GPIO INTERRUPT VIA REGISTERS

Handong university

Jong-won Lee

2022.03.27

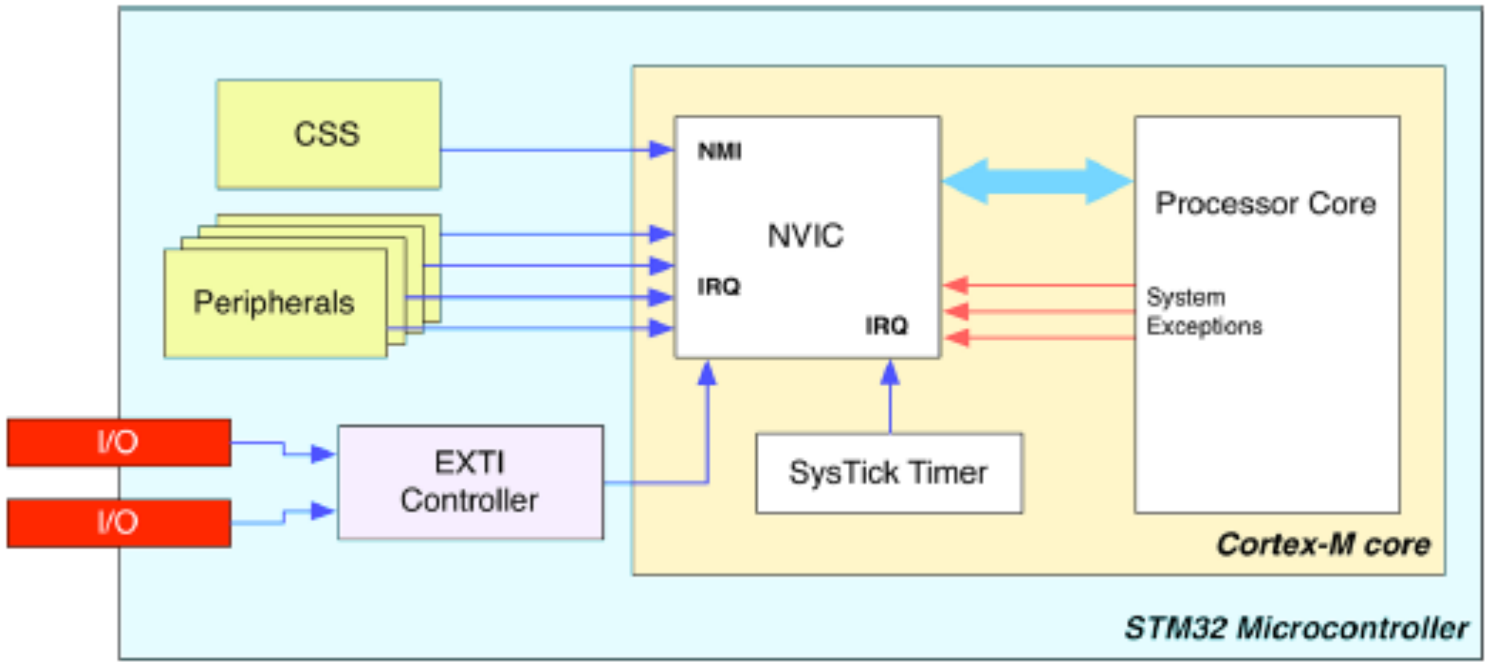
References

- RM0383 Reference manual, STM32F411xC/E advanced Arm®-based 32-bit MCUs, Sep. 2017.
- STM32F411xC STM32F411xE Datasheet, Dec. 2017.

