INTERRUPT LAB (ASSEMBLY)

ECE30070 Microprocessor Application

HGU

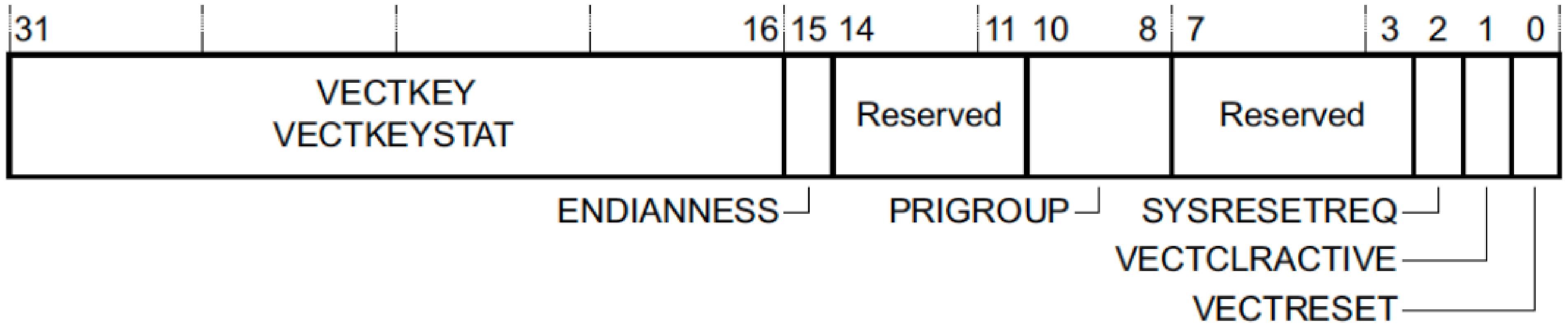
Registers used to configure Cortex-M4 Exceptions

- Application Interrupt and Reset Control Register (AIRCR)
- Nested Vectored Interrupt Controller (NVIC) Registers
  - ISER*n*, ICER*n*, ISPR*n*, ICPR*n*, IABR*n* *(n= 0 or 1)*
  - ICTR
  - IPR*n* *(n= 0 .. 15*)
- GPIO Task and Event (GPIOTE) Registers
  - CONFIG[n] (n=0..7)
  - EVENTS_IN[n] (n=0..7)
  - INTINCLR
  - INTINSET

AIRCR (Application Interrupt & Reset Control Register)

- Base address **0xE000ED0C**
- In order to Write the register VECTKEY should be set to 0x05FA
- SYSRESETREQ : Writing 1 invokes system Soft Reset
- **PRIGROUP** - This field lets you split exception priorities into two parts known as the ***group priority*** and ***subpriority***

AIRCR bit assignments:



Registers of NVIC

- NVIC base address = 0xE000E000
- NVIC_ISER*n* (Interrupt Set Enable): 0xE000E100 + 4*n
- NVIC_ICER*n* (Interrupt Clear-Enable): 0xE000E180 + 4*n
- NVIC_ISPR*n* (Interrupt Set-Pending): 0xE000E200 + 4*n
- NVIC_ICPR*n* (Interrupt Clear-Pending): 0xE000E280 + 4*n
- NVIC_IABR*n* (Interrupt Active Bit): 0xE000E300 + 4*n

Each Register has a bit corresponding to 32 interrupts.

n= 0,1 for nRF52840,
since nRF52840 has 64 external interrupts

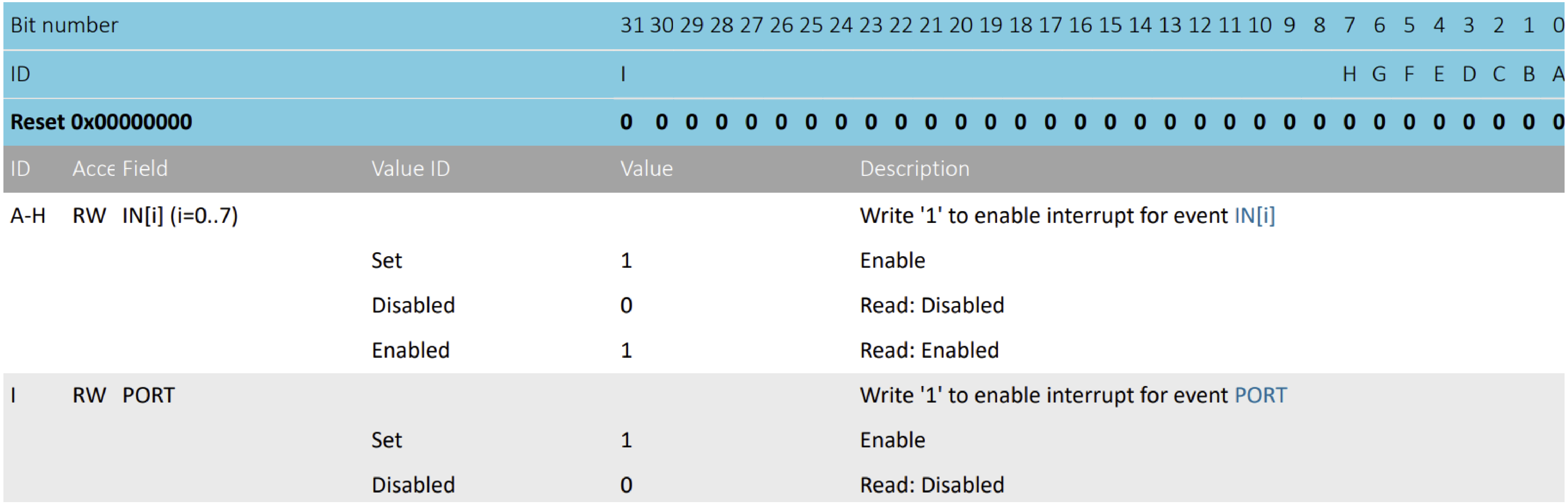
- NVIC_IPR*n* (Interrupt Priority): 0xE000E400 + 4*n (***n=0~15*** for nRF52840 *since 64 interrupts*)
  - Each register has four 8-bit fields, each corresponding to one interrupt
  - interrupt priority ranges 0 ~ 7 (8 levels)
- ICTR (Interrupt Controller Type Register) : E000E004
  - INTLINESNUM (3:0) : number of interrupts supported = (INTLINESNUM + 1) * 32
  - Read-only

Registers of GPIOTE

- GPIOTE module provides functionality for accessing GPIO pins using Tasks and Events. Each GPIOTE channel can be assigned to one pin.
- Number of GPIOTE Channels : 8
- GPIOTE base address = 0x40006000
- INTINCLR Register (0ffset=0x304): enable interrupt for each channel
- INTINSET (offset-0x308): disable interrupt for each channel
- CONFIG*[n] (offset = 0x510+4*n, n = 0~7) : configure for channel*
- EVENTS_IN*[n] (n=0~7)*

GPIOTE Register: INTENSET

- Offset= 0x304
- Writing ‘1’ to **Enable interrupt** for IN[n] (n=0..7) event (A..H) and PORT (I) event



Bit number	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
ID	I																								H	G	F	E	D	C	B	A
Reset 0x00000000	**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**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-H	RW	IN[i] (i=0..7)			Write '1' to enable interrupt for event IN[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
I	RW	PORT			Write '1' to enable interrupt for event PORT
			Set	1	Enable
			Disabled	0	Read: Disabled

GPIOTE Register: INTENCLR

- Offset=0x308
- Writing ‘1’ **Disable Interrupt** for IN[n] (n=0.. 7) event or PORT event

Bit number	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
ID	I																								H	G	F	E	D	C	B	A
Reset 0x00000000	**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**	**0**	**0**	**0**	**0**	**0**	**0**

ID	RW	Field	Value ID	Value	Description
A	RW	IN0			Write '1' to disable interrupt for IN[0] event See EVENTS_IN[0]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

GPIOTE Register: CONFIG[n] (n=0..7)

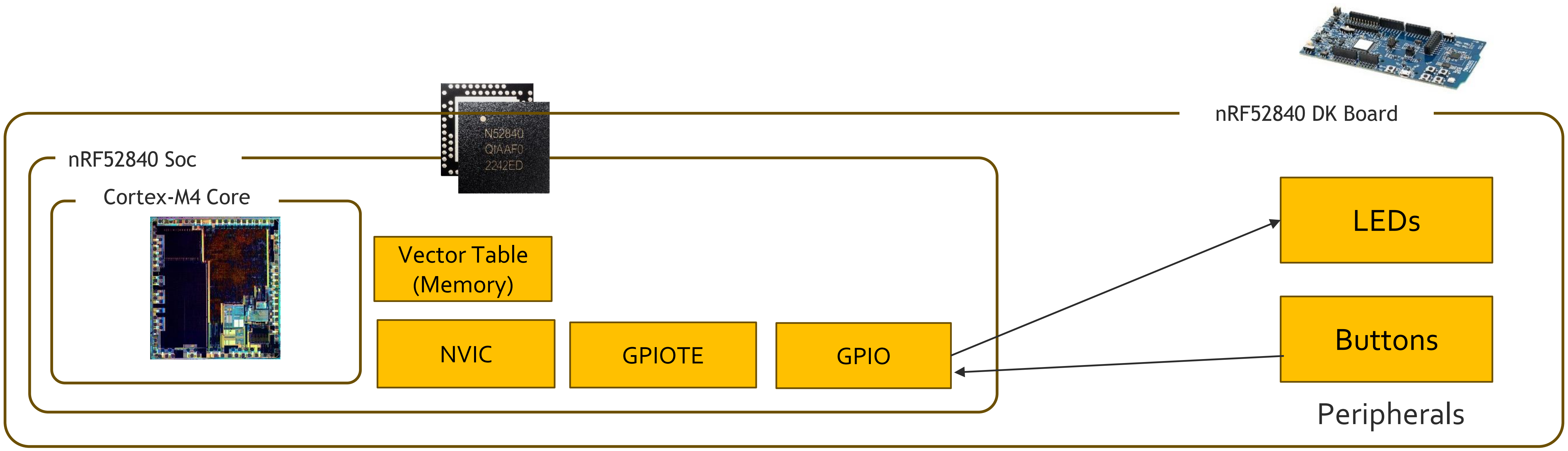
- Offset=0x510 + (n x 4)
- Configuration for OUT[n], SET[n],and CLR[n] Task and IN[n] event

Bit number	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
ID												E			D	D			C	B	B	B	B	B							A	A
Reset 0x00000000	**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**	**0**	**0**	**0**	**0**	**0**	**0**

