Candace Caimol

**CPE301 – SPRING 2016**

Design Assignment 2

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

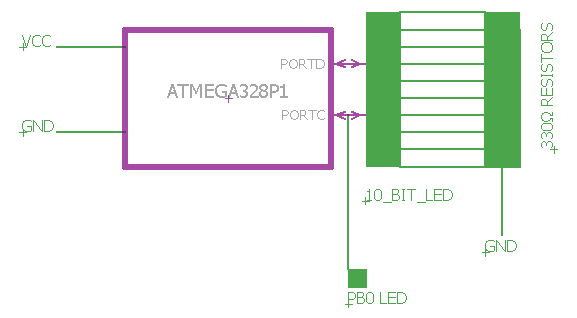
The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 1. | INITIAL CODE OF TASK 1/A |  |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 5. |  |  |  |
| 6. | SCHEMATICS |  |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 8. | SCREENSHOT OF EACH DEMO |  |  |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |

Components Used

* Breadboard
* Atmel Xplained Mini with onboard Atmega328P and programmer
* 10-bit LED bar
* 1 red LED bulb
* 330 Ω resistors
* USB cable to connect to Desktop (power source)



It should be noted that Timer1 was used for the delay in this assignment. To find the contents of TCNT, I used the following equation:

TCNT = 2^N – x/s

* Number of bits in timer = 16
* X = delay \* MC frequency = 0.25s \* 8 MHz
* S = prescaler = 64

Solving this equation and converting to hexadecimal, the contents of TCNT are found to be 0x85EE. Since normal mode is used, TCCR1A is 0x00. TCCR1B is populated with 0x03 due to the prescaler being 64.

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | INITIAL CODE OF TASK 1/A |  |  |

ASSEMBLY CODE

/\*

\* AssemblerApplication1.asm

\*

\* Created: 3/7/2016 1:28:25 PM

\* Author: caimol

\*/

LDI R16, 1 ;R16 = 1 (bit 0)

SBI DDRC, 0 ;Set PC0 to o/p

LDI R17, 0 ;R17 = 0

OUT PORTC, R17 ;initialize PC0 to off

BEGIN:

RCALL DELAY\_SUBR ;call delay

EOR R17, R16 ;toggle bit 0 of R17

OUT PORTC, R17 ;output to PORTC

RJMP BEGIN ;infinitely loop back through main routine

DELAY\_SUBR: ;delay subroutine

LDI R20, 0x85 ;R20 = 0x85

STS TCNT1H, R20 ;load TCNT1H

LDI R20, 0xEE ;R20 = 0xEE

STS TCNT1L, R20 ;load TCNT1L

LDI R20, 0x00 ;R20 = 0x00

STS TCCR1A, R20 ;normal mode

LDI R20, 0x03 ;R20 = 0x03

STS TCCR1B, R20 ;prescaler = 64

AGAIN:

IN R20, TIFR1 ;gets TOV1 flag from TIFR1

SBRS R20, TOV1 ;skip necxt instruction if OV

RJMP AGAIN ;else, loop until OV

LDI R20, 0x00 ;R20 = 0x00

STS TCCR1B, R20 ;reset TCCR1B

LDI R20, 0x01 ;R20 = 0x01 to reset TIFR

OUT TIFR1, R20 ;reset TIFR

RET

C CODE  
/\*

\* DA2T1\_C.c

\*

\* Created: 3/13/2016 2:25:22 PM

\* Author : user

\*/

#include <avr/io.h>

int main(void)

{

DDRC |= 0x01; //enable PB0 as output

PORTC |= (1 << PC0); //initialize PB0 to 1

while (1)

{

TCNT1 = 0x85EE; //starting value of Timer1

TCCR1A = 0x00; //Normal mode

TCCR1B = 0x03; //Prescaler value 64

while (!(TIFR1 & (1<<TOV1))) //while no overflow

{

//keep cycling through timer

}

TCCR1B = 0x00; //reset TCCR1B to 0

TIFR1 = 0x01; //clear overflow flags by loading with 0x01

PORTC ^= 1; //toggle PB0

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |

ASSEMBLY CODE

/\*

\* AssemblerApplication1.asm

\*

\* Created: 3/7/2016 1:28:25 PM

\* Author: caimol

\*/

;;;;;;;;initializations for counter;;;;;;;;;;

SBI DDRD, 0 ;enable PB0 as output

SBI DDRD, 1 ;enable PB1 as output

SBI DDRD, 2 ;enable PB2 as output

SBI DDRD, 3 ;enable PB3 as output

SBI DDRD, 4 ;enable PB4 as output

SBI DDRD, 5 ;enable PB5 as output

SBI DDRD, 6 ;enable PB6 as output

SBI DDRD, 7 ;enable PB7 as output

LDI R19, 0 ;counter initialized to 0 (R19 is designated register)

OUT PORTD, R19 ;output the counter value

;;;;;;;;initializations for PC0 output (blinks with counter);;;;;;;;;;;;;

LDI R16, 1 ;R16 = 1 (bit 0)

SBI DDRC, 0 ;Set PC0 to o/p

LDI R17, 0 ;R17 = 0

OUT PORTC, R17 ;initialize PC0 to off

START:

RCALL DELAY\_SUBR ;call delay

EOR R17, R16 ;toggle bit 0 of R17

;;;;;;;;;;for counter;;;;;;;;;

OUT PORTC, R17 ;output R17 to PORTC

SBIS PORTC, 0 ;if PB0 is set, skip next instruction

JMP OUTCOUNT ;jump to OUTCOUNT without incrementing

INC R19 ;else, increment current count in counter

OUTCOUNT:

OUT PORTD, R19 ;output current counter value to PORTB

;;;;;;;;;;;;end counter stuff;;;;;;;;;;;;;;

OUT PORTC, R17 ;output to PORTC

RJMP START ;infinitely loop back through main routine

DELAY\_SUBR: ;delay subroutine

LDI R20, 0x85 ;R20 = 0x85

STS TCNT1H, R20 ;load TCNT1H

LDI R20, 0xEE ;R20 = 0xEE

STS TCNT1L, R20 ;load TCNT1L

LDI R20, 0x00 ;R20 = 0x00

STS TCCR1A, R20 ;normal mode

LDI R20, 0x03 ;R20 = 0x03

STS TCCR1B, R20 ;prescaler = 64

AGAIN:

IN R20, TIFR1 ;gets TOV1 flag from TIFR1

SBRS R20, TOV1 ;skip necxt instruction if OV

RJMP AGAIN ;else, loop until OV

LDI R20, 0x00 ;R20 = 0x00

STS TCCR1B, R20 ;reset TCCR1B

LDI R20, 0x01 ;R20 = 0x01 to reset TIFR

OUT TIFR1, R20 ;reset TIFR

RET

C CODE  
/\*

\* DA2\_C.c

\*

\* Created: 3/10/2016 12:35:29 AM

\* Author : user

\*/

#include <avr/io.h>

int main(void)

{

DDRC |= 0x01; //enable PC0 as output

DDRD |= 0xFF; //enable PB0-PB7 as output

PORTC |= (1 << PC0); //initialize PC0 to 0

PORTD = 0x00; //initialize binary counter to 0

while (1)

{

TCNT1 = 0x85EE; //starting value of Timer1

TCCR1A = 0x00; //Normal mode

TCCR1B = 0x03; //Prescaler value

while (!(TIFR1 & (1<<TOV1))) //while no overflow

{

//keep cycling through timer

}

TCCR1B = 0x00; //reset TCCR1B to 0

TIFR1 = 0x01; //clear overflow flags by loading with 0x01

PORTC ^= 1; //toggle PC0

if ((PORTC & (1 << 0)) == 1) //if PC0 is set

{

PORTD += 1; //increment binary counter

}

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |

ASSEMBLY CODE

/\*

\* AssemblerApplication1.asm

\*

\* Created: 3/7/2016 1:28:25 PM

\* Author: caimol

\*/

SBI DDRC, 5 ;enable PC5 as output

SBI DDRC, 4 ;enable PC4 as output

SUB R5, R5 ;initialize count5 to 0

SUB R10, R10 ;initialize count10 to 0

LDI R16, 5 ;R16 = 5

ADD R1, R16 ;used to compare to count5

LDI R16, 10 ;R16 = 10

ADD R2, R16 ;used to compare to count10

LDI R16, 32 ;R16 = 32

ADD R6, R16 ;2^5 (causes PC5 to light up)

LDI R16, 16 ;R16 = 16

ADD R7, R16 ;2^4 (causes PC4 to light up)

SUB R8, R8 ;initialize PC5 to 0 (off)

;;;;;;;;initializations for counter;;;;;;;;;;

SBI DDRD, 0 ;enable PB0 as output

SBI DDRD, 1 ;enable PB1 as output

SBI DDRD, 2 ;enable PB2 as output

SBI DDRD, 3 ;enable PB3 as output

SBI DDRD, 4 ;enable PB4 as output

SBI DDRD, 5 ;enable PB5 as output

SBI DDRD, 6 ;enable PB6 as output

SBI DDRD, 7 ;enable PB7 as output

LDI R19, 0 ;counter initialized to 0 (R19 is designated register)

OUT PORTD, R19 ;output the counter value

;;;;;;;;initializations for PC0 output (blinks with counter);;;;;;;;;;;;;

LDI R16, 1 ;R16 = 1 (bit 0)

SBI DDRC, 0 ;enable PC0 as output

OUT PORTC, R8 ;output 0 to PORTC (initialized as off)

START:

RCALL DELAY\_SUBR ;call delay subroutine

EOR R8, R16 ;XOR to toggle bit 0 of R8

;;;;;;;for counter;;;;;;;;;;;;

OUT PORTC, R8 ;output R8 to PORTC

SBIS PORTC, 0 ;compare current value at PB0 to check if high (rising edge). if so skip next instruction

JMP OUTCOUNT ;jump to OUTCOUNT (don't increment)

INC R19 ;else, increment current count in counter

INC R5 ;increment R5 (count5)

INC R10 ;increment R10 (count10)

CP R5, R1 ;compare count5 to 5

BRLO CHECK10 ;branch if lower than

EOR R8, R6 ;else, toggle PC5

SUB R5, R5 ;reset count5 to 0

CHECK10:

CP R10, R2 ;compare count10 to 10

BRLO OUTCOUNT ;branch if lower than

EOR R8, R7 ;else, toggle PC4

SUB R10, R10 ;reset count10 to 0

OUTCOUNT:

OUT PORTD, R19 ;output current counter value to PORTB

;;;;;;;;;;;;end counter stuff;;;;;;;;;;;;;;

OUT PORTC, R8 ;output value to PORTC

RJMP START ;loop back through main routine

DELAY\_SUBR:

LDI R20, 0x85 ;R20 = 0x85

STS TCNT1H, R20 ;load TCNT1H (top byte of starting value of timer)

LDI R20, 0xEE ;R20 = 0xEE

STS TCNT1L, R20 ;R20 = 0xEE (lower byte of starting value of timer)

LDI R20, 0x00 ;R20 = 0x00

STS TCCR1A, R20 ;0x00 = Normal mode

LDI R20, 0x03 ;R20 = 0x03

STS TCCR1B, R20 ;prescaler = 64

AGAIN:

IN R20, TIFR1 ;gets TOV1 flag from TIFR1

SBRS R20, TOV1 ;skip next instruction if OV

RJMP AGAIN ;else loop until OV

LDI R20, 0x00 ;R20 = 0x00

STS TCCR1B, R20 ;Reset TCCR1B

LDI R20, 0x01 ;R20 = 0x01 to reset TIFR

OUT TIFR1, R20 ;Reset TIFR

RET ;return from function call

C CODE

/\*

\* DA2T3\_C.c

\*

\* Created: 3/13/2016 2:54:53 PM

\* Author : user

\*/

#include <avr/io.h>

int main(void)

{

DDRC |= 0x31; //enable PC0 as output

DDRD |= 0xFF; //enable PB0-PB7 as output

PORTC |= (0 << PC0); //initialize PC0 to 0

PORTD = 0x00; //initialize binary counter to 0

static int count = 0; //used to count number of cycles

while (1)

{

TCNT1 = 0x85EE; //starting value of Timer1

TCCR1A = 0x00; //Normal mode

TCCR1B = 0x03; //Prescaler value 64

while (!(TIFR1 & (1<<TOV1))) //while no overflow

{

//keep cycling through timer

}

TCCR1B = 0x00; //reset TCCR1B to 0

TIFR1 = 0x01; //clear overflow flags by loading with 0x01

PORTC ^= 1; //toggle PC0

if (PORTC & (1 << 0)) //if PC0 is set

{

PORTD += 1; //increment binary counter

count++; //increment count

if (count % 5 == 0) //for every 5th count

{

PORTC ^= (1 << PC5); //toggle PC5

}

if (count % 10 == 0) //for every 10th count

{

PORTC ^= (1 << PC4); //toggle PC4

}

}

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |

ASSEMBLY CODE

/\*

\* AssemblerApplication1.asm

\*

\* Created: 3/7/2016 1:28:25 PM

\* Author: caimol

\*/

.ORG 0 ;location for reset

JMP MAIN ;jump to main program, skipping the jump to interrupt subroutine

.ORG 0x08 ;location for PCINT1

JMP EX1\_ISR ;jump to interrupt subroutine

MAIN:

SBI DDRC, 5 ;enable PC5 as output

SBI DDRC, 4 ;enable PC4 as output

SUB R5, R5 ;initialize count5 to 0

SUB R10, R10 ;initialize count10 to 0

LDI R16, 5 ;R16 = 5

ADD R1, R16 ;used to compare to count5

LDI R16, 10 ;R16 = 10

ADD R2, R16 ;used to compare to count10

LDI R16, 32 ;R16 = 32

ADD R6, R16 ;2^5 (causes PC5 to light up)

LDI R16, 16 ;R16 = 16

ADD R7, R16 ;2^4 (causes PC4 to light up)

SUB R8, R8 ;initialize PC5 to 0 (off)

;;;;;;;;initializations for counter;;;;;;;;;;

SBI DDRD, 0 ;enable PB0 as output

SBI DDRD, 1 ;enable PB1 as output

SBI DDRD, 2 ;enable PB2 as output

SBI DDRD, 3 ;enable PB3 as output

SBI DDRD, 4 ;enable PB4 as output

SBI DDRD, 5 ;enable PB5 as output

SBI DDRD, 6 ;enable PB6 as output

SBI DDRD, 7 ;enable PB7 as output

LDI R19, 0 ;counter initialized to 0 (R19 is designated register)

OUT PORTD, R19 ;output the counter value

;;;;;;;;initializations for PC0 output (blinks with counter);;;;;;;;;;;;;

LDI R16, 1 ;R16 = 1 (bit 0)

SBI DDRC, 0 ;enable PC0 as output

OUT PORTC, R8 ;output 0 to PORTC (initialized as off)

;;;;;;;;;;;;;;;;;;;;;;;setting conditions for interrupt;;;;;;;;;;;;;;;;;;;

LDI R20, 1 << PCIE1 ;enable PCIE1

STS PCICR, R20 ;change on enabled pin on PCINT[14:8] will cause interrupt

LDI R20, 1 << PCIF1 ;enable PCIF1

STS PCIFR, R20 ;PCIF1 becomes set on logic change in PCINT[14:8]

