Interrupts and Interrupt Structure for SAMD21

Lance Ragas

What is an interrupt

- Signal sent for event that needs immediate attention
- Sent from either hardware or software
- Received by operating system
- Computers today are interrupt-driven
    - Keep running down todo list until you reach end or are interrupted
- Mediated by processor and handled by kernel

What can interrupts be triggered by?

Hardware

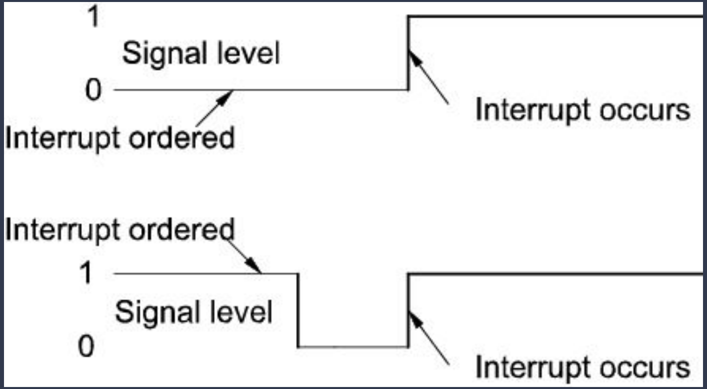
- Any device connected to computer, internal and external
- Electronic alerting signal sent to processor

Software

- Caused by either exceptional condition or an instruction in the instruction set
- Exceptional condition
    - trap/ exception
    - Errors or events occurring that cannot be handled by program

Types of Interrupts

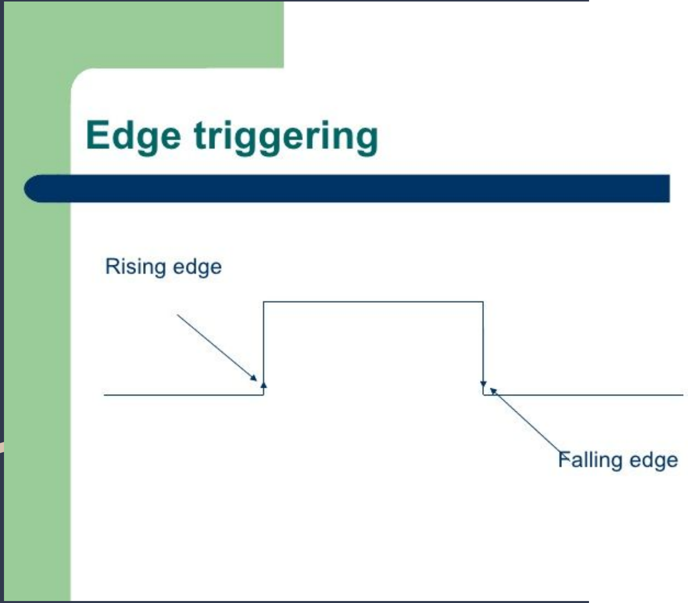
Level-Triggered



- Triggered by maintaining a high or low logic
- Interrupt then occurs during the rising or falling edge of the clock
  - Edge can be configured
- Line remains asserted if interrupt is not serviced

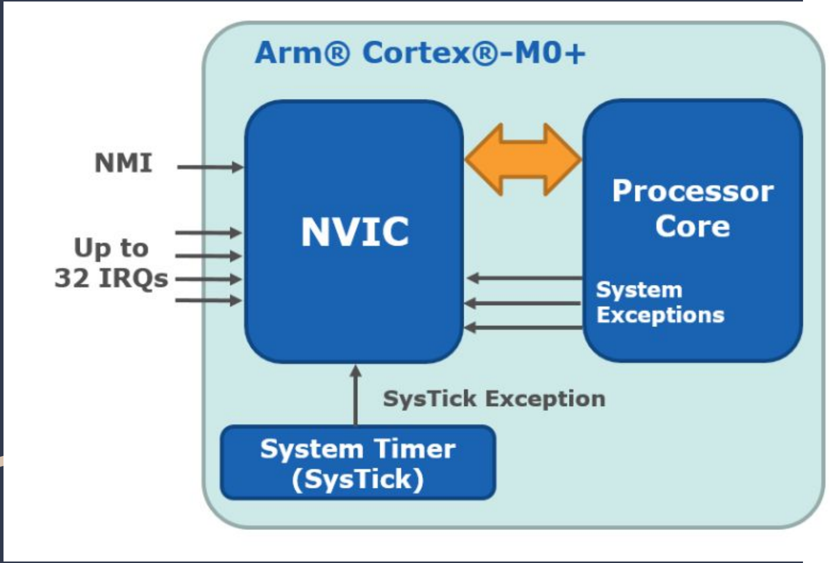
Edge Triggered

- Triggered by interrupt line switching levels
- Either falling edge or rising edge


