Lab 1: Interfacing Push-button and LED
Instructor: Prof. Yifeng Zhu
Spring 2015

Goals

1. Get familiar with the Keil uVision software development environment
2. Create a C project for STM32L Discovery Kit and program the kit
3. Learn basics of GPIO input and output configuration
4. Perform simple digital I/O input (button) and output (LED)
5. Understand polling I/O (busy waiting) and its inefficiency

Grading Rubrics (Total = 20 points)

1. Pre-lab assignment (2 points)
2. Documentation and Maintainability (5 points)
3. Functionality and Correctness (5 points)
4. Lab Demonstration (5 points)
5. Something cool (3 points)

NOTE*: Do* ***NOT*** *connect the Discovery Kit into your PC or laptop before installing the software. Windows might associate the kit with an incorrect USB device driver.*

Pre-lab assignment

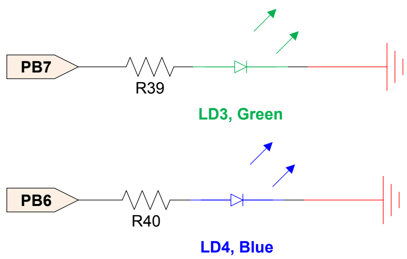
1. Windows Operation Systems is required in ECE 271. Our development software, Keil uVision, can only run Windows machines or virtual machines.
2. Download free Keil uVision (MDK-Lite Edition) and install it. It is free but limits your data and code to 32 KB, which is not an issue for all homework and lab assignments in ECE 271.
3. Follow the tutorial of setting up Gitlab sever and download your lab repository
4. Review: Read Textbook Chapter 4.6 to review bitwise operations.

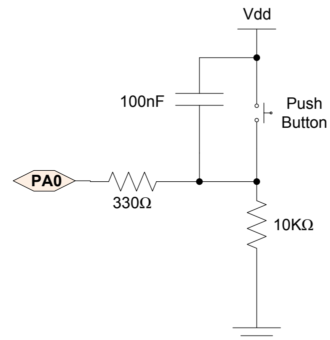
Lab assignment

- Following Book Chapter 14 and implement a C program that toggles both Blue and Green LED when the user button is pressed.
- Do something cool. The following gives a few examples but you are not limited to this. ***Creative ideas are always encouraged.***
  - Using an oscilloscope to show the voltage output of LED and the voltage of the pushbutton pin. Find out the latency between the button pressed and LED lighting up.
  - Using the software logic analyzer provided in MDK-KEIL to analyze the digital input and output signals.
  - Using an oscilloscope to show the GPIO output signal difference when the GPIO have different output speeds.
  - Using GPIO a LED to send out SOS in Morse code (· · · – – – · · ·) if the user button is pressed.

Blue and Green LEDs on the Board

There are two LEDs on the STM32L Discovery board, which are connected to the GPIO Port B Pin 6 (PB6) and the GPIO Port B Pin 7 (PB7) pin of the STM32L processor, respectively. To light up a LED, the corresponding LED pin must be set up to 1, i.e., active high.





Note: At ambient temperature, the GPIOs (general purpose input/outputs) can sink or source up to **±8 mA**.

PIN Connections

STM32L152RBT6 is based on LQFP64 package and has 64 pins.

- USER Pushbutton: connected to PA0 (GPIO Port A, PIN 0), CLK RCC_AHBENR_GPIOAEN
- RESET Pushbutton: connected RESET
- GREEN LED: connected to PB7 (GPIO Port B, PIN 7), CLK RCC_AHBENR_GPIOBEN
- BLUE LED: connected to PB6 (GPIO Port B, PIN 6), CLK RCC_AHBENR_GPIOBEN
- Linear touch sensor/touchkeys: PA6, PA7 (group 2), PC4, PC5 (group 9), PB0, PB1 (group 3)

