

# PSI3441 – Arquitetura de Sistemas Embarcados

---

## Interrupções

---

Escola Politécnica da Universidade de São Paulo

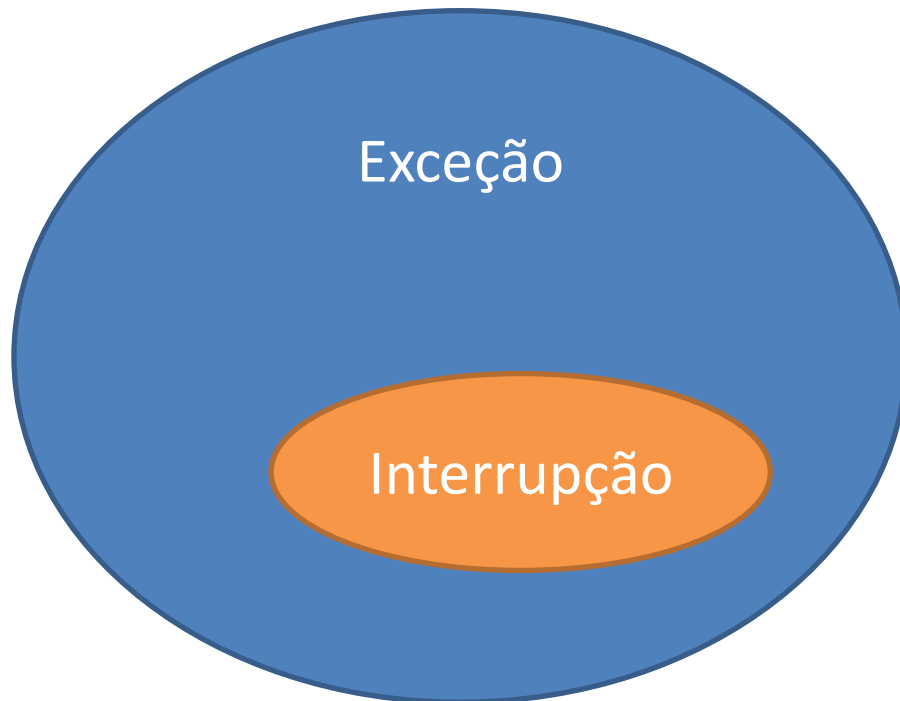
Prof. Gustavo Rehder – [grehder@lme.usp.br](mailto:grehder@lme.usp.br)





# Exceção – Interrupção

- Exceção: Qualquer evento que altera o fluxo normal de execução de instruções.



Exceção: eventos anormais:

- Instruções inválidas
- Acesso ilegal

Interrupção: Sinal de hardware

- Sinais externos
- Flags de periféricos



# Pooling vs. Interrupção

- Pooling

// Monitora continuamente o flag do registrador.

```
//No main
for (;;)
{
if (SysTick_PDD_ReadControlStatusReg(
SysTick_DEVICE) & 0x10000)
{
Bit1_NegVal();
}
}
```

- Interrupção

//Espera uma interrupção para executar a função.

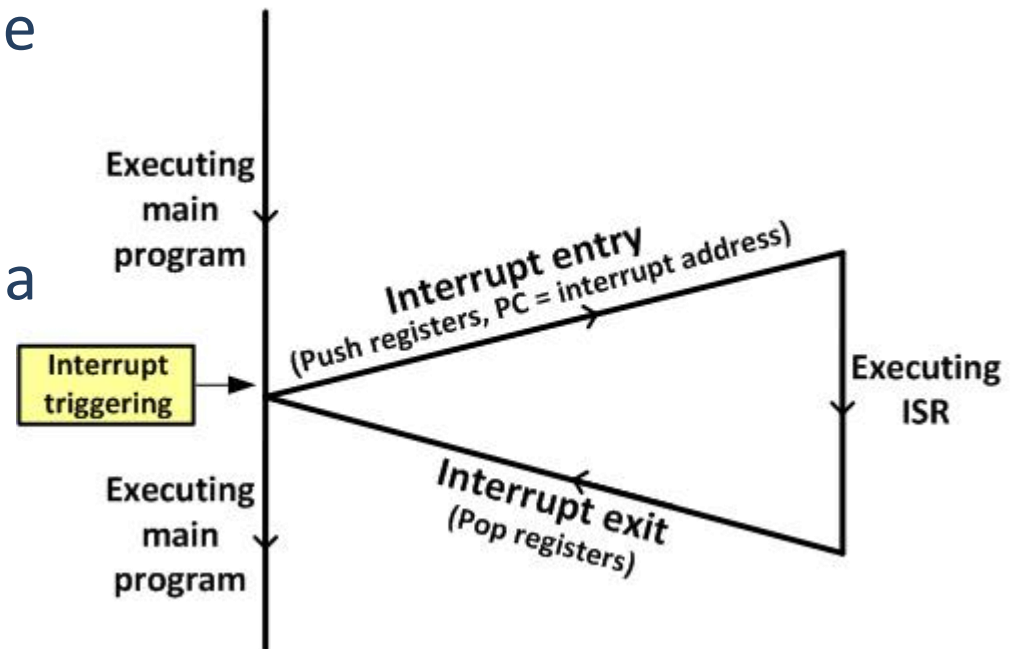
```
//No main
for (;;)
{
}

//No Events (Interrupt service routine (ISR))
void TI1_OnInterrupt(LDD_TUserData
*UserDataPtr)
{
Bit1_NegVal();
}
```