Edge triggering

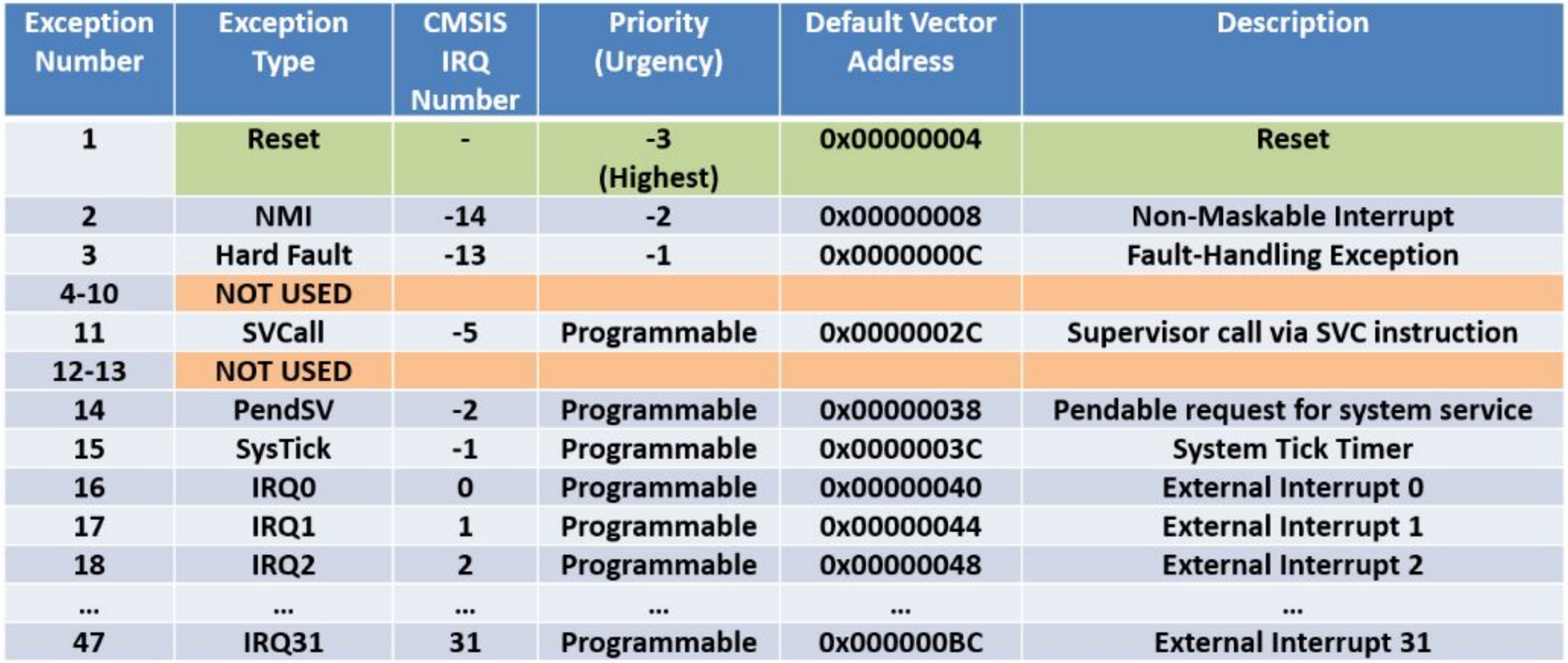
SAMD21

Nested Vectored Interrupt Controller (NVIC)



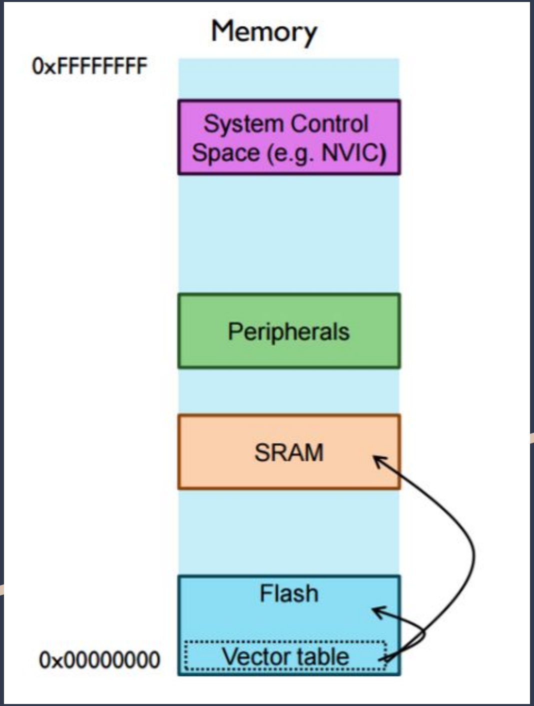
- Interrupt controller
- Flexible interrupt priority management
- Programmable priority levels
- Automatic nested interrupt support
- Supports up to 32 external interrupt request inputs (IRQs)
  - Each IRQ has 4 programmable priority levels
- Internal Interrupts
  - SysTick Exception
  - Processor Core Exception

