TUTORIAL: Digital I/O

LED Toggle with Push-Button

Name: Yechan Kim ID: 22100153

# I. Introduction

We will learn how to control digital I/O of GPIOs of the MCU board to turn on/off an LED with a push-button input. The LED should be turned on when the button is pressed.

The objectives of this lab are to learn how to

* Read and configure registers of digital GPIO of MCU
* Program firmware to control digital input/output pins
* Create your own functions for GPIOs

### Hardware

NUCLEO -F411RE

### Software

Keil uVision IDE, CMSIS, EC\_HAL

### Documentation

[STM32 Reference Manual](https://ykkim.gitbook.io/ec/stm32-m4-programming/hardware/nucleo-f411re" \l "manual-documentation)

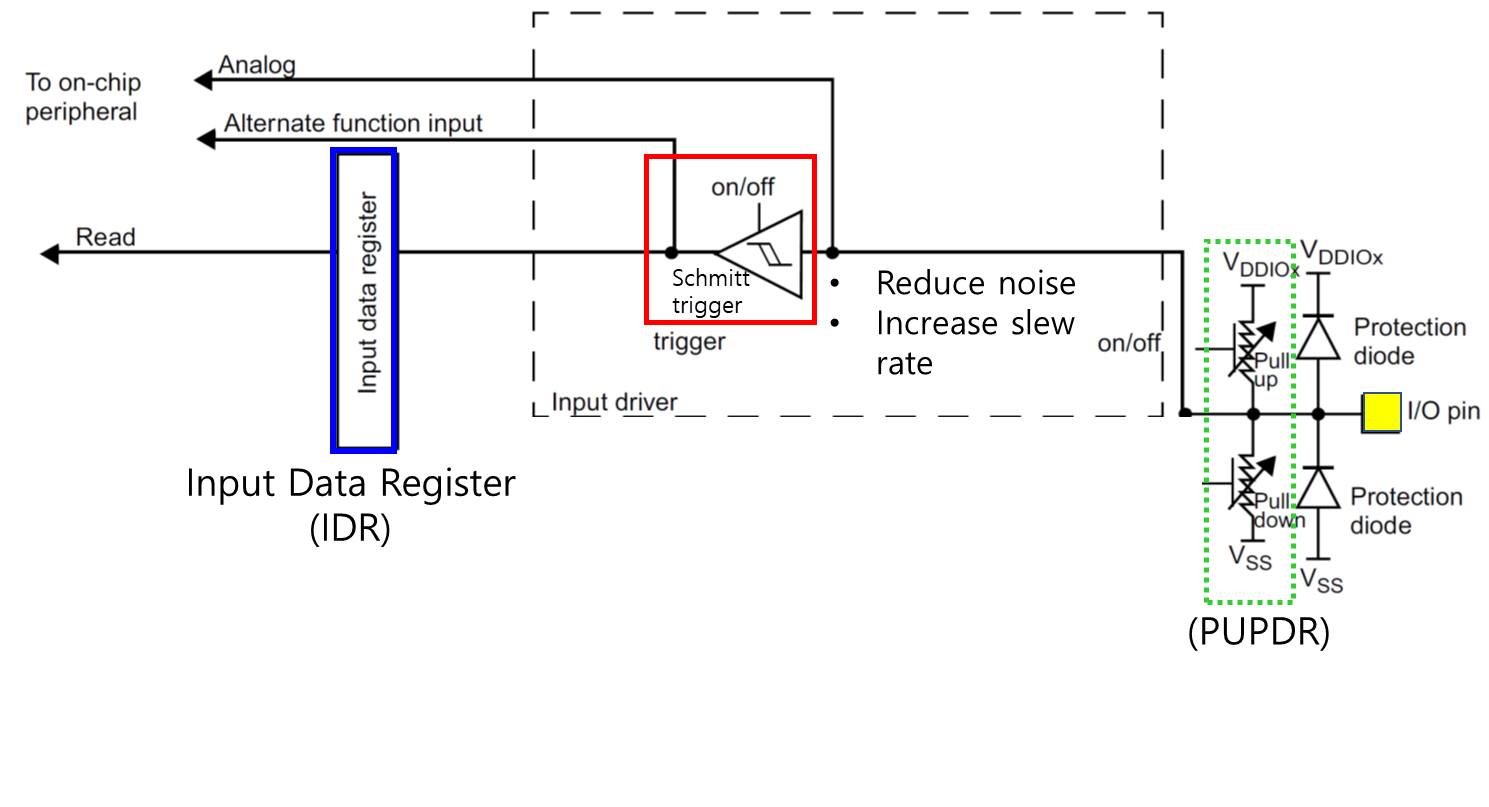
# II. Basics of GPIO IN

## A. GPIO Digital Out Register

List GPIO registers for this LAB

|  |  |  |
| --- | --- | --- |
| Type | Register Name | Description |
| GPIO | MODER | Mode: Input |
|  | PUPDR | Pull-Up Pull-Down: |
|  | IDR | Input Data Register |

Schematic



Process of GPIOx register initiation

|  |
| --- |
| 0. Enable Peripheral Clock (**AHB1ENR**)  1. Configure as Digital Input (**MODER**)  2. Configure pull-up/down resistors (**PUPDR**)  3. Read Data **(IDR)** |

# III. Tutorial

## A. Register Configuration

**1. GPIO: Digital In - Pin Initialization & Read PushButton**

Port C Pin 13 / Input // Pull-Up

use **#define BUTTON\_PIN 13**

* **MODER:** Input (MODER5[1:0]=[0 0])

|  |
| --- |
| GPIOC ->MODER &= ~(3<<( BUTTON\_PIN \*2)); // clear bits at both [26] and [27] |

|  |
| --- |
| 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** |
| **Initial** | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| **Logic** | Bitwise AND | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **mask** | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **Value** | x | x | x | x | 0 | 0 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |

* **PUPDR:** pull-up (PUPDR13[1:0]=[0 1])

|  |
| --- |
| GPIOC->PUPDR &= ~(3<<( BUTTON\_PIN \*2));  GPIOC->PUPDR |= (1<<( BUTTON\_PIN \*2)); |

|  |
| --- |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | |
| **Initial** | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | |
| **Logic** | Bitwise OR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **mask** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **Value** | x | x | x | x | 0 | 1 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | |

* **IDR:** Read Push-Button Value (IDR13= 1)

|  |
| --- |
| GPIOC->IDR =(IDR>>BUTTON\_PIN) & (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** | |
| **Initial** | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | |
| **Logic** | Bitwise OR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **mask** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **Value** | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 1 | x | x | x | x | x | x | x | x | x | x | x | x | x | |

## B. Programming

**Preparation**

* + Open the program ‘Keil uVision5’ and create a new project named as ‘**TU\_GPIO\_Digital\_InOut\_LED\_Button’**.

*“../****repos/EC/Tutorial/TU\_GPIO\_Digital\_InOut\_LED\_Button/****”*

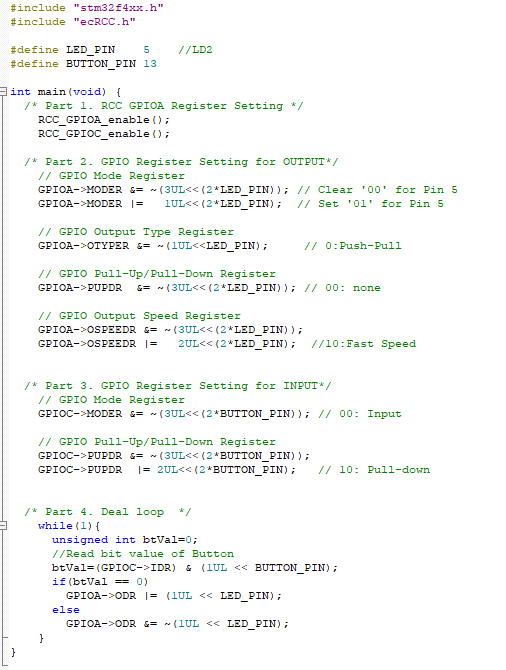
* + Create a new item called ‘**TU\_GPIO\_Digital\_InOut\_LED\_Button.c’**
  + Copy and paste from the source ‘[TU\_GPIO\_Digital\_InOut\_LED\_Button\_student.c](https://github.com/ykkimhgu/EC-student/tree/main/tutorial/tutorial-student)**[’](https://github.com/ykkimhgu/EC-student/tree/main/tutorial/tutorial-student)**.
  + Include provided **ecRCC2.h** and **ecRCC2.c** library files in your project.