GPIO Interrupt

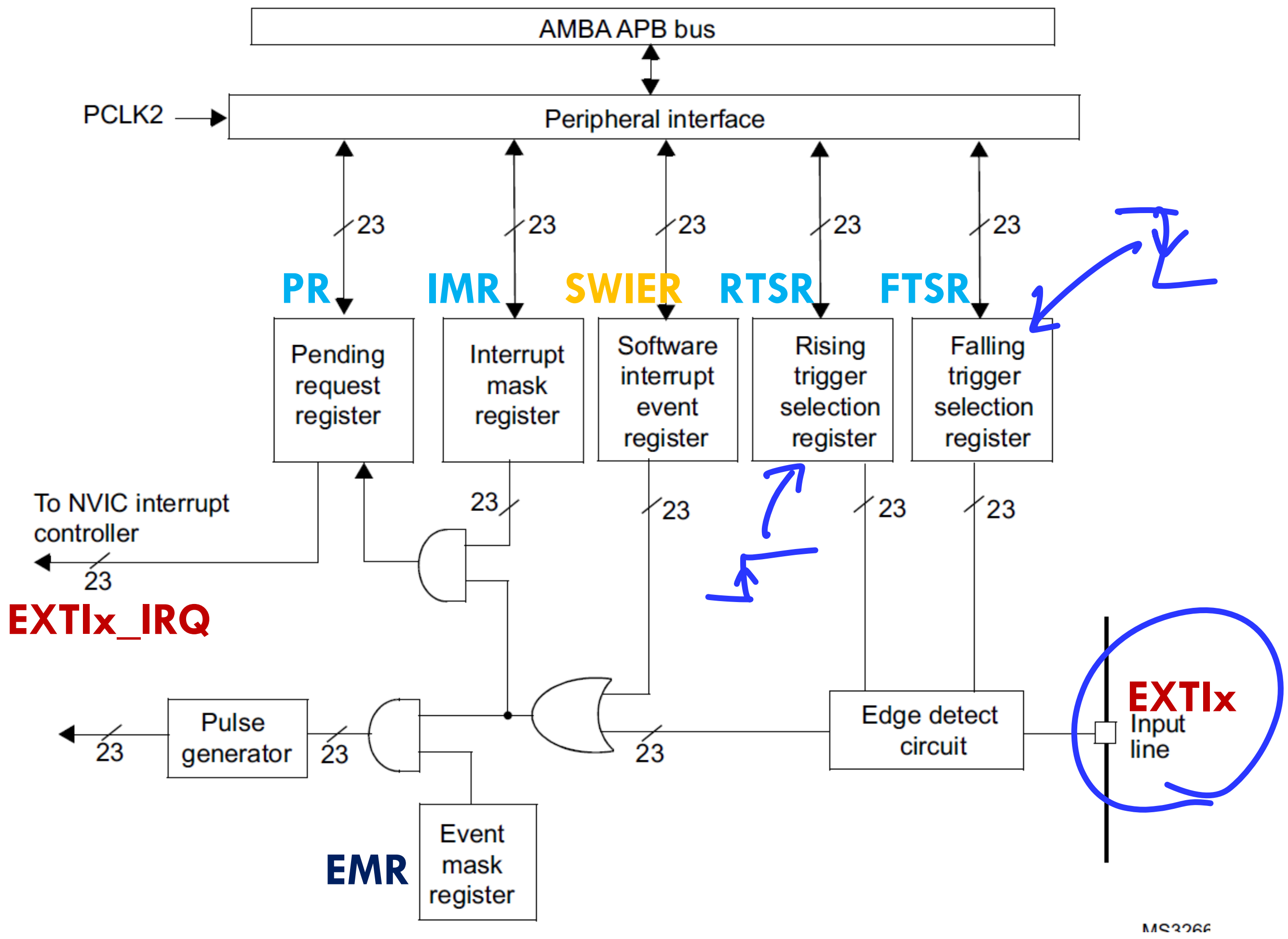
EXTI Controller

- External interrupt/event controller (EXTI)