Cortex Microcontroller Software Interface Standard (CMSIS) IRQ



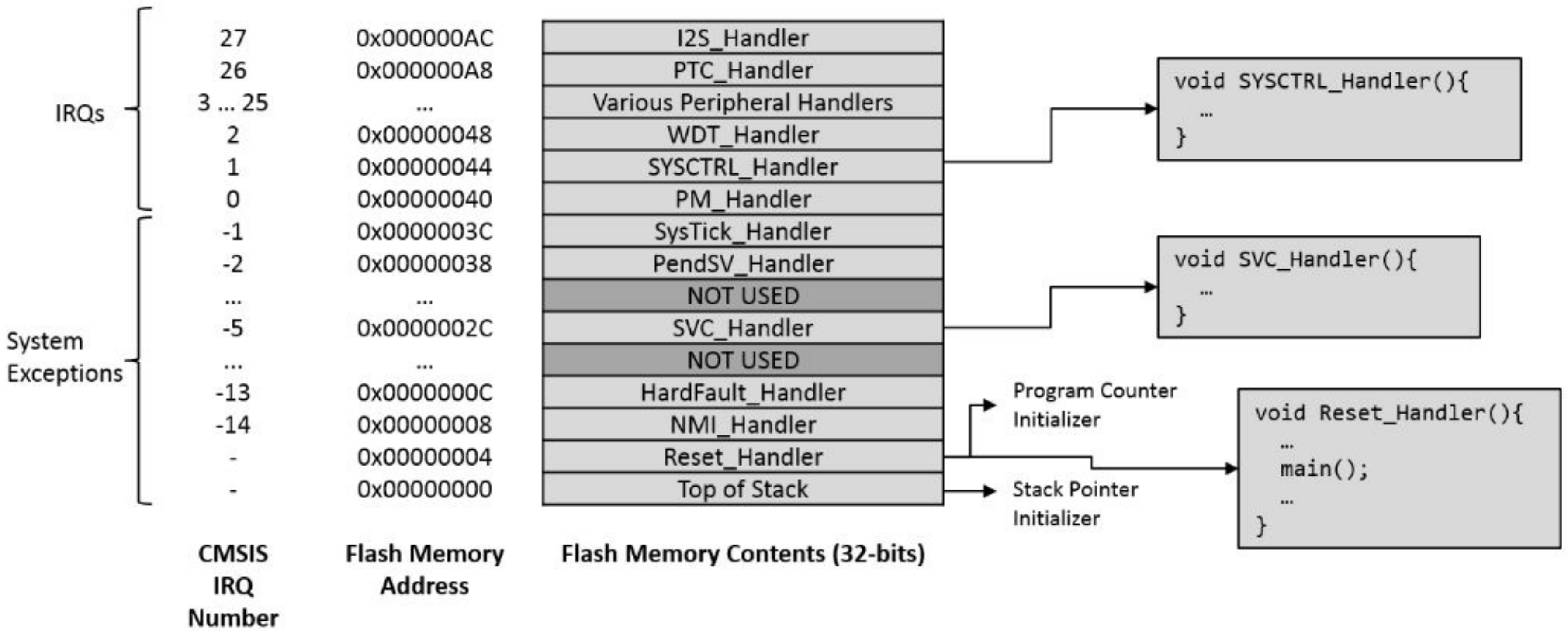
Exception Number	Exception Type	CMSIS IRQ Number	Priority (Urgency)	Default Vector Address	Description
1	Reset	-	-3 (Highest)	0x00000004	Reset
2	NMI	-14	-2	0x00000008	Non-Maskable Interrupt
3	Hard Fault	-13	-1	0x0000000C	Fault-Handling Exception
4-10	NOT USED				
11	SVCall	-5	Programmable	0x0000002C	Supervisor call via SVC instruction
12-13	NOT USED				
14	PendSV	-2	Programmable	0x00000038	Pendable request for system service
15	SysTick	-1	Programmable	0x0000003C	System Tick Timer
16	IRQ0	0	Programmable	0x00000040	External Interrupt 0
17	IRQ1	1	Programmable	0x00000044	External Interrupt 1
18	IRQ2	2	Programmable	0x00000048	External Interrupt 2
...	...	...	...	...	...
47	IRQ31	31	Programmable	0x000000BC	External Interrupt 31

Vector Table



- Memory is stored in vectors
- Hardware automatically determines which interrupt or exception routine to execute
- Useful for:
  - Bootloader applications
  - Faster vector table fetch
  - Dynamic changing of handlers
  - Executing programs from RAM