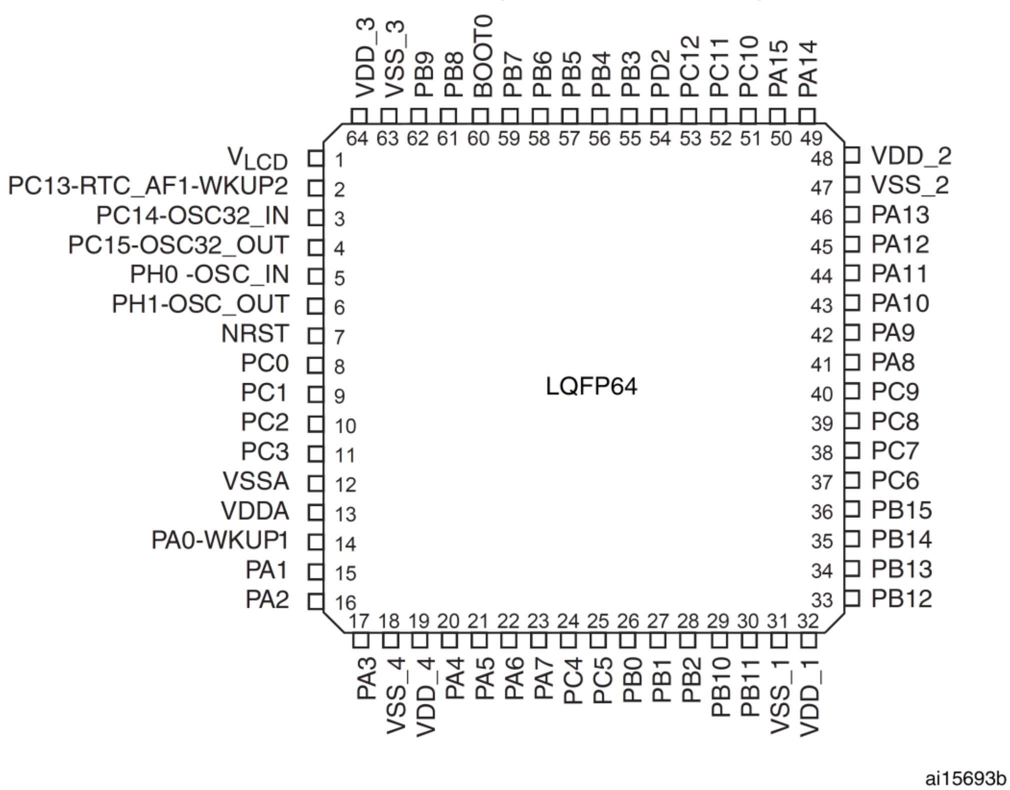


Figure 1 LQFP64 Package of STM32L152RBT6 used in STM32L Discovery Kit

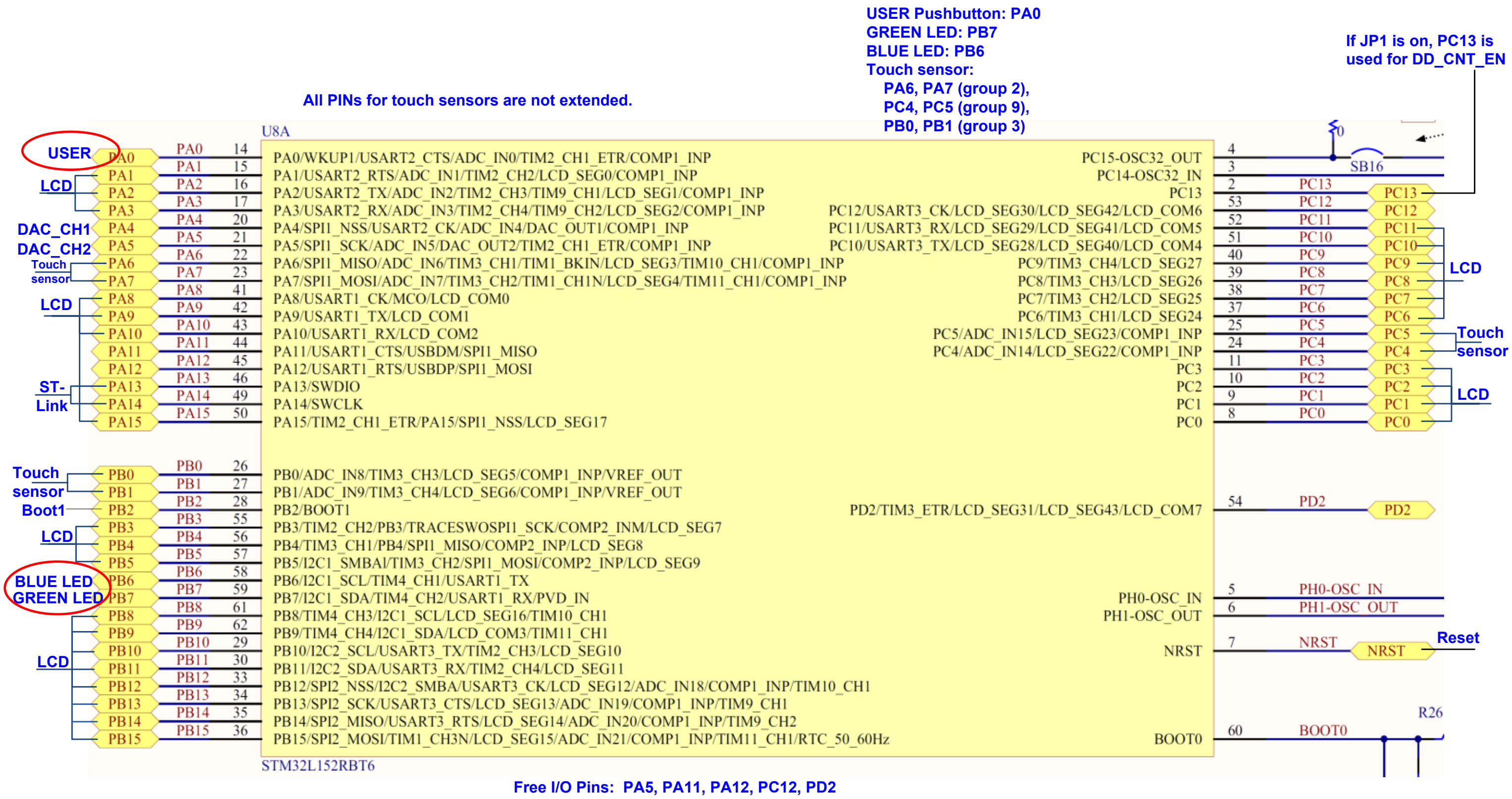


Figure 2. PIN Connection of STML-Discovery Board

Clock Configuration

There are two major types of clocks: system clock and peripheral clock.

System Clock

In order to meet the requirement of performance and energy-efficiency for different applications, the processor core can be driven by four different clock sources, including (1) HSI (high-speed internal) oscillator clock, (2) HSE (high-speed external) oscillator clock, (3) PLL clock, (4) MSI (multispeed internal) oscillator clock. A faster clock provides better performance but usually consumes more power, which is not appropriate for battery-powered systems.

The default system clock is MSI (multispeed internal) oscillator clock. The MSI is used as system clock source after startup from Reset, wake-up from Stop or Standby low power modes. The MSI clock signal is generated from an internal RC oscillator. Its frequency range can be adjusted by software by using the MSIRANGE[2:0] bits in the RCC_ICSCR register. Seven frequency ranges are available: 65.536 kHz, 131.072 kHz, 262.144 kHz, 524.288 kHz, 1.048 MHz, ***2.097 MHz (default value)*** and 4.194 MHz.

Peripheral Clock

All peripherals require to be clocked to function. However, ***clocks of all peripherals are turned off by default in order to reduce power consumption***. Figure 3 shows the diagram of STM32L152RBT6, the processor used in the STM32L Discovery kit. The clock sources in the domain of Advanced High-performance Bus (*AHB*), low-speed Advanced Peripheral Bus 1 (*APB1*) and high-speed Advanced Peripheral Bus 2 (*APB2*) can be switched on or off independently when it is not used. Figure 4 shows the clock tree of STM32L152RBT6. Since the green and blue LEDs are connected to the peripheral GPIO Port B, we need to set the corresponding bit of the RCC_AHBENR to enable the clock of GPIO Port B.

