TUTORIAL: External Interrupt

LED Toggle with Push-Button

Name: 김은찬 ID: 21801017

I. Introduction

In this tutorial, we will learn how to use External Interrupt. We will create functions that capture the falling edge trigger by pushing a button using an external interrupt.

The objectives of this lab are to learn how to

- Configure External input (EXTI) interrupt with NVIC
- Create your own functions for configuration of interrupts

Hardware

NUCLEO -F411RE

Software

Keil uVision IDE, CMSIS, EC_HAL

Documentation

STM32 Reference Manual

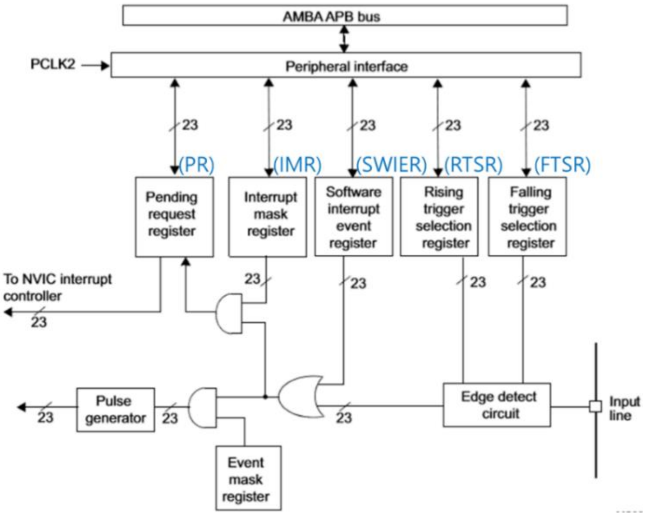
II. Basics of External Interrupt (EXTI)

A. Register List

List of external interrupt (EXTI) registers used in this tutorial [Reference Manual ch7, ch10.2]

Type	Register Name	Description
SYSCFG	SYSCFG_EXTICRx	External Interrupt Configuration, x=1 to 4 EXTICR1: for pin0~pin3 , EXTICR2: for pin4~pin7, etc
EXTI_	EXTI_IMR	Interrupt Mask
	EXTI_FTSR EXTI_RTSR	Falling/Rising Trigger Selection

Schematic



B. Register Setting

(Digital Input Setting)

• Enable GPIO peripheral clock	**RCC->AHB1ENR**
• Configure DigitalIn pin	

(EXTI Setting)

• Enable SYSCFG peripheral clock.	**RCC->APB2ENR**
• Connect the corresponding external line to GPIO	**SYSCFG->EXTICR**
• Configure the trigger edge.	**EXTI->FTSR/RTSR**
• Configure Interrupt mask	**EXTI->IMR**
• Enable EXTI.	**EXTI->IMR**

(NVIC Setting)

• Configure the priority of EXTI interrupt request.	**NVIC_SetPriority(**)
• Enable EXTI interrupt request.	**NVIC_EnableIRQ(**)

(EXTI Use)

• Create user codes in handler	**EXTIx_IRQHandler()**
• Clear pending bit after interrupt call	

