Microchip Studio Intro, Lab 2

Embedded System Design - CS3813301
Group 6
Tobias Vera - tobiasvera2011@gmail.com (F10915126),
Gerardo Fisch - gerarfisch@gmail.com (F10915108)
October 26th, 2021

I. INTRODUCTION

The purpose of this lab is to get familiar with the assembly coding of the Output ports of the ATMega328p in order to make a Binary Coded Decimal Counter using a 4-digit seven-segment display.

II. PROCEDURE

- Setup the Arduino Uno board on the Microchip Studio app through the External Tools windows, adding the respective Command (C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avrdude.exe) and Arguments (-C"C:\Program Files (x86) \Arduino\hardware\tools\avr/etc/avrdude.conf" -v -patmega328p -carduino -PCOM4 -b115200 -D -Uflash:w:"$(ProjectDir)Debug\$(TargetName).hex":i). The Command and the Arguments can vary depending on each user's path.
- Wire-up the system as the schematic of Figure 1 on the Experimental Data section.
- Look at the mapping of the ATMega328p onto the Arduino ports in order to get a table like Table 1 on Experimental Data section, which shows which values to load into the output ports to represent each of the different digits on the display.
- Do the coding as shown and explained on the Appendix A.

Obs.: The count should count at a rate of 0.25s to 0.5s per count.
Obs.: Only the rightmost 2 segments should count. The leftmost 2 segments should display, 00 with the decimal point on, for the second 0, i.e., possible count values are 00.00 … 00.15 … 00.99.

III. EXPERIMENTAL DATA

A. Schematic of the Circuit

The schematic of the circuit can be seen on the Figure 1. From the schematic it can be seen that the pins used for this circuit are the pins from 2 to 13. From the schematic of the Arduino UNO[1], it can be seen that the only output ports used on the original circuit are the PORT B and PORT D, since the pins from 2 to 7 correspond to the PORT D and the pins from 8 to 13 correspond to the PORT B. After that, the values of PORT B and PORT D required to represent each digit were computed and the results can be seen on the Table I.

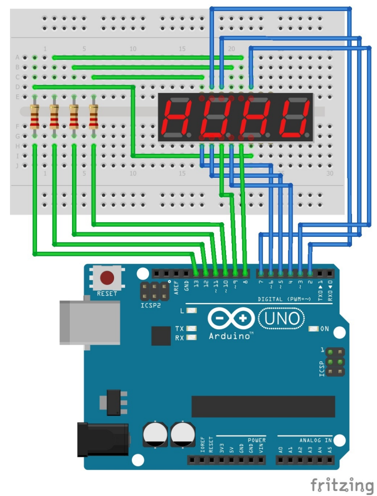

Fig. 1. Schematic of the circuit

B. Photo of the wired-up circuit

Based on the schematic of the circuit, the same circuit was reproduced. The circuit can be seen on Figure 2.

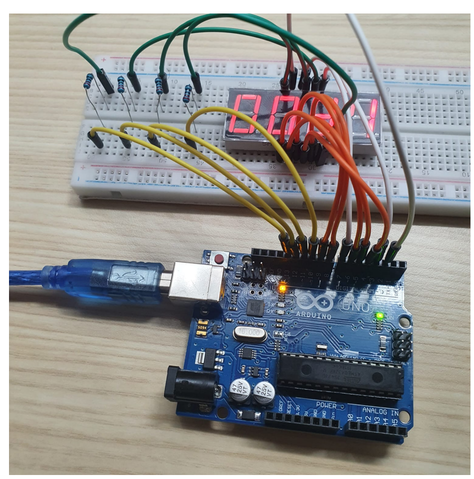

Fig. 2. Wired-up circuit

C. Assembly Code

The Assembly Code can be seen on the Appendix. A flow diagram of the code can be seen on the Figure 3.

