

# ELF52 - Sistemas Microcontrolados

## Interrupções

**Professor:**

Prof. Marcos Eduardo

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

# Interrupções

# Interrupções

- Problema da latência:
  - Tempo que o microcontrolador demora para acessar um periférico que já disponibilizou um dado útil.
- Formas de acesso aos periféricos:
  - Ciclo cego (*blind cycle*): estimar o tempo e tentar ler no tempo certo;
  - *Busy wait* ou *polling*: leitura constante;
  - *Interrupção*: a execução é interrompida quando o dado está disponível;
  - DMA (*direct memory access*): dado copiado automaticamente.

# Interrupções

- O que é uma interrupção?



# Interrupções

- Definição:
  - Qualquer evento **interno** ou **externo** que obriga o microcontrolador a suspender o que está fazendo para atender o evento que o interrompeu.
- Para que serve?
  - Executar uma tarefa de **prioridade mais alta**.
- Funcionamento:
  - O programa é desviado para um outro ponto da memória de programa onde se encontra a rotina de atendimento à interrupção (como uma subrotina);
  - Após executar a rotina, o microcontrolador **volta ao ponto imediatamente seguinte** de onde foi interrompido.

# Interrupções

- Qual a diferença entre um **BL** e uma **interrupção**?
- O **BL** é uma instrução programada no software para acontecer em um momento específico. Já a interrupção, pode acontecer a qualquer momento.

# Interrupções

- Qual a diferença entre um **BL** e uma **interrupção**?
- O **BL** é uma instrução programada no software para acontecer em um momento específico. Já a interrupção, pode acontecer a qualquer momento.

# Interrupções

- Aspectos fundamentais:
  - Há uma chave geral para ativar TODAS as interrupções (bit **I**, do registrador **PRIMASK**): 0 para habilitar, 1 para desabilitar;
  - O controlador de interrupções separa cada um dos periféricos em fontes de interrupção;
  - Existe uma fonte interrupção independente para cada periférico disponível no microcontrolador PortA, PortB, ..., PortQ, Timers, UARTs, I2Cs, SSIs, etc;
  - Cada fonte de interrupção pode ser ativada ou desativada independentemente por software.



# Interrupções

- Cada interrupção pode ter uma prioridade, em que o registrador **BASEPRI** previne interrupções com prioridades menores:
  - Exemplo: Se **BASEPRI** estiver configurado para 3, somente pedidos de interrupção com prioridade 0, 1 ou 2 serão atendidos, enquanto que maior ou igual a 3 serão postergados;
  - Se **BASEPRI** estiver configurado em 0, todas as prioridades são permitidas.

## Interrupções

- As interrupções são controladas pelo **NVIC** (*Nested Vectored Interrupt Controller*);
- Descrito na seção 3.4 Datasheet;
- Cada interrupção tem um bit separado para armá-la, para que o *software* possa ativá-la ou desativá-la;
- Cada fonte de interrupção tem um ID ( número da interrupção ) associado (Tabela 2 9 da página 116 datasheet);
- Para habilitar uma interrupção em um periférico, é necessário habilitar a sua fonte e configurar sua prioridade precisa ser configurada a prioridade e habilitada a fonte no **NVIC**.

## Interrupções (condições)

- Para uma interrupção acontecer → 5 condições:
  - 1 Bit **I** no registrador **PRIMASK** deve ser 0;
  - 2 Habilite a fonte no *Nested Vectored Interrupt Controller* (NVIC) (prioridade e *enable*);
  - 3 O nível da interrupção (no NVIC) deve ser menor que o **BASEPRI**;
  - 4 Armar a interrupção do periférico no *interrupt mask register* (IMR) (por bit e no registrador do periférico);
  - 5 Evento externo da interrupção deve acontecer.