III. Tutorial

A. Register Configuration

1. Pin Initialization & Set LED and Push-button

LED: Port A Pin 5 / Output / Push-Pull / No Pull-Up & No Pull-Down

Push-button: Port C Pin 13 / Input / No Pull-Up & No Pull-Down

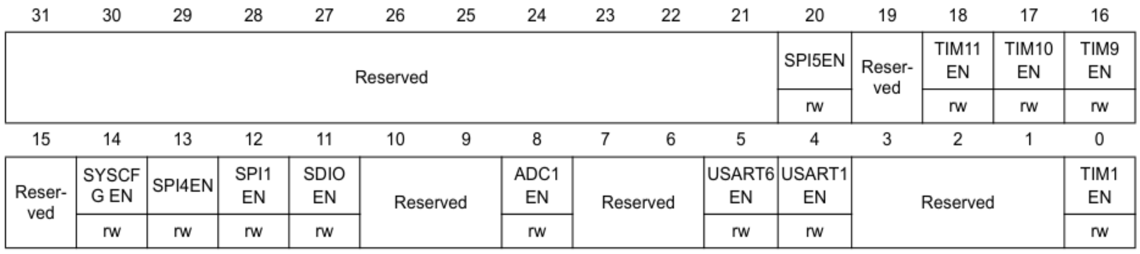
```
// code using your library functions
GPIO_init(GPIOA, 5,OUTPUT);
GPIO_pupd(GPIOA, 5, EC_NONE);
GPIO_init(GPIOC, 13,INPUT);
GPIO_pupd(GPIOC, 13, EC_NONE);
```

2. Enable Peripheral Clock: SYSCFGEN

- **RCC_APB2ENR:** Enable SYSCFG

RCC->APB2ENR |= 1<<14

// Paste RCC_APB2ENR register map



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											SPI5EN	Reserved	TIM11 EN	TIM10 EN	TIM9 EN
											rw		rw	rw	rw

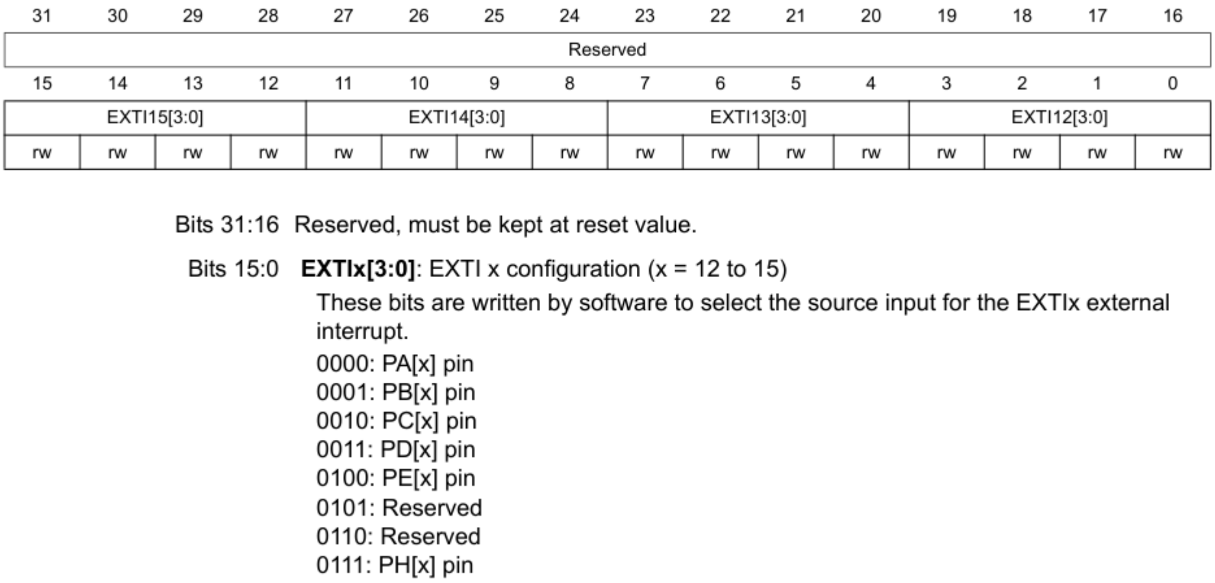
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SYSCFG EN	SPI4EN	SPI1 EN	SDIO EN	Reserved		ADC1 EN	Reserved		USART6 EN	USART1 EN	Reserved			TIM1 EN
	rw	rw	rw	rw			rw			rw	rw				rw

3. EXTI Initialization & Connect Push-button to EXTI line

- **SYSCFG_EXTICR4:** Connect PC_13(push-button) to EXTI13 line

```
SYSCFG->EXTICR[3] &=~ 15≪4                    // clear bits [3:0]
SYSCFG->EXTICR[3] |= 2<<4    // set to 0010 for PC[13]
```