	PORT B		PORT D	
	d4 d3	d2 d1 dp g	f e d c	b a
	- - 13 12	11 10 9 8	7 6 5 4	3 2 - -
0XXX	0 0 1 1	1 0 0 0	1 1 1 1	1 1 0 0
X0XX	0 0 1 1	0 1 1 0	1 1 1 1	1 1 0 0
XXX0	0 0 0 1	1 1 0 0	1 1 1 1	1 1 0 0
XXX1	0 0 0 1	1 1 0 0	0 0 0 1	1 0 0 0
XXX2	0 0 0 1	1 1 0 1	0 1 1 0	1 1 0 0
XXX3	0 0 0 1	1 1 0 1	0 0 1 1	1 1 0 0
XXX4	0 0 0 1	1 1 0 1	1 0 0 1	1 0 0 0
XXX5	0 0 0 1	1 1 0 1	1 0 1 1	0 1 0 0
XXX6	0 0 0 1	1 1 0 1	1 1 1 1	0 1 0 0
XXX7	0 0 0 1	1 1 0 0	0 0 0 1	1 1 0 0
XXX8	0 0 0 1	1 1 0 1	1 1 1 1	1 1 0 0
XXX9	0 0 0 1	1 1 0 1	1 0 1 1	1 1 0 0
XX0X	0 0 1 0	1 1 0 0	1 1 1 1	1 1 0 0
XX1X	0 0 1 0	1 1 0 0	0 0 0 1	1 0 0 0
XX2X	0 0 1 0	1 1 0 1	0 1 1 0	1 1 0 0
XX3X	0 0 1 0	1 1 0 1	0 0 1 1	1 1 0 0
XX4X	0 0 1 0	1 1 0 1	1 0 0 1	1 0 0 0
XX5X	0 0 1 0	1 1 0 1	1 0 1 1	0 1 0 0
XX6X	0 0 1 0	1 1 0 1	1 1 1 1	0 1 0 0
XX7X	0 0 1 0	1 1 0 0	0 0 0 1	1 1 0 0
XX8X	0 0 1 0	1 1 0 1	1 1 1 1	1 1 0 0
XX9X	0 0 1 0	1 1 0 1	1 0 1 1	1 1 0 0

TABLE I
VALUES OF PORT B AND PORT D FOR EACH DIGIT

D. Counting Rate

To compute the counting rate, a break-point was added on the code shown in the Appendix. The break-point was at the start of the "update_decimal" label. Then, on the Processor Status window of the Microchip Studio, the counting rate of the decimal digit was computed. To access this window go to Debug>Windows>Processor Status. The result was $4.950.474, 88\mu s \approx 4.95s$. Therefore, the counting rate of each number is $4.95s/10 = 0.495ms$. The counting rate can be seen of Figure 4.

E. Inline Assembly

To convert this code to inline assembly, it is needed to follow the corresponding syntax, and put every line of the code inside the function asm();
Example:

Assembly		Inline Assembly
ldi r16, 0b0011111	$\rightarrow$	asm("ldi r16, 0b0011111");

F. Results

The Figures 5, 6 and 7 show the final result.

IV. DISCUSSION

The first two digits remained unchanged and the last two worked as the counter, from 00 to 99 and starting back, as required by the instructor. The experience was useful for learning about the input/output through the ports of the microchip. This process consists of loading the values of the pins to be used into a register, and then loading this register's value into the output port, which will finally turn on the required sections of the display. The labor of setting up the seven-segment display has to be done for each digit

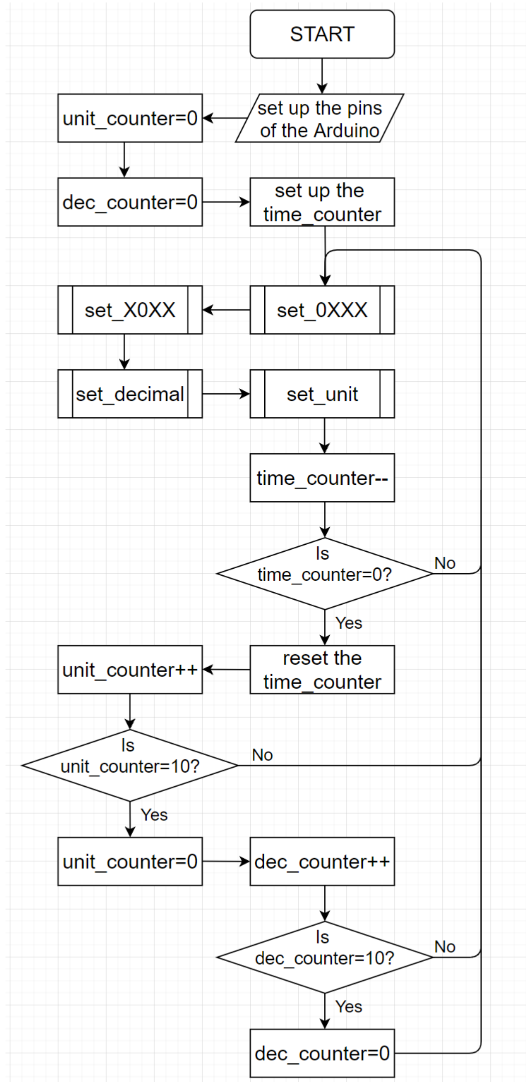

Fig. 3. Flow Diagram of the Program

to be represented on both PORT B and PORT D, which are the output-ports used for this experiment.
It was also useful to learn about some unknown assembler commands, as well as to learn about the assembly's limitations in calling functions and how to work around these limitations through jumps.