Filed	Value	Description
AA	o 1 3	**Mode** : Disabled **<u>Event mode</u>:** the pin specified by **PSEL** is **input** and IN[n] event is generated if POLARITY occurs Task mode: the pin specified by **PSEL** is **output** and triggering the SET[n], CLR[n],OUT[n] task will perform the operation specified by POLARITY on the pin. The pin acquired by GPIOTE module task cannot be a regular output.
BBBBB	0...31	**PSEL:** GPIO number associated with SET[n], CLR[n],and OUT[n] tasks and IN[n] event
C	0,1	**PORT:** port number
DD	0,..,3	**POLARITY**: None(0), LoTOHi (1), HiToLo (2), Toggle(3) (OUT[n] task pin값 변경 or IN[n] event 발생)
E	0, 1	OUTINIT: in task mode, initial value of pin (0 or 1), in event mode: No effect

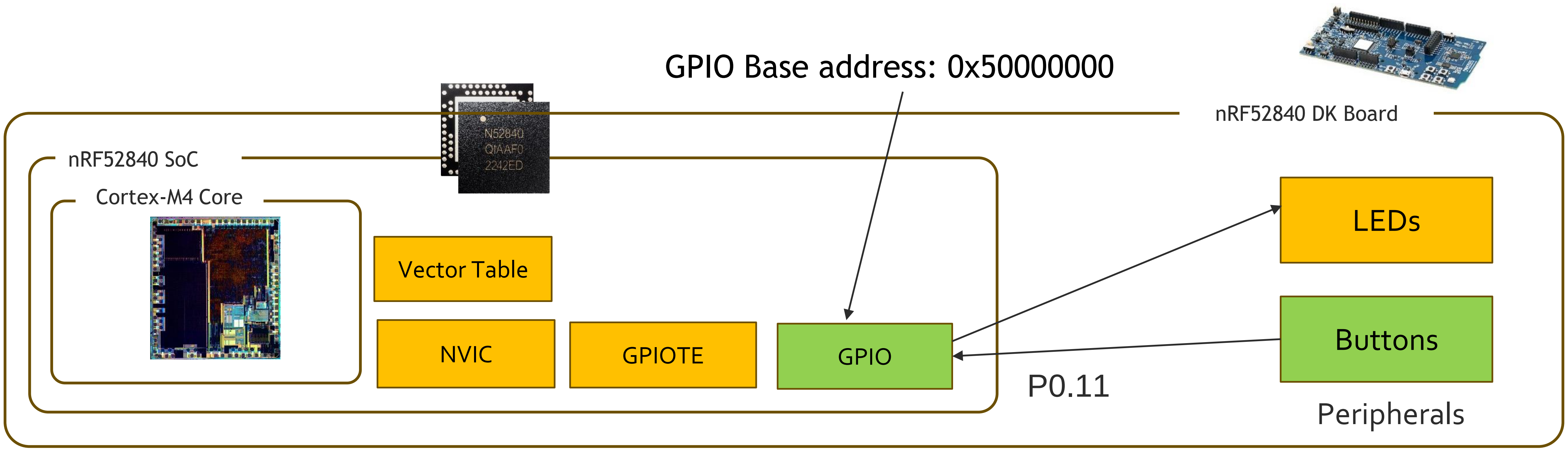
Example Code

- **nRF52840 DK Button1 Interrupt**
  - Toggle LED-1 by the Button-1 Push Event
  - By invoking interrupt service routine



Example Code

- **nRF52840 DK Button1 Interrupt**



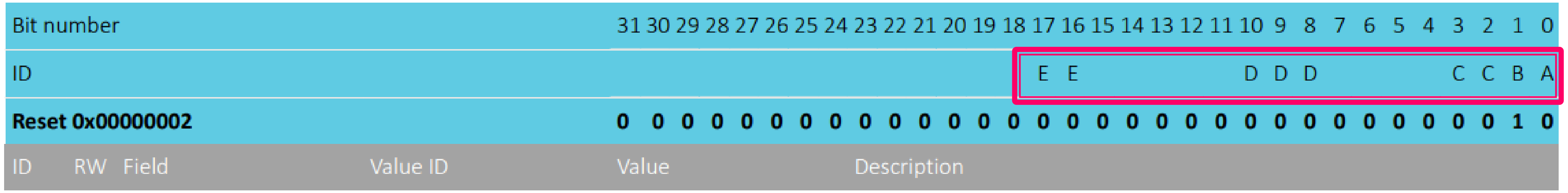
Example Code

- **nRF52840 DK Button1 Interrupt**
  - (1) Make Button1 ready for Input

```
.equ BTN1_MASK, 0x00800
.equ GPIO_P0_BASE, 0x50000000
.equ GPIO_PIN_CNF_11_OFFSET,    0x72C   // P0.11 pin configuration address offset (BTN1)
```

```
 // Button1 setting
LDR R1, =0x0003000c    // Sense Low, Pull-Up, Set as Connect Input
STR R1, [R0, #GPIO_PIN_CNF_11_OFFSET]
```

PIN_CONFIG[n]
(n=0~31)



Bit number	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
ID															E	E						D	D	D					C	C	B	A
Reset 0x00000002	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	0	0	0	0	1	0

ID	RW	Field	Value ID	Value	Description

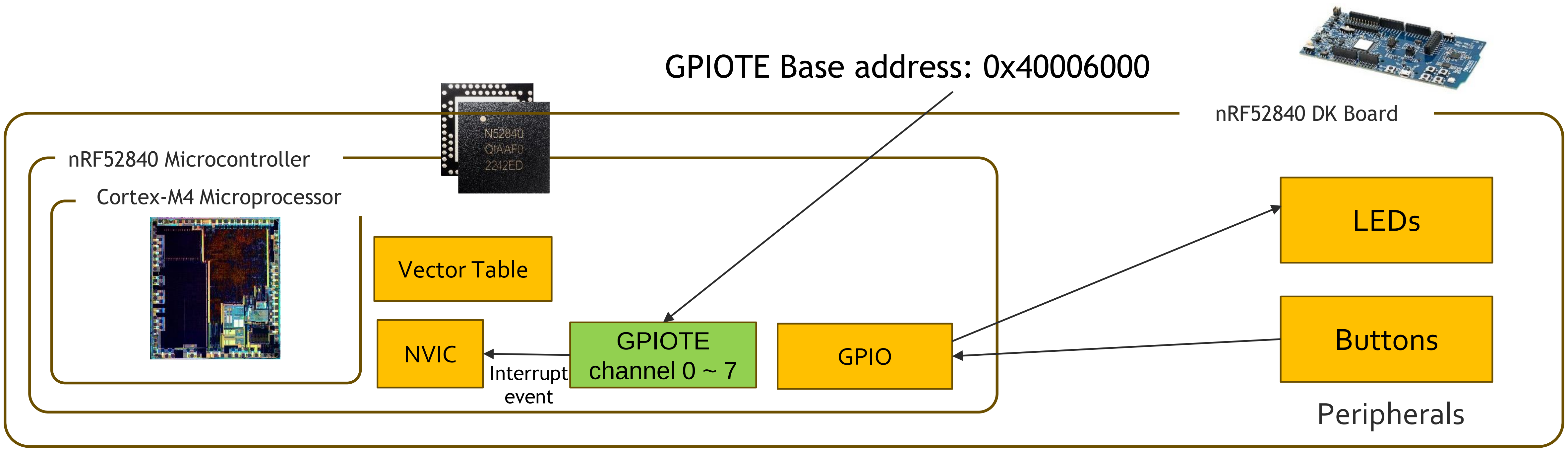
- GPIO PIN Configuration register
- Addr Offset = 0x700 + n * 4
- n = 11 for BTN1 pin (P0.11)

A=0 (input)
B=0 (Connect Input Beffer))
CC = 3 (Pull-Up)

DDD = 000 (Standard Drive Strength)
EE = 11 (Sense Low)

Example Code

- **nRF52840 DK Button1 Interrupt**



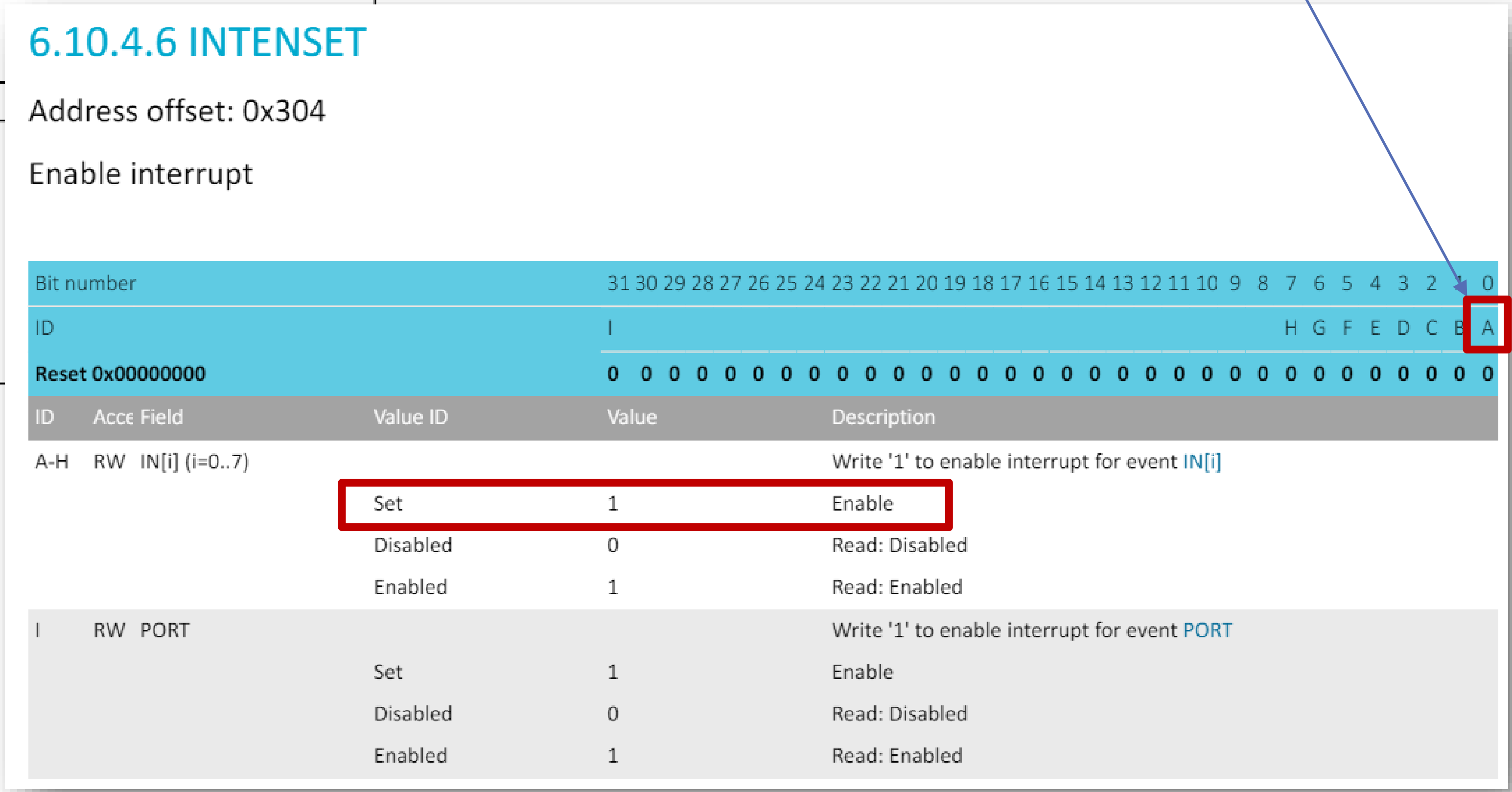
Example Code (GPIO.INTENSET Register)