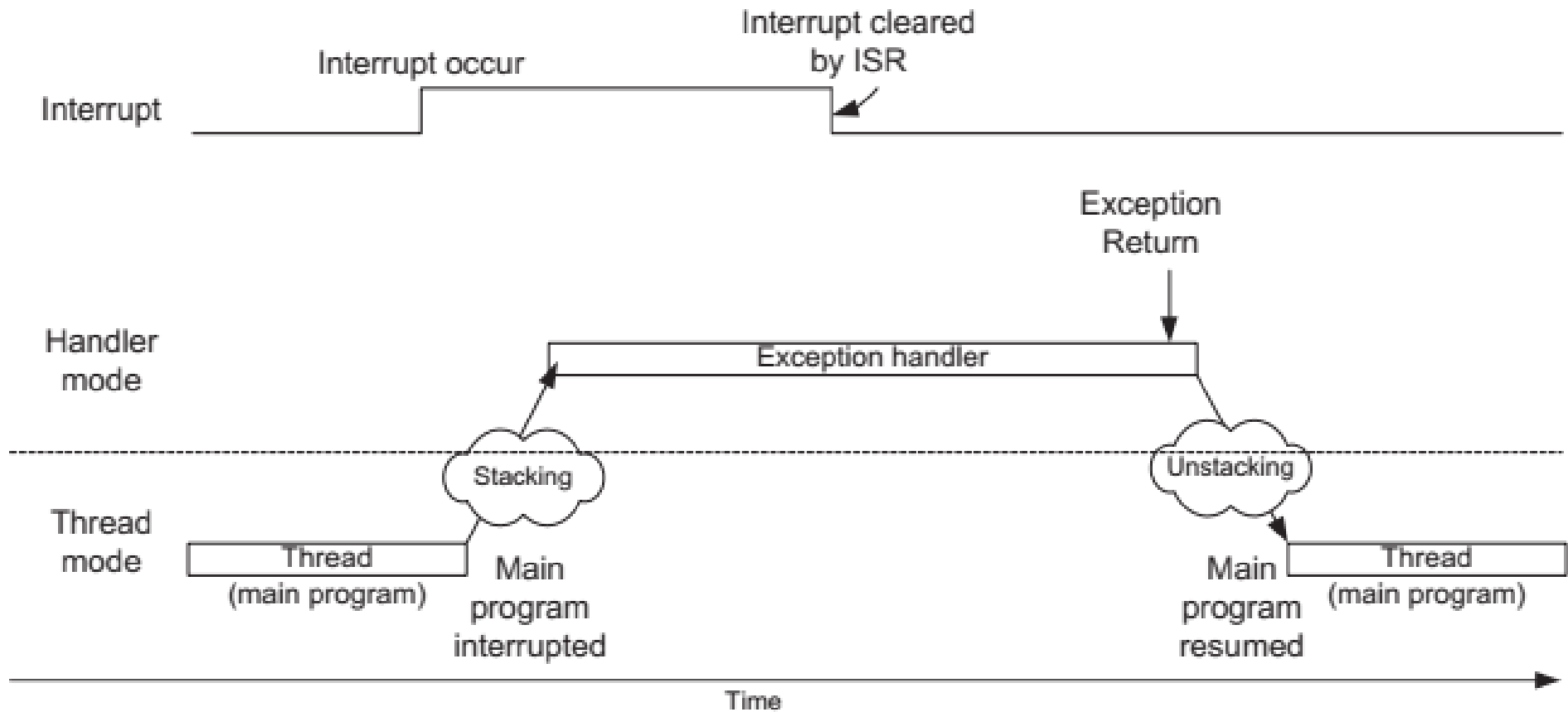
# O que acontece quando ocorre uma interrupção