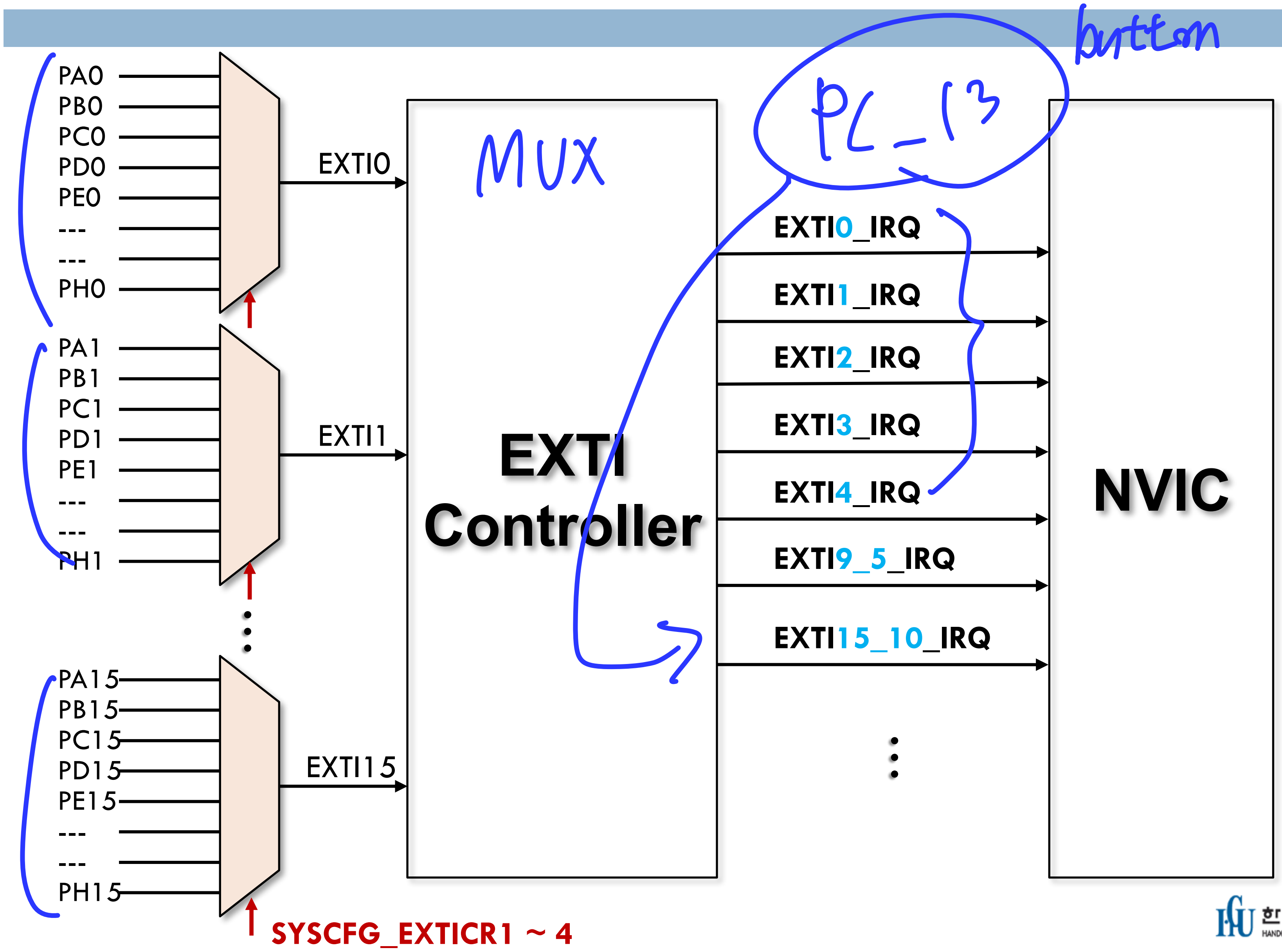


EXTI Controller

- EXTI block diagram



From GPIO Input Signal to NVIC



SYSCFG_EXTICR1/2

configuration

- SYSCFG external interrupt configuration register 1,2 (SYSCFG_EXTICR1/2)

1

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
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			