V. CONCLUSION

The results of the experiment were satisfactory, going as expected. The first two digits of the system remained unchanged, and the last two digits functioned as a counter, going from 00 to 99 and starting back. The limitations presented by the assembler when calling functions were solved successfully through the use of jumps. The program can also meet the time parameters requested by the instructor through delay programming.

REFERENCES

[1] https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf

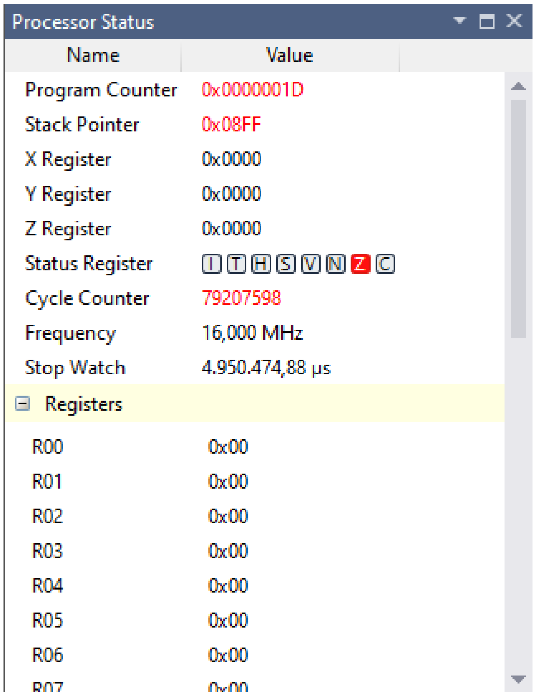

Fig. 4. It can be seen that the counting rate of the decimal digit is $4.950.474, 88\mu s$