- Periférico gera um pedido de interrupção (interrupt request – IRQ)
- O processador detecta e aceita o IRQ. O programa que está rodando é suspenso e a CPU salva no stack estado atual (contadores, registradores, flags, etc);
- CPU busca o endereço das rotinas de tratamento de interrupção (Interrupt Service Routine - ISR) em uma tabela (Vector Table);
- CPU vai para o endereço do ISR e executa seu código (função);
- CPU retorna ao estado anteriormente salvo e continua a execução do programa anterior.





# Exceção/Interrupção execução



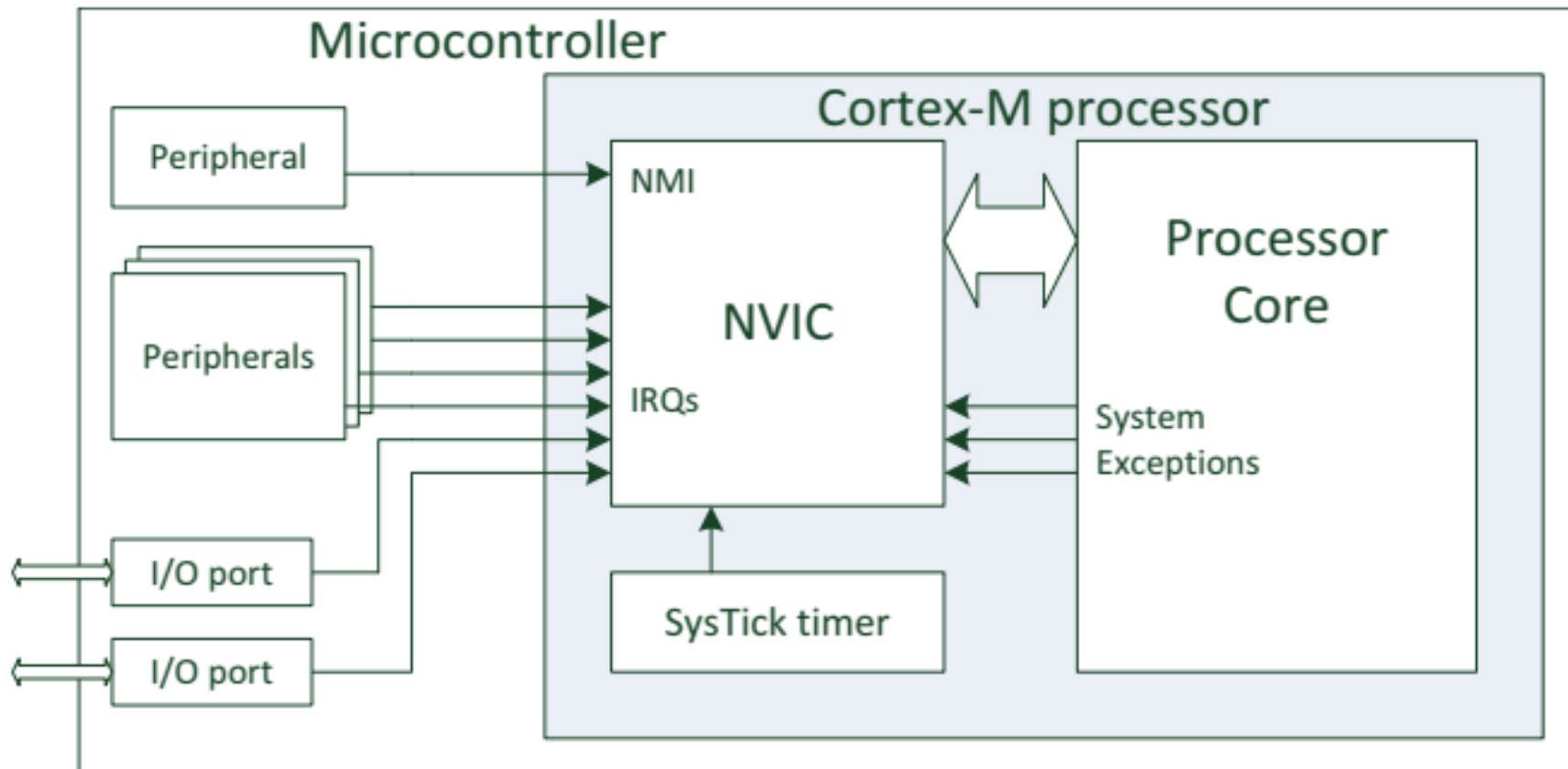


# Definições

- *Interrupt Requests (IRQs)* – Interrupções associadas a periféricos externos, ex. GPIOs. O Cortex M0+ possui 32 IRQs.
- *Non-Maskable Interrupt (NMI)* – IRQs de alta prioridade, ex. *Watchdog*.
- *Handlers* – função que é executada quando uma exceção ocorre. Caso seja uma interrupção também é conhecido como *interrupt handler* ou *Interrupt Service Routine (ISR)*.