## Interrupções (Tratamento)

- Quando uma interrupção acontecer, a execução do código principal é interrompida e o programa vai para a rotina de tratamento de interrupção (ISR);
- A rotina de tratamento de interrupção deve ser executada **o mais rápido possível** (evitar laços e iterações).
- Deve-se sair dela utilizando a instrução no assembly **BX LR** ou **return** em C;
- Ao ser tratada um interrupção, o seu *flag* de disparo deve ser limpado dentro da rotina de tratamento (ISR), para que outra interrupção do mesmo tipo possa ocorrer (como um ACK);

## Interrupções no Cortex-M

- Cada fonte de interrupção (assim como as exceções) está associada a uma posição na memória ROM de 32 bits chamada de vetor;
- Cada vetor deste, aponta para uma função de tratamento de interrupção: o endereço da ISR é escrito nestas posições da memória ROM;
- Esses vetores estão no início da memória ROM, normalmente no arquivo `startup.s`;
- Há até 240 possibilidades de fontes de interrupção que são listados a partir do endereço `0x0000.0040` (antes disso são as exceções).

## Interrupções no Cortex-M

- Algumas posições das exceções (Extraído do startup.s):

```
1      EXPORT  __Vectors
2  __Vectors ; address ISR
3  DCD StackMem + Stack; 0x00000000 Top of Stack
4  DCD Reset_Handler ; 0x00000004 Reset Handler
5  DCD NMI_Handler ; 0x00000008 NMI Handler
6  DCD HardFault_Handler; 0x0000000C Hard Fault Handler
7  DCD MemManage_Handler; 0x00000010 MPU Fault Handler
8  DCD BusFault_Handler ; 0x00000014 Bus Fault Handler
9  DCD UsageFault_Handler; 0x00000018 Usage Fault Handler
10 ...
```

## Interrupções no Cortex-M

- Algumas posições das exceções (Extraído do startup.s):

```
1  ...
2  DCD SVC_Handler ; 0x0000002C SVCcall Handler
3  DCD DebugMon_Handler ; 0x00000030 Debug Monitor
   Handler
4  DCD 0 ; 0x00000034 Reserved
5  DCD PendSV_Handler ; 0x00000038 PendSV Handler
6  DCD SysTick_Handler ; 0x0000003C SysTick Handler
7  DCD GPIOPortA_Handler ; 0x00000040 GPIO Port A
8  DCD GPIOPortB_Handler ; 0x00000044 GPIO Port B
9  DCD GPIOPortC_Handler ; 0x00000048 GPIO Port C
10 DCD GPIOPortD_Handler ; 0x0000004C GPIO Port D
11 DCD GPIOPortE_Handler ; 0x00000050 GPIO Port E
12 ...
```

## Interrupções no Cortex-M

- A rotina de tratamento da interrupção pode ser declarada em qualquer arquivo e em qualquer lugar do código;
- Declarar um label com o mesmo nome do vetor, para declarar a função de tratamento da interrupção (ISR);
- Lembrar de fazer o EXPORT desta função no arquivo que for feita a declaração.



## Interrupções no Cortex-M

- Uma **interrupção aninhada** (*nested interrupt*) acontece quando uma interrupção de maior prioridade suspende uma ISR;
- A **prioridade** define a ordem de execução se duas interrupções acontecerem ao mesmo tempo. Ela define também se uma interrupção pode suspender outra de menor prioridade.

# Interrupções

- Se:
  - O flag de interrupção estiver ativado, mas as interrupções estiverem desabilitadas ( $I=1$ );
  - O nível de prioridade não é alto o suficiente;
  - A fonte não está habilitada.
- O pedido não é perdido, ele vai para fila de **interrupções pendentes**.

## Interrupções (Troca de Contexto)

- O que acontece quando ocorre uma interrupção?
  - 1 A instrução em execução é terminada;
  - 2 A execução do programa corrente é suspensa (entrando no modo de exceção ou modo *handler*) empilhando 8 registradores na pilha (R0, R1, R2, R3, R12, LR, PC e PSR com R0 no topo);
  - 3 LR é setado para um valor específico significando que uma rotina de serviço de interrupção (ISR) está sendo tratada (bits [31:8] para 0xFFFFFFFF e bits [7:1] especificam o tipo de interrupção, bit 0 sempre será 1);
  - 4 IPSR é setado para o número da interrupção processada;
  - 5 PC é carregado com o endereço do ISR.

