Lab 1

William Watkins, 31 Aug 21, 5067

Tuesday, August 31, 2021 21:04

1. Yes, everything eventually worked. Initially the LCD contrast was too low, but was quickly remedied.

2. From the schematic, going from the bottom pin to the top; the first pin connects to pin RF6, the next receives the V_{out} from the LM35, the top connects to RA3. Both RF6 & RA3 are inputs to the PIC18F87K22. They are also connected to the RF6 & RA3 buttons. Pressing RA3 overrides the LM35, and the LCD instead outputs a temperature corresponding to the voltage through the button (1.73V, in this case).

3. In SW4, switch 4 is active, connecting the buzzer & RB6. An input from the EXT ICD port, MCU PGD, shares a pin w/ RB6 on the MCU pin headers. So, the buzzer would be activated by data flowing from the EXT ICD port

4. When the switch for RG7 is pushed to high, the LED turns on. However, it turns off in the middle & down positions. Similarly pressing RG7 pulls the LED high, making it turn on. Behind the scenes, a pull-down resistor on SW3 is pulling the output (LED) low. By placing the switch in the up position or pressing the button, you are pulling RG7 high.

5. In a high-level programming language, like C, you are able to write code in pseudo-english, and are able to write code for multiple different MCUs. A low-level language, on the other hand, is developed by a manufacturer for a specific processor & is hard to

manufacturer for a specific processor & is hard to read. It does, however, have the benefit of being much more space-efficient than high-level languages.

6. Address Bus - Carry addresses of memory or I/O devices according to instructions from MPU. Unidirectional.

Data Bus - Carry data between MPU & memory or I/O devices. Bidirectional for R/W capability.

Control Lines - Carry timing signals that initiate various R/W operations of MPU

7. Address Bus is unidirectional b/c MPU uses it to address where next command is sent. Data Bus is bidirectional to enable both reading & writing data.

8. $1\text{KB} = 1024 \text{ bytes} \cdot \frac{8 \text{ bits}}{1 \text{ byte}} = 8,192 \text{ bits}$

9. 4MB mem
   → Assume each register is 8 bits wide

   $4\text{MB} = 1024\text{KB} = 1,048,576 \text{ bytes} - 1 \text{ (First register @ 000000)}$
   $= \text{register } 1,048,575$

$16\overline{)1,048,576}$ $\boxed{0\text{xFFFFF}}$
$\overline{)65,535}$
$\overline{)4,095}$
$\overline{)255}$
$\overline{)15}$
0

10. $FFFF_H = 64KB$, so 1FFFF = 128KB

11. The program counter is a 21-bit register that holds the program memory address while executing programs.

12.

2)42 0 → **101010B**

21 1

10 0

5 1

2 0

1 1

16)42 A → **0x2A**

2 2

0

2)76 0 → **1001100B**

38 0

19 1

9 1

4 0

2 0

1 1

0

16)76 C → **0x4C**

4 4

0

2)130 0 → **10000010**

65 1

32 0

16 0

8 0

4 0

2 0

1 1

0

16)130 2 → **0x82**

8 8

0

2)142 0 → **10001110B**

71 1

35 1

17 1

8 0

4 0

16)142 E → **0x8E**

8 8

0

8 0
4 0
2 0
1 1
0

2|245 1 → 11110101B

16|245 5 → 0xF5
15 F
0

-117 = 10001011B

-> invert

01110100 + 1

-117 = 01110101B → group binary into 4's → hex → 0x75

-76

76 = 01001100B -> invert 10110011 + 1 = 10110100B = -76

-> hex -> 0xB4

group binary into 4's

-255

2|255 1 11111111B -> 00000000B + 1 = 00000001B = -255

127 1

63 1 -> hex 0x01

31 1

15 1

7 1

3 1

1 1

0

$13_{16} \times 30 \rightarrow 3 \cdot 16^1 + 0 \cdot 16^0 = 48$

signed $\rightarrow 2\overline{)48}$ 0 $\quad 110000_{2's} - 1 \rightarrow 101111_B \rightarrow 010000_B$

24 0 $\quad \rightarrow 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$

12 0 $\quad = 16$

6 0

3 1

1 1

0

$0_{16} \times 22 \rightarrow 2 \cdot 16^1 + 2 \cdot 16^0 = 34$

signed $\rightarrow 2\overline{)34}$ 0 $\quad 100010_B - 1 \rightarrow 100001_B \rightarrow 011110_B$

17 1 $\quad \rightarrow 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$

8 0 $\quad = -30$

4 0

2 0

1 1

0

$0_{16} \times 72 \rightarrow 7 \cdot 16^1 + 2 \cdot 16^0 = 114$

signed $\rightarrow 2\overline{)114}$ 0 $\quad 1110010_B - 1 \rightarrow 1110001_B \rightarrow 0001110_B$

57 1 $\quad \rightarrow 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = -14$

28 0

14 0

7 1

3 1

1 1

0

$0_{16} \times 4D \rightarrow 4 \cdot 16^1 + 13 \cdot 16^0 = 77$

signed $2\overline{)77}$ 1 $\quad 1001101_{2's} - 1 \rightarrow 1001100_B \rightarrow 0110011_B$

38 0 $\quad \rightarrow 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -51$

19 1

9 1

4 0

4 ¢
2 ¢
1 1
¢

¢ x AB -> 10·16¹ + 11·16⁰ = 171

signed 2|171 | 10101011B - 1 -> 10101010B -> 01010101B
85 | -> 1·2⁶ + 1·2⁴ + 1·2² + 1·2⁰ = -85
42 ¢
2 1 1
10 ¢
5 1
2 ¢
1 1
¢

14. The 3.3V voltage regulator on the board steps the 5V supply down to 3.3V when the PIC18F87K22 ENVREG pin is tied to V_{DD}.

15. Voltage = 250.01 mV
gain = 10 mV/°C
Temp = 25°C = 77°F.
This makes sense, as it was a warm night, & is roughly (-1°F) the same as my apartment thermostat.

16. The noise, as measured by a max/min measurement in Waveforms, was ±10mV, which is ±1°C! An acceptable solution may be to take the average of the signal, or using a low-pass filter, or creating a hardware filter.

17. PORTB is 25mA, while PORTG is 2mA. This means an LED that draws 20mA should never go on PORTG!

an LED that draws 40mA should never go on PURIST!