**Project 1: AVR Counter**

**Tim Crawford, Ben Wantland, Andrew Leise**

**Project Description:**

The goal of CpE 3150 Project 1 was to create a counter using the ATmega324pb microcontroller

and accompanying Simon Board. To display the counter, the Simon Board had to be utilized to light up

LEDs 1-4 representing a 4-bit binary number. Upon pressing the increment key (switch) on the board, the

value of the counter would be incremented and the new count would be displayed on the LEDs. Upon

pressing the decrement key (switch) on the board, the value of the counter would be decremented and the

new count would be displayed on the LEDs. Since the counter was utilizing a display with four LEDs, the

range of values was only 0 to 15. Thus, the counter would overflow upon incrementing at 15 and

underflow upon decrementing at 0. When this occurred, the counter was reset to its corresponding value

(0 on overflow, and 15 on underflow). To signify that the counter has overflowed or underflowed, an

alarm sound was implemented to play for the event. This concluded the core operations of the AVR

counter. After completion of the main counter functionality, each group member added additional

subroutines to give more depth to the counter. These are explained below at the end of the report.

The only major problem that was encountered during the implementation of the project was the

debouncing of the switches. Upon pressing the increment or decrement switches, multiple inputs would

be interpreted. That is, for one button press the counter would be incremented or decremented multiple

times. To remedy this, a new solution for the button presses had to be developed. This was done by

changing the way the switch presses were detected. Instead of using a delay function for checking, the

program logic was switched to wait until the button was released before the program continued. This

ensured the program would only continue for the button press if the switch was released, guaranteeing

only one increment operation is performed at a time.

The main program consists of a loop that runs through six checks to see if switches 1-6 are

pressed, and then executes the subroutine associated with that switch. Switches 3-6 are utilized by each

team member’s individual subroutine, which can each be found in the following sections of the report.

The main portion of the project involves switches 1 and 2 which increment and decrement the counter

respectively. Once the button press has been detected, the respective increment or decrement function is

called. In these functions, the necessary actions to increment or decrement the counter as well as display

the new count are implemented. First, the count register is incremented or decremented, then compared

with the respective overflow/underflow value. These values were 0x10 for an increment overflow and

0xFF for a decrement underflow. If the BREQ of this compare is true, then the register has overflowed or

underflowed, and the respective speaker function is called. It should be noted that two different speaker

functions, one for increment overflow and another for decrement underflow, were created. This was done

in case different sounds were to be used for each option, although this was never realized in the program.

If an increment overflow or decrement underflow is encountered, an alarm will play the first four

notes of the Megalovania subroutine. See the Megalovania section of the report for the calculation of

delays for the alarm. A simplified version of the LED display that is used in the Lightshow section of the

report also flashes when the counter overflows or underflows.

Following the overflow and underflow check, the count is prepared for output. Due to the

orientation of the LEDs with respect to the Simon Board, the four bits utilized in the counter had to be

reversed for correct output. The least significant bit of the counter is the most significant bit in the output,

and likewise with the rest of the bits. Thus, the counter value had to be flipped to be outputted properly.

To do this, a FLIP function was created.

In the FLIP function, the R18 register is initialized to 0 and the value of register R17 is copied

into R19. Then, through a series of RORs and ROLs, the value in register R19 is rotated into R18,

creating the reversed value for output. Note that in this process the value of register R17 is not affected in

any way, thus the integrity of the counter is kept intact. After four rotations, the function returns.

The last operation to perform before outputting the count is to invert (complement) the value of

R18. This was done because the LEDs on the Simon Board are all active low, meaning they light when

given a value of 0. After doing this, the value of register R18 is outputted to PORTD, lighting up the

current value of the counter on the LEDs. This finishes the operation of the increment or decrement

function, and the program returns to the main loop of checking the different switches.

**Pin Functions and the ATmega324pb:**

Various components of the ATmega324pb microcontroller were utilized for the implementation

of Project 1. The first five bits of PORTA were used for gathering input from switches 1-4 and 6. The

lower four bits of PORTD were also utilized. These outputted the 4-bit counter value on the Simon Board.

Bits 4-6 of PORTE were also used for the speaker, LED5, and switch 5 respectively.