- **nRF52840 DK Button1 Interrupt**
  - (2) Set GPIOTE for enabling GPIO interrupt event generation
    - ✓Let's use **Channel 0.**

```
.equ GPIOTE_BASE, 0x40006000
.equ GPIOTE_INTENSET_OFFSET, 0x304
```

```
 // GPIOTE setting
 LDR R0, =GPIOTE_BASE
 MOV R1, #0x01 // channel = 0
 STR R1, [R0, #GPIOTE_INTENSET_OFFSET]
```

for channel 0



6.10.4.6 INTENSET

Address offset: 0x304

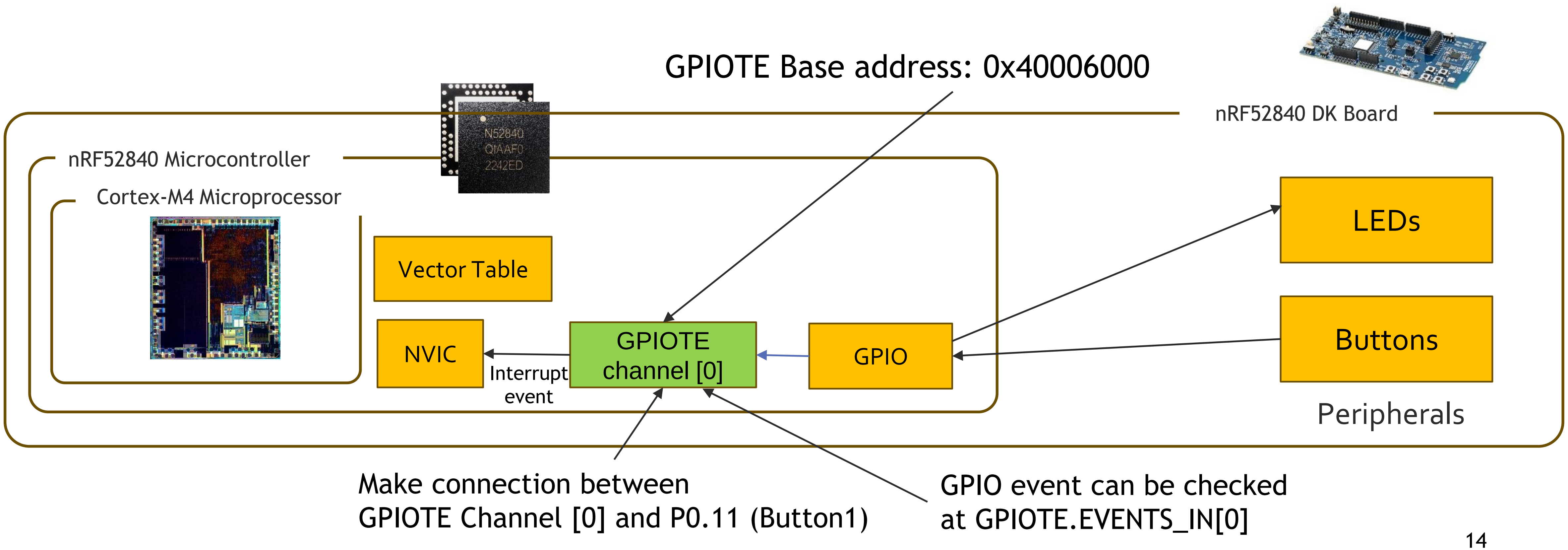
Enable interrupt

Bit number	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
ID	I																								H	G	F	E	D	C	B	A
Reset 0x00000000	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	0	0	0	0	0	0

ID	Acce	Field	Value ID	Value	Description
A-H	RW	IN[i] (i=0..7)			Write '1' to enable interrupt for event IN[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
I	RW	PORT			Write '1' to enable interrupt for event PORT
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

Example Code

- **nRF52840 DK Button1 Interrupt**



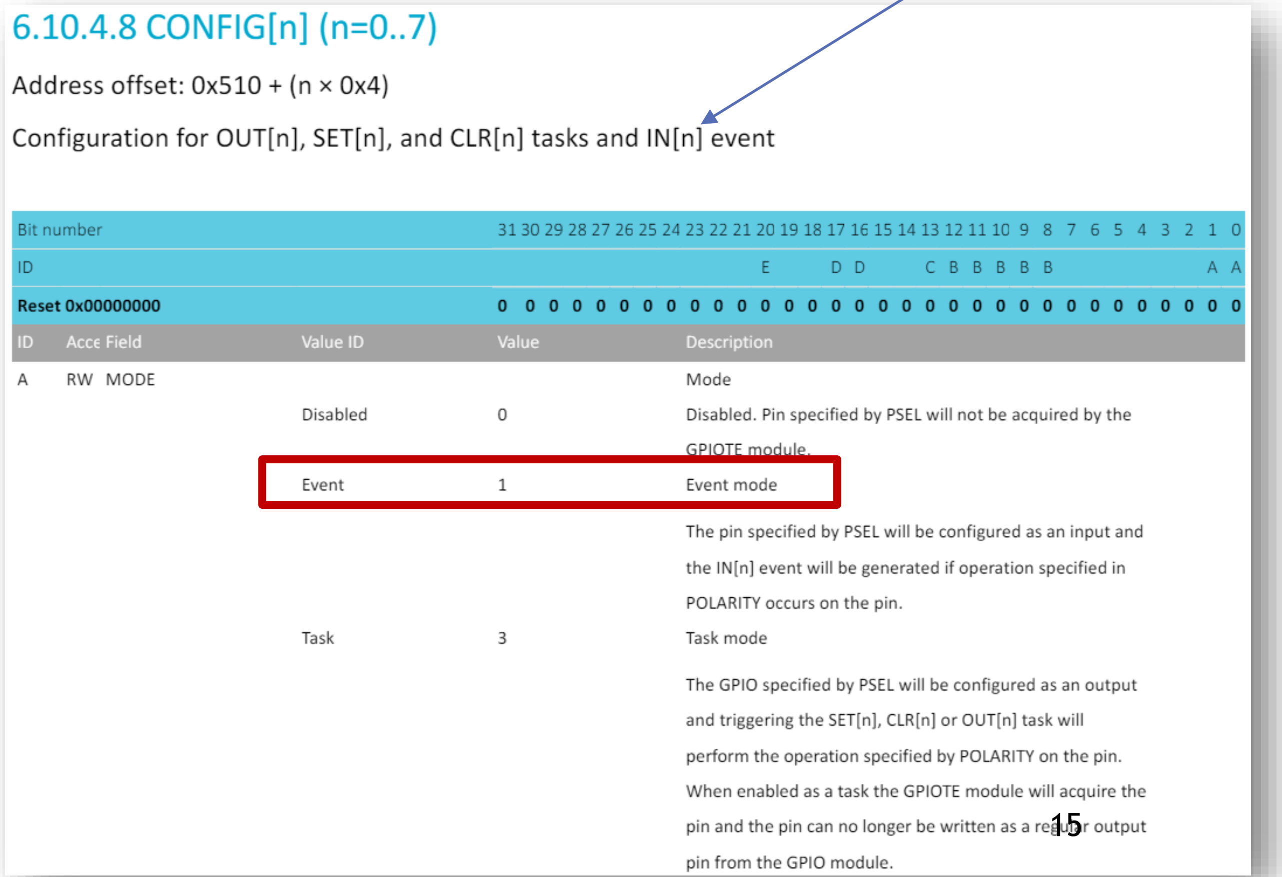
Example Code (GPIOTE.CONFIG[0] Register)

- **nRF52840 DK Button1 Interrupt**
  - (3) Set GPIOTE channel 0's configuration.
    - ✓GPIOTE.CONFIG[0]

```
.equ GPIOTE_BASE, 0x40006000
.equ GPIOTE_CONFIG0_OFFSET, 0x510 // GPIOTE channel 0
```

```
 // GPIOTE setting
 LDR R0, =GPIOTE_BASE
 MOV R1, #0x01 // mode = event (A)
 …
```

for channel 0 $(n = 0)$



6.10.4.8 CONFIG[n] (n=0..7)

Address offset: 0x510 + (n × 0x4)

Configuration for OUT[n], SET[n], and CLR[n] tasks and IN[n] event

Bit number	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
ID												E			D	D			C	B	B	B	B	B							A	A
Reset 0x00000000	**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**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	MODE			Mode
			Disabled	0	Disabled. Pin specified by PSEL will not be acquired by the GPIOTE module.
			Event	1	Event mode
					The pin specified by PSEL will be configured as an input and the IN[n] event will be generated if operation specified in POLARITY occurs on the pin.
			Task	3	Task mode
					The GPIO specified by PSEL will be configured as an output and triggering the SET[n], CLR[n] or OUT[n] task will perform the operation specified by POLARITY on the pin. When enabled as a task the GPIOTE module will acquire the pin and the pin can no longer be written as a regular output pin from the GPIO module.

