to ~70% to slow down the cooling


00006b 910f        pop r16



                ;*****************************************************
                ;*
                ;* "bcd_7seg" - Subroutine converts value into segment
                ;* ATMega16
                ;*
                ;* Description: obtains value from adc conversion turns
                ;* them into segment value used for displaying the LED
                ;* the values are loaded onto a table
                ;* Author: Syed Ahmed and Neil Vaitoonkait
                ;* Version: 0.0
                ;*
                ;*
                ;* Parameters:
                ;* r16 - value to be converted
                ;* r30 low z pointer
                ;* r31 high z pointer
                ;*
                ;*****************************************************

                bcd_7seg:
00006c 931f        push r17        ;copy register
00006d e0f0        ldi ZH, high(hextable * 2)  ;load high byte
00006e eeea        ldi ZL, low (hextable * 2)  ;load high byte
00006f e010        ldi r17, $00        ;load 0's
000070 1ff1        adc ZH, r17         ;load 0's to zh
000071 0fe0        add ZL, r16         ;read the value coming in
000072 9104        lpm r16, Z          ;load hex value from table
000073 911f        pop r17             ;restore
000074 9508        ret                 ;return back
000075 4f01
000076 0612
000077 244c
000078 0f60
```

```
000079 0c00
00007a 8008
00007b 81b1
00007c b8b0         hextable: .db $01, $4f, $12, $06, $4c, $24, $60, $0f, $00, $0c, $08, $
80, $b1, $81, $b0, $b8

                ;*****************************************************
                ;*
                ;* "mux_display" - subroutine for display
                ;*
                ;* Description: subroutine outputs value to portb
                ;* value is multiplexed so digits are visible each
                ;* digit is turned on with its respective bcd value
                ;* delay fixes the flicker frequency
                ;*
                ;* Author: Syed Ahmed Neil Vaitoonkait
                ;*
                ;* Parameters:
                ;* r0 - digit 0
                ;* r1 - digit 1
                ;* r2 - digit 2
                ;* r20 - counter
                ;*
                ;*****************************************************
                mux_display:
00007d 93cf         push r28        ;copy register
00007e b7cf         in r28, SREG    ;copy sreg
00007f 93cf         push r28        ;copy register
000080 934f         push r20        ;copy register

000081 ef4e         ldi r20, $FE        ;turn on porta transistor
000082 bb4b         out PORTA, r20      ;turn on porta
000083 ba08         out PORTB, r0       ;output value for r0
                    //dig0
000084 d00d         rcall var_delay     ;delay for mux display
000085 ef4d         ldi r20, $fd        ;turn on porta transistor
000086 bb4b         out porta, r20      ;turn on porta transistor
000087 ba18         out portb, r1       ;output value for r1
                    //dig1
000088 d009         rcall var_delay     ;delay for mux display
000089 ef4b         ldi r20, $fb        ;turn on porta transistor
00008a bb4b         out porta, r20      ;turn on porta transistor
00008b ba28         out portb, r2       ;output value for r2
                    //dig2
00008c d005         rcall var_delay     ;delay for mux display
00008d 914f         pop r20             ;restore register
00008e 91cf         pop r28             ;restore register
00008f bfcf         out SREG, r28       ;restore sreg
000090 91cf         pop r28             ;restore sreg
000091 9518         reti                ;return for isr
                ;*****************************************************
                ;*
                ;* "var_delay" - subroutine for delay
                ;*
                ;* Description: subroutine creates a delay so there's
                ;* no flicker frequency and displays are lit with no
                ;* ghosting the delay is created through occupying
                ;* clock cycles
                ;*
                ;* Author: Syed Ahmed Neil Vaitoonkait
                ;*
                ;* Parameters:
                ;* r21 - outter loop
                ;* r27 - innter loop
                ;*
                ;*****************************************************
                var_delay:
000092 935f         push r21        ;copy register
```

```
000093 93bf        push r27        ;copy register
000094 e25d        ldi r21, 45     ;45 decrements
                outter_loop:
000095 e2bd        ldi r27, 45     ;decrement until 0
                inner_loop:
000096 95ba        dec r27         ;decrement until 0
000097 f7f1        brne inner_loop ;branch out if 0
000098 955a        dec r21         ;decrement until 0
000099 f7d9        brne outter_loop ;branch out if 0
00009a 91bf        pop r27         ;restore register
00009b 915f        pop r21         ;restore register
00009c 9508        ret             ;return to where subroutine called

                var_delay1:
00009d 931f            push r17    ;copy register
00009e 930f            push r16    ;copy register
00009f ef16            ldi r17, 246 ;246 decrements
                   outer_loop1:
0000a0 ef06            ldi r16, 246 ;246 decrements
                   inner_loop1:
0000a1 950a            dec r16     ;keep decrementing
0000a2 f7f1            brne inner_loop1 ;branch out when 0
0000a3 951a            dec r17     ;keep decrementing
0000a4 f7d9            brne outer_loop1    ;branch out when 0
0000a5 910f            pop r16     ;restore register
0000a6 911f            pop r17     ;restore register
0000a7 9508            ret         ;return subroutine
                ;***************************************************************************
***
                ;* Author: ATMEL
                ;* Modified/Used by: Syed Ahmed & Neil Vaitoonkait
                ;* "mpy8u" - 8x8 Bit Unsigned Multiplication
                ;*
                ;* This subroutine multiplies the two register variables mp8u and mc8u.
                ;* The result is placed in registers m8uH, m8uL
                ;*
                ;* Number of words :9 + return
                ;* Number of cycles    :58 + return
                ;* Low registers used  :None
                ;* High registers used  :4 (mp8u,mc8u/m8uL,m8uH,mcnt8u)
                ;*
                ;* Note: Result Low byte and the multiplier share the same register.
                ;* This causes the multiplier to be overwritten by the result.
                ;*
                ;***************************************************************************
***

