Lab 2: Buttons and LEDs

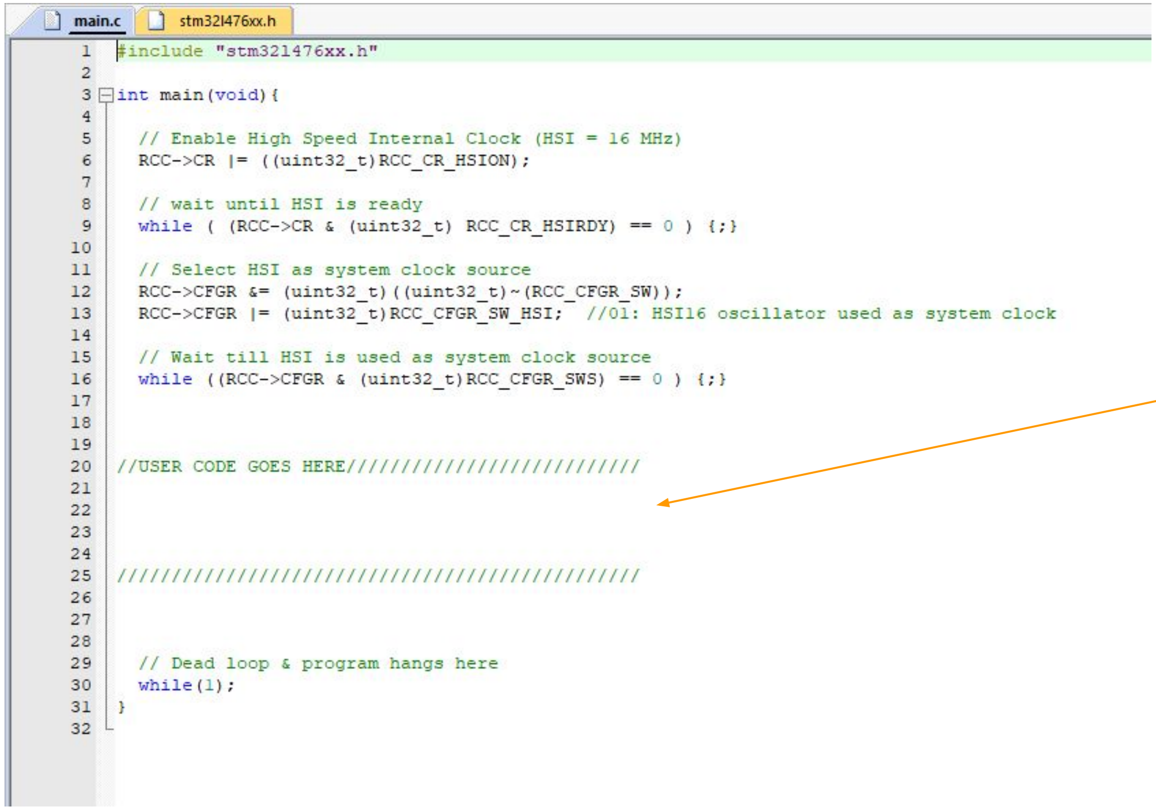
ECE 2020

Lab Outline

1. Download C template project from courseweb
2. Enable peripheral clocks for buttons and LEDs
3. Set configuration registers for buttons and LEDS
4. Write a loop that does: **if** button is pressed ➔ **then** turn on LED’s

- ➢ Show us the project on your board (checkoff)
- ➢ Manually toggle LED in debug mode (checkoff)
- ➢ Submit code and pre lab answers on courseweb