// past register map here



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 12 to 15)

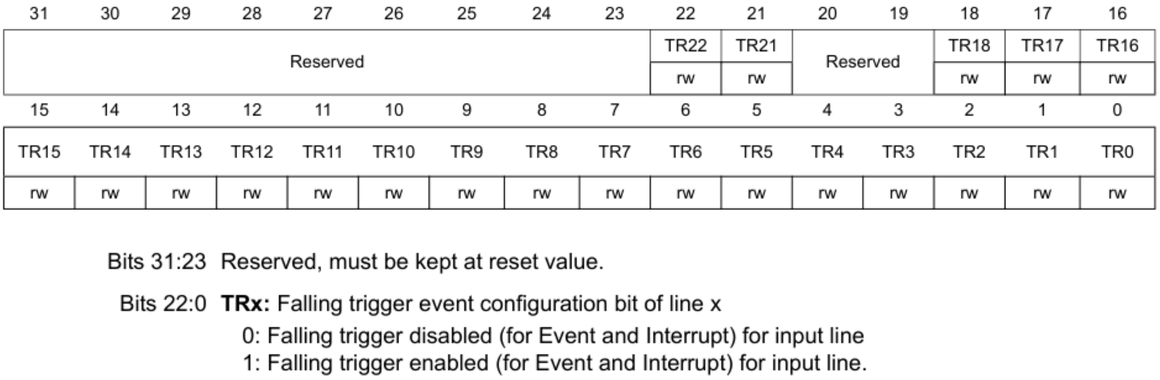
These bits are written by software to select the source input for the EXTIx external interrupt.

0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: Reserved
0110: Reserved
0111: PH[x] pin

- **EXTI_FTSR:** Enable Falling Trigger

```
EXTI->FTSR |= 1<<13       // TR13=1
```

// past register map here



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	Reserved		TR18	TR17	TR16
									rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

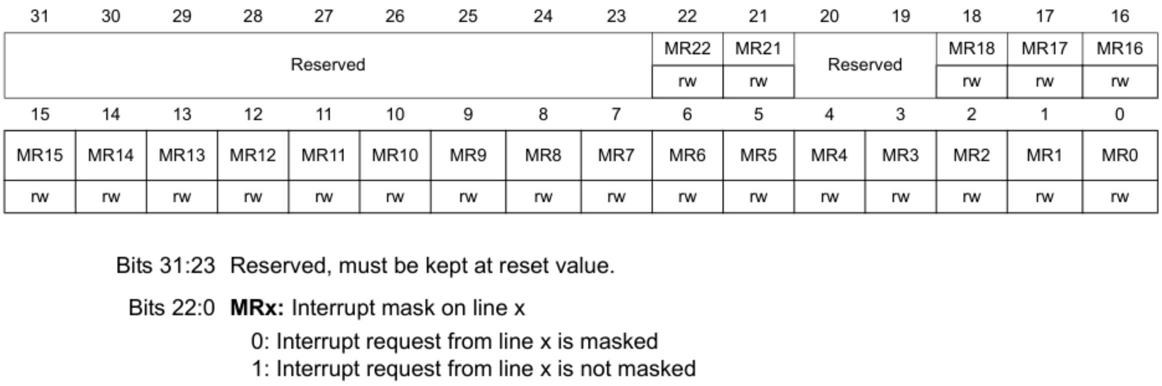
Bits 22:0 **TRx:** Falling trigger event configuration bit of line x

0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

- **EXTI_IMR:** Interrupt NOT masked (Enable)

EXTI->IMR |= 1<<13 // MR13 = 1

// past register map here



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	Reserved		MR18	MR17	MR16
									rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **MRx:** Interrupt mask on line x