LDI R20, 1 << PCINT8;enable PCINT8 (PC0)

STS PCMSK1, R20 ;pin change interrupt enabled on PCINT8

SEI ;set I (enable interrupts)

;;;;;;;;;;;;;;;;;;;;;;;end ISR settings;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

START:

RCALL DELAY\_SUBR ;call delay subroutine

EOR R8, R16 ;XOR to toggle bit 0 of R8

;;;;;;;for counter;;;;;;;;;;;;

OUT PORTC, R8 ;output R8 to PORTC

SBIS PORTC, 0 ;compare current value at PB0 to check if high (rising edge). if so skip next instruction

JMP OUTCOUNT ;jump to OUTCOUNT (don't increment)

INC R19 ;else, increment current count in counter

OUTCOUNT:

OUT PORTD, R19 ;output current counter value to PORTB

;;;;;;;;;;;;end counter stuff;;;;;;;;;;;;;;

OUT PORTC, R8 ;output value to PORTC

RJMP START ;loop back through main routine

DELAY\_SUBR:

LDI R20, 0x85 ;R20 = 0x85

STS TCNT1H, R20 ;load TCNT1H (top byte of starting value of timer)

LDI R20, 0xEE ;R20 = 0xEE

STS TCNT1L, R20 ;R20 = 0xEE (lower byte of starting value of timer)

LDI R20, 0x00 ;R20 = 0x00

STS TCCR1A, R20 ;0x00 = Normal mode

LDI R20, 0x03 ;R20 = 0x03

STS TCCR1B, R20 ;prescaler = 64

AGAIN:

IN R20, TIFR1 ;gets TOV1 flag from TIFR1

SBRS R20, TOV1 ;skip next instruction if OV

RJMP AGAIN ;else loop until OV

LDI R20, 0x00 ;R20 = 0x00

STS TCCR1B, R20 ;Reset TCCR1B

LDI R20, 0x01 ;R20 = 0x01 to reset TIFR

OUT TIFR1, R20 ;Reset TIFR

RET ;return from function call

;;;;;;;;;;;;;;;;;;;;;;interrupt subroutine;;;;;;;;;;;;;;;;;;;;;;;;;

EX1\_ISR: ;interrupt subroutine

OUT PORTC, R8

SBIS PORTC, 0

JMP CHECK5

INC R5 ;increment R5 (count5)

INC R10 ;increment R10 (count10)

CHECK5:

CP R5, R1 ;compare count5 to 5

BRLO CHECK10 ;branch if lower than

EOR R8, R6 ;else, toggle PC5

SUB R5, R5 ;reset count5 to 0

CHECK10:

CP R10, R2 ;compare count10 to 10

BRLO ISR\_DONE ;branch if lower than

EOR R8, R7 ;else, toggle PC4

SUB R10, R10 ;reset count10 to 0

ISR\_DONE:

LDI R20, 1 << INTF1 ;write logical 1 to INTF1

OUT EIFR, R20 ;clear interrupt flag

RETI ;return from interrupt subroutine

C CODE  
/\*

\* DA2T4\_C.c

\*

\* Created: 3/13/2016 3:06:42 PM

\* Author : user

\*/

#include <avr/io.h>

#include <avr/interrupt.h>

int main(void)

{

DDRC |= 0x31; //enable PC0 as output

DDRD |= 0xFF; //enable PB0-PB7 as output

PORTC |= (0 << PC0); //initialize PC0 to 0

PORTD = 0x00; //initialize binary counter to 0

PCICR |= (1 << PCIE1); //change on enabled pin on PCINT[14:8] will cause interrupt

PCIFR |= (1 << PCIF1); //PCIF1 becomes set on logic change in PCINT[14:8]

PCMSK1 |= (1 << PCINT8);//pin change interrupt enabled on PCINT8 (PC0)

sei(); //set interrupts

while (1)

{

TCNT1 = 0x85EE; //starting value of Timer1

TCCR1A = 0x00; //Normal mode

TCCR1B = 0x03; //Prescaler value 64

while (!(TIFR1 & (1<<TOV1))) //while no overflow

{

//keep cycling through timer

}

TCCR1B = 0x00; //reset TCCR1B to 0

TIFR1 = 0x01; //clear overflow flags by loading with 0x01

PORTC ^= 1; //toggle PC0

if (PORTC & (1 << 0)) //if PC0 is set

{

PORTD += 1; //increment binary counter

}

}

}

ISR (PCINT1\_vect) //PC0 in the PCINT1\_vect

{

static int count = 0; //used to count cycles

if (PORTC & (1 << 0)) //if PC0 is set

{

count++;