**Exercise**

Fill in the empty spaces in the code. Then, compile(F7) and flash(F8) the source code

on the MCU board.

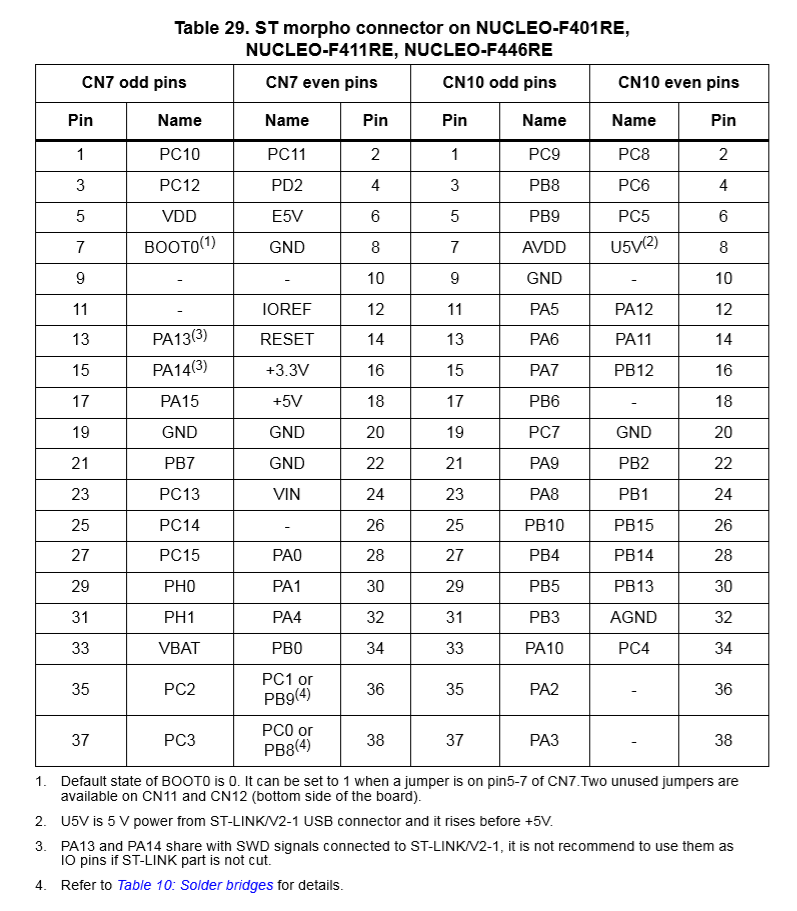
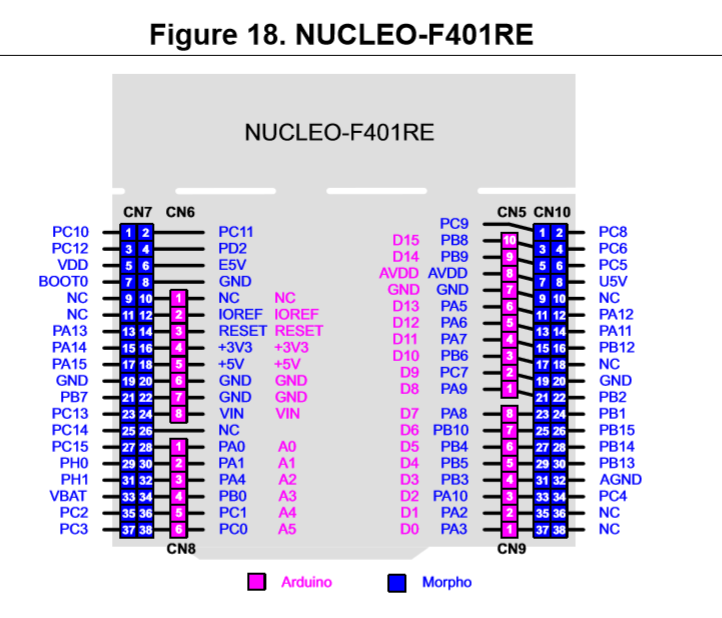
**Solution**