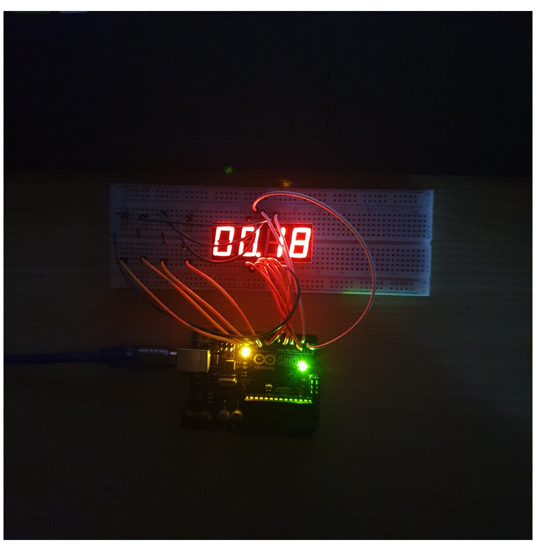

Fig. 5. Example of the circuit when the output is 00.18

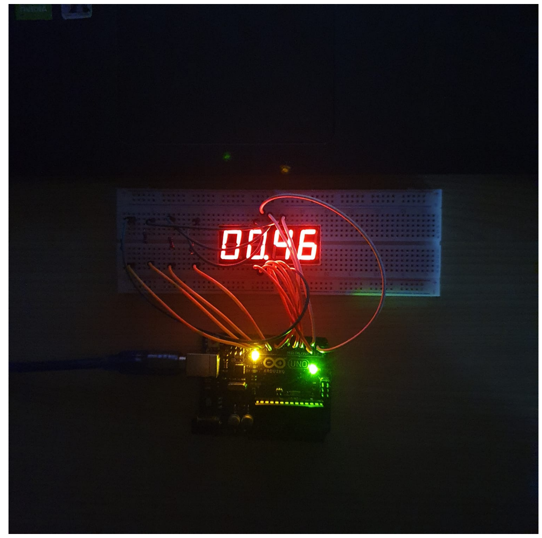

Fig. 6. Example of the circuit when the output is 00.46

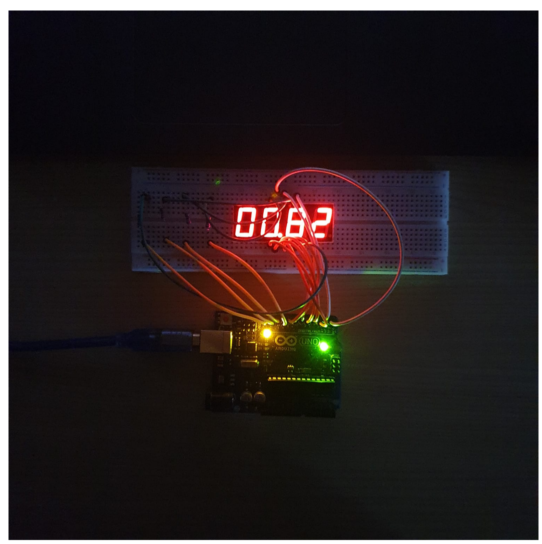

Fig. 7. Example of the circuit when the output is 00.62

APPENDIX

```
; Configure Direction PORT B
ldi r16, 0b00111111 ; digital pins from 8 to 13 of the Arduino
out DDRB, r16
; Configure Direction PORT D
ldi r17, 0b11111100 ; digital pins from 2 to 7 of the Arduino
out DDRD, r17

; r18 and r19 register used for PORT B and PORT D, respectively
ldi r20, 0x00 ; register used for the unit counter
ldi r21, 0x00 ; register used for the decimal counter
ldi r22, 64   ; register used for the timer
ldi r23, 20   ; register used for the timer
ldi r24, 2    ; register used for the timer

loop:
  call set_0XXX               ; call subroutine that turn on the 1st digit
  call set_X0XX               ; call subroutine that turn on the 2nd digit
  rjmp set_decimal_long_jump  ; jump to a branch closer to the subroutine set_decimal
  jump_back_decimals:         ; branch used to return after set_decimal
  rjmp set_unit_long_jump     ; jump to a branch closer to the subroutine set_unit
  jump_back_units:            ; branch used to return after set_unit
  subi r22, 1
  sbci r23, 0
  sbci r24, 0
  cpi r24,0x00               ; r24 == 0 means that we need to update r20 (unit counter)
  breq update_time_counter   ; if r24 == 0 go to update_time_counter
  rjmp loop                  ; else repeat the loop

  update_time_counter:
    ldi r22, 64             ; reset the values of the registers
    ldi r23, 20             ; used for the timer
    ldi r24, 2
    inc r20
    cpi r20, 0x0A           ; r20 == 10 means that we need to update the decimal counter
    breq update_decimal     ; if r20 == 10 go to update_decimal
    rjmp loop               ; else repeat the loop

  update_decimal:
    ldi r20, 0x00                  ; r20 is reset again to 0
    inc r21
    cpi r21, 0x0A                  ; r21 == 10 means that we need to reset r20 to 0
    breq set_decimal_counter_to_0  ; if r21 == 10 go to set_decimal_counter_to_0
    rjmp loop                      ; else repeat the loop

  set_decimal_counter_to_0:
    ldi r21, 0x00           ; r21 is reset again to 0
    rjmp loop               ; repeat the loop

; each of the subroutines set_XXXX change the values of
; r18 and r19 to the values needed to show that digit
set_0XXX:
  ldi r18, 0b00111100      ; turn off the LEDs and the segments
  out PORTB, r18           ; of the 4 digit 7 segment digit display
  ldi r19, 0b00000000
  out PORTD, r19
  ldi r18, 0b00111000
  out PORTB, r18
  ldi r19, 0b11111100
  out PORTD, r19
  ret

set_X0XX:
  ldi r18, 0b00111100     ; turn off the LEDs and the segments
  out PORTB, r18          ; of the 4 digit 7 segment digit display
  ldi r19, 0b00000000
  out PORTD, r19
  ldi r18, 0b00110110
  out PORTB, r18
  ldi r19, 0b11111100
  out PORTD, r19
  ret

set_decimal_long_jump:
```

```
  call set_decimal            ; call subroutine that turn on the 3nd digit
  rjmp jump_back_decimals     ; jump back to the original part of the code

set_decimal:
  ldi r18, 0b00111100         ; turn off the LEDs and the segments
  out PORTB, r18              ; of the 4 digit 7 segment digit display