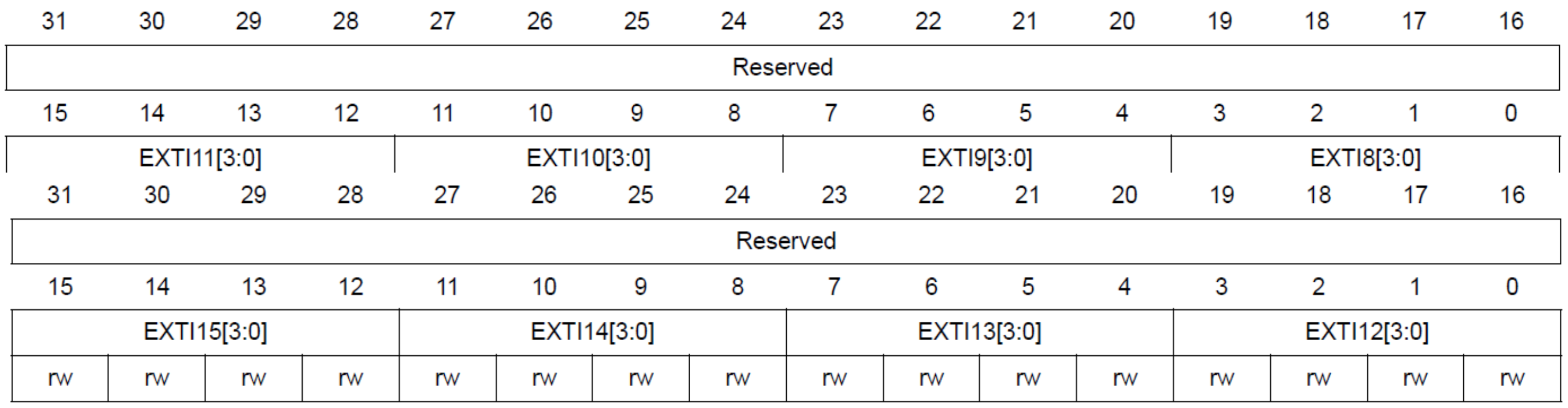
2

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
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3) (x = 4 to 7)
  - 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

SYSCFG_EXTICR3/ 4

- SYSCFG external interrupt configuration register 3,4 (SYSCFG_EXTICR3/4)



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
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			

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 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 8 to 11)(x = 12 to 15)
  - 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

SYSCFG_EXTICR1 ~ 4

- In stm32f411xe.h

```
typedef struct
{
  __IO uint32_t MEMRMP;       /*!< SYSCFG memory remap register,                      Address offset: 0x00      */
  __IO uint32_t PMC;          /*!< SYSCFG peripheral mode configuration register,     Address offset: 0x04      */
  __IO uint32_t EXTICR[4];    /*!< SYSCFG external interrupt configuration registers, Address offset: 0x08-0x14 */
  uint32_t      RESERVED[2];  /*!< Reserved, 0x18-0x1C                                                          */
  __IO uint32_t CMPCR;        /*!< SYSCFG Compensation cell control register,         Address offset: 0x20      */
} SYSCFG_TypeDef;
```

```
#define SYSCFG              ((SYSCFG_TypeDef *) SYSCFG_BASE)
```

  - SYSCFG->EXTICR[0] ~ SYSCFG->EXTICR[3]

EXTI Registers

Mask ?

interrupt

- Interrupt mask register (EXTI_IMR)

interrupt enable or disable

  - Reset value: 0x0000_0000

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 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
  - Cf.
    - EXTI line 16 is connected to the PVD output
    - EXTI line 17 is connected to the RTC Alarm event
    - EXTI line 18 is connected to the USB OTG FS Wakeup event
    - EXTI line 21 is connected to the RTC Tamper and TimeStamp events
    - EXTI line 22 is connected to the RTC Wakeup event

EXTI Registers

event (딱히 상관 없음)

- Event mask register (EXTI_EMR)
  - Reset value: 0x0000_0000

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 22:0 MRx: Event mask on line x
  - 0: Event request from line x is masked
  - 1: Event request from line x is not masked

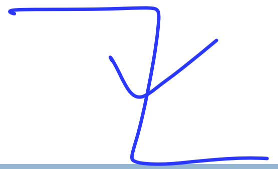
EXTI Registers

- Rising trigger selection register (EXTI_RTSR)
  - Reset value: 0x0000_0000

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 22:0 TRx: Rising trigger event configuration bit of line x
  - 0: Rising trigger disabled (for Event and Interrupt) for input line
  - 1: Rising trigger enabled (for Event and Interrupt) for input line

EXTI Registers



원하는 기거 ?