                bin2bcd8:
0000a8 2711        clr tBCDH       ;clear result MSD
0000a9 507a     bBCD8_1:subi   fbin,10     ;input = input - 10
0000aa f010        brcs    bBCD8_2     ;abort if carry set
0000ab 9513        inc tBCDH       ;inc MSD
                ;-----------------------------------------------------------------------
---
                ;              ;Replace the above line with this one
                ;              ;for packed BCD output
                ;  subi    tBCDH,-$10  ;tBCDH = tBCDH + 10
                ;-----------------------------------------------------------------------
---
0000ac cffc        rjmp    bBCD8_1     ;loop again
0000ad 5f76     bBCD8_2:subi   fbin,-10    ;compensate extra subtraction
                ;-----------------------------------------------------------------------
---
                ;              ;Add this line for packed BCD output
                ;  add fbin,tBCDH
                ;-----------------------------------------------------------------------
---
0000ae 9508        ret
```

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

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

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

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

ATmega16 register use summary:
r0 :   2 r1 :   2 r2 :   2 r3 :   0 r4 :   0 r5 :   0 r6 :   0 r7 :   0
r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   0 r13:   0 r14:   0 r15:   0
r16:  38 r17:  21 r18:   6 r19:   4 r20:   8 r21:   4 r22:   0 r23:   4
r24:   6 r25:   5 r26:   4 r27:   7 r28:   6 r29:   2 r30:   2 r31:   2
x  :   0 y  :   0 z  :   1
Registers used: 19 out of 35 (54.3%)

ATmega16 instruction use summary:
.lds  :   0 .sts  :   0 adc   :   1 add   :   1 adiw  :   0 and   :   0
andi  :   4 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   1 break :   0 breq  :   2 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :   4 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   1 cbi   :   2 cbr   :   0
clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   1 cls   :   0
clt   :   0 clv   :   0 clz   :   0 com   :   0 cp    :   0 cpc   :   0
cpi   :   2 cpse  :   0 dec   :   4 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :   6 inc   :   1 jmp   :   0
ld    :   0 ldd   :   0 ldi   :  32 lds   :   0 lpm   :   2 lsl   :   0
lsr   :   2 mov   :   6 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   0 or    :   0 ori   :   0 out   :  28 pop   :  11
push  :  11 rcall :   9 ret   :   4 reti  :   2 rjmp  :   8 rol   :   0
ror   :   4 sbc   :   0 sbci  :   0 sbi   :   4 sbic  :   0 sbis  :   2
sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   0 sec   :   0 seh   :   0
sei   :   2 sen   :   0 ser   :   0 ses   :   0 set   :   0 sev   :   0
sez   :   0 sleep :   0 spm   :   0 st    :   0 std   :   0 sts   :   0
sub   :   0 subi  :   2 swap  :   0 tst   :   0 wdr   :   0
Instructions used: 29 out of 113 (25.7%)

ATmega16 memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
```

```
[.cseg] 0x000000 0x00015e    318     16    334   16384   2.0%
[.dseg] 0x000060 0x000060      0      0      0    1024   0.0%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors, 0 warnings
```


U4
DIG2
U3
DIG1
U2
DIG0
V +
DIG2
DIG1
DIG0
14
333
RN1
U1
ATMega16-DIP40
PB0/(XCK/T0)
PB1/(T1)
PB2/(INT2/AIN0)
PB3/(OC0/AIN1)
PB4/(SS)
PB5/(MOSI)
PB6/(MISO)
PB7/(SCK)
RESET
XTAL2
XTAL1
PD0/(RXD)
PD1/(TXD)
PD2/(INT0)
PD3/(INT1)
PD4/(OC1B)
PD5/(OC1A)
PD6/(ICP)
PD7/(OC2)
PA0/(ADC0)
PA1/(ADC1)
PA2/(ADC2)
PA3/(ADC3)
PA4/(ADC4)
PA5/(ADC5)
PA6/(ADC6)
PA7/(ADC7)
AREF
AVCC
PC7/(TOSC2)
PC6/(TOSC1)
PC5/(TDI)
PC4/(TDO)
PC3/(TMS)
PC2/(TCK)
PC1/(SDA)
PC0/(SCL)
VCC
GND
5V
Q3
ZTX555/TO
332
R4
Q2
331
R3
Q1
330
R2
SW9
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
U5
74HC148
I0
I1
I2
I3
I4
I5
I6
I7
EI
A0
A1
A2
EO
GS
TEC
C/H
DPS
LED
D1
LED
R5
330
R1
10K
U6
LM34
OUT
VS
GND
J2
10-PIN HEADER
T_HS
T_TEC
EN/TEC
EN/Fan
DPS
C/H
TEC
FAN
DPS
C/H
J1
JTAG
TDI
TDO
TMS
TCK
VCC
nSRST
N/C
GND
RESET
Title
LAB 11 SCHEMATIC 1
Size
A
Document Number
SYED AHMED & NEIL VIATKOONKIAT
Rev
Date:
Sheet 1 of 1