  ldi r19, 0b00000000         ; the same as set_0XXX and set_X0XX
  out PORTD, r19
  cpi r21, 0x01               ; compare r21 with values between 0-9 to decide
  breq set_XX1X               ; which number to show on the third digit
  cpi r21, 0x02
  breq set_XX2X
  cpi r21, 0x03
  breq set_XX3X
  cpi r21, 0x04
  breq set_XX4X
  cpi r21, 0x05
  breq set_XX5X
  cpi r21, 0x06
  breq set_XX6X
  cpi r21, 0x07
  breq set_XX7X
  cpi r21, 0x08
  breq set_XX8X
  cpi r21, 0x09
  breq set_XX9X
  cpi r21, 0x00
  breq set_XX0X

set_XX1X:
  ldi r18, 0b00101100
  out PORTB, r18
  ldi r19, 0b00011000
  out PORTD, r19
  ret

set_XX2X:
  ldi r18, 0b00101101
  out PORTB, r18
  ldi r19, 0b01101100
  out PORTD, r19
  ret

set_XX3X:
  ldi r18, 0b00101101
  out PORTB, r18
  ldi r19, 0b00111100
  out PORTD, r19
  ret

set_XX4X:
  ldi r18, 0b00101101
  out PORTB, r18
  ldi r19, 0b10011000
  out PORTD, r19
  ret

set_XX5X:
  ldi r18, 0b00101101
  out PORTB, r18
  ldi r19, 0b10110100
  out PORTD, r19
  ret

set_XX6X:
  ldi r18, 0b00101101
  out PORTB, r18
  ldi r19, 0b11110100
  out PORTD, r19
  ret

set_XX7X:
  ldi r18, 0b00101100
  out PORTB, r18
  ldi r19, 0b00011100
```

```
    out PORTD, r19
    ret

set_XX8X:
    ldi r18, 0b00101101
    out PORTB, r18
    ldi r19, 0b11111100
    out PORTD, r19
    ret

set_XX9X:
    ldi r18, 0b00101101
    out PORTB, r18
    ldi r19, 0b10111100
    out PORTD, r19
    ret

set_XX0X:
    ldi r18, 0b00101100
    out PORTB, r18
    ldi r19, 0b11111100
    out PORTD, r19
    ret

set_unit_long_jump:
    call set_unit              ; call subroutine that turn on the 4nd digit
    rjmp jump_back_units       ; jump back to the original part of the code

set_unit:
    ldi r18, 0b00111100        ; turn off the LEDs and the segments
    out PORTB, r18             ; of the 4 digit 7 segment digit display
    ldi r19, 0b00000000        ; the same as set_0XXX and set_X0XX
    out PORTD, r19
    cpi r20, 0x01              ; compare r20 with values between 0-9 to decide
    breq set_XXX1              ; which number to show on the fourth digit
    cpi r20, 0x02
    breq set_XXX2
    cpi r20, 0x03
    breq set_XXX3
    cpi r20, 0x04
    breq set_XXX4
    cpi r20, 0x05
    breq set_XXX5
    cpi r20, 0x06
    breq set_XXX6
    cpi r20, 0x07
    breq set_XXX7
    cpi r20, 0x08
    breq set_XXX8
    cpi r20, 0x09
    breq set_XXX9
    cpi r20, 0x00
    breq set_XXX0

set_XXX1:
    ldi r18, 0b00011100
    out PORTB, r18
    ldi r19, 0b00011000
    out PORTD, r19
    ret

set_XXX2:
    ldi r18, 0b00011101
    out PORTB, r18
    ldi r19, 0b01101100
    out PORTD, r19
    ret

set_XXX3:
    ldi r18, 0b00011101
    out PORTB, r18
    ldi r19, 0b00111100
    out PORTD, r19
    ret
```

```
set_XXX4:
  ldi r18, 0b00011101
  out PORTB, r18
  ldi r19, 0b10011000
  out PORTD, r19
  ret

set_XXX5:
  ldi r18, 0b00011101
  out PORTB, r18
  ldi r19, 0b10110100
  out PORTD, r19
  ret

set_XXX6:
  ldi r18, 0b00011101
  out PORTB, r18
  ldi r19, 0b11110100
  out PORTD, r19
  ret

set_XXX7:
  ldi r18, 0b00011100
  out PORTB, r18
  ldi r19, 0b00011100
  out PORTD, r19
  ret

set_XXX8:
  ldi r18, 0b00011101
  out PORTB, r18
  ldi r19, 0b11111100
  out PORTD, r19
  ret

set_XXX9:
  ldi r18, 0b00011101
  out PORTB, r18
  ldi r19, 0b10111100
  out PORTD, r19
  ret

set_XXX0:
  ldi r18, 0b00011100
  out PORTB, r18
  ldi r19, 0b11111100
  out PORTD, r19
  ret
```