1. Download C template



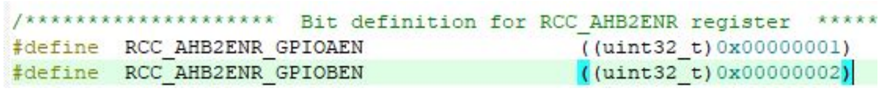
```c
#include "stm32l476xx.h"

int main(void){

  // Enable High Speed Internal Clock (HSI = 16 MHz)
  RCC->CR |= ((uint32_t)RCC_CR_HSION);

  // wait until HSI is ready
  while ( (RCC->CR & (uint32_t) RCC_CR_HSIRDY) == 0 ) {;}

  // Select HSI as system clock source
  RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
  RCC->CFGR |= (uint32_t)RCC_CFGR_SW_HSI;  //01: HSI16 oscillator used as system clock

  // Wait till HSI is used as system clock source
  while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) == 0 ) {;}



//USER CODE GOES HERE//////////////////////////



//////////////////////////////////////////////////



  // Dead loop & program hangs here
  while(1);
}
```

Only code you should change

2. Configuring Clock

RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN; Reset and Clock Control (RCC)

A number defined in stm32l476xx.h that gives the mask to enable port B's clock



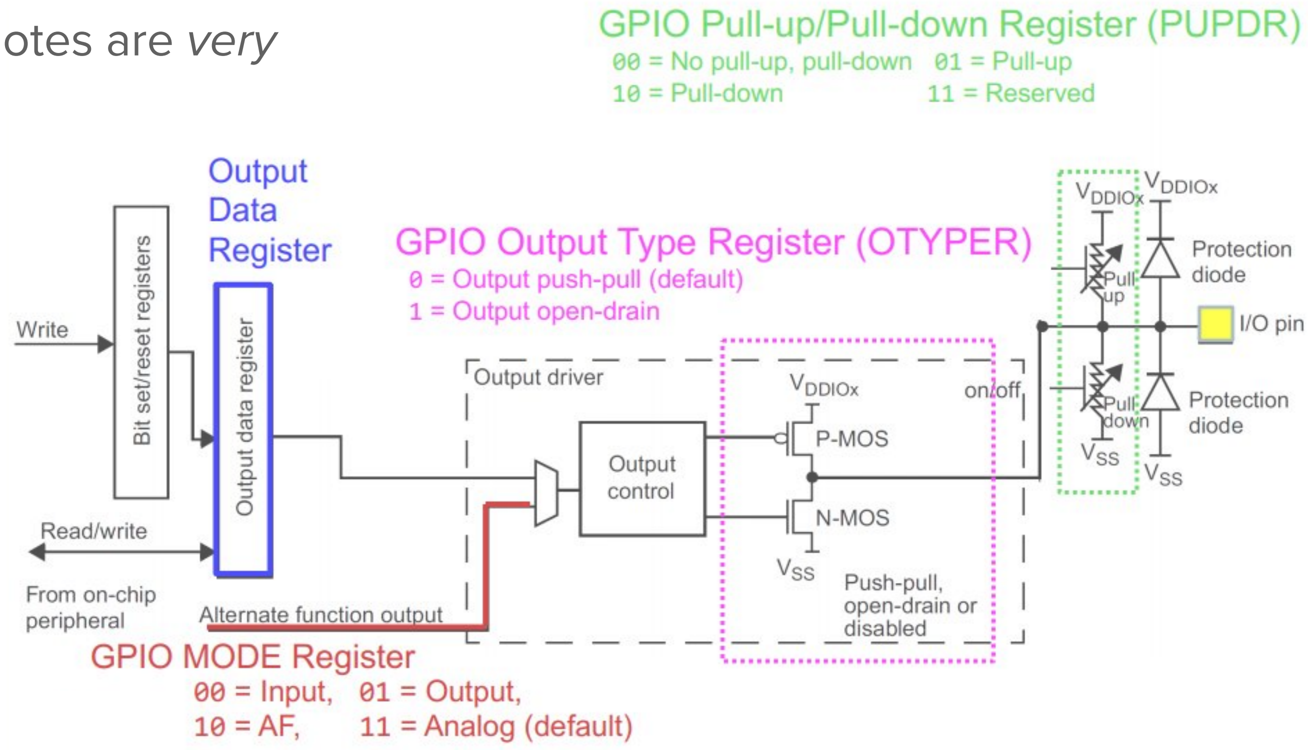
```
/********************  Bit definition for RCC_AHB2ENR register  *****
#define  RCC_AHB2ENR_GPIOAEN                         ((uint32_t)0x00000001)
#define  RCC_AHB2ENR_GPIOBEN                         ((uint32_t)0x00000002)
```