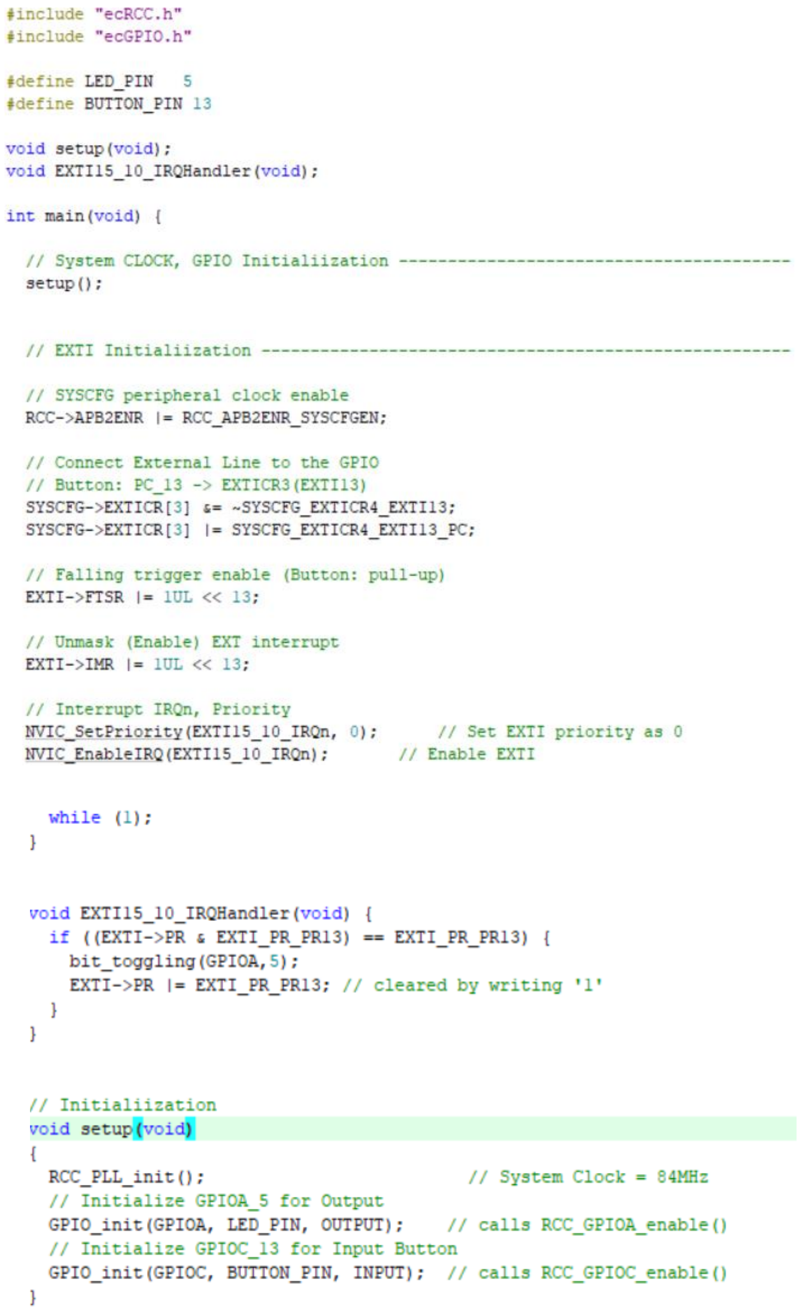
0: Interrupt request from line x is masked