Vector Tables



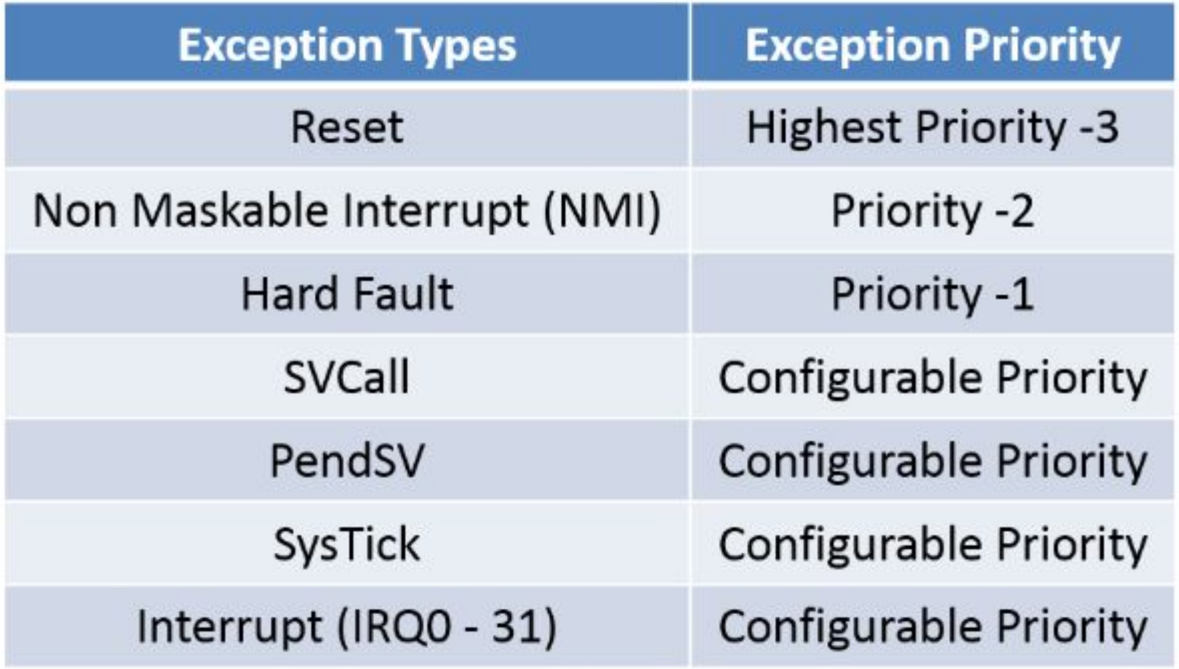
CMSIS IRQ Number	Flash Memory Address	Flash Memory Contents (32-bits)
27	0x000000AC	I2S_Handler
26	0x000000A8	PTC_Handler
3 … 25	…	Various Peripheral Handlers
2	0x00000048	WDT_Handler
1	0x00000044	SYSCTRL_Handler
0	0x00000040	PM_Handler
-1	0x0000003C	SysTick_Handler
-2	0x00000038	PendSV_Handler
…	…	NOT USED
-5	0x0000002C	SVC_Handler
…	…	NOT USED
-13	0x0000000C	HardFault_Handler
-14	0x00000008	NMI_Handler
-	0x00000004	Reset_Handler
-	0x00000000	Top of Stack

```
void SYSCTRL_Handler(){
  …
}
```

```
void SVC_Handler(){
  …
}
```

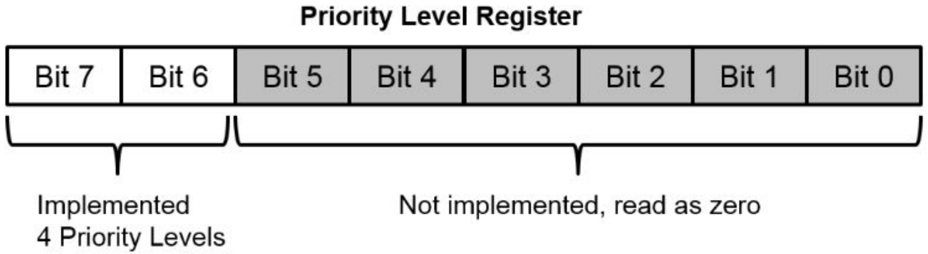
```
void Reset_Handler(){
  …
  main();
  …
}
```

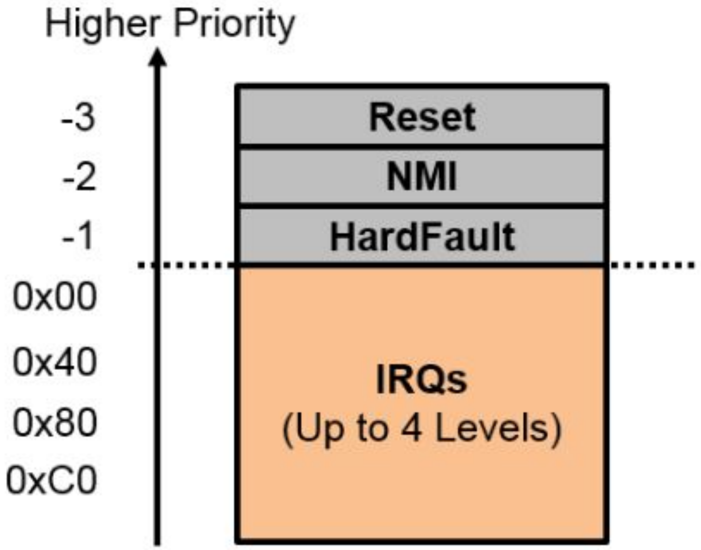
Priority Levels



Exception Types	Exception Priority
Reset	Highest Priority -3
Non Maskable Interrupt (NMI)	Priority -2
Hard Fault	Priority -1
SVCall	Configurable Priority
PendSV	Configurable Priority
SysTick	Configurable Priority
Interrupt (IRQ0 - 31)	Configurable Priority