Struct defined in stm32l476xx.h, stands for "Reset and Clock Control"

Register found in this struct
Stands for: "Advanced High-Performance Bus 2 Enable Register"

3. Configuring Peripherals: LED

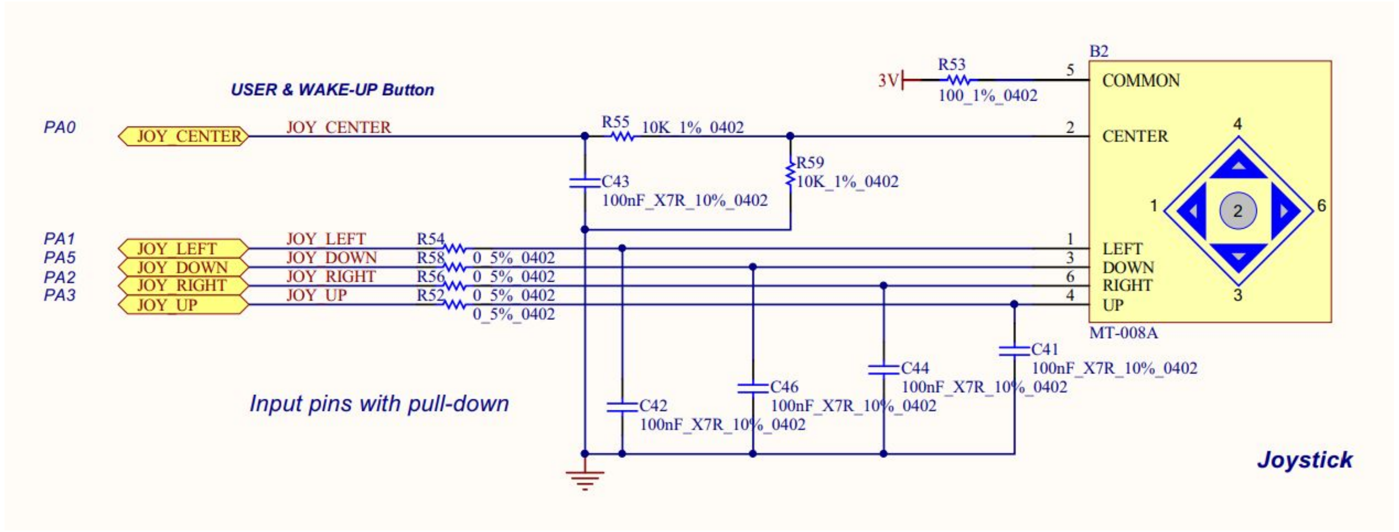
- GPIOB/E->...your specific config register
- Pre lab and lecture notes are *very* helpful