A large variety of registers, including R16-R19 and R24-R31, were also incorporated into the

core design. Registers R16-R19 were used mainly for the increment and decrement functions of the

counter. Registers R24-R26, and R28 were used for several different delay functions throughout the

program. The timer functions specific to the ATmega324pb architecture could have been used here,

however, that material was not yet covered in class by the time the design was finalized. Lastly, registers

R29-R31 were used in the Megalovania function and R27, and R29 were used in the Lightshow and

Flash\_Lights functions. See the below table for a full description of the ports utilized.

Pin Device Function

PD0 LED1 Display counter/Lightshow

PD1 LED2 Display counter/Lightshow

PD2 LED3 Display counter/Lightshow

PD3 LED4 Display counter/Lightshow

PE5 LED5 Lightshow/Megalovania

PD4 LED6 Lightshow

PD5 LED7 Lightshow

PD6 LED8 Lightshow

PD7 LED9 Lightshow

PA0 SW1 Increment counter

PA1 SW2 Decrement counter

PA2 SW3 Play Megalovania

PA3 SW4 Increment counter for the amount of seconds held

PE6 SW5 Decrement counter for the amount of seconds held

PA4 SW6 Play Lightshow

PE4 Speaker Alarm sound/Megalovania

**Table 1:** ATmega324pb pin descriptions and functions

**Commented Code:**

The following code consists of the routines performed during the operation of the main counter. Note that

the following contains more than just the main counter functions. The different individual functions

additionally check for switch inputs in the main loop, so they also appear here to avoid confusion or

missing code. Switches 0 and 1 are the two switches used for the main counter operation as well as the

various functions called inside them.

; Set up main registers necessary for program setup .ORG 0 LDI R16,0xFF LDI R17,0x00 LDI R20,0x00

; Define PORTD pins used for counter as output OUT DDRD,R16 OUT PORTD,R16

; Define PORTA pins as input OUT DDRA,R20 LDI R16, 0b00011111 OUT PORTA,R16

; Define PORTE pins 4, 5 for output, 6 for input SBI DDRE,4 SBI DDRE,5 SBI PORTE,5 CBI DDRE,6 SBI PORTE,6

; Check for button press on switch index 0 (increment button function) CHECK\_SW0:

; If button is not held, check next switch SBIC PINA,0 RJMP CHECK\_SW1 ; If it is, increment counter, loop while button is still held, then move to next switch check CALL INCREMENT SW0\_Hold: SBIS PINA,0

RJMP SW0\_Hold RJMP CHECK\_SW1

; Check for button press on switch index 1 (decrement button function) CHECK\_SW1:

; If button is not held, check next switch SBIC PINA,1 RJMP CHECK\_SW2 ; If it is, decrement counter, loop while button is still held, then move to next switch check CALL DECREMENT SW1\_Hold: SBIS PINA,1

RJMP SW1\_Hold RJMP CHECK\_SW2

; Check for button press on switch index 2 (Music button function) CHECK\_SW2:

; If button is not held, check next switch SBIC PINA,2 RJMP CHECK\_SW3 ; If it is, call Megalovania function, loop while button is held, and move to next switch check CALL Megalovania SW2\_Hold: SBIS PINA,2

RJMP SW2\_Hold RJMP CHECK\_SW3

; Check for button press on switch index 3 (Timer increment function) CHECK\_SW3:

; If button is not held, check next switch SBIC PINA,3 RJMP CHECK\_SW4 ; If it is, load a value of 0 into R16, and increment it for every second button is held ; Add this to the counter register and display the new value LDI R16, 0x00 SW3\_Hold: INC R16

CALL SEC\_DELAY SBIS PINA,3 RJMP SW3\_Hold

ADD R17, R16 DEC R17 ANDI R17, 0x0F ; Clear upper nibble of R17 in case addition result is greater than 15 CALL INCREMENT RJMP CHECK\_SW4

; Check for button press on switch index 4 (Timer Decrement function) CHECK\_SW4:

; If button is not held, check next switch (note sw5 is on PINE bit 6) SBIC PINE,6 RJMP CHECK\_SW5 ; If it is, load a value of 0 into R16, and increment it for every second button is held ; Add this to the counter register and display the new value LDI R16, 0x00 SW4\_Hold: INC R16

CALL SEC\_DELAY SBIS PINE,6 RJMP SW4\_Hold

SUB R17, R16 INC R17 ANDI R17, 0x0F ; Clear upper nibble of R17 in case subtraction result is greater than 15 CALL DECREMENT RJMP CHECK\_SW5

; Check for button press on switch index 5 (Light Show function) ;ADDED CHECK\_SW5:

; If button is not held, check next switch SBIC PINA,4 RJMP CHECK\_SW0 ; If it is, call Megalovania function, loop while button is held, then move to next switch check CALL LIGHT\_SHOW

SW5\_Hold: SBIS PINA,4

RJMP SW5\_Hold RJMP CHECK\_SW0

; Subroutine to handle the increment and display of the counter INCREMENT:

; Increment R17, check for overflow with 0x10 INC R17 CPI R17, 0x10 BREQ SPEAKER\_INC ; If no overflow, flip the value for display, complement it for active low, then display and return I\_RET: CALL FLIP

COM R18 OUT PORTD, R18 RET

; Subroutine to handle the decrement and display of the counter DECREMENT:

; Decrement R17, check for underflow with 0xFF DEC R17 CPI R17, 0xFF BREQ SPEAKER\_DEC ; If no overflow, flip the value for display, complement it for active low, then display and return D\_RET: CALL FLIP

COM R18 OUT PORTD, R18 RET

; Subroutine to flip the value of R17 into R18 FLIP:

; Set R18 to 0 and R19 equal to R17 (this is so the value of the counter itself is not changed) ; Continually use ROR and ROL to rotate each bit of R19 into R18 to flip the value, then return LDI R18, 0x00 MOV R19, R17 CLC ROR R19 ROL R18 CLC ROR R19 ROL R18 CLC ROR R19 ROL R18 CLC ROR R19 ROL R18 RET

; Subroutine to handle the speaker call after an incremental overflow SPEAKER\_INC:

; Call corresponding functions for the notes, reset R17 to 0x00 (from 0x10 of overflow), and jump back CALL LD16 CALL LD16 CALL D16 CALL Rst16 CALL A8 LDI R17, 0x00 CALL FLASH\_LIGHTS RJMP I\_RET ; Subroutine to handle the speaker call after a decremental underflow SPEAKER\_DEC:

; Call corresponding functions for the notes, reset R17 to 0x0F (from 0xFF of overflow), and jump back CALL LD16 CALL LD16 CALL D16 CALL Rst16 CALL A8 LDI R17, 0x0F CALL FLASH\_LIGHTS RJMP D\_RET

**Work Distribution:**

The workload was split into three parts: Counter, Lights, and Sound. Andrew Leise wrote the

code for the counter, Tim Crawford implemented the buzzer for the overflow and underflow cases, and

Ben Wantland used the LEDs on the board and added them to complement the buzzer in the overflow and

underflow. In addition, each member wrote a unique function showcasing their portion of the project.

Andrew wrote an alternate counter increment and decrement function that changes the count based on

how long the switch is held. Ben created a unique lightshow the flashes the LEDs in an appealing pattern.

Tim translated the first four measures of the popular song, Megalovania, into machine code. See the

below flowchart for a further description of the work effort. Overall, the work effort was split evenly and

every group member properly contributed to the project. Each team member contributed 33% to the

overall project.

**Figure 1:** Flowchart of work contributions

**Individual Reports:**

**Lightshow and Flash Lights - Ben Wantland:**

The Lightshow function uses all the LEDs of the board to display a short light show. The

different functions within Lightshow are used to make different shapes and lines, and the functions could

be easily modified to go for a longer or shorter period of time. When each section of Lightshow is called,

R27 is modified and outputted to PORTD with a short delay. This was done so the show can be seen and

shapes, like arrows, can appear to move across the LEDs. PORTE’s 5th bit is also modified to enable the

use of LED five.