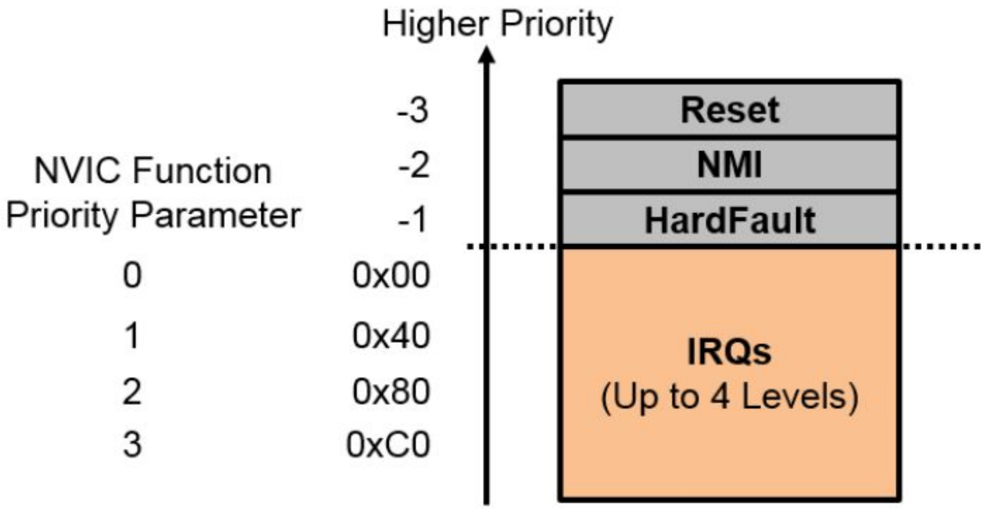
Priority Level



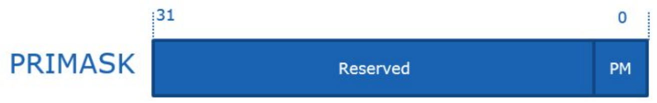


Priority Level Configuration

```
void NVIC_SetPriority(IRQn_t IRQn, uint32_t priority);
```