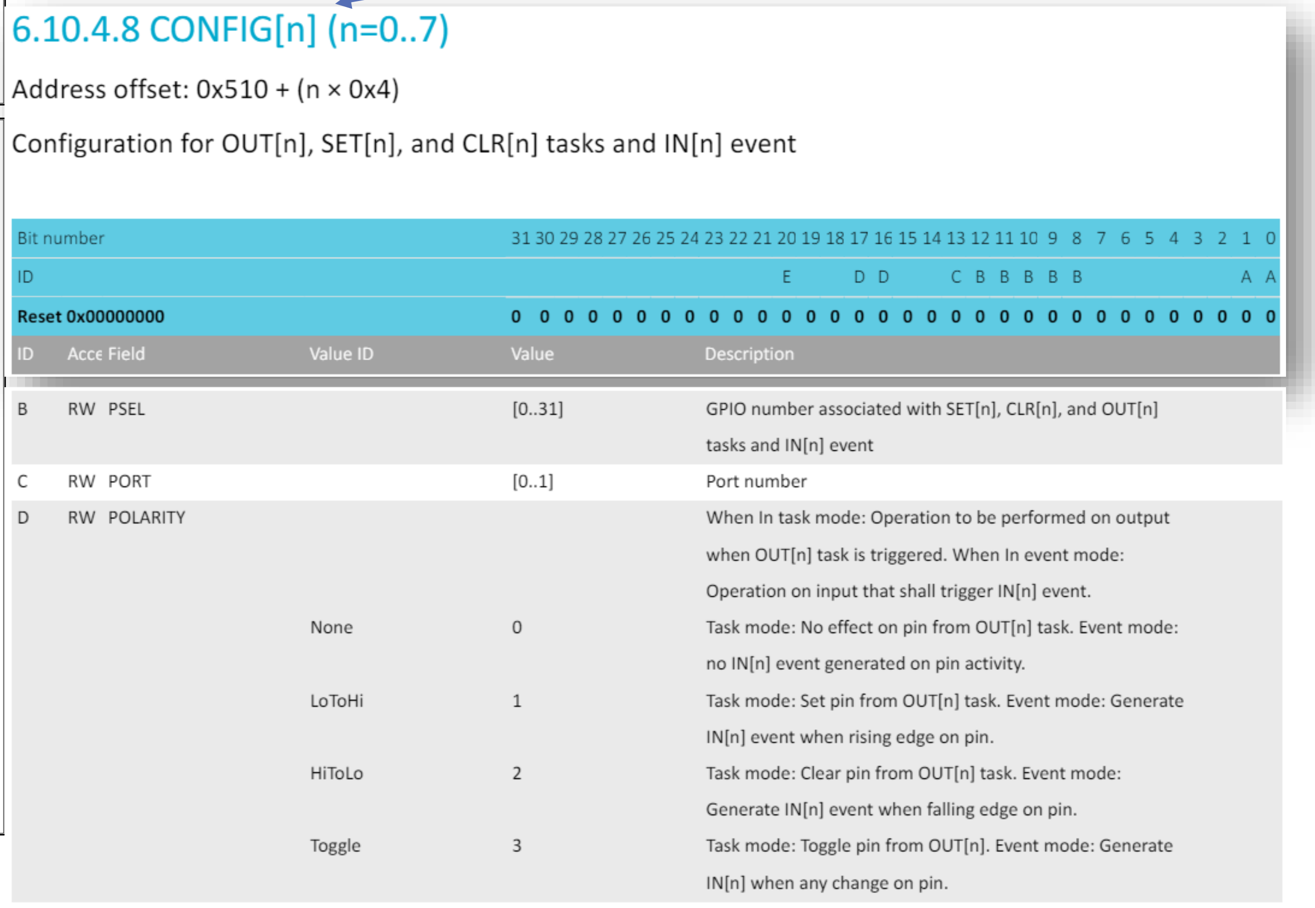
Example Code: GPIOTE.CONFIG[n]

- **nRF52840 DK Button1 Interrupt**
  - (3) Set GPIOTE channel 0's configuration.
    - ✓GPIOTE.CONFIG[0]

```
.equ GPIOTE_BASE, 0x40006000
.equ GPIOTE_CONFIG0_OFFSET, 0x510 // GPIOTE channel 0
```

```
// GPIOTE setting
LDR R0, =GPIOTE_BASE
MOV R1, #0x01 // mode = event (A)
MOV R2, #11 // pin number = 11
LSL R2, R2, #8 // shift bits to PSEL place (B)
ORR R1, R1, R2
MOV R2, #0x00 // port number = 0
LSL R2, R2, #13 // shift bits to PORT place (C)
ORR R1, R1, R2
MOV R2, #0x02 // polarity = HiToLo
LSL R2, R2, #16 // shift bits to PORT place (D)
ORR R1, R1, R2
STR R1, [R0, #GPIOTE_CONFIG0_OFFSET]
```

for channel 0 ($n = 0$)



6.10.4.8 CONFIG[n] (n=0..7)

Address offset: 0x510 + (n × 0x4)

Configuration for OUT[n], SET[n], and CLR[n] tasks and IN[n] event

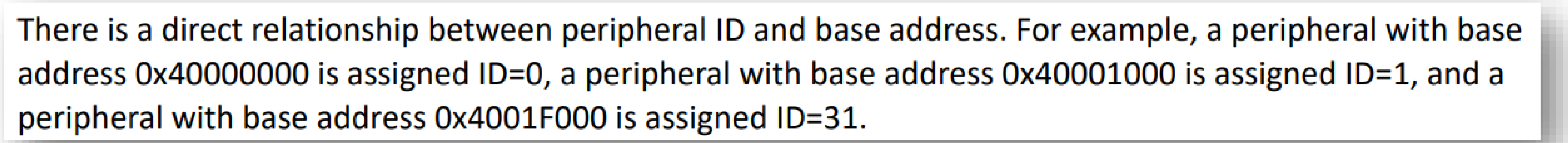
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID												E			D	D			C	B	B	B	B	B							A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
B	RW	PSEL		[0..31]	GPIO number associated with SET[n], CLR[n], and OUT[n] tasks and IN[n] event
C	RW	PORT		[0..1]	Port number
D	RW	POLARITY			When In task mode: Operation to be performed on output when OUT[n] task is triggered. When In event mode: Operation on input that shall trigger IN[n] event.
			None	0	Task mode: No effect on pin from OUT[n] task. Event mode: no IN[n] event generated on pin activity.
			LoToHi	1	Task mode: Set pin from OUT[n] task. Event mode: Generate IN[n] event when rising edge on pin.
			HiToLo	2	Task mode: Clear pin from OUT[n] task. Event mode: Generate IN[n] event when falling edge on pin.
			Toggle	3	Task mode: Toggle pin from OUT[n]. Event mode: Generate IN[n] when any change on pin.

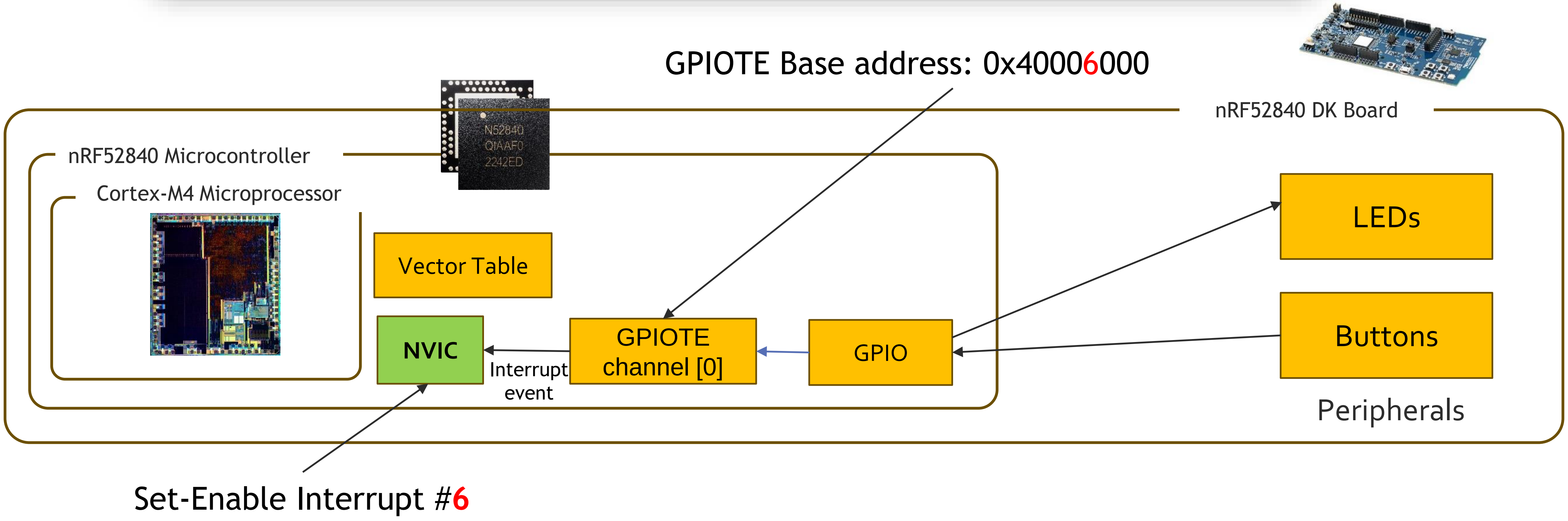
Example Code

- **nRF52840 DK Button1 Interrupt**

from nRF52840 document



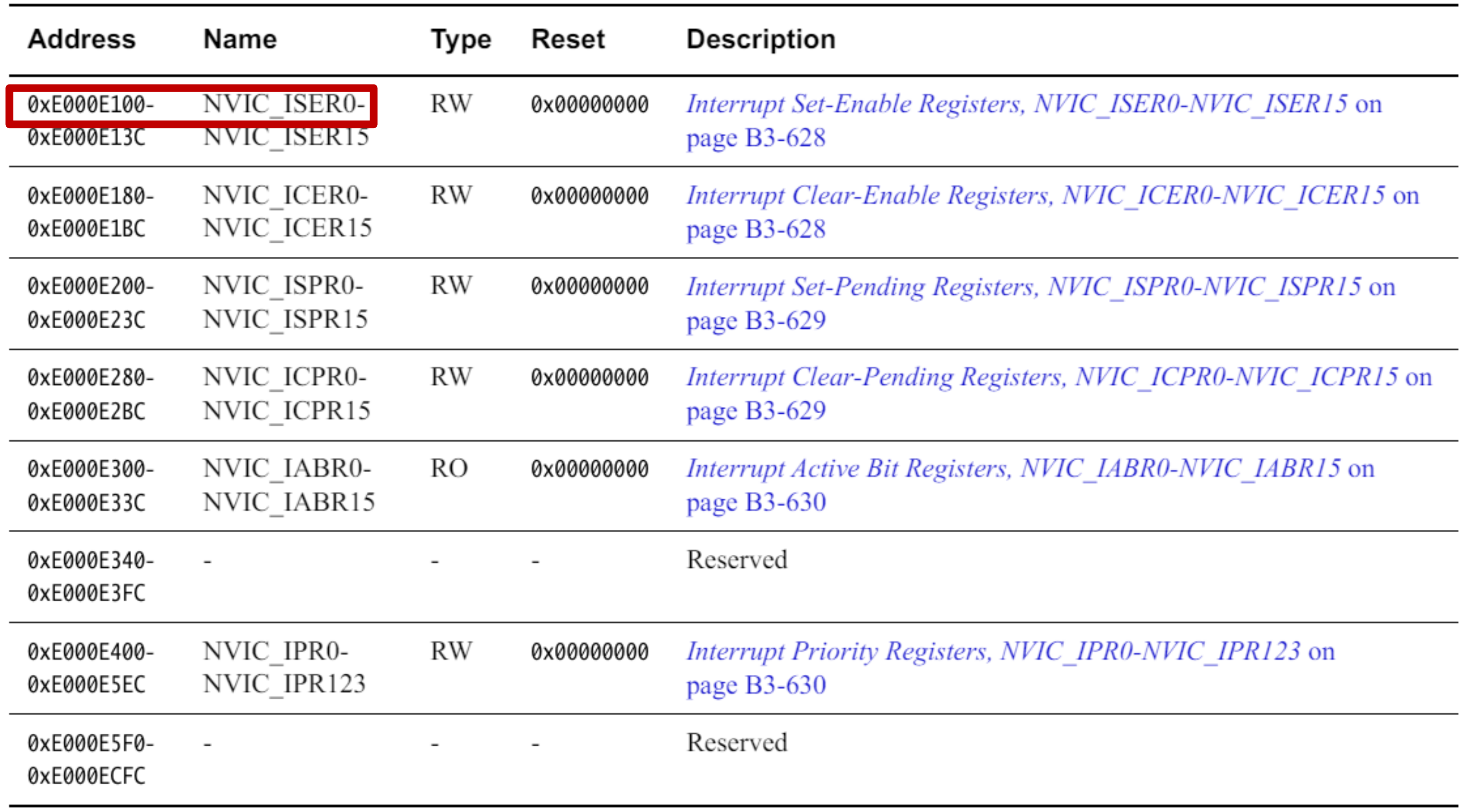
There is a direct relationship between peripheral ID and base address. For example, a peripheral with base address 0x40000000 is assigned ID=0, a peripheral with base address 0x40001000 is assigned ID=1, and a peripheral with base address 0x4001F000 is assigned ID=31.