- *Nested Interrupts* – por causa dos diferentes níveis de prioridade, uma interrupção pode ocorrer quando um *handler* está sendo executado. Isto é um *nested interrupt*.



# Nested Vectored Interrupt Controller (NVIC)



NVIC – Nested Vectored Interrupt Controller: Controla todas as interrupções externas



# Vector Table

**Non-Maskable Interrupt** – não pode ser desabilitado. Falta de energia e watchdog.

**HardFault** – instruções inválidas, erros no bus, na memória ou operações ilegais

**SVCcall** (Supervisor Call) – modo de supervisão usado em sistemas operacionais

**Pendable Service Call** – usado por sistemas operacionais

**System Tick Timer** – interrupção do timer

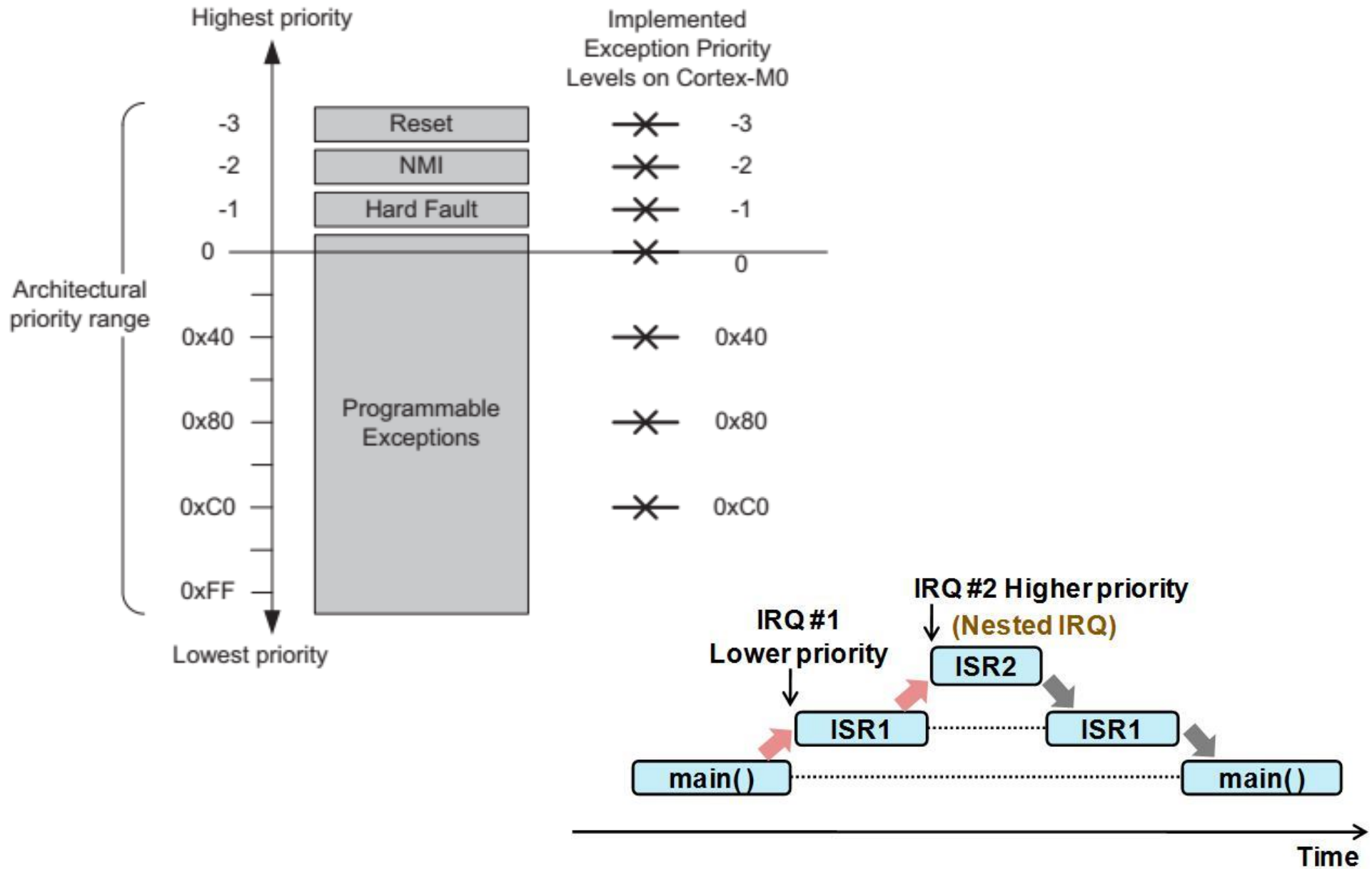
**Interrupts** – interrupção de outros periféricos (depende do fabricante)