The Delay function is used for both the counter overflow/underflow and the Lightshow to provide

a short delay of around .2 seconds. R26, R27, and R28 are used for the delay loops and their values are all

100. This value is rounded for simplicity. Making “n” equal to:

n = 3030304 → (3030304)(1s/(16(106cc))) = 0.18939 second time delay

While this time delay is not exactly .2 seconds it is easy to implement and is close enough to the desired

time to fulfill its purpose.

Implementation of the lighting functions didn’t cause any issues because the mapping of the

registers and functions had been planned ahead of time.

Below is the code for the Flash\_Lights, Delay, and also part of the Lightshow function.

Flash\_Lights has the same initial loaded values and uses some of the same functions as Lightshow but is

not as time consuming. The section of the Lightshow function below shows how the Lightshow function

operates and the rest of the function works the same way. The Delay function is also shown and is used

repeatedly in the other functions.

;In ; Flashes the LEDs after the short segment of megalovania, when

Lightshow overflow or underflow is triggered.

;flash horizontal lines FLASH\_LIGHTS: LDI R27, 0b11111111

LDI R29, 2 OUT DDRD, R27

LOOP: LDI R27, 0b11010110 ; left col LDI R27, 0b1111111

OUT PORTD, R27 OUT DDRA, R27

SBI PORTE, 5 LDI R27, 0b01000000

CALL Delay OUT PORTA, R27

LDI R27, 0b10111101 ; mid col LDI R27, 0b11111111

OUT PORTD, R27 OUT PORTD, R27

CBI PORTE, 5 SBI DDRE, 5

CALL Delay SBI PORTE, 5

LDI R27, 0b01101011 ; right col ;toggle outside with middle light

OUT PORTD, R27 LDI R29, 5

SBI PORTE, 5 LOOP8: LDI R27, 0b01011010

CALL Delay OUT PORTD, R27

LDI R27, 0b11111111 ; all off CBI PORTE, 5

OUT PORTD, R27 CALL Delay

CALL Delay LDI R27, 0b10100101

LDI R27, 0b01101011 ; right col OUT PORTD, R27

OUT PORTD, R27 CALL Delay

CALL Delay DEC R29

LDI R27, 0b10111101 ; mid col BRNE LOOP8

OUT PORTD, R27 ;turn off all lights

CBI PORTE, 5 LDI R27, 0b11111111

CALL Delay OUT PORTD, R27

LDI R27, 0b11010110 ; left col SBI PORTE, 5

OUT PORTD, R27 RET

SBI PORTE, 5 ; Delay used for use of FLASH\_LIGHTS and LIGHT\_SHOW

CALL Delay Delay:

LDI R27, 0b11111111 ; all off LDI R26, 100

OUT PORTD, R27 loopA: LDI R27, 100

CALL Delay loopB: LDI R28, 100

DEC R29 loopC: DEC R28

BRNE LOOP BRNE loopC DEC R27 BRNE loopB DEC R26 BRNE loopA RET

**Figure 2:** Code for different light show subroutines

**Time-Based Increment/Decrement - Andrew Leise:**

The time-based increment and decrement functionalities add and subtract from the current count

based on how long the button has been pressed. The time-based increment function is a part of the

“Check\_SW3” and the time-based increment function is a part of the “Check\_SW4” subroutines

respectively. If switch 4 (noted as index 3 in “Check\_SW3”) is pressed, then the program enters a loop

that checks if the button is still pressed every second. Every time the loop is entered, the counter is

incremented. After the button is released, the loop is exited and the time value is added to the counter. It

should also be noted that the upper nibble of the counter is cleared to avoid any overflow issues going

forward in the program. Lastly, the counter is decremented and the Increment subroutine is called to

properly output the new value to PORTD. The subroutine for the time-based decrement function is

exactly the same, however, instead of adding R16 to R17, R16 is subtracted from R17. Additionally, the

counter is incremented before the Decrement function is called for displaying the output to PORTD. This

effectively decreases the counter by the value of the button press length. The following calculations

explain the loop for the 1s delay.