Example Code (NVIC.NBIC_ISER0 Register)

- **nRF52840 DK Button1 Interrupt**
  - (4) Enable the interrupt for GPIOTE event by changing NVIC registers.
    - ✓It is predefined that GPIOTE event has the Interrupt number 6.

Interrupt 6
can be enabled
using NVIC_ISER0

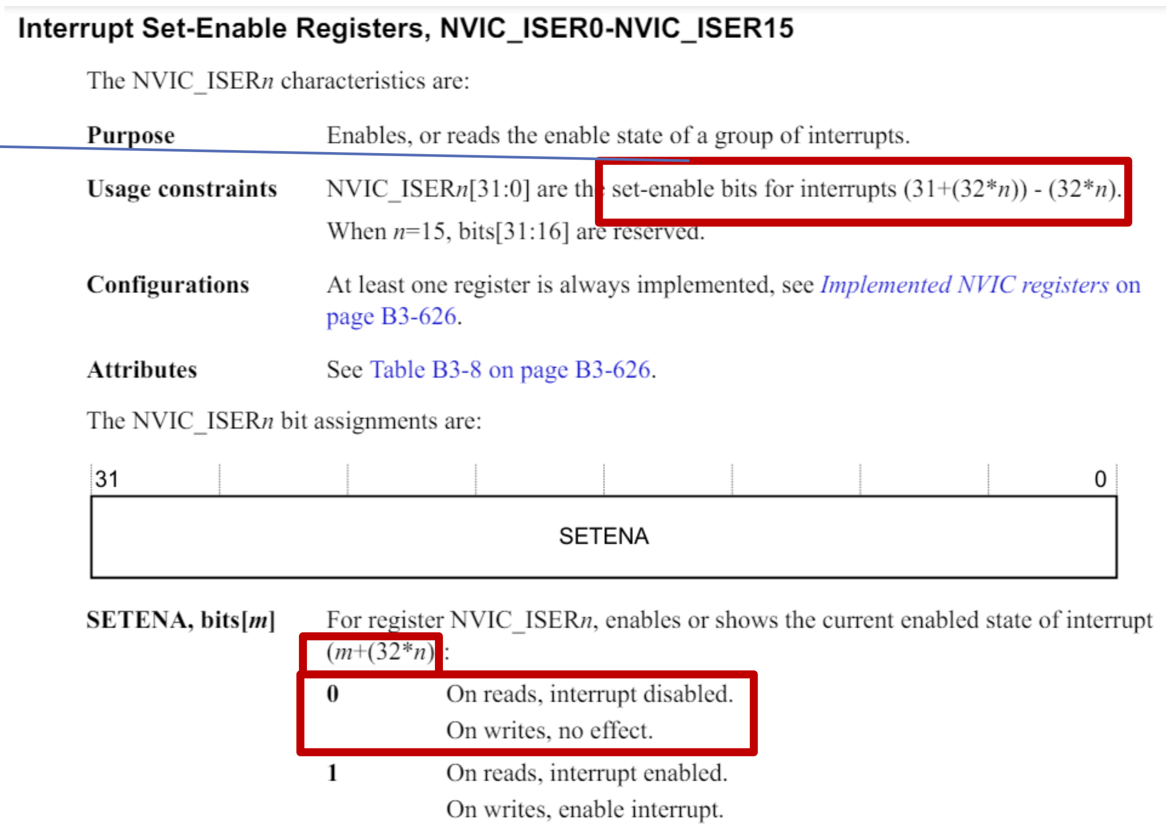


Address	Name	Type	Reset	Description
0xE000E100-0xE000E13C	NVIC_ISER0-NVIC_ISER15	RW	0x00000000	*Interrupt Set-Enable Registers, NVIC_ISER0-NVIC_ISER15* on page B3-628
0xE000E180-0xE000E1BC	NVIC_ICER0-NVIC_ICER15	RW	0x00000000	*Interrupt Clear-Enable Registers, NVIC_ICER0-NVIC_ICER15* on page B3-628
0xE000E200-0xE000E23C	NVIC_ISPR0-NVIC_ISPR15	RW	0x00000000	*Interrupt Set-Pending Registers, NVIC_ISPR0-NVIC_ISPR15* on page B3-629
0xE000E280-0xE000E2BC	NVIC_ICPR0-NVIC_ICPR15	RW	0x00000000	*Interrupt Clear-Pending Registers, NVIC_ICPR0-NVIC_ICPR15* on page B3-629
0xE000E300-0xE000E33C	NVIC_IABR0-NVIC_IABR15	RO	0x00000000	*Interrupt Active Bit Registers, NVIC_IABR0-NVIC_IABR15* on page B3-630
0xE000E340-0xE000E3FC	-	-	-	Reserved
0xE000E400-0xE000E5EC	NVIC_IPR0-NVIC_IPR123	RW	0x00000000	*Interrupt Priority Registers, NVIC_IPR0-NVIC_IPR123* on page B3-630
0xE000E5F0-0xE000ECFC	-	-	-	Reserved

Example Code (NVIC_ISER0 Register)

- **nRF52840 DK Button1 Interrupt**
  - (4) Enable the interrupt for GPIOTE event by changing NVIC registers.
    - ✓It is predefined that GPIOTE event has the Interrupt number 6.
    - ✓NVIC_ISER0
      - $n = 0$
      - Interrupt range: from $(31 + (32 \cdot n))$ to $32 \cdot n$ e.g. from 31 to 0 where interrupt 6 is in.
      - bit 6 needs to be set to 1

```
LDR R0, =0xE000E100 // interrupt set-enable register
MOV R1, #(1<<6) // GPIOTE interrupt number = 6
STR R1, [R0]
```



Interrupt Set-Enable Registers, NVIC_ISER0-NVIC_ISER15

The NVIC_ISER*n* characteristics are:

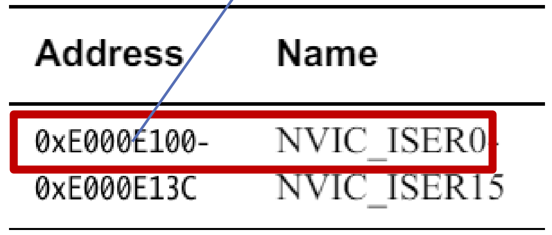
Purpose	Enables, or reads the enable state of a group of interrupts.
Usage constraints	NVIC_ISER*n*[31:0] are th set-enable bits for interrupts (31+(32**n*)) - (32**n*). When *n*=15, bits[31:16] are reserved.
Configurations	At least one register is always implemented, see *Implemented NVIC registers* on page B3-626.
Attributes	See Table B3-8 on page B3-626.

The NVIC_ISER*n* bit assignments are:

31 … 0
SETENA

SETENA, bits[*m*] For register NVIC_ISER*n*, enables or shows the current enabled state of interrupt (*m*+(32**n*)):