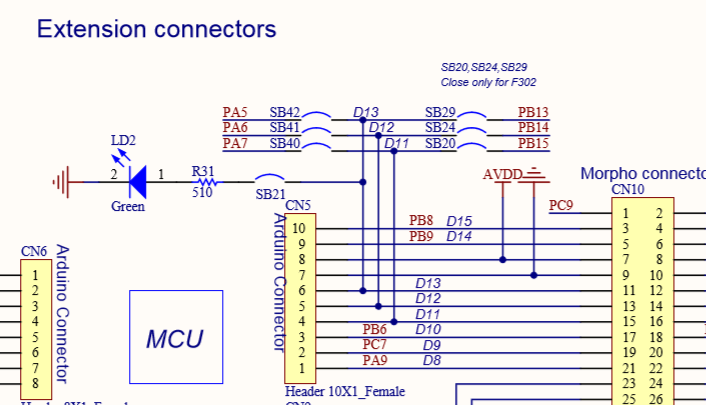
## Appendix

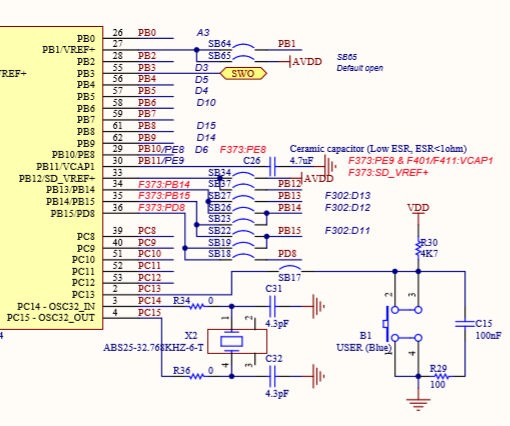
[See here for MCU resources](https://ykkim.gitbook.io/ec/resource/nucleo-f411re)

1. Pin Configuration of NUCLE-F401RE

****

1. LED/Button Circuit Diagram





## Result

**Source Code**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @author  SSSLAB

\* @Mod       2024-09-06 by YKKIM

\* @brief   Embedded Controller:  Tutorial Digital In

\*                    - Turn on LED LD2 while Button B1 is pressing

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

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

// GPIO Speed            : Low speed (00), Medium speed (01), Fast speed (10), High speed (11)

// GPIO Output Type: Output push-pull (0, reset), Output open drain (1)

// GPIO Push-Pull    : No pull-up, pull-down (00), Pull-up (01), Pull-down (10), Reserved (11)

#include "stm32f4xx.h"

#include "ecRCC2.h"

#define LED\_PIN    5        //LD2

#define BUTTON\_PIN 13

int main(void) {

    /\* Part 1. RCC GPIOA Register Setting \*/

        RCC\_HSI\_init();

        RCC\_GPIOA\_enable();

        RCC\_GPIOC\_enable();

    /\* Part 2. GPIO Register Setting for OUTPUT\*/

        // GPIO Mode Register

        GPIOA->MODER &=  ~(3<<(2\*LED\_PIN));                                     // Clear '00' for Pin 5

        GPIOA->MODER |=  (1<<(2\*LED\_PIN));                                      // Set '01' for Pin 5

        // GPIO Output Type Register

        GPIOA->OTYPER &= ~(1<<LED\_PIN);                                     // 0:Push-Pull

        // GPIO Pull-Up/Pull-Down Register

        GPIOC->PUPDR  &=     ~(3<<( LED\_PIN \*2));                                       // 00: none

        // GPIO Output Speed Register

        GPIOA->OSPEEDR &=  ~(3<<LED\_PIN);

        GPIOA->OSPEEDR |=  (2<<LED\_PIN);                                     //10:Fast Speed

    /\* Part 3. GPIO Register Setting for INPUT\*/

        // GPIO Mode Register

        GPIOC->MODER &= ~(3<<(BUTTON\_PIN\*2));                                       // 00: Input

        // GPIO Pull-Up/Pull-Down Register

        GPIOC->PUPDR &= ~(3<<(BUTTON\_PIN\*2));

        GPIOC->PUPDR |= (2<<(BUTTON\_PIN\*2));                                    // 10: Pull-down

    /\* Button Value Initialization \*/

        unsigned int btVal=0;

    /\* Part 4. Deal loop  \*/

        while(1){

            //Read bit value of Button

            btVal= (GPIOC->IDR) & (1UL << BUTTON\_PIN);

            if(btVal == 0)

                GPIOA->ODR |= (1UL << LED\_PIN);

            else

                GPIOA->ODR &= ~(1UL << LED\_PIN);

        }

}

**Screen Shot**