- Falling trigger selection register (EXTI_FTSR)
  - Reset value: 0x0000_0000

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

1

- 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

$<<13$

EXTI Registers

not H/W

- Software interrupt event register (EXTI_SWIER)
  - Reset value: 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									SWIER 22	SWIER 21	Reserved		SWIER 18	SWIER 17	SWIER 16
									rw	rw			rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- Bits 22:0 SWIERx: Software Interrupt on line x
  - If interrupt are enabled on line x in the EXTI_IMR register, writing '1' to SWIERx bit when it is set at '0' sets the corresponding pending bit in the EXTI_PR register, thus resulting in an interrupt request generation.
  - This bit is **cleared by writing 1 to the corresponding bit in EXTI_PR**.

clear

여기서 안해도 됨.

EXTI Registers

pending (clear)

Pending register (EXTI_PR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PR22	PR21	Reserved		PR18	PR17	PR16
									rc_w1	rc_w1			rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

- Bits 22:0 PRx: Pending bit
  - 0: No trigger request occurred
  - 1: selected trigger request occurred
- This bit is set when the selected edge event arrives on the external interrupt line.
- This bit is **cleared by programming it to '1'.**

EXTI Registers

- In stm32f422xe.h

```
typedef struct
{
  __IO uint32_t IMR;    /*!< EXTI Interrupt mask register,            Address offset: 0x00 */
  __IO uint32_t EMR;    /*!< EXTI Event mask register,                Address offset: 0x04 */
  __IO uint32_t RTSR;   /*!< EXTI Rising trigger selection register,  Address offset: 0x08 */
  __IO uint32_t FTSR;   /*!< EXTI Falling trigger selection register, Address offset: 0x0C */
  __IO uint32_t SWIER;  /*!< EXTI Software interrupt event register,  Address offset: 0x10 */
  __IO uint32_t PR;     /*!< EXTI Pending register,                   Address offset: 0x14 */
} EXTI_TypeDef;
```

```
#define EXTI              ((EXTI_TypeDef *) EXTI_BASE)
```

  - EXTI->IMR

Interrupt Number in STM32F411re

```
typedef enum
{
  …
  RCC_IRQn                = 5,    /*!< RCC global Interrupt                  */
  EXTI0_IRQn              = 6,    /*!< EXTI Line0 Interrupt                  */
  EXTI1_IRQn              = 7,    /*!< EXTI Line1 Interrupt                  */
  EXTI2_IRQn              = 8,    /*!< EXTI Line2 Interrupt                  */
  EXTI3_IRQn              = 9,    /*!< EXTI Line3 Interrupt                  */
  EXTI4_IRQn              = 10,   /*!< EXTI Line4 Interrupt                  */
  DMA1_Stream0_IRQn       = 11,   /*!< DMA1 Stream 0 global Interrupt        */
  DMA1_Stream1_IRQn       = 12,   /*!< DMA1 Stream 1 global Interrupt        */
  …
  DMA1_Stream6_IRQn       = 17,   /*!< DMA1 Stream 6 global Interrupt        */
  ADC_IRQn                = 18,   /*!< ADC1, ADC2 and ADC3 global Interrupts */
  EXTI9_5_IRQn            = 23,   /*!< External Line[9:5] Interrupts         */
  TIM1_BRK_TIM9_IRQn      = 24,   /*!< TIM1 Break interrupt and TIM9 global interrupt */
  …
  USART2_IRQn             = 38,   /*!< USART2 global Interrupt               */
  EXTI15_10_IRQn          = 40,   /*!< External Line[15:10] Interrupts       */
  RTC_Alarm_IRQn          = 41,   /*!< RTC Alarm (A and B) through EXTI Line Interrupt */
  …
} IRQn_Type;
```