Global IRQs Enabling

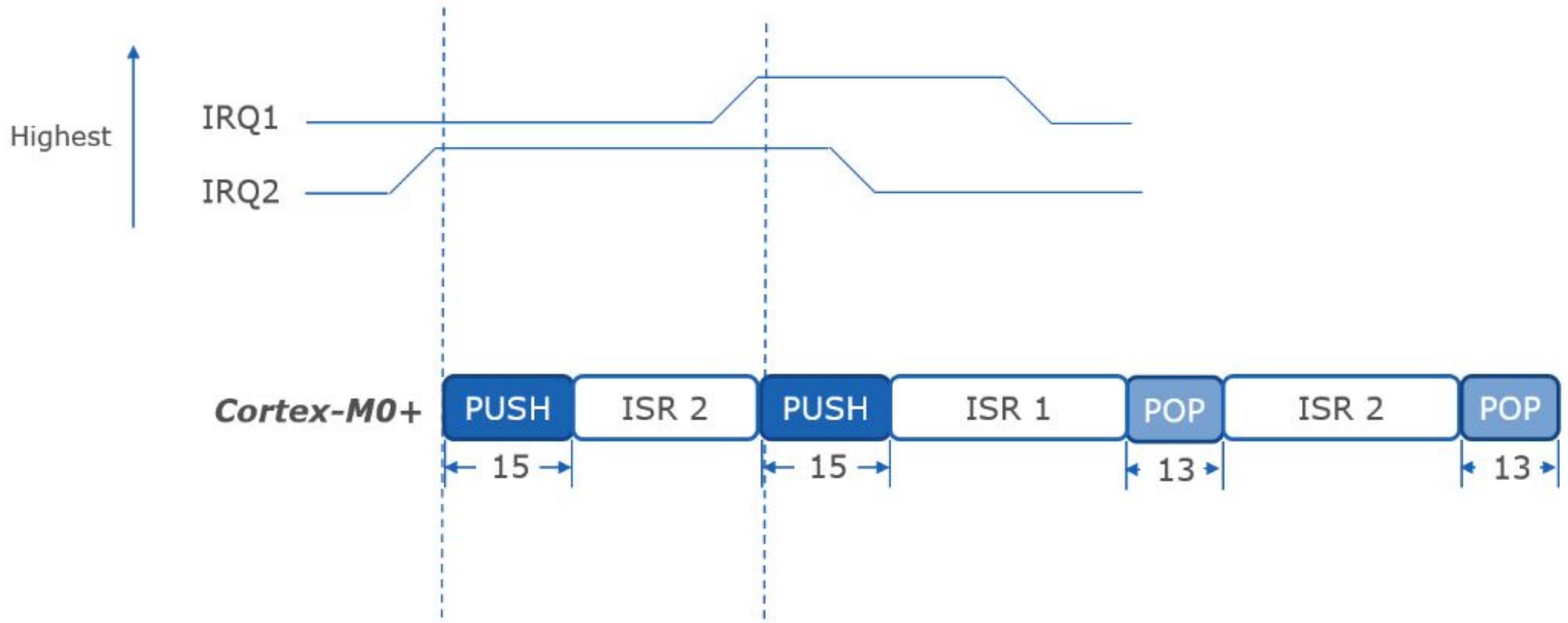


bit 0 **PM:** Prioritizable Interrupt Mask

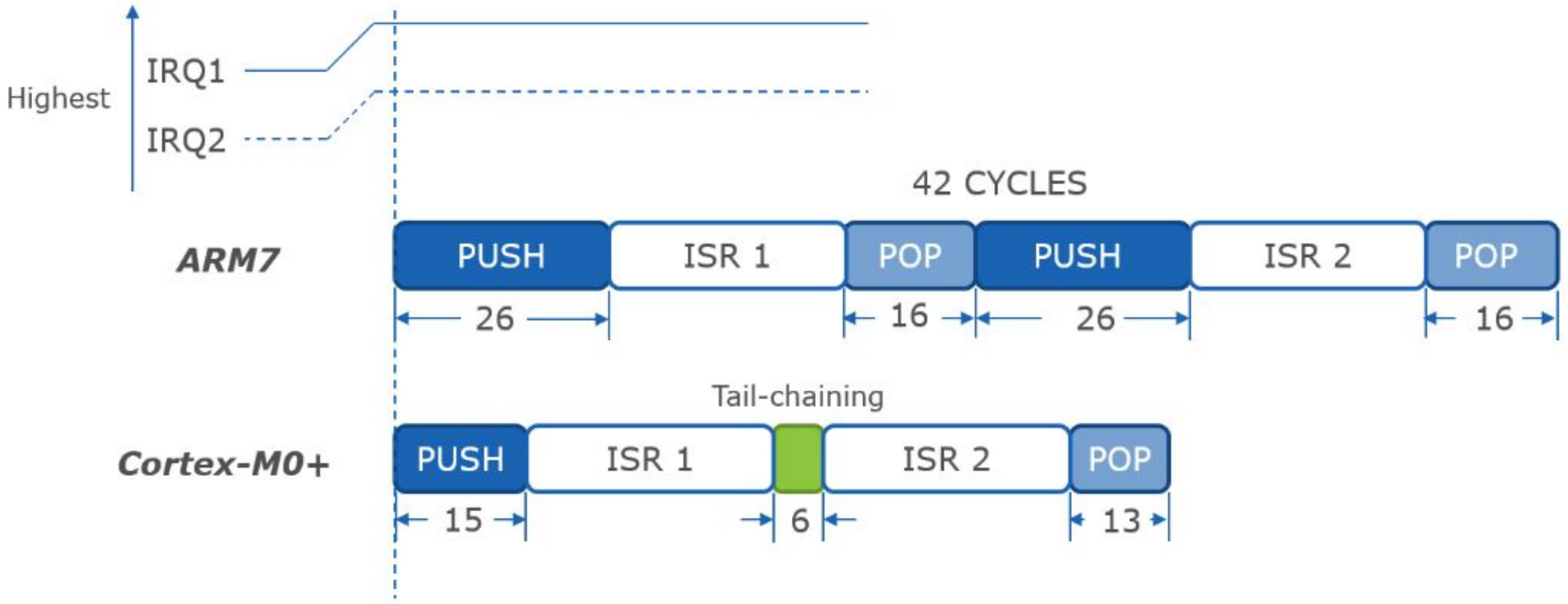
1 = all exceptions with configurable priority are disabled
0 = (default) all IRQs enabled

```
/* Disable Interrupts */
void __disable_irq(void);

/* Enable Interrupts */
void __enable_irq(void);
```