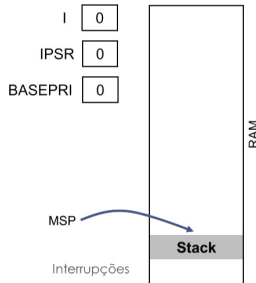
## Modos de Execução

- 2 Modos de execução do microcontrolador:
  - Modo **Thread**:
    - Programa principal;
    - Entra em execução depois do *reset*;
    - Identificável quando IPSR é 0.
  - Modo **Handler**:
    - Entra em execução quando acontece uma interrupção;
    - Identificável quando o IPSR é diferente de 0.

## Exemplo de Interrupção

- Interrupção gerada por borda na porta C:
  - Supor que a interrupção da Porta C seja configurada como prioridade nível 2.

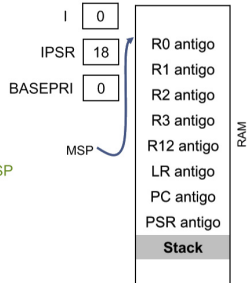
Antes da interrupção



### Troca de contexto

- Termina a instrução
  - Empilha os registradores (R0, R1, R2, R3, R12, LR, PC e PSR)
  - PC={0x00000048}
  - Seta IPSR=18
  - Seta LR=0xFFFFFFF9
- F9→ Ao retornar, volta para o modo normal (*thread*) e usa o MSP como *stack pointer*

Depois da interrupção



## Interrupções no Cortex-M

- Para ligar (padrão) e desligar a chave geral das interrupções há duas funções já declaradas no arquivo startup.s. Basta chamá-las do código;
- Para desligar:

```
1 DisableInterrupts    CPSID I    ;set I
2                      BX         LR
```

- Para ligar:

```
1 EnableInterrupts    CPSIE I    ;disable I
2                      BX         LR
```

## Registradores do NVIC

## Registradores do NVIC

- As interrupções são controladas pelo **NVIC** (*Nested Vectored Interrupt Controller*);
- Seção 3.4 do *Datasheet*;
- Para habilitar uma fonte de interrupção precisa ser **configurada a prioridade** e **habilitada a fonte** no NVIC. (Tabela 2-9 da página 166 do *Datasheet* mostra todas as fontes de interrupção).



# Registadores do NVIC

## 1 Interrupt Set Enable: ENx

- Os registadores de **enable** controlam a habilitação das fontes de interrupção. Cada bit habilita uma interrupção, que cujo número correspondente pode ser encontrado na tabela 2-9 da página 116 do *datasheet*. **OBS: Para desabilitar utiliza-se outro registrador.**

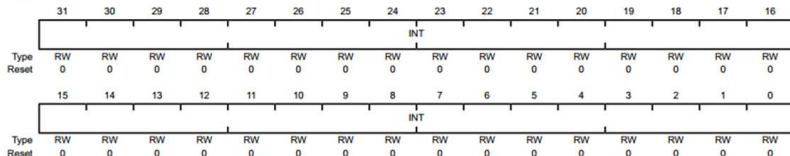
### Exemplo:

Interrupt 0-31 Set Enable (EN0)

Base 0xE000.E000

Offset 0x100

Type RW, reset 0x0000.0000



Address	31	30	29-7	6	5	4	3	2	1	0	Name
0xE000E100	G	F	...	UART1	UART0	E	D	C	B	A	NVIC_EN0_R
0xE000E104			...						UART2	H	NVIC_EN1_R

## Registradores do NVIC

1 Interrupt Set Enable: **ENx**

Registrador	Interrupções	Endereço
EN0	0 a 31	0xE000.E100
EN1	32 a 63	0xE000.E104
EN2	64 a 95	0xE000.E108
EN3	96 a 113	0xE000.E10C

## Registradores do NVIC

### 2 Interrupt Clear Enable: DISx

- Os registradores de *disable* controlam o desligamento das fontes de interrupção. Cada bit desabilita uma interrupção, que cujo número correspondente pode ser encontrado na tabela 2-9 da página 116 do *datasheet*.