if (count % 5 == 0) //for every 5th count

{

PORTC ^= (1 << PC5); //toggle PC5

}

if (count % 10 == 0) //for every 10th count

{

PORTC ^= (1 << PC4); //toggle PC4

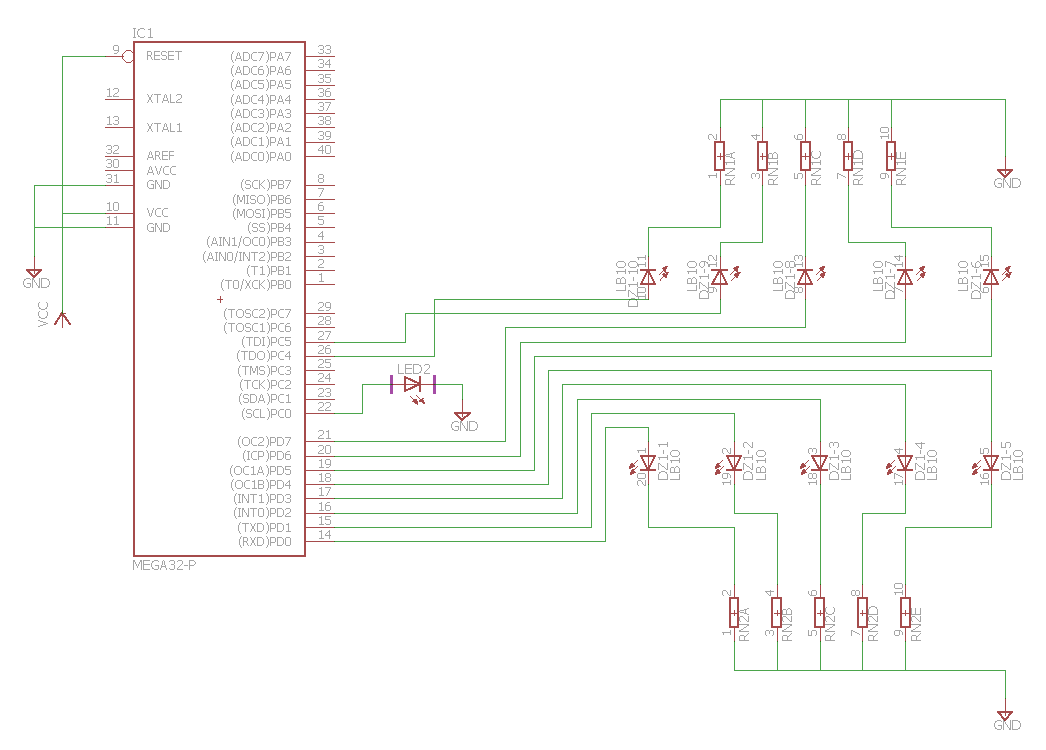
}

}

return;

}

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | SCHEMATICS |  |  |



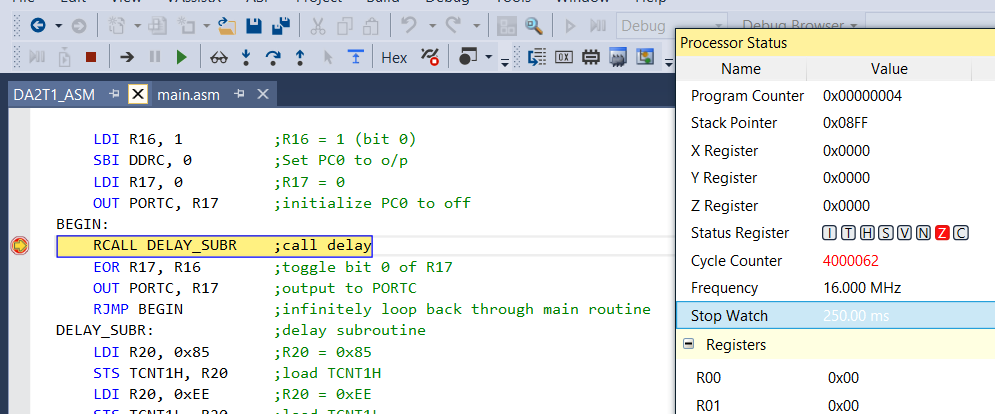
It’s important to note that PORTD was used instead of PORTB when outputting to the LEDs for the binary counter. This is because the Atmel Xplained Mini did not contain the adequate number of bits in PORTB (only went from 0 to 5). It’s also important to note that PC4 was used instead of PC6 when outputting to the 10th LED bit. This is because PC6 is used as a reset pin, and connecting it to a signal would render the board and its connected components inactive.

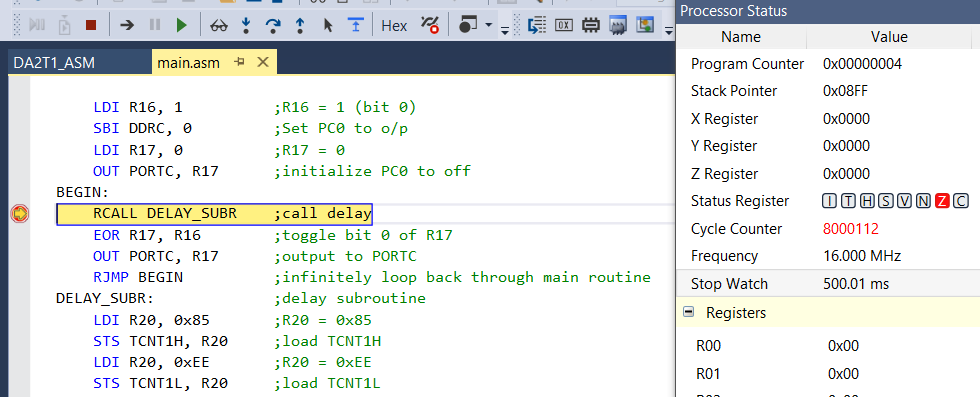
|  |  |  |  |
| --- | --- | --- | --- |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