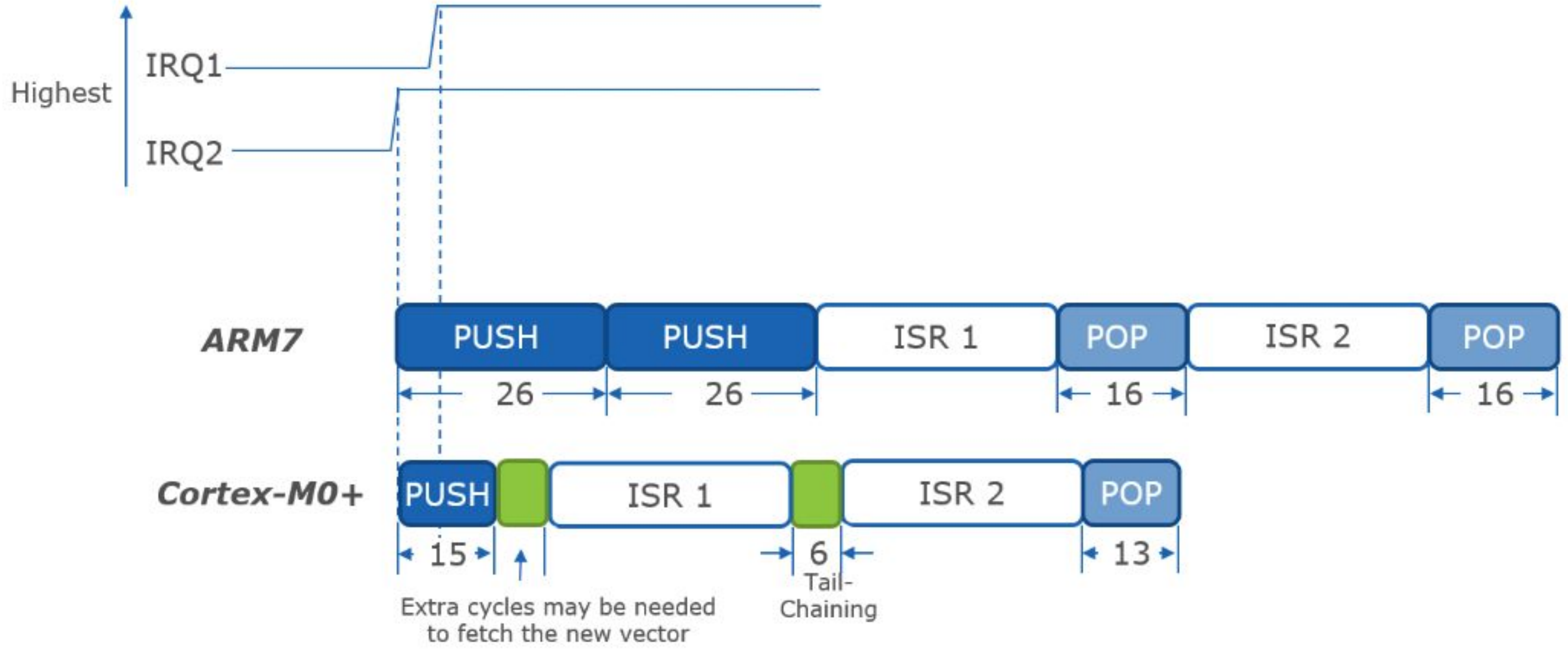
Interrupt Nesting



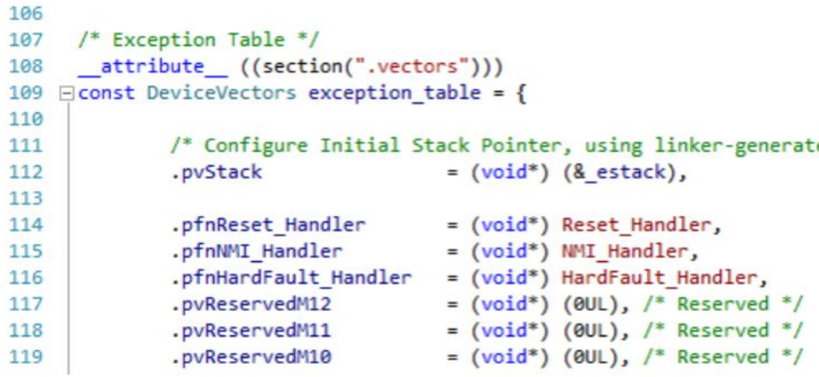
Tail-Chaining



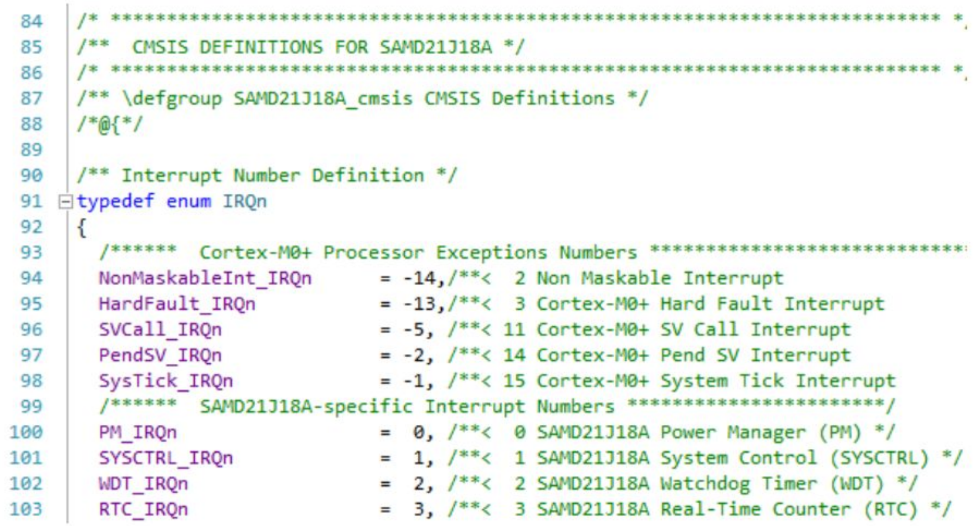
Late Arrival



Vector table and IRQ declartions



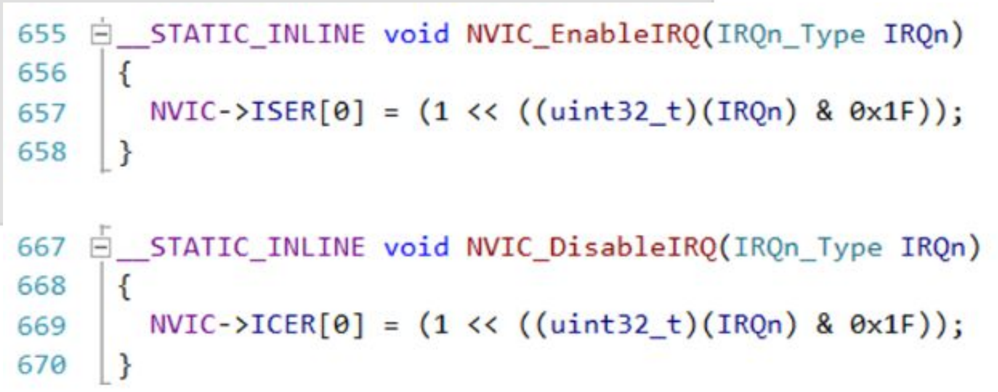
```
/* Exception Table */
__attribute__ ((section(".vectors")))
const DeviceVectors exception_table = {

        /* Configure Initial Stack Pointer, using linker-generated symbols */
        .pvStack                = (void*) (&_estack),

        .pfnReset_Handler       = (void*) Reset_Handler,
        .pfnNMI_Handler         = (void*) NMI_Handler,
        .pfnHardFault_Handler   = (void*) HardFault_Handler,
        .pvReservedM12          = (void*) (0UL), /* Reserved */
        .pvReservedM11          = (void*) (0UL), /* Reserved */
        .pvReservedM10          = (void*) (0UL), /* Reserved */
```



```
/* ************************************************************************** *
/**  CMSIS DEFINITIONS FOR SAMD21J18A */
/* ************************************************************************** *
/** \defgroup SAMD21J18A_cmsis CMSIS Definitions */
/*@{*/

/** Interrupt Number Definition */
typedef enum IRQn
{
  /******  Cortex-M0+ Processor Exceptions Numbers ****************************
  NonMaskableInt_IRQn      = -14,/**<  2 Non Maskable Interrupt