OBS: Para habilitar utiliza-se o registrador anterior.

- Exemplo:

Interrupt 0-31 Clear Enable (DIS0)

Base 0xE000.E000

Offset 0x180

Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INT															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	INT															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## Registradores do NVIC

② Interrupt Clear Enable: DISx

Registrador	Interrupções	Endereço
DIS0	0 a 31	0xE000.E180
DIS1	32 a 63	0xE000.E184
DIS2	64 a 95	0xE000.E188
DIS3	96 a 113	0xE000.E18C

## Registadores do NVIC

### 3 Interrupt Priority: **PRIx**

- Os registadores de prioridade configuram a prioridade de 4 em 4 fontes de interrupção (8 bits para cada fonte, mas **somente os bits entre 5 a 7 são usados**). Algumas delas são:

Address	31 – 29	23 – 21	15 – 13	7 – 5	Name
0xE000E400	GPIO Port D	GPIO Port C	GPIO Port B	GPIO Port A	NVIC_PRI0 R
0xE000E404	SSI0, Rx Tx	UART1, Rx Tx	UART0, Rx Tx	GPIO Port E	NVIC_PRI1 R
0xE000E408	PWM Gen 1	PWM Gen 0	PWM Fault	I2C0	NVIC_PRI2 R
0xE000E40C	ADC Seq 1	ADC Seq 0	Quad Encoder	PWM Gen 2	NVIC_PRI3 R
0xE000E410	Timer 0A	Watchdog	ADC Seq 3	ADC Seq 2	NVIC_PRI4 R
0xE000E414	Timer 2A	Timer 1B	Timer 1A	Timer 0B	NVIC_PRI5 R
0xE000E418	Comp 2	Comp 1	Comp 0	Timer 2B	NVIC_PRI6 R
0xE000E41C	GPIO Port G	GPIO Port F	Flash Control	System Control	NVIC_PRI7 R
0xE000E420	Timer 3A	SSI1, Rx Tx	UART2, Rx Tx	GPIO Port H	NVIC_PRI8 R
0xE000E424	CAN0	Quad Encoder 1	I2C1	Timer 3B	NVIC_PRI9 R
0xE000E428	Hibernate	Ethernet	CAN2	CAN1	NVIC_PRI10 R
0xE000E42C	uDMA Error	uDMA Soft Tfr	PWM Gen 3	USB0	NVIC_PRI11 R
0xE000ED20	SysTick	PendSV	--	Debug	NVIC_SYS_PRI3 R

## Registradores do NVIC

### 3 Interrupt Priority: **PRIx**

Registrador	Interrupções	Endereço
PRI0	0 a 3	0xE000.E400
PRI1	4 a 7	0xE000.E404
PRI2	8 a 11	0xE000.E408
...	...	...
PRI26	104 a 107	0xE000.E468
PRI27	108 a 111	0xE000.E46C
PRI28	112 e 113	0xE000.E470

## Registradores do NVIC

- Além de configurar no NVIC, cada periférico possui sua própria configuração para habilitar a interrupção.

# Interrupções de GPIO



## Interrupções nos GPIO

- Utilizadas para reconhecer quando um hardware altera o estado de 1 para 0 ou 0 para 1;
- Há dois tipos:
  - Por borda;
  - Por nível.
- Qual a diferença entre esperar por *polling* e interrupção?

## Interrupções nos GPIO

- Além dos registradores do NVIC, os seguintes registradores do GPIO também realizam o controle das interrupções a nível de **periférico**.
- Similarmente a outros registradores do GPIO, cada bit controla um pino do port.