0	On reads, interrupt disabled. On writes, no effect.
1	On reads, interrupt enabled. On writes, enable interrupt.

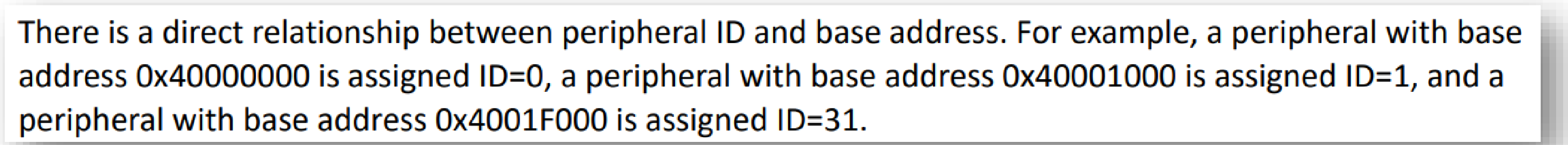


Address	Name
0xE000E100-0xE000E13C	NVIC_ISER0-NVIC_ISER15

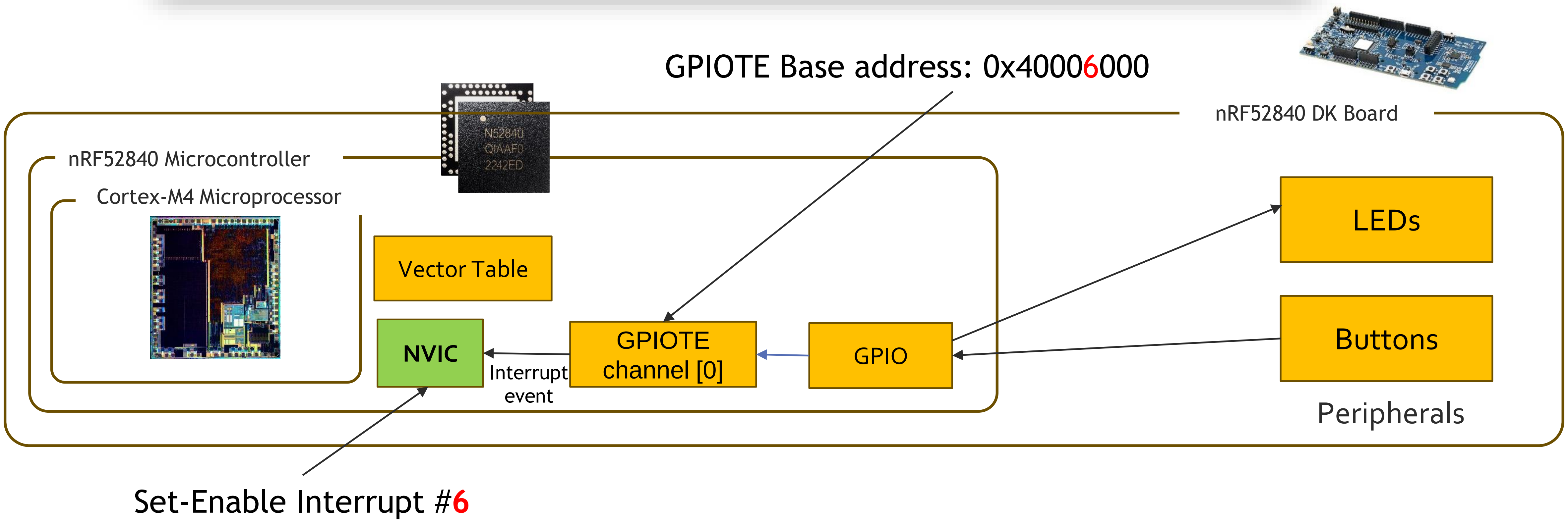
Example Code

- **nRF52840 DK Button1 Interrupt**

from nRF52840 document



There is a direct relationship between peripheral ID and base address. For example, a peripheral with base address 0x40000000 is assigned ID=0, a peripheral with base address 0x40001000 is assigned ID=1, and a peripheral with base address 0x4001F000 is assigned ID=31.



Example Code (NVIC_IPR[1] Register)

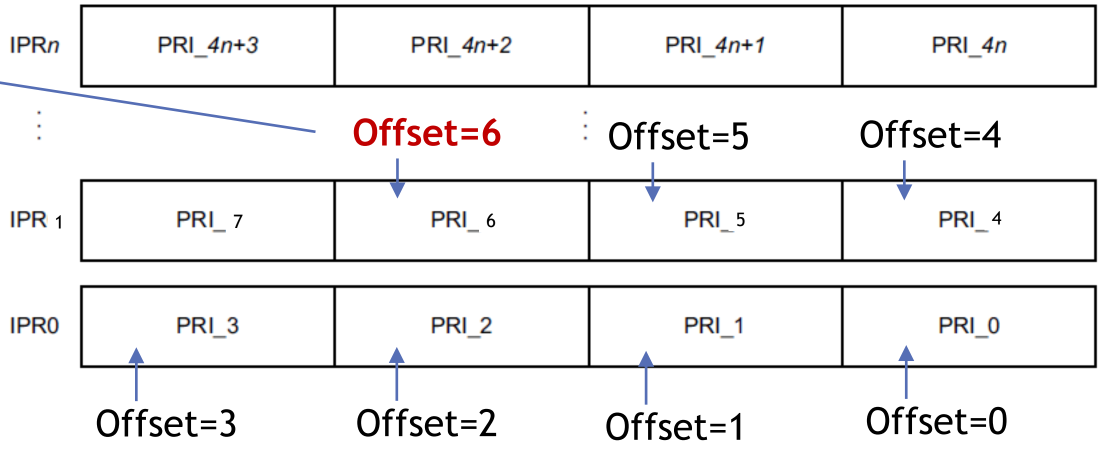
- **nRF52840 DK Button1 Interrupt**
  - (5) Set the priority of the interrupt 6
    - ✓It is predefined that GPIOTE event has the Interrupt number 6.
    - ✓NVIC_IPR0 + #6
    - ✓Priority value: the lower the value, the higher the priority

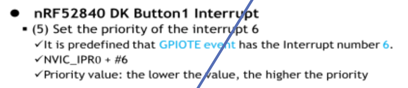
```
LDR R0, =0xE000E400 // interrupt priority register
MOV R1, #2           // My Interrupt Priority (2 for GPIOTE)
STR R1, [R0 , #6]      // GPIOTE interrupt number = 6
```

Interrupt Priority Registers, NVIC_IPR0-NVIC_IPR123

The NVIC_IPR*n* Register characteristics are:

Purpose	Sets or reads interrupt priorities.



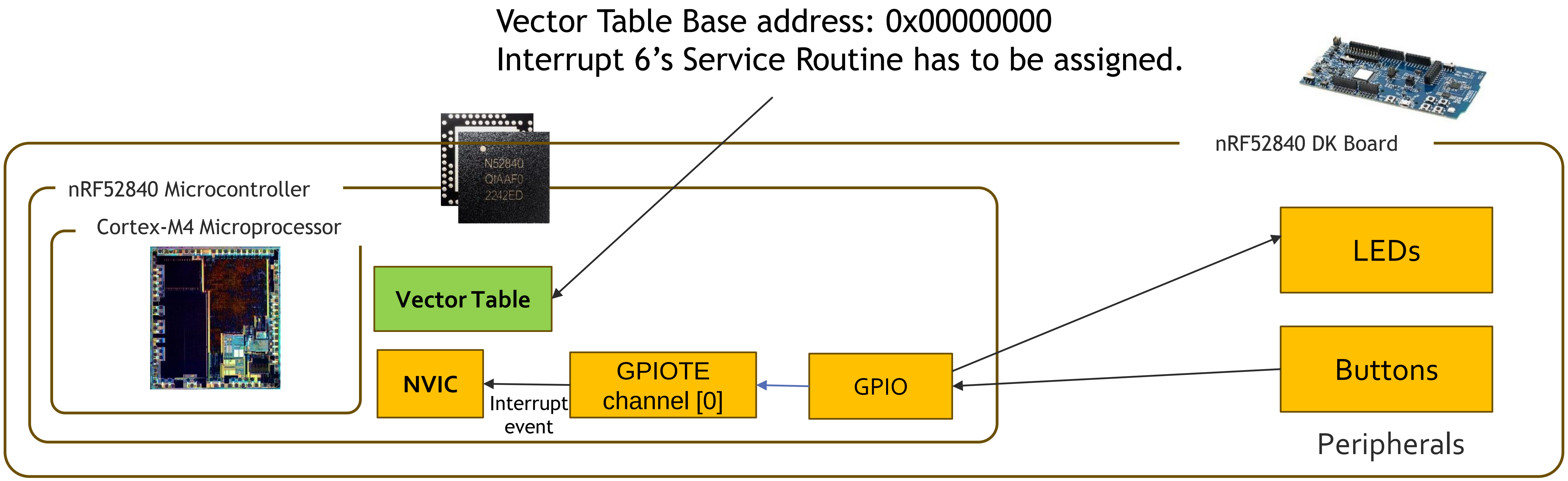


- nRF52840 DK Button1 Interrupt
  - (5) Set the priority of the interrupt 6
    - ✓It is predefined that GPIOTE event has the Interrupt number 6.
    - ✓NVIC_IPR0 + #6
    - ✓Priority value: the lower the value, the higher the priority

0xE000E400-	NVIC_IPR0-
0xE000E5EC	NVIC_IPR123

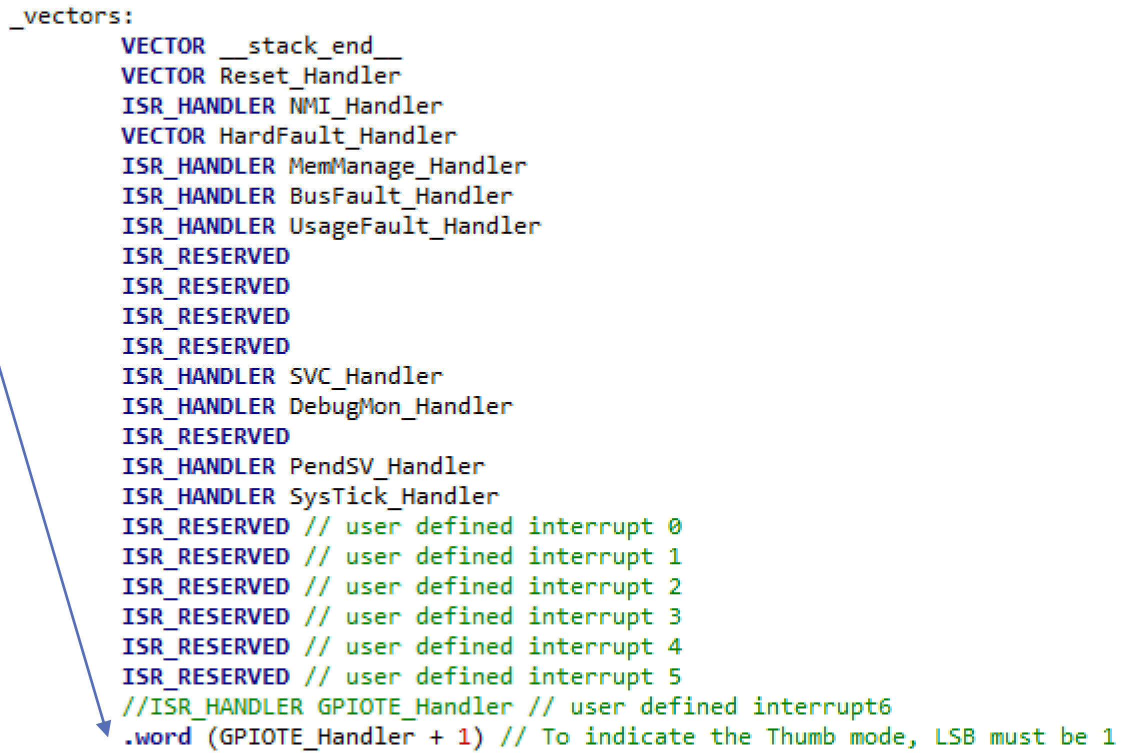
Example Code

- **nRF52840 DK Button1 Interrupt**



Example Code (Interrupt Vector Table Setting)