Memory Address		Exception Number
0x0000004C	Interrupt#3 vector	19
0x00000048	Interrupt#2 vector	18
0x00000044	Interrupt#1 vector	17
0x00000040	Interrupt#0 vector	16
0x0000003C	SysTick vector	15
0x00000038	PendSV vector	14
0x00000034	Not used	13
0x00000030	Not used	12
0x0000002C	SVC vector	11
0x00000028	Not used	10
0x00000024	Not used	9
0x00000020	Not used	8
0x0000001C	Not used	7
0x00000018	Not used	6
0x00000014	Not used	5
0x00000010	Not used	4
0x0000000C	HardFault vector	3
0x00000008	NMI vector	2
0x00000004	Reset vector	1
0x00000000	MSP initial value	0





# Prioridade





# Vector Table

Core

SysTick

Address	Vector	IRQ <sup>1</sup>	NVIC IPR register number <sup>2</sup>	Source module	Source description
0x0000_0000	0	—	—	ARM core	Initial Stack Pointer
0x0000_0004	1	—	—	ARM core	Initial Program Counter
0x0000_0008	2	—	—	ARM core	Non-maskable Interrupt (NMI)
0x0000_000C	3	—	—	ARM core	Hard Fault
0x0000_0010	4	—	—	—	—
0x0000_0014	5	—	—	—	—
0x0000_0018	6	—	—	—	—
0x0000_001C	7	—	—	—	—
0x0000_0020	8	—	—	—	—
0x0000_0024	9	—	—	—	—
0x0000_0028	10	—	—	—	—
0x0000_002C	11	—	—	ARM core	Supervisor call (SVCall)
0x0000_0030	12	—	—	—	—
0x0000_0034	13	—	—	—	—
0x0000_0038	14	—	—	ARM core	Pendable request for system service (PendableSrvReq)
0x0000_003C	15	—	—	ARM core	System tick timer (SysTick)

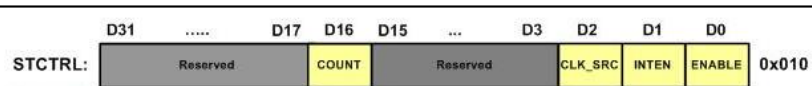
### Non-Core Vectors

0x0000_0040	16	0	0	DMA	DMA channel 0 transfer complete and error
0x0000_0044	17	1	0	DMA	DMA channel 1 transfer complete and error
0x0000_0048	18	2	0	DMA	DMA channel 2 transfer complete and error
0x0000_004C	19	3	0	DMA	DMA channel 3 transfer complete and error
0x0000_0050	20	4	1	—	—
0x0000_0054	21	5	1	FTFA	Command complete and read collision
0x0000_0058	22	6	1	PMC	Low-voltage detect, low-voltage warning
0x0000_005C	23	7	1	LLWU	Low Leakage Wakeup

⋮

0x0000_00A4	41	25	6	DAC0	
0x0000_00A8	42	26	6	TSIO	
00_00AAC	43	27	6	MCG	
00_00B0	44	28	7	LPTMR0	
00_00B4	45	29	7	—	
00_00B8	46	30	7	Port control module	Pin detect (Port A)
00_00BC	47	31	7	Port control module	Pin detect (Port D)