TASK 1/A:

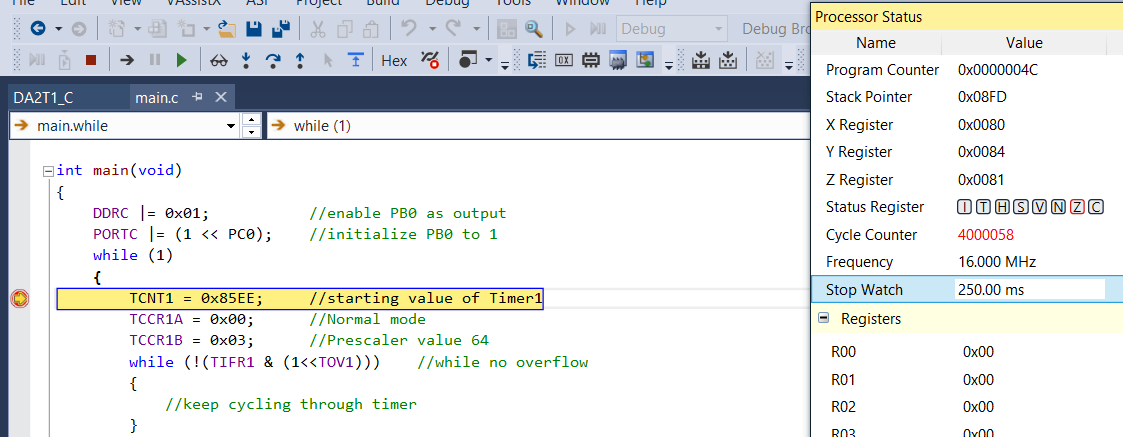
Verify duty cycle and period: 50% duty cycle, period = 0.5 second

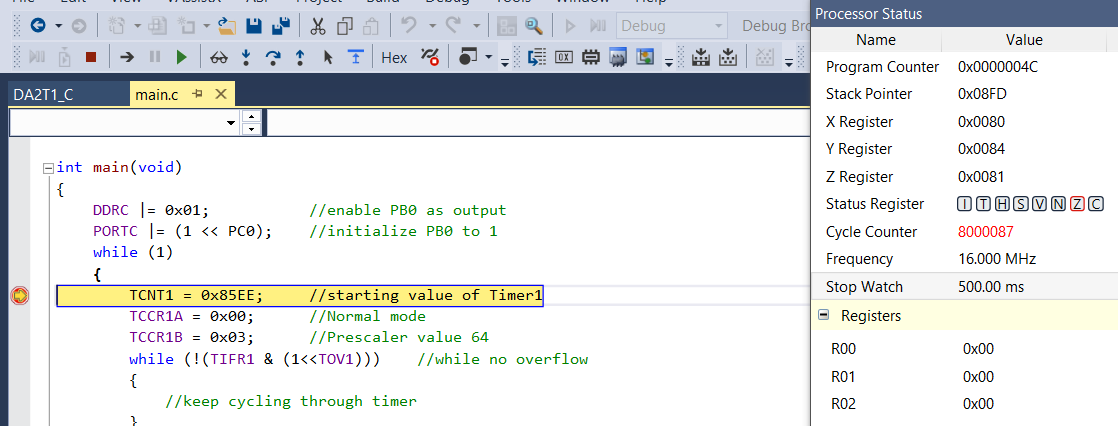
Assembly code:  
At halfway through the period, the stopwatch was at 250 ms. At the end of the full period, the time doubled to 500 ms and the value in the cycle counter exactly doubled (indicating 50% duty cycle).





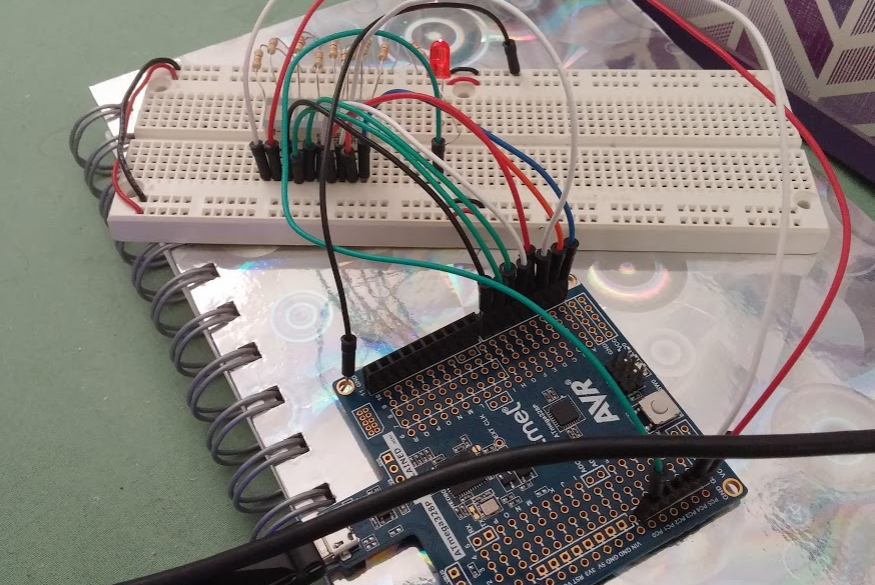
C code:  
The exact same thing occurred during the simulation for the C code version of Task 1/A. Therefore, this has also been verified for a 50% duty cycle and 0.5 second period.



