Block 03

GPIO continued, C in microcontrollers

2023

EmLab @ FI MUNI

Financováno Evropskou unií
NextGenerationEU

Národní plán obnovy

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Seminar on Digital System Architecture

Why use C

- higher level abstractions
- human readable
- mantainability
- still low level enough

PIC18 toolchain

- proprietary XC8 toolchain
- can be launched from command line
- compatible flags with GCC[1]
- documentation here or in study materials

[1] not fully

C disassembly

- C can be easily translated to assembly
- let's look at some basic constructs
- dissassembly can be viewed in MPLAB

What not to use

- recursion (or lot of nested function calls)
- float/double unless needed
- dynamic allocation
- Why these?

What is normal in embedded C

- global variables
- macros
- pass by pointer, output arguments
- function pointers
- only a subset of standard library
- Why these?

PIC18 registers in C

- to read/write register, we need to know it's address
- will be unreadable
- include the `xc.h` and device specific header file
- we then can access the registers easily by their name

Example

```
    #include <xc.h>
    #include <pic18f44k22.h>
    void main(void) {
        PORTB = 0x02;
        uint8_t var = LATCHB;
        PORTAbits.RA4 = 1;
    }

```

LED output in C

- same steps needed as in ASM

Task

Open the `c_template` project and try displaying any value on the LEDs. Then try displaying some kind of animation.

GPIO continued

GPIO registers

- we already know two:
    - PORT
    - TRIS
- the rest:
    - LAT - output latch
    - ANSEL - digital buffer input enable
    - SLRCON - slew rate config

How it actually works

- all required info in datasheet
- Let’s look at it!

Button input

- reverse of LED output
- need to setup GPIO peripheral for input
- read PORT register to obtain value of input
- `c_template` project already has setup done
- Mind the ANSEL register, why?

Level and edge detect

- important difference between the two
- What is the difference?

Tasks

- Write code that increments a counter value displayed on LEDs every time you push the button.
- First try level detection, then try edge detection.
- Think about which one you want to use in which situation.

(De)bouncing

- sometimes you detect more edges than expected
- Why?
- multiple ways to deal with it
  - hardware - capacitor, schmidt triggers
  - software - counting, timing

Task

Write a counter based button debouncing (one button is enough).

Homework

Mandatory

Nothing.

Optional

Draw a schematic of button matrix (ie. a numpad) and analyze how it works. Write pseudocode of suitable code to read the matrix input.