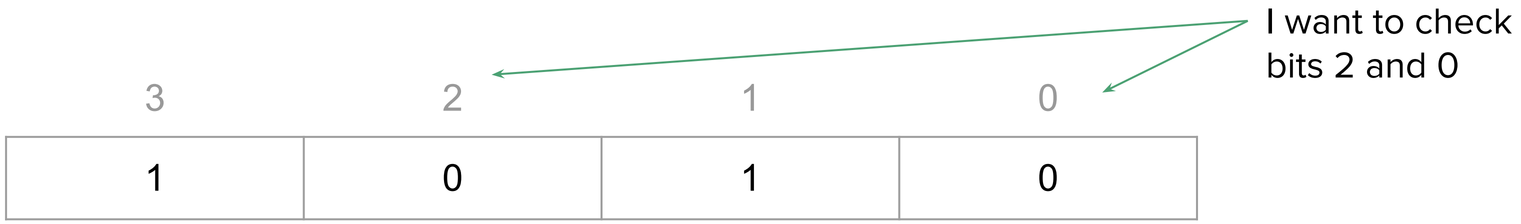
GPIO Pull-up/Pull-down Register (PUPDR)
00 = No pull-up, pull-down 01 = Pull-up
10 = Pull-down 11 = Reserved
Output Data Register
GPIO Output Type Register (OTYPER)
0 = Output push-pull (default)
1 = Output open-drain
Write
Bit set/reset registers
Output data register
Read/write
From on-chip peripheral
Alternate function output
Output driver
Output control
V_DDIOx
P-MOS
N-MOS
V_SS
on/off
Push-pull, open-drain or disabled
Pull up
Pull down
Protection diode
I/O pin
GPIO MODE Register
00 = Input, 01 = Output,
10 = AF, 11 = Analog (default)

3. Configuring Peripherals: Buttons

- GPIOA->...your specific config register
- Will need to set your mode (MODER), resistors (PUPDR), and read from the input register (IDR)



Reading from the IDR Register



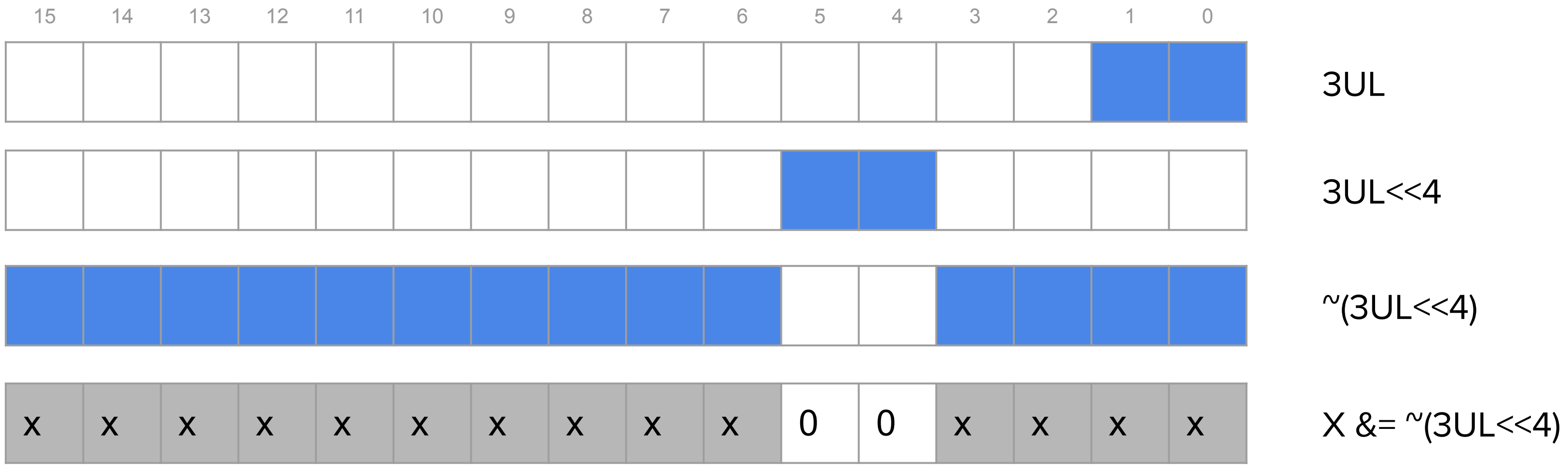
Int X = (Input Register & (0b0101))

X = 0b0000

Boolean ON = (X!=0)

Masking

Example: GPIOB->MODER &= ~(3UL<<4);



Coding Style & Submission Tips

- It helps us if you submit your code as .c/.s files (i.e. “text”), please don’t send screenshots
- Comments are necessary to get full credit (plus its a good habit to form now)