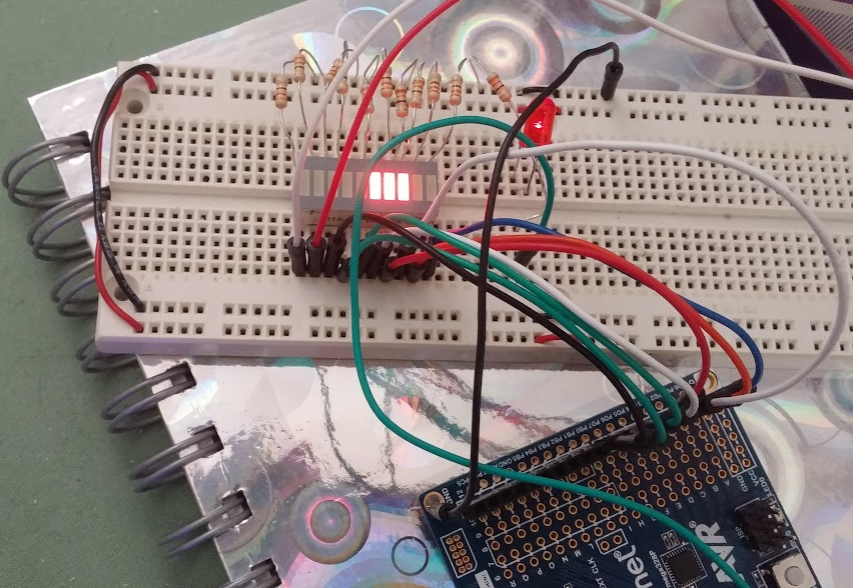


|  |  |  |  |
| --- | --- | --- | --- |
| 8. | SCREENSHOT OF EACH DEMO |  |  |

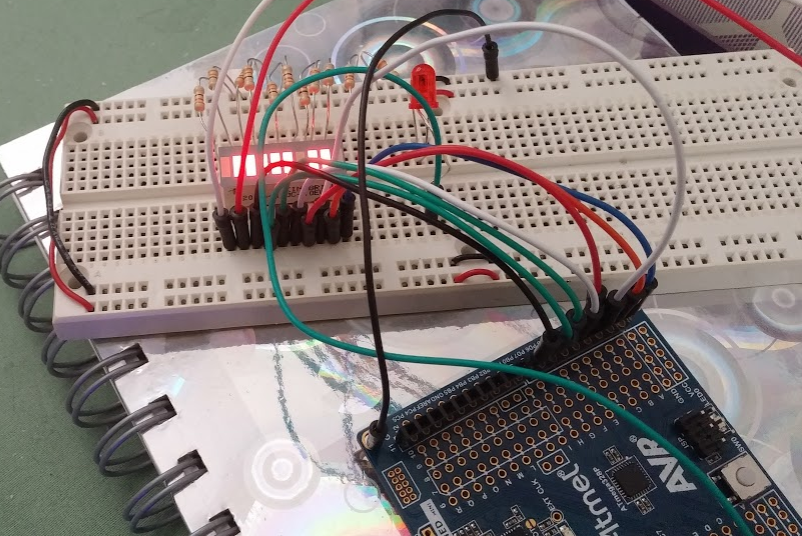
TASK 1/A: Waveform generated on PORTC.0 with 50% DC and 0.5 second period. Red LED bulb connected to this bit toggles every 0.25 seconds. (See video)



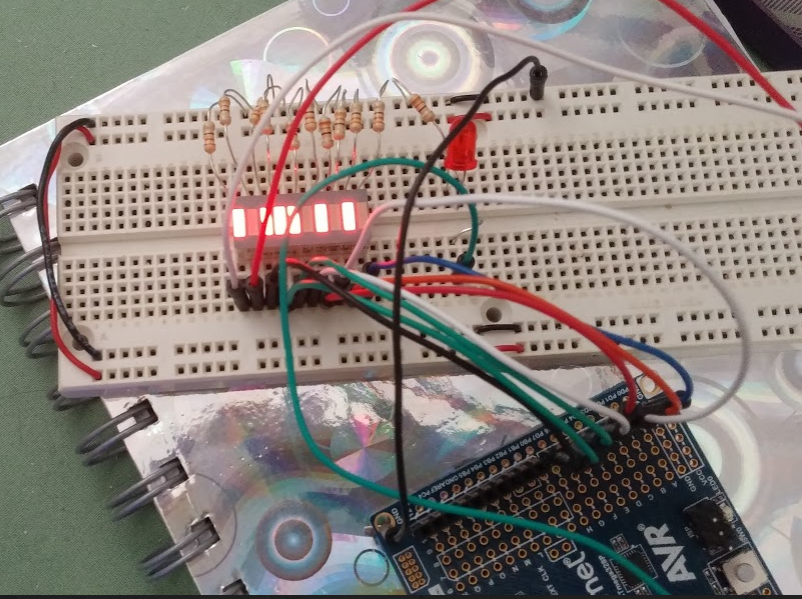
TASK 2/B: 8-bit counter displayed on first 8 bits of the 10-bit LED bar (read right to left). It counts up on every rising edge of the previous waveform (every time the LED toggles on). This is outputted through PORTD instead of PORTB because the board used did not have all 8 bits for PORTB. See video for full counting sequence.