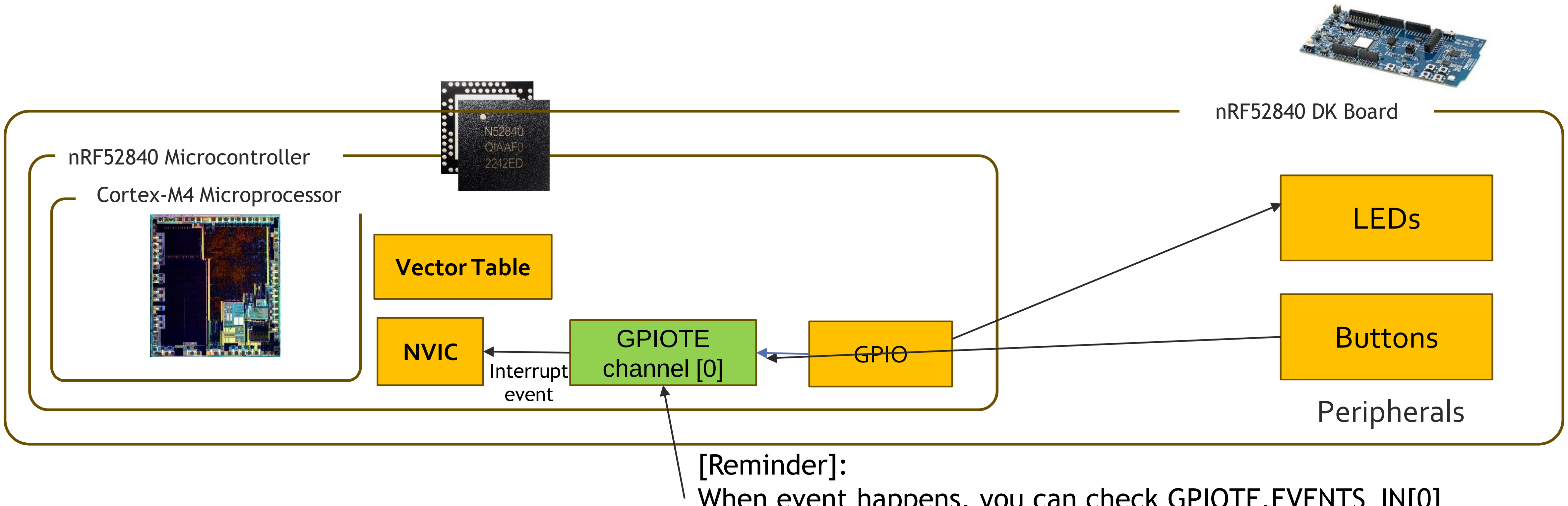
- **nRF52840 DK Button1 Interrupt**
  - (6) Update the vector table defined in “Cortex_M_Startup.s”.
    - ✓There are 16 predefined interrupts(and exceptions).
    - ✓Interrupt 6 goes to the 16+7th place (external or user defined interrupt 6)
    - ✓The linked ISR(Interrupt Service Routine)’s address must have LSB equal to 1.



```
_vectors:
        VECTOR __stack_end__
        VECTOR Reset_Handler
        ISR_HANDLER NMI_Handler
        VECTOR HardFault_Handler
        ISR_HANDLER MemManage_Handler
        ISR_HANDLER BusFault_Handler
        ISR_HANDLER UsageFault_Handler
        ISR_RESERVED
        ISR_RESERVED
        ISR_RESERVED
        ISR_RESERVED
        ISR_HANDLER SVC_Handler
        ISR_HANDLER DebugMon_Handler
        ISR_RESERVED
        ISR_HANDLER PendSV_Handler
        ISR_HANDLER SysTick_Handler
        ISR_RESERVED // user defined interrupt 0
        ISR_RESERVED // user defined interrupt 1
        ISR_RESERVED // user defined interrupt 2
        ISR_RESERVED // user defined interrupt 3
        ISR_RESERVED // user defined interrupt 4
        ISR_RESERVED // user defined interrupt 5
        //ISR_HANDLER GPIOTE_Handler // user defined interrupt6
        .word (GPIOTE_Handler + 1) // To indicate the Thumb mode, LSB must be 1
```

Example Code

- **nRF52840 DK Button1 Interrupt**



[Reminder]:
When event happens, you can check GPIOTE.EVENTS_IN[0] for channel 0 event

Example Code (Interrupt Handler for GPIOTE Event)

- **nRF52840 DK Button1 Interrupt**
  - (7) Define the interrupt service routine in “main.s”.
    - ✓GPIOTE_Handler

```
.word (GPIOTE_Handler + 1)
```

    - ✓Notes
      - Must include the line “. global GPIOTE_Handler”
      - Must clear the interrupt event not to repeat the ISR.

GPIOTE base addr = 0x40006000
GPIOTE.EVENTS_IN[0] addr = 0x100

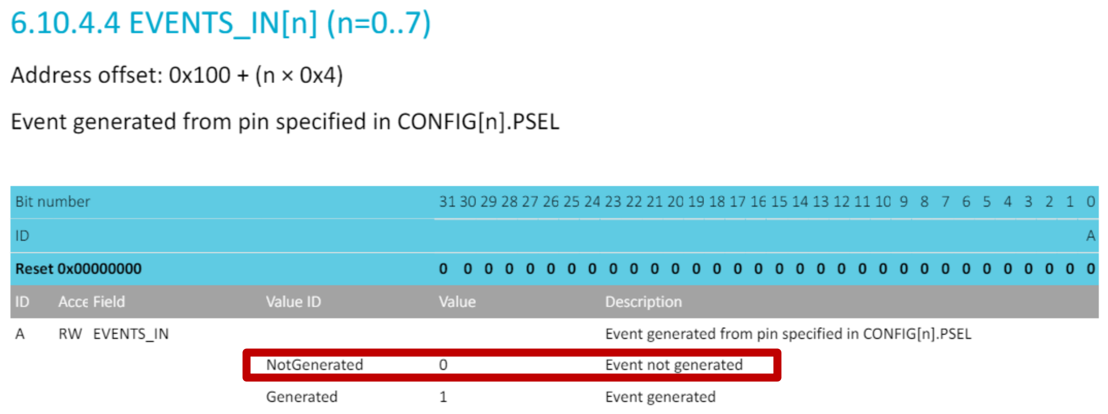
```
.equ GPIOTE_EVENT_IN0, 0x40006100
```

```
.global GPIOTE_Handler
GPIOTE_Handler:
  PUSH {R0-R7, LR}

  // Instructions
  …

  // event clear
  LDR R0, =GPIOTE_EVENT_IN0
  MOV R1, #0
  STR R1, [R0] // GPIOTE.EVENTS_IN[0] clear

  POP {R0-R7, LR}

  BX LR
```

storing register values for the previous process

clearing all the events in Channel 0

retrieving register values for the previous process



6.10.4.4 EVENTS_IN[n] (n=0..7)

Address offset: 0x100 + (n × 0x4)

Event generated from pin specified in CONFIG[n].PSEL

Bit number	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
ID																																A
Reset 0x00000000	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	0	0	0	0	0	0

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_IN			Event generated from pin specified in CONFIG[n].PSEL
			NotGenerated	0	Event not generated
			Generated	1	Event generated

Example Code (main Routine)

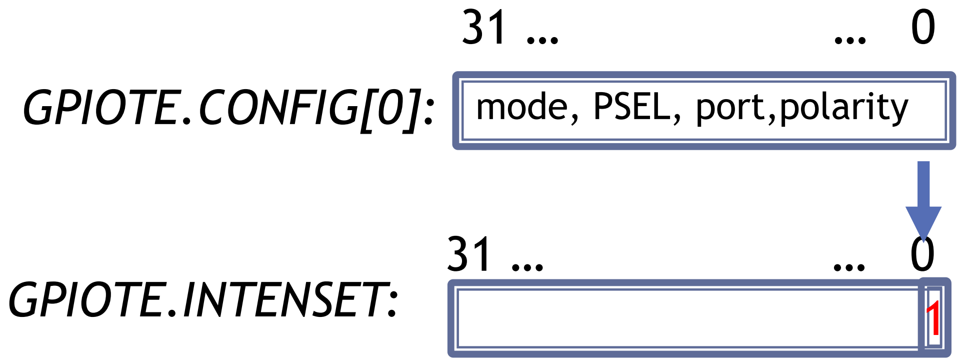
- **nRF52840 DK Button1 Interrupt**
  - (8) Test the interrupt
    - ✓Making the device in the sleep mode using the WFI (Wait for Interrupt) instruction.
    - ✓See if the device awakes when an interrupt occurs.

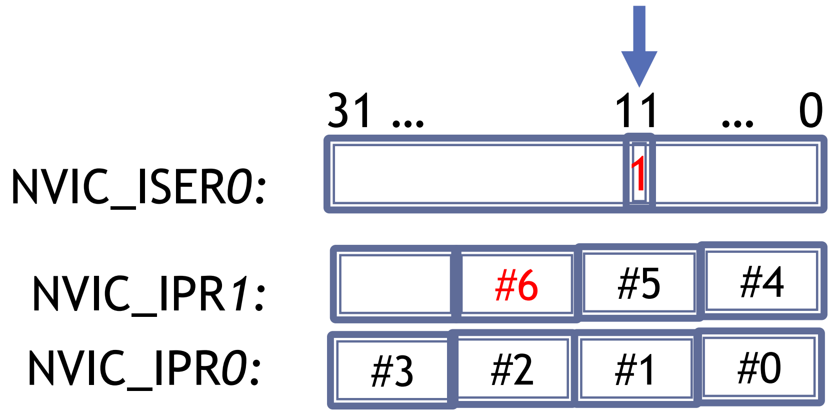
```
main:
  … // interrupt setting part

  WFI // The device goes to sleep.

  BL . // infinite loop to make the processor stay at here
```

    - ✓While sleeping, i.e. WFI is executed, the processor is not in the infinite loop,
      but it goes into the infinite loop once Button1 is pressed and the interrupt service routine is processed.

Summary: Register Configuration for the Interrupt Handling

1. GPIOTE Registers:
   - GPIOTE.INTENSET : Enable interrupt from each Channel
   - GPIOTE.CONFIG[n]: Configures the GPIOTE channel, such as the mode (e.g. event mode) and the triggering event condition (e.g., Hi2Lo event).
2. NVIC Registers:
   - NVIC_ISER*n*: Enables the GPIOTE interrupt (Interrupt Num=6)in the NVIC. (n=0)
   - NVIC_IPR*n*: Sets the priority level for the GPIOTE interrupt. (n=1)
3. GPIO Registers:
   - GPIO_PIN_CNF[n]: Configures the GPIO pin direction and input/output behavior. (n=11)



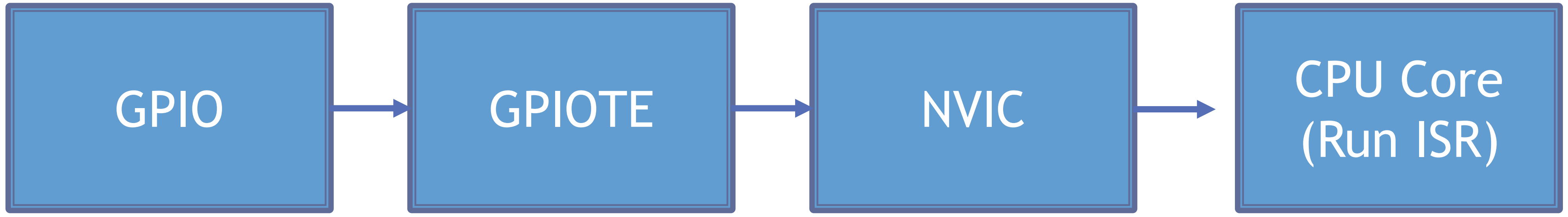


Control Flow for Button Event Interrupt Handling

1. **Button Input Event:**
   - The user presses the button on the nRF52840DK board.
   - This triggers a GPIO input signal change, which is detected by the GPIOTE peripheral.
2. **GPIOTE (GPIO Task and Event) Interrupt Trigger**:
   - The GPIOTE peripheral recognizes the input signal change as a configured event
   - The GPIOTE peripheral generates an interrupt request and sends it to the NVIC.
3. **NVIC(Nested Vectored Interrupt Controller) Interrupt Handling:**
   - The NVIC receives the interrupt request from the GPIOTE peripheral (Interrupt number=6).
   - The NVIC checks the priority of the GPIOTE interrupt and determines if it should preempt any currently executing interrupts.
   - The NVIC then saves the current context (e.g., CPU registers) and jumps to the appropriate Interrupt Service Routine (ISR) address in the interrupt vector table.
4. **ISR (interrupt Service Routine) Execution**:
   - The processor starts executing the GPIOTE_IRQHandler, which is the ISR for the GPIOTE interrupt.
   - In the ISR, you can perform the desired actions, such as toggling an LED.
   - Once the ISR completes, the NVIC restores the previous context and returns control to the main program.