1: Interrupt request from line x is not masked

B. Programming

Procedure

- Create a new folder ‘**EC/Tutorial/TU_EXTI/**’
- Open the program ‘Keil uVision5’ and create a new project.
- Name the project as ‘**TU_EXTI**’.
- Create a new item called ‘TU_EXTI.c’ and use the given source code Click here to download
- This is an example code for turning LED on/off with the button input trigger.
- Fill in the empty spaces in the code.
- Run the program and check your result.
- Your tutorial report must be submitted to LMS



```
#include "ecRCC.h"
#include "ecGPIO.h"

#define LED_PIN    5
#define BUTTON_PIN 13

void setup(void);
void EXTI15_10_IRQHandler(void);

int main(void) {

  // System CLOCK, GPIO Initialiization ----------------------------------------
  setup();


  // EXTI Initialiization ------------------------------------------------------

  // SYSCFG peripheral clock enable
  RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;

  // Connect External Line to the GPIO
  // Button: PC_13 -> EXTICR3(EXTI13)
  SYSCFG->EXTICR[3] &= ~SYSCFG_EXTICR4_EXTI13;
  SYSCFG->EXTICR[3] |= SYSCFG_EXTICR4_EXTI13_PC;

  // Falling trigger enable (Button: pull-up)
  EXTI->FTSR |= 1UL << 13;

  // Unmask (Enable) EXT interrupt
  EXTI->IMR |= 1UL << 13;

  // Interrupt IRQn, Priority
  NVIC_SetPriority(EXTI15_10_IRQn, 0);      // Set EXTI priority as 0
  NVIC_EnableIRQ(EXTI15_10_IRQn);       // Enable EXTI


    while (1);
  }


  void EXTI15_10_IRQHandler(void) {
    if ((EXTI->PR & EXTI_PR_PR13) == EXTI_PR_PR13) {
      bit_toggling(GPIOA,5);
      EXTI->PR |= EXTI_PR_PR13; // cleared by writing '1'
    }
  }


  // Initialiization
  void setup(void)
  {
    RCC_PLL_init();                         // System Clock = 84MHz
    // Initialize GPIOA_5 for Output
    GPIO_init(GPIOA, LED_PIN, OUTPUT);    // calls RCC_GPIOA_enable()
    // Initialize GPIOC_13 for Input Button
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);  // calls RCC_GPIOC_enable()
  }
```

Appendix

See here for MCU resources

1. Pin Configuration of NUCLE-F401RE

Figure 18. NUCLEO-F401RE

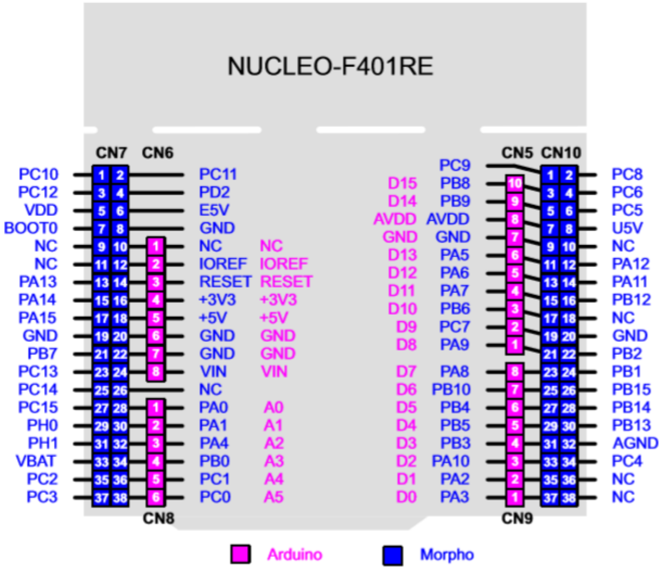
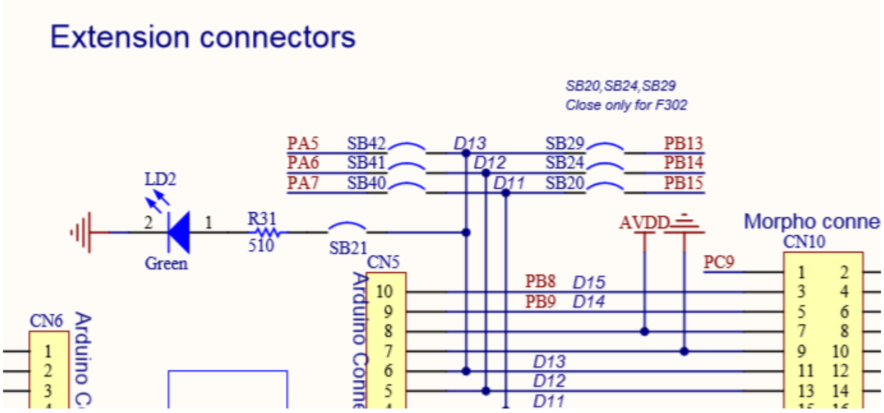