## Interrupções nos GPIO

- Para configurar as interrupções nos GPIO, além dos NVIC, os seguintes registradores controlam:
  - **GPIOIS** (*Interrupt Sense*): Borda ou nível;
  - **GPIOIBE** (*Interrupt Both Edges*): Uma borda apenas ou ambas as bordas;
  - **GPIOIEV** (*Interrupt Event*): Borda de subida ou borda de descida, nível alto ou nível baixo;
  - **GPIOIM** (*Interrupt Mask*): Habilita a interrupção;
  - **GPIORIS** (*Raw Interrupt Status*): Indica se as condições para a interrupção aconteceram mesmo se não está habilitada no GPIOIM;
  - **GPIOMIS** (*Masked Interrupt Mask*): Indica que as condições engatilharam uma interrupção no periférico. Neste caso, ela está habilitada no GPIOIM;
  - **GPIOICR** (*Interrupt Clear Register*): Ao setar o bit, realiza a limpeza do GPIORIS e GPIOMIS, (ACK da interrupção) permitindo uma nova interrupção.

## Configuração

- Para configurar se a interrupção é por borda ou nível e qual(is) da(s) borda(s) utiliza-se os seguintes registradores:

GPIOS	GPIOIBE	GPIOIEV	Modo
0	0	0	Borda de descida
0	0	1	Borda de subida
0	1	-	Ambas as bordas
1	0	0	Nível Baixo
1	0	1	Nível alto

## Configuração

- Para habilitar/desabilitar a interrupção utilizar o registrador **GPIOIM**, setar o bit do respectivo pino para habilitar a interrupção e limpar para desabilitar a interrupção;
- Quando uma condição de interrupção acontece, o sinal do estado da interrupção pode ser visto em dois locais:
  - **GPIORIS**: As condições foram atendidas, mas a interrupção não foi necessariamente enviada para o controlador de interrupções;
  - **GIOMIS**: Mostra somente as condições que são permitidas ser passadas para o controlador de interrupções.

## Configuração

- Quando o **GPIOICR** é setado, o **GPIORIS** e o **GPIOMIS** são limpos. Estes últimos são READONLY, ou seja, não podem ser escritos, somente lidos. Este processo é o *Acknowledgement*.
- Uma interrupção irá acontecer quando:
  - O bit no **GPIOIM** estiver setado (pino da porta escutando int.);
  - O nível da interrupção do GPIO for menor que o **BASEPRI**;
  - A interrupção do GPIO for habilitada no **NVIC\_EN**;
  - O bit 0 do registrador **PRIMASK** for 0.
  - O flag no **GPIORIS** estiver setado (evento ocorreu);

## Configuração

- Sequência para habilitar uma interrupção no GPIO (além da inicialização do GPIO):
  - 1 Desabilite a interrupção no registrador GPIOIM (`GPIO_PORTx_AHB_IM_R`);
  - 2 Configure o tipo de interrupção (0=borda, 1=nível) no registrador GPIOIS (`GPIO_PORTx_AHB_IS_R`);
  - 3 Configure GPIOIBE (0=borda única, 1=borda dupla) (`GPIO_PORTx_AHB_IBE_R`), GPIOIEV (0= nível baixo ou borda de descida, 1=nível alto ou borda de subida) (`GPIO_PORTx_AHB_IEV_R`);
  - 4 Limpe o registrador GPIORIS, setando o bit no (`GPIO_PORTx_AHB_ICR_R`);
  - 5 Habilite a interrupção no registrador GPIOIM (`GPIO_PORTx_AHB_IM_R`);
  - 6 Sete a prioridade no NVIC (`NVIC_PRIyy_R`);
  - 7 Habilite a interrupção no NVIC (`NVIC_ENz_R`);
  - 8 Habilite todas as interrupções (chave geral).