Lab-1: Button-1 Event Interrupt

- Lab-1 mission
  - Toggle LED-1 by the Button-1 Push Event
  - Use External Interrupt Service instead of Polling Method
  - Write the code in Thumb-2 Assembly Language Code
  - Set The GPIOTE Interrupt number as 6 (Exception number = 7+15 = 22)
  - Set the Interrupt Priority as 2
- For Interrupt Service Addition, Related Modules



Lab-1 Source Code (1) : Vector Table Update

```
        .balign 4
        .global _vectors
_vectors:
        VECTOR          __stack_end__
        VECTOR          Reset_Handler
        ISR_HANDLER     NMI_Handler
        VECTOR          HardFault_Handler
        ISR_HANDLER     MemManage_Handler
        ISR_HANDLER     BusFault_Handler
        ISR_HANDLER     UsageFault_Handler
        ISR_RESERVED
        ISR_RESERVED
        ISR_RESERVED
        ISR_RESERVED
        ISR_HANDLER     SVC_Handler
        ISR_HANDLER     DebugMon_Handler
        ISR_RESERVED
        ISR_HANDLER     PendSV_Handler
        ISR_HANDLER     SysTick_Handler
        ISR_RESERVED // user defined interrupt 0
        ISR_RESERVED // user defined interrupt 1
        ISR_RESERVED // user defined interrupt 2
```

```
ISR_RESERVED // user defined interrupt 3
ISR_RESERVED // user defined interrupt 4
ISR_RESERVED // user defined interrupt 5
                   // user defined interrupt6
.word (GPIOTE_Handler + 1) // To indicate the Thumb mode,
                           // LSB must be 1
```

Lab-1 Source Code (2) : GPIOTE Handler

```
.global GPIOTE_Handler
// Interrupt Handler for GPIOTE Event
GPIOTE_Handler:
  PUSH {LR}
  // Check if the interrupt source is Button1 Event (GPIOTE channel 0)
  LDR R0, =GPIOTE_EVENT_IN0
  LDR R1, [R0]      // Read GPIOTE Event In0
  TST R1, #1        // Check if Event In0 occurred
  BEQ EXIT_GPIOTE_Handler  // If No Button1 Event, Skip the Handler

  // Perform LED toggle Task
  BL led1Toggle

  // clear Event in GPIOTE Event Reg
  LDR R0, =GPIOTE_EVENT_IN0
  MOV R1, #0
  STR R1, [R0] // GPIOTE.EVENTS_IN[0] clear

EXIT_GPIOTE_Handler:
  POP {LR}
  BX LR
```

```
led1Toggle:
  // Tpggles LED ON and OFF

 // Read LED OUT state
  LDR R0, =GPIO_P0_BASE
  LDR R1, [R0, #GPIO_OUT_REG_OFFSET]

 // Toggle LED1 value and write it
 EORS R1, R1, #LED1_MASK
  STR R1, [R0, #GPIO_OUT_REG_OFFSET]

  BX LR //MOV PC, LR // return
```

Lab-1 Source Code (3) : GPIOTE & NVIC Setup

```
GPIOTE_SETUP:
  // GPIOTE setting
  LDR R0, =GPIOTE_BASE
  MOV R1, #0x01 // channel = 0
  STR R1, [R0, #GPIOTE_INTENSET_OFFSET]

// setup GPIOTE.CONFIG[0] for INT Channel 0
  MOV R1, #0x01 // mode = event
  MOV R2, #11   // pin number = 11
  LSL R2, R2, #8 // shift bits to PSEL place (BBBBB)
  ORR R1, R1, R2
  MOV R2, #0x00 // port number = 0
  LSL R2, R2, #13 // shift bits to PORT place (C)
  ORR R1, R1, R2
  MOV R2, #0x02 // polarity = HiToLo
  LSL R2, R2, #16 // shift bits to PORT place (DD)
  ORR R1, R1, R2
  STR R1, [R0, #GPIOTE_CONFIG0_OFFSET]

  BX LR // return
```

```
NVIC_SETUP:
  // NVIC setting
  LDR R0, =0xE000E100 // interrupt set-enable register
  MOV R1, #(1<<6) // GPIOTE interrupt number = 6
  STR R1, [R0]
  LDR R0, =0xE000E400 // interrupt priority register
  MOV R1, #2 // It doesn't have to be at the top priority
  STR R1, [R0 , #6] // GPIOTE interrupt number = 6

  BX LR  // return
```

Lab-1 Source Code (4) : main & GPIO Setup

```
.thumb
.thumb_func

.equ BTN1_MASK, 0x00800
.equ BTN2_MASK, 0x01000
.equ LED1_MASK, 0x02000
.equ LED2_MASK, 0x04000
.equ LED3_MASK, 0x08000
.equ LED4_MASK, 0x10000

.equ GPIO_P0_BASE,              0x50000000
.equ GPIO_DIRSET_REG_OFFSET,    0x518
.equ GPIO_OUT_REG_OFFSET,       0x504
.equ GPIO_PIN_CNF_11_OFFSET,    0x72C
              // P0.11 pin configuration address offset (BTN1)

.equ GPIOTE_BASE, 0x40006000
.equ GPIOTE_INTENSET_OFFSET,    0x304
.equ GPIOTE_CONFIG0_OFFSET,     0x510 // GPIOTE channel 0
.equ GPIOTE_EVENT_IN0,          0x40006100
```

```
.section .text
.global main

main:
  BL GPIO_SETUP
  BL GPIOTE_SETUP
  BL NVIC_SETUP
  WFI
  B .

GPIO_SETUP:
  // LED1 setting
  LDR R0, =GPIO_P0_BASE
  LDR R1, =LED1_MASK
  STR R1, [R0, #GPIO_DIRSET_REG_OFFSET]

  // Button1 setting
  LDR R1, =0x0003000c
  //LDR R1, =0x0000000c
  STR R1, [R0, #GPIO_PIN_CNF_11_OFFSET]
  MOV PC,LR
```

Quiz

1. What happen if you delete the two lines in GPIOTE_Handler ?
   - push {lr}
   - pop {lr}

2. What is the value of LR in the GPIOTE_Handler entry?

1. What is the value of stack pointer in the main routine and in the GPIOTE Handler Entry? Explain the difference

Exercise 1,2

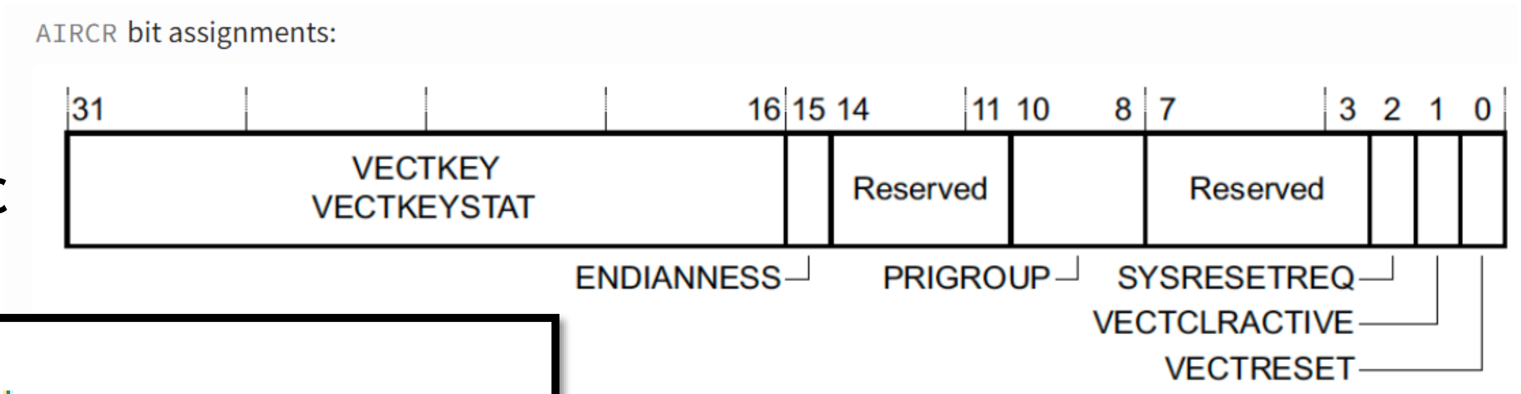
- Exercise 1: Change the GPIOTE_Handler (ISR)
  - When Button-1 is Pressed, LED-1 and LED-2 blink at 1Hz for 5 seconds and return to man routine

- Exercise 2: Add Button-2 Event Interrupt
  - When Button-2 is pressed LED-2 will Blink at 10Hz rates for 5 sseconds
  - Set Button-2 Interrupt Priority Lower than Button-1 Interrupt (set PRI=3)

Exercise 3

- Change the Assembly code to support the Soft Reset by Button 3 push event
  - Following code is the Soft reset code by setting AIRCR.SYSRESETREQ bit (2nd bit of AIRCR) = 1
  - In order to Write the register VECTKEY should be set to 0x05FA

AIRCR base address : 0xE000ED0C



```
reset:
  LDR R0, =0xE000ED0C // AIRCR address
  LDR R1, [R0]
  ORR R1, R1, #(1 << 2) // SYSRESETREQ bit is set to 1
  LDR R2, =0x0000FFFF
  AND R1, R1, R2
  LDR R2, =0x05FA0000 // by the rule
  ORR R1, R1, R2
  STR R1, [R0]
  B .
```