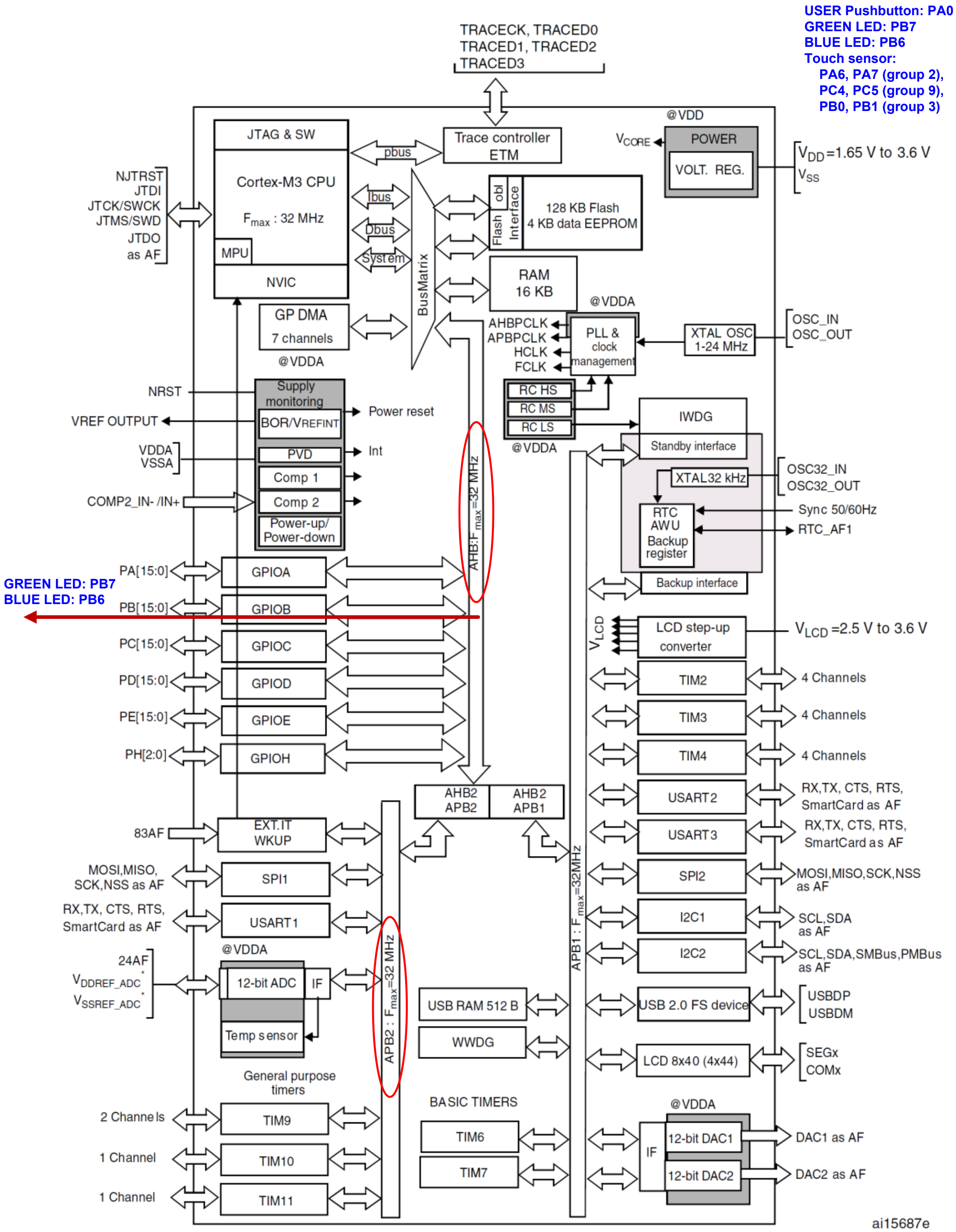


Figure 3 Diagram of STM32L152RBT6. The clock sources of the AHB, APB2, and APB1 domain are off by default.