The clock frequency of the AtMega342PB is 16 MHz and every machine cycle takes one clock cycle. Let

n be the number of machine cycles required for this operation. Thus, the number of machine cycles that

need to execute for a 1s delay is

n = 16,000,000

The following equation was determined by the structure of the SEC\_DELAY function in the code. This

was used to derive the necessary number of iterations for each loop. It should also be noted that the

outermost loop was preset to 250 for simplicity in calculating the other values, and to give an even outer

number.

n = (1 + (1 + (1 + (1 + 1 + 1 + 2) \* x1 - 1 + 1 + 2) \* x2 - 1 + 1 + 2) \* x3 - 1 + 4)

With n = 16,000,000 and x3 = 250.

16,000,000 = (1 + (1 + (1 + (1 + 1 + 1 + 2) \* x1 - 1 + 1 + 2) \* x2 - 1 + 1 + 2) \* 250 - 1 + 4)

There are numerous different solutions to this equation, thus the closest, near-whole values were chosen.

Solving for x1 and x2 yielded:

x1 = 149.926

x2 = 85

Which were then rounded to...

x1 = 150

x2 = 85

Utilizing these values causes the delay function to take n = 16,002,004 machine cycles, which is

extremely close to the desired value of 16,000,000. Solving for the real time delay yielded:

t = 16,002,004 \* (1s/(16x106cc)) = 1.00013s time delay.

As stated previously, this delay function is called every time the button press (for indices SW3 and SW4)

is rechecked. The number of total loops is counted and then added or subtracted to the counter, creating

the time-based increment and decrement operations. The additional overhead of the increment/decrement

operation in the switch loop and the branch checks should be noted, although is likely negligible. This

would only add three additional clock cycles per loop, which would not have a significant impact.

Various ATmega324pb features were utilized and implemented for this functionality. Registers

R16 and R17 were utilized for storing the added/subtracted value and counter respectively. The different

instructions utilized included INC, SBIS, RJMP, ADD, DEC, ANDI, and CALL. These were used to

perform the different operations needed to output the new count to PORTD. It should be noted that the

on-board timer functions could have been utilized for the 1 second delay function. If this project were to

be redone, these would most likely be used, however, the timer functions were taught just after the design

and implementation had already been finalized. Thus, the general delay function that was calculated for

above was utilized. PINE bit 6 and PINA bit 3 were also used, as they were the indices for the index 4

and 3 switches respectively. See the code snippet below for the two functions.

; Check for button press on switch index 3 (Timer increment function) CHECK\_SW3:

; If button is not held, check next switch SBIC PINA,3 RJMP CHECK\_SW4 ; If it is, load a value of 0 into R16, and increment it for every second button is held ; Add this to the counter register and display the new value LDI R16, 0x00 SW3\_Hold: INC R16

CALL SEC\_DELAY SBIS PINA,3 RJMP SW3\_Hold

ADD R17, R16 DEC R17 ; Clear upper nibble of R17 in case addition caused value greater than 15 ANDI R17, 0x0F CALL INCREMENT RJMP CHECK\_SW4

; Check for button press on switch index 4 (Timer Decrement function) CHECK\_SW4:

; If button is not held, check next switch (note sw5 is on PINE bit 6) SBIC PINE,6 RJMP CHECK\_SW5 ; If it is, load a value of 0 into R16, and increment it for every second button is held ; Add this to the counter register and display the new value LDI R16, 0x00

SW4\_Hold: INC R16

CALL SEC\_DELAY SBIS PINE,6 RJMP SW4\_Hold

SUB R17, R16 INC R17 ; Clear upper nibble of R17 in case subtraction caused value greater than 15 ANDI R17, 0x0F CALL DECREMENT RJMP CHECK\_SW5

; Subroutine to handle the delay of 1 second SEC\_DELAY:

; Simply iterate through three loops then return LDI R24,250 D0: LDI R25,85 D1: LDI R26,150 D2: NOP NOP DEC R26 BRNE D2 DEC R25 BRNE D1 DEC R24 BRNE D0 RET

**Megalovania - Tim Crawford:**

The Megalovania subroutine plays the first four measures of Megalovania and increments the

counter on every beat. Pressing switch 3, which is located on PA2, will run the Megalovania subroutine

and turn on the blue LED located on PE5 for the duration of the song. The notes used are low B flat