## Tratamento da Interrupção

- A função de tratamento da interrupção (**ISR**) pode ser declarada em qualquer arquivo, no entanto deve ter o mesmo nome que as declarações no arquivo startup.s. Não esquecer do **EXPORT**;
- Ela deve escrever no registrador **GPIOICR**. Isto é o *acknowledgement* para limpar o **GPPIORIS** e o **GPPIOMIS**;
- Como as portas têm mais de um pino, se houver interrupções em mais de um pino na porta, verifique através do registrador **GPPIORIS** para saber de qual pino é a interrupção e depois realize o tratamento para a interrupção naquele pino.



## Interrupções no Cortex-M

- Exemplo de tratamento de interrupção de GPIO para o pino PC4 (pode ser declarado em qualquer arquivo desde que faça o **EXPORT**):

```
1 GPIOPortC_Handler
2   LDR R1, =GPIO_PORTC_ICR_R
3   MOV R0, #0x10      ;bit 4 do PortC 00010000b
4   STR R0, [R1]       ;limpando a interrupção (ack)
5   ;
6   ; código qualquer
7   ;
8   BX LR              ;retorno
```

## Exemplo

## Exemplo

- Programar uma interrupção por borda de descida na USR\_SW1 (**PJ0**) e uma interrupção por borda de subida na USR\_SW2 (**PJ1**). Para que quando o usuário pressione a chave USR\_SW1 acenda o LED1 (**PN1**) e quando soltar a chave USR\_SW2 apague o LED1.

## Exemplo

- A seguir será demonstrado o passo-a-passo para a configuração da interrupção no GPIO Port J;
- Antes de configurar a interrupção, deve-se realizar a configuração dos GPIOs conforme a aula de GPIOs.

## Exemplo (passo-a-passo)

❶ Configurar os ports **J** e **N**, conforme a aula de GPIO.

- **J0** e **J1** → Entrada;
- **N1** → Saída.

## Exemplo (passo-a-passo)

- Antes de configurar as interrupções, devemos desabilitar (para depois habilitar novamente ao final) no registrador **GPIOIM**. Como vamos usar os pinos **J0** e **J1**, desabilitar os dois bits:

GPIO\_PORTJ\_AHB\_IM\_R

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DMA/ME		R/E					
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
															0	0
															J1	J0

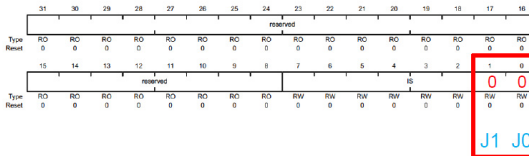
Zerar estes bits

Pag 764

## Exemplo (passo-a-passo)

- 2 Como vamos capturar a interrupção durante o pressionamento ou liberação das chaves, a interrupção deve ser configurada como borda em ambos os pinos no registrador **GPIOIS**:

GPIO\_PORTJ\_AHB\_IS\_R



Zerar estes bits

Pag 761

## Exemplo (passo-a-passo)

- 3 a) Configurar borda única em ambos pinos no registrador **GPIOIBE**:

GPIO\_PORTJ\_AHB\_IBE\_R

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								IS							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
															0	0
															J1	J0

Zerar estes bits

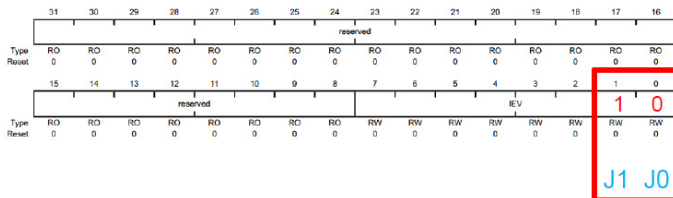
Pag 762



## Exemplo (passo-a-passo)

- 3 b) Configurar borda de descida para J0 e borda de subida para J1 no registrador **GPIOIEV**:

GPIO\_PORTJ\_AHB\_IEV\_R

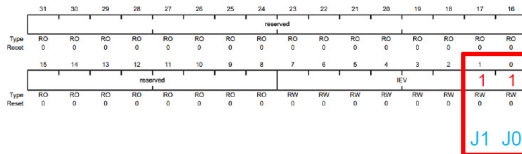


Pag 763

## Exemplo (passo-a-passo)

- Garantir que a interrupção será atendida limpando o GPIORIS e GPIOMIS, realizando o ACK no registrador GPIOICR para ambos os pinos (J1 e J0). Setar os bits:

GPIO\_PORTJ\_AHB\_ICR\_R



Setar estes bits

J1 J0

Pag 769

## Exemplo (passo-a-passo)

- 5 Ativar a interrupção em ambos os pinos do Port J, no registrador  
GPIOIM:

GPIO\_PORTJ\_AHB\_IM\_R

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				DMAIMR				IMR				J1		J0	
Type	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Setar estes bits

J1 J0

Pag 764

## Exemplo (passo-a-passo)

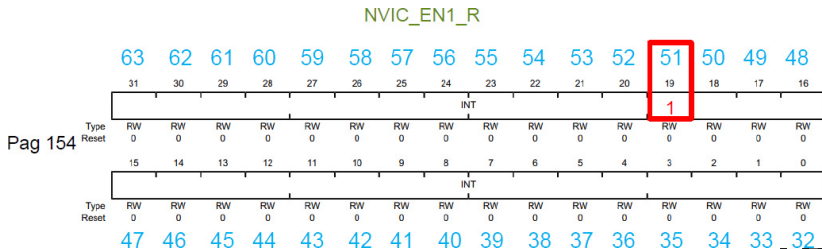
- 6 a) Ativar a fonte de interrupção no NVIC. Primeiramente consultar a Tabela 2-9 que começa na página 116 do datasheet, para saber o número da interrupção do Port J. A fonte do Port J é a número **51**:

**Table 2-9. Interrupts (continued)**

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
62	46	0x0000.00F8	ADC1 Sequence 0
63	47	0x0000.00FC	ADC1 Sequence 1
64	48	0x0000.0100	ADC1 Sequence 2
65	49	0x0000.0104	ADC1 Sequence 3
66	50	0x0000.0108	EPI 0
67	51	0x0000.010C	GPIO Port J
68	52	0x0000.0110	GPIO Port K
69	53	0x0000.0114	GPIO Port L
70	54	0x0000.0118	SSI 2
71	55	0x0000.011C	SSI 3

## Exemplo (passo-a-passo)

- 6 b) Ativar a fonte de interrupção no NVIC. Sabendo que o número da fonte de interrupção é o 51, encontrar qual ENx habilitará a fonte de interrupção do Port J. Da tabela 3-8 (página 146) encontra-se que o **EN1** habilita as interrupções 32 a 61. Setar o bit deste registrador que habilita a interrupção, no bit **19**.



7 Configurar a prioridade da fonte de interrupção no NVIC. Sabendo que o número da fonte de interrupção é o 51, encontrar qual PRIx configura a prioridade da fonte de interrupção do Port J. Da tabela 38 (página 146), encontra-se que o **PRI12** configura a prioridade das interrupções 48 a 51. Configurar os bits 29 a 31, com a prioridade desejada. Vamos supor que queremos setar a prioridade 5.



MOV Rx, #5

LSL Rx, #29

## Exemplo (passo-a-passo)

- ISR:
  - Verificar no arquivo startup.s qual é a função de tratamento de interrupção;
  - No caso é a **GPIOPortJ\_Handler**;
  - Declarar o label **GPIOPortJ\_Handler** em qualquer arquivo que não seja o startup.s, por exemplo gpio.s;
  - Primeiramente realizar um teste para saber qual interrupção foi gerada, se foi causada pelo J0 ou pelo J1, por meio do **GPJORIS**;
  - Se o bit 0 estiver setado, foi causada pelo J0. Se o bit 1 estiver setado, foi causada pelo J1;
  - Realizar o ACK da interrupção escrevendo 1 no bit respectivo do registrador **GPIOICR**;
  - Escrever no pino para acender ou apagar o LED;
  - Sair da interrupção com **BX LR**.

# Dúvidas?