Timer



bit	Name	Description
0	ENABLE	Enable (0: the counter is disabled, 1: enables SysTick to begin counting down)
1	INTEN	Interrupt Enable 0: Interrupt generation is disabled, 1: when SysTick counts to 0 an interrupt is generated
2	CLK_SRC	Clock Source 0: System clock divided by 16 1: System clock
16	COUNT	Count Flag 0: the SysTick has not counted down to zero since the last time this bit was read 1: the SysTick has counted down to zero <i>Note: this flag is cleared by reading the STRCTL or writing to STCURRENT register.</i>

Figure 5-8: STCTRL (System Tick Control)

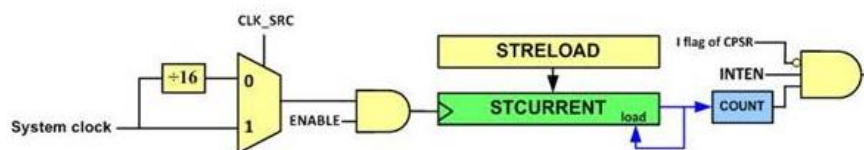


Figure 6-16: SysTick Internal Structure

# SysTick – Interrupt Handler

C/C++ - systick\_interrupt/Generated\_Code/Vectors.c - CodeWarrior Development Studio

File Edit Source Refactor Search Project MQX Tools Processor Expert Run Window Help

CodeWarrior Projects

File Name Build

- PE\_Error.h ✓
- PE\_LDD.c ✓
- PE\_LDD.h ✓
- PE\_Types.h ✓
- SysTick.c ✓
- SysTick.h ✓
- Vectors.c ✓
- ProcessorExpert.pe
- Project\_Headers
- bme.h ✓
- Project\_Settings
- ProjectInfo.xml
- Sources
- Events.c ✓

Components - systick\_interr...

- Generator\_Configurations
- FLASH
- OSs
- Processors
- Cpu:MKL25Z128VLK4
- Components
- SysTickInit\_SysTick
- myTick\_ISR Name of INT\_SysTick

Commander

- Project Creation
- Build

Component Inspector - SysTick

Properties Methods Clock Diagram

Name	Value	Details
Device	SysTick	SysTick
<b>Settings</b>		
Clock source	External clock	
Reload value	0	D
Counter period	STOP	
<b>Interrupts</b>		
Interrupt	INT_SysTick	INT_SysTick
Interrupt priority	0 (Highest)	
ISR Name	myTick	myTick

Events.c main.c Vectors.c main.c Vectors.c

```

(tIsrFunc) &Cpu_Interrupt, /* 0x06 0x00000018 - ivINT_Reserved6 unused by PE
(tIsrFunc) &Cpu_Interrupt, /* 0x07 0x0000001C - ivINT_Reserved7 unused by PE
(tIsrFunc) &Cpu_Interrupt, /* 0x08 0x00000020 - ivINT_Reserved8 unused by PE
(tIsrFunc) &Cpu_Interrupt, /* 0x09 0x00000024 - ivINT_Reserved9 unused by PE
(tIsrFunc) &Cpu_Interrupt, /* 0x0A 0x00000028 - ivINT_Reserved10 unused by PE
(tIsrFunc) &Cpu_Interrupt, /* 0x0B 0x0000002C - ivINT_SVCall unused by PE
(tIsrFunc) &Cpu_Interrupt, /* 0x0C 0x00000030 - ivINT_Reserved12 unused by PE
(tIsrFunc) &Cpu_Interrupt, /* 0x0D 0x00000034 - ivINT_Reserved13 unused by PE
(tIsrFunc) &Cpu_Interrupt, /* 0x0E 0x00000038 - ivINT_PendableSrvReq unused by PE
(tIsrFunc) &myTick, /* 0x0F 0x0000003C 0 ivINT_SysTick used by PE */
(tIsrFunc) &Cpu_Interrupt, /* 0x10 0x00000040 - ivINT_DMA0 unused by PE
(tIsrFunc) &Cpu_Interrupt, /* 0x11 0x00000044 - ivINT_DMA1 unused by PE
    
```

Problems Console Memory

0 items

Description	Resource	Path	Location	Type

Writable Smart Insert 92 : 1



# referência

- Yiu, Joseph. *The Definitive Guide to Arm<sup>®</sup> Cortex<sup>®</sup>-m0 and Cortex-m0+ Processors*. Academic Press, 2015.