  HardFault_IRQn           = -13,/**<  3 Cortex-M0+ Hard Fault Interrupt
  SVCall_IRQn              = -5, /**< 11 Cortex-M0+ SV Call Interrupt
  PendSV_IRQn              = -2, /**< 14 Cortex-M0+ Pend SV Interrupt
  SysTick_IRQn             = -1, /**< 15 Cortex-M0+ System Tick Interrupt
  /******  SAMD21J18A-specific Interrupt Numbers ***********************/
  PM_IRQn                  =  0, /**<  0 SAMD21J18A Power Manager (PM) */
  SYSCTRL_IRQn             =  1, /**<  1 SAMD21J18A System Control (SYSCTRL) */
  WDT_IRQn                 =  2, /**<  2 SAMD21J18A Watchdog Timer (WDT) */
  RTC_IRQn                 =  3, /**<  3 SAMD21J18A Real-Time Counter (RTC) */
```

NVIC Specific Functions

```
/* Set the priority for an interrupt */
void NVIC_SetPriority(IRQn_t IRQn, uint32_t priority);

/* Enable a device specific interrupt */
void NVIC_EnableIRQ (IRQn_Type IRQn);

/* Disable a device specific interrupt */
void NVIC_DisableIRQ (IRQn_Type IRQn)
```



```
__STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
{
  NVIC->ISER[0] = (1 << ((uint32_t)(IRQn) & 0x1F));
}

__STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)
{
  NVIC->ICER[0] = (1 << ((uint32_t)(IRQn) & 0x1F));
}
```

Configuring Interrupts

http://microchipdeveloper.com/32arm:samd21-nvic-configuration