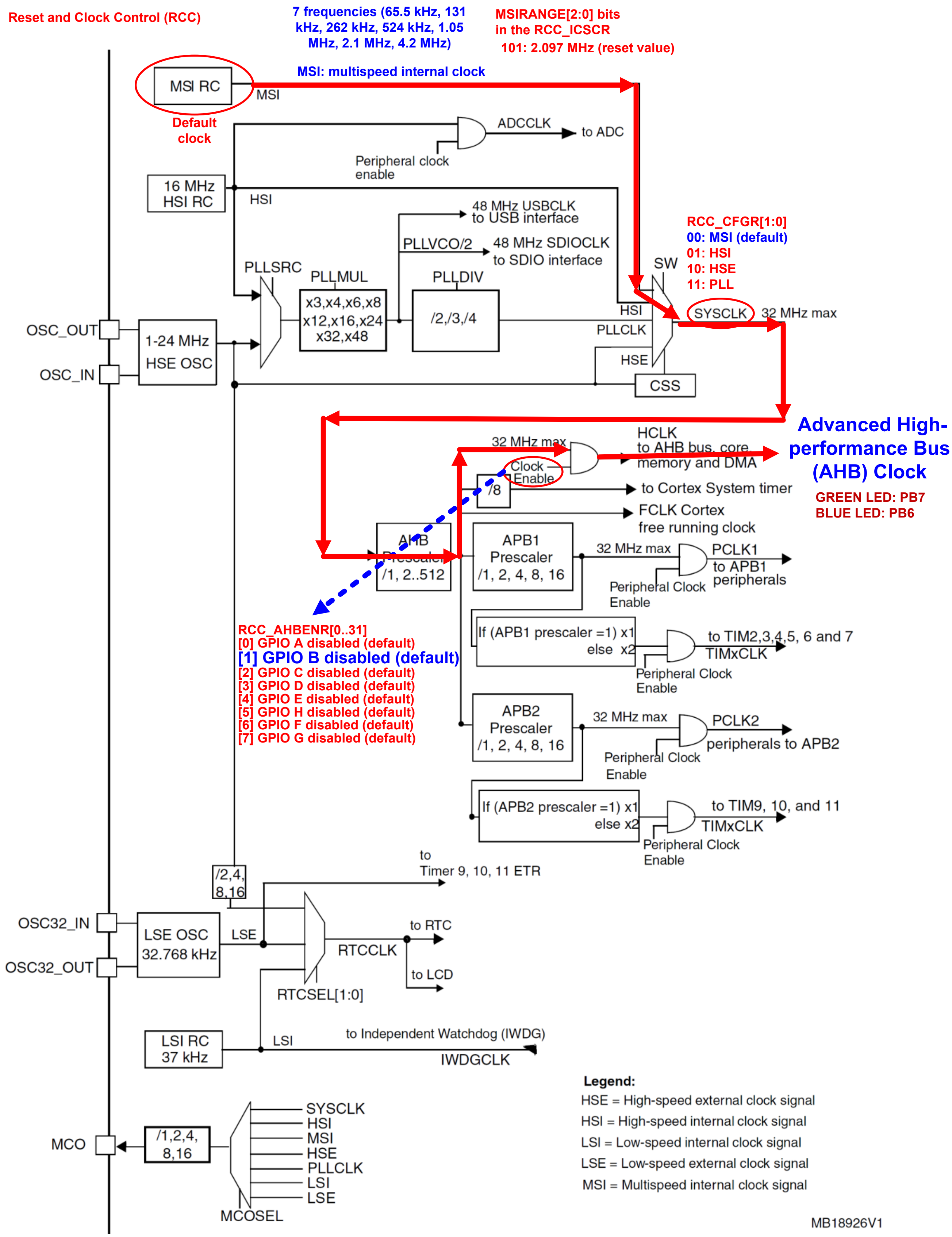
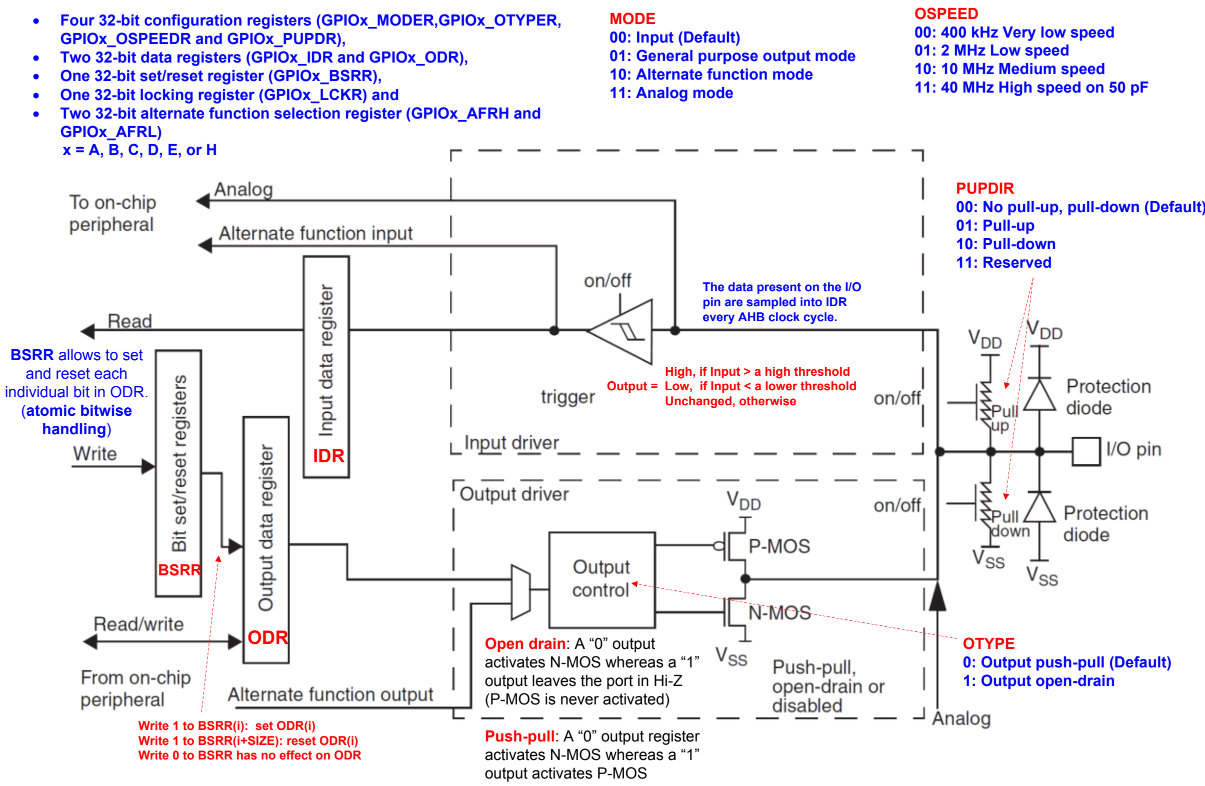