(116.54 Hz), low B (123.47 Hz), low C (130.81 Hz), low D (146.83 Hz), low F (174.61 Hz), low G (196

Hz), A flat (207.65 Hz), A (220 Hz), and D (293.66 Hz). Calculating the required number of machine

cycles to produce a square wave for each note involves finding the period of the wave and the amount of

time it takes to complete one machine cycle. Half of T divided by tMC is equal to the number of machine

cycles required for the delay to produce the frequency for each note. The number of MCs to play a low B

flat 16th note is:

T = 1/f = 1/(116.54 Hz) = 8.581 ms

tMC = (1cc/1MC)(1s/(16x106cc)) = 0.0625 μs

nLBF = (T/2)/tMC = (8.581 ms/2)/0.0625 μs = 68646 MCs

The rest of the notes can be calculated in the same way:

nLB = 64,793 MCs nLC = 61158 MCs nLD = 54484 MCs nLF = 45816 MCs

nLG = 40816 MCs nAF = 38526 MCs nA = 36364 MCs nD = 27242 MCs

Megalovania is played at 120 bpm which means a quarter note is played every 0.5s. The length of

time of a quarter note in MCs can be found with tMC.

tMC 8th note = (1cc/1MC)(0.5s/0.0625μs) = 8 million MCs

Since a quarter note is 8 million MCs, 8th notes are 4 million MCs and 16th notes are 2 million MCs. The

number of loops required to create the delay for each note function can be calculated by solving for each

x where n is the number of MCs for each note and tMC is the number of MCs for the length of the note:

n = 1 + 1 + (1 + (1 + 2) \* x1 + 2) \* x2 - 1

tMC = 1 + (2n + 1 + 2) \* x3 - 1 + 4

The above functions were derived by calculating the delay of the LB16 subroutine which is in figure 2

below. The highlighted section of the code represents the delay of half the period. The delay for the 16th

rest subroutine utilizes three nested loops. Each x for the number of loops in the 16th rest function is

calculated in a similar way:

2000000 MCs = 1 + (1 + (1 + 2) \* 100 + 2) \* x1 + 2) \* x2 - 1 + 4

Megalovania: CBI PORTE,5

CALL MINCREMENT CALL LD16;M1 CALL MINCREMENT CALL LD16 CALL MINCREMENT CALL D16 CALL Rst16 CALL MINCREMENT CALL A8 CALL Rst16 CALL MINCREMENT CALL AF8 CALL MINCREMENT CALL LG8 CALL MINCREMENT CALL LF8 CALL MINCREMENT CALL LD16 CALL MINCREMENT CALL LF16 CALL MINCREMENT CALL LG16 ...

LB16: LDI R29,15;16th Low B Note LB16\_Loop: LDI R30,142

SBI PORTE,4 LB16\_Loop\_set\_2: LDI R31,151 LB16\_Loop\_set\_1: DEC R31

BRNE LB16\_Loop\_set\_1 DEC R30 BRNE LB16\_Loop\_set\_2 LDI R30,142 CBI PORTE,4 LB16\_Loop\_clr\_2: LDI R31,151 LB16\_Loop\_clr\_1: DEC R31

BRNE LB16\_Loop\_clr\_1 DEC R30 BRNE LB16\_Loop\_clr\_2 DEC R29 BRNE LB16\_Loop RET

Rst16: LDI R29,100;16th Rest Rst16\_Loop\_3: LDI R30,237 Rst16\_Loop\_2: LDI R31,28 Rst16\_Loop\_1: DEC R31

BRNE Rst16\_Loop\_1 DEC R30 BRNE Rst16\_Loop\_2 DEC R29 BRNE Rst16\_Loop\_3 RET

**Figure 3:** First measure of Megalovania (left) along with low B 16th note and 16th rest (right).

This concludes the main report for CpE 3150 Project 1. Overall, the project

implementation was a success. The primary counter was corrected programed and tested (see the

attached video in drive submission), numerous additional functions were provided by the three

group members, and the ATmega324pb was correctly utilized. Numerous skills were gained

throughout the development process, displaying that the project was a success. In total, counter

and its subsequent functions were fully implemented and tested, completing the first CpE 3150

project.