NUCLEO-F401RE
CN7
CN6
CN5
CN10
CN8
CN9
PC10 PC12 VDD BOOT0 NC NC PA13 PA14 PA15 GND PB7 PC13 PC14 PC15 PH0 PH1 VBAT PC2 PC3
PC11 PD2 E5V GND NC IOREF RESET +3V3 +5V GND GND VIN NC PA0 PA1 PA4 PB0 PC1 PC0
NC IOREF RESET +3V3 +5V GND GND VIN A0 A1 A2 A3 A4 A5
PC9
D15 PB8
D14 PB9
AVDD AVDD
GND GND
D13 PA5
D12 PA6
D11 PA7
D10 PB6
D9 PC7
D8 PA9
D7 PA8
D6 PB10
D5 PB4
D4 PB5
D3 PB3
D2 PA10
D1 PA2
D0 PA3
PC8 PC6 PC5 U5V NC PA12 PA11 PB12 NC GND PB2 PB1 PB15 PB14 PB13 AGND PC4 NC NC
Arduino
Morpho

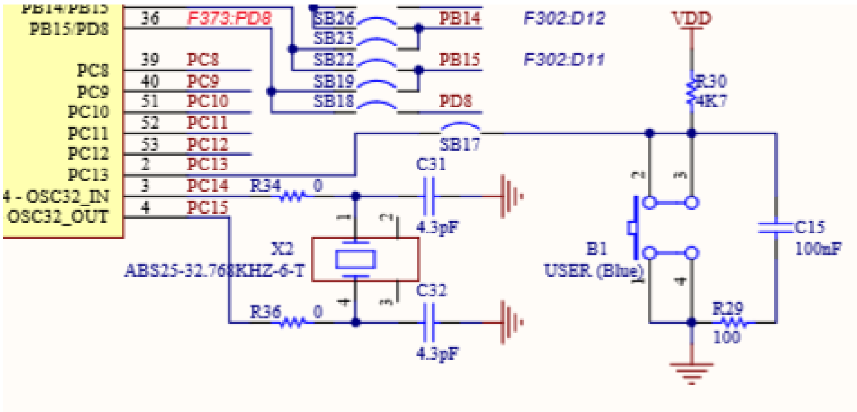
Table 29. ST morpho connector on NUCLEO-F401RE, NUCLEO-F411RE, NUCLEO-F446RE

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0[1]	GND	8	7	AVDD	U5V[2]	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13[3]	RESET	14	13	PA6	PA11	14
15	PA14[3]	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	-	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9[4]	36	35	PA2	-	36
37	PC3	PC0 or PB8[4]	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

2. LED/Button Circuit Diagram


Extension connectors
SB20,SB24,SB29
Close only for F302
PA5
PA6
PA7
SB42
SB41
SB40
D13
D12
D11
SB29
SB24
SB20
PB13
PB14
PB15
LD2
2
1
Green
R31
510
SB21
AVDD
Morpho conne
CN10
PC9
CN5
Arduino Conne
10
9
8
7
6
5
PB8
PB9
D15
D14
D13
D12
D11
CN6
Arduino C
1
2
3
1
2
3
4
5
6
7
8
9
10
11
12
13
14


PB14/PB13
PB15/PD8
PC8
PC9
PC10
PC11
PC12
PC13
4 - OSC32_IN
OSC32_OUT
36
39
40
51
52
53
2
3
4
F373:PD8
PC8
PC9
PC10
PC11
PC12
PC13
PC14
PC15
SB26
SB23
SB22
SB19
SB18
PB14
PB15
PD8
F302:D12
F302:D11
SB17
R34
0
R36
0
C31
4.3pF
C32
4.3pF
X2
ABS25-32.768KHZ-6-T
VDD
R30
4K7
B1
USER (Blue)
C15
100nF
R29
100