Figure 4. Enable the clock to GPIO Port B

Each of the GPIO pins can be configured by software as output (push-pull or open-drain), as input (with or without pull-up or pull-down) or as peripheral alternate function. ***In this lab, we will configure PB7 as push-pull output.***



Code Comments and Documentation

Program comments are used to improve code readability, and to assist in debugging and maintenance. A general principal is “Structure and document your program the way you wish other programmers would” (McCann, 1997). The book titled “the Elements of Programming Style” by Brian Kernighan and P. J. Plauger gives good advices for beginners.

1. Format your code well. Make sure it's easy to read and understand. Comment where needed but don't comment obvious things it makes the code harder to read. If editing someone else's code, format consistently with the original author.
2. Every program you write that you intend to keep around for more than a couple of hours ought to have documentation in it. Don't talk yourself into putting off the documentation. A program that is perfectly clear today is clear only because you just wrote it. Put it away for a few months, and it will most likely take you a while to figure out what it does and how it does it. If it takes you a while to figure it out, how long would it take someone else to figure it out?
3. Write Clearly - don't be too clever - don't sacrifice clarity for efficiency.
4. Don’t over comment. Use comments only when necessary.
5. Format a program to help the reader understand it. Always Beautify Code.
6. Say what you mean, simply and directly.
7. Don't patch bad code - rewrite it.
8. Make sure comments and code agree.
9. Don't just echo code in comments - make every comment meaningful.
10. Don't comment bad code - rewite it.
11. The single most important factor in style is consistency. The eye is drawn to something that "doesn't fit," and these should be reserved for things that are actually different.