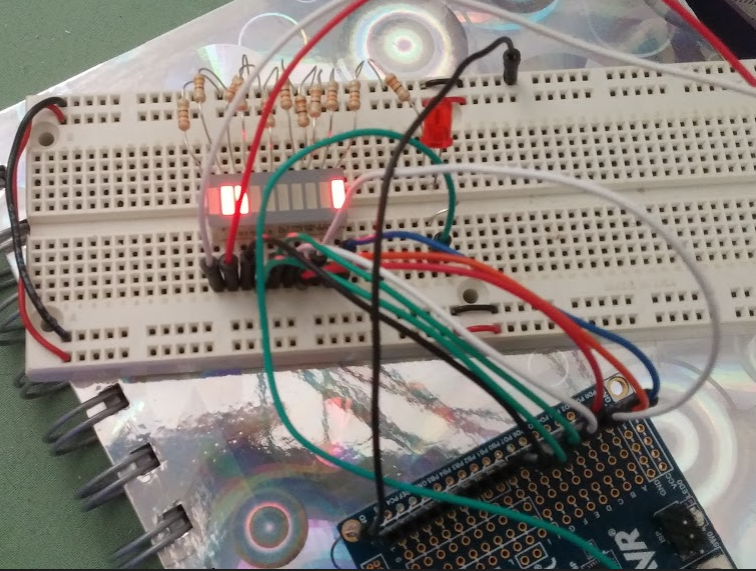
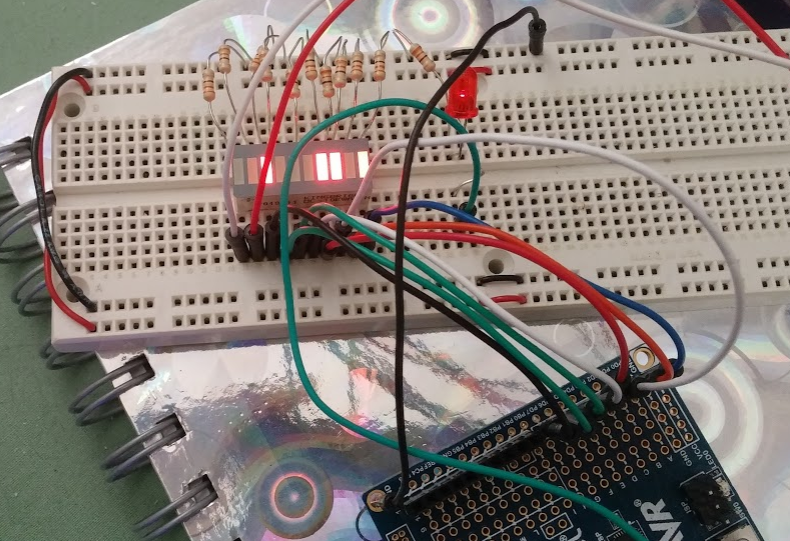
TASK 3/C: On every 5th count of the 8-bit counter (rising pulse), the 9th bit of the LED bar toggles. This signal is outputted through PORTC.5.

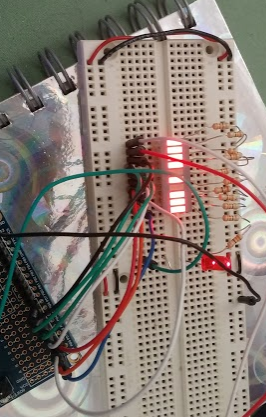


On every 10th count, the 10th bit of the LED bar is toggled. This is outputted through PORTC.4 (not PORTC.6 because this is a reset pin).



TASK 4/D: Interrupts are used to update the status of/toggle the 9th and 10th LED bits.





|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| T1 (ASM): <https://youtu.be/QcXlOF0fQkI>  T2 (ASM): <https://youtu.be/jXOnXkDt5os>  T3 (ASM): <https://youtu.be/UQo3ECkzFdM>  T4 (ASM): <https://youtu.be/KlC7LNWoCVg>  T1 (C): <https://youtu.be/y4bGzbTcZQ4>  T2 (C): <https://youtu.be/YQbI8O4BI2M>  T3 (C): <https://youtu.be/kKzMIpZ1X0w>  T4 (C): <https://youtu.be/xb8m-W1UDaU> | | | |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
| <https://github.com/candacecaimol/Design_Assignments.git> | | | |

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“*This assignment submission is my own, original work*”.

CANDACE CAIMOL