NVIC

NVIC Registers in STM32F411xx

- NVIC registers to control interrupts from peripherals
  - In core_cm4.h

```
typedef struct
{
  __IOM uint32_t ISER[8U];             /*!< Offset: 0x000 (R/W)  Interrupt Set Enable Register */
        uint32_t RESERVED0[24U];
  __IOM uint32_t ICER[8U];             /*!< Offset: 0x080 (R/W)  Interrupt Clear Enable Register */
        uint32_t RESERVED1[24U];
  __IOM uint32_t ISPR[8U];             /*!< Offset: 0x100 (R/W)  Interrupt Set Pending Register */
        uint32_t RESERVED2[24U];
  __IOM uint32_t ICPR[8U];             /*!< Offset: 0x180 (R/W)  Interrupt Clear Pending Register */
        uint32_t RESERVED3[24U];
  __IOM uint32_t IABR[8U];             /*!< Offset: 0x200 (R/W)  Interrupt Active bit Register */
        uint32_t RESERVED4[56U];
  __IOM uint8_t  IP[240U];             /*!< Offset: 0x300 (R/W)  Interrupt Priority Register (8Bit wide) */
        uint32_t RESERVED5[644U];
  __OM  uint32_t STIR;                 /*!< Offset: 0xE00 ( /W)  Software Trigger Interrupt Register */
} NVIC_Type;
```

```
#define SCS_BASE        (0xE000E000UL)                       /*!< System Control Space Base Address */
#define NVIC_BASE       (SCS_BASE +  0x0100UL)               /*!< NVIC Base Address */
#define NVIC            ((NVIC_Type      *)     NVIC_BASE     )   /*!< NVIC configuration struct */
```

NVIC Registers in STM32F411xx

- NVIC registers to control interrupts from peripherals
  - not system exceptions

Address	Name	CMSIS-Core symbol	Property	Function
0xE000E100 ~ 0xE000E10B	NVIC_ISER0 ~ NVIC_ISER2	NVIC->ISER[0] ~ NVIC->ISER[2]	RW (w1s)	Interrupt Set-Enable Register
0xE000E180 ~ 0xE000E18B	NVIC_ICER0 ~ NVIC_ICER2	NVIC->ICER[0] ~ NVIC->ICER[2]	RW (w1c)	Interrupt Clear-Enable Register
0xE000E200 ~ 0xE000E20B	NVIC_ISPR0 ~ NVIC_ISPR2	NVIC->ISPR[0] ~ NVIC->ISPR[2]	RW (w1s)	Interrupt Set-Pending Register
0xE000E280 ~ 0xE000E28B	NVIC_ICPR0 ~ NVIC_ICPR2	NVIC->ICPR[0] ~ NVIC->ICPR[2]	RW (w1c)	Interrupt Clear-Pending Register
0xE000E300 ~ 0xE000E30B	NVIC_IABR0 ~ NVIC_IABR2	NVIC->IABR[0] ~ NVIC->IABR[2]	RO	Interrupt Active Bit Register
0xE000E400 ~ 0xE000E453	NVIC_IPR0 ~ NVIC_IPR20	NVIC->IP[0] ~ NVIC->IP[20]	RW	Interrupt Priority Register
0xE000EF00	NVIC_STIR	NVIC->STIR	WO	Software Trigger Interrupt Register

GPIO INTERRUPT VIA REGISTERS

Handong university

Jong-won Lee

2020.04.02

Lab. 3-1

- Control the blinking rate of the LED 2 by the User button.
  - Whenever the user button is pressed, change the blinking rate as follows: 500 ms on/off → 1 sec on/off → 2 sec on/off → 500 ms on/off → …
- Use interrupt to detect press the user button
- Implement the following functions by directly handling registers.

```
void led2_init(void);
void led2_toggle(void);

void button_init(void);

void button_Handler(void);      // ISR of EXTI15_10_IRQn
```

Lab. 3-1

- A skeleton code.

```
#include "mbed.h"

volatile int interval;

void led2_init(void);
void button_init(void);
void led2_toggle(void);

void button_Handler(void);
```

```

// main() runs in its own thread in the OS
int main() {

    led2_init();
    button_init();

    NVIC_SetVector(EXTI15_10_IRQn, (uint32_t)button_Handler);

    interval = 500;

    while (true) {

        led2_toggle();

        ThisThread::sleep_for((std::chrono::milliseconds)(interval));
    }
}
```