Lab 1: Pre-Lab Assignment (2 points)
Spring 2015

Student Name: ____________________

TA: ____________________

Time & Date: ____________________

1. Enable the clock of GPIO Port A (for user push-button) and Port B (for Blue and Green LEDs)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHBENR	Reserved	FSMCEN	Reserved		AESEN	Reserved	DMA2EN	DMA1EN	Reserved								FLITFEN	Reserved		CRCEN	Reserved				GPIOPGEN	GPIOPFEN	GPIOPHEN	GPIOPEEN	GPIOPDEN	GPIOPCEN	GPIOPBEN	GPIOPAEN
Mask																																
Value																																

2. GPIO Output Pins Initialization (PB.6 Blue LED and PB.7 Green LED)

a. Configure PB 6 (Blue LED), PB 7(Green LED) as Output

GPIO Mode: Input (00, reset), Output (01), AlterFunc (10), Analog (11)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOB MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
Mask																																
Value																																

GPIOB Mode Register MASK Value = 0x____________________ (in HEX)

GPIOB Mode Register Value = 0x____________________ (in HEX)

b. Configure PB 6 (Blue LED), PB 7(Green LED) Output Type as Push-Pull

Push-Pull (0, reset), Open-Drain (1)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTYPER	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
Mask																																
Value																																

GPIOB Output Type Register MASK Value = 0x____________________ (in HEX)

GPIOB Output Type Register Value = 0x____________________ (in HEX)

c. Configure PB 6 (Blue LED), PB 7(Green LED) Output Type as No Pull-up No Pull-down

NO PUPD (00, reset), Pullup (01), Pulldown (10), Reserved (11)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
Mask																																
Value																																

GPIOB Pull-up Pull-down Register MASK Value = 0x________________________________ (in HEX)

GPIOB Pull-up Pull-down Register Value = 0x________________________________ (in HEX)

3. GPIO Input Pin Initialization (PA.0 for the user push button)

a. Configure PA.0 as Input

GPIO Mode: Input (00, reset), Output (01), AlterFunc (10), Analog (11)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOA MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
Mask																																
Value																																

GPIOA Mode Register MASK Value = 0x________________________________ (in HEX)

GPIOA Mode Register Value = 0x________________________________ (in HEX)

b. Configure PA.0 as No Pull-up No Pull-down

NO PUPD (00, reset), Pullup (01), Pulldown (10), Reserved (11)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
Mask																																
Value																																

GPIOA Pull-up Pull-down Register MASK Value = 0x________________________________ (in HEX)

GPIOA Pull-up Pull-down Register Value = 0x________________________________ (in HEX)

4. Set the output bits for PB 6 (Blue LED), PB 7(Green LED) to light them up.

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR	Reserved																ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
Mask																																
Value																																

GPIOB Data Output Register MASK Value = 0x____________________ (in HEX)
GPIOB Data Output Register Value = 0x____________________ (in HEX)

5. Read digital input (Push Button) and Write digital output (LED)

The digital outputs of each GPIO port are saved in the Output Data Register (GPIOx_ODR).
The digital inputs of each GPIO port are saved in the Input Data Register (GPIOx _IDR).

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR	Reserved																ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
Mask																																
Value																																

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR	Reserved																IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
Mask																																
Value																																

Lab 1: Lab Demo (1 point)

You should be able to correctly answer the following questions to TAs.

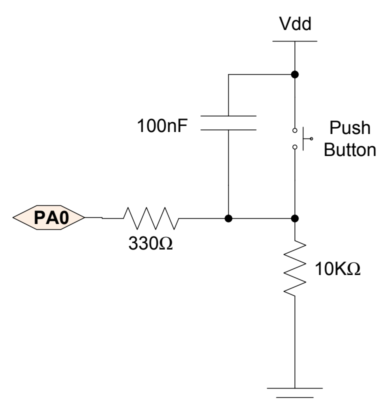
1. Why did we configure PB.6 and PB.7 as push-pull instead of open-drain?
2. What is GPIO speed? Did you notice any difference of you choose different speeds in this lab assignment?
3. Show TA that you can toggle a LED in the debug environment by directly changing the value of the Output Data Register (ODR) of a GPIO port.
4. Show the voltage signal of Pin PA.0 on an oscilloscope when the user button is pressed and then released.

Lab 1: Lab Code Submission

1. Submit and maintain your code in Gitlab server
2. Make sure to comment your codes appropriately
3. Make sure to complete the Readme.Md file

Lab 1: Post-Lab Assignment (1 point)

1. The push button on the STM32L board has a hardware debouncing circuit. Explain briefly how the hardware debouncing circuit work. Find out a typical solution for software debouncing.



(Answer this question in the file Readme.md and submit it to the Gitlab Server)