

O Supremo STM32|Interrupções

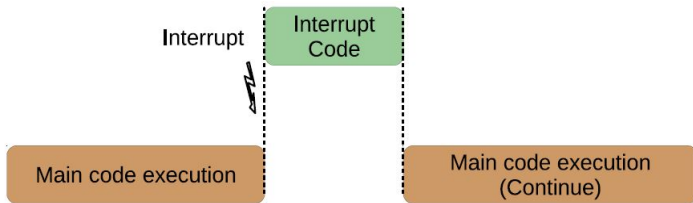
Luiz Fernando Pinto de Oliveira

7 de julho de 2018

■ Características:

- ▶ 43 canais de interrupção de máscara (não incluindo as 16 linhas de interrupção do Cortex-M3);
- ▶ 16 níveis de prioridade programáveis;
- ▶ Exceção de baixa latência e manipulação de interrupções;
- ▶ *Interrupt tail-chaining*.

- Quando configurada e ativada, uma interrupção força o microcontrolador a interromper a execução normal do código e executar a rotina especificada para essa finalidade;
- As interrupções são uma exceção à operação de código normal, portanto, deve ser uma rotina o mais curta possível;



Interrupções - Fontes de interrupção do STM32

Position	Priority	Acronym	Description	Address
-	-	-	Reserved	0x0000_0000
-3	-	Reset	Reset	0x0000_0004
-2	-	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
-1	-	HardFault	All class of fault	0x0000_000C
0	0	MemManage	Memory management	0x0000_0010
1	1	BusFault	Pre-fetch fault, memory access fault	0x0000_0014
2	2	UsageFault	Undefined instruction or illegal state	0x0000_0018
-	-	-	Reserved	0x0000_001C 0x0000_002B
3	3	SVCall	System service call via SWI instruction	0x0000_002C
4	4	Debug Monitor	Debug Monitor	0x0000_0030
-	-	-	Reserved	0x0000_0034
5	5	PendSV	Pendable request for system service	0x0000_0038
6	6	SysTick	System tick timer	0x0000_003C

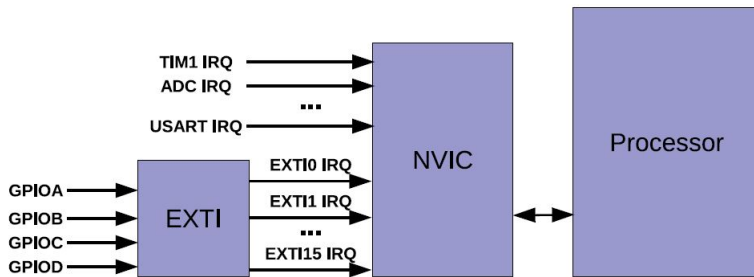
Interrupções - Fontes de interrupção do STM32

Position	Priority	Acronym	Description	Address
0	7	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	TAMPER	Tamper interrupt	0x0000_0048
3	10	RTC	RTC global interrupt	0x0000_004C
4	11	FLASH	Flash global interrupt	0x0000_0050
5	12	RCC	RCC global interrupt	0x0000_0054
6	13	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088
19	26	CAN1_TX	CAN1 TX interrupts	0x0000_008C
20	27	CAN1_RX0	CAN1 RX0 interrupts	0x0000_0090
21	28	CAN1_RX1	CAN1 RX1 interrupt	0x0000_0094
22	29	CAN1_SCE	CAN1 SCE interrupt	0x0000_0098
23	30	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	TIM1_BRK	TIM1 Break interrupt	0x0000_00A0
25	32	TIM1_UP	TIM1 Update interrupt	0x0000_00A4

Interrupções - Fontes de interrupção do STM32

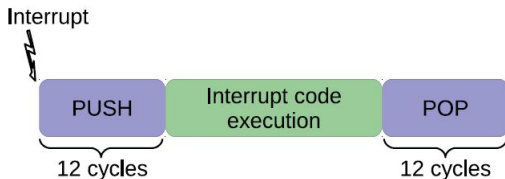
Position	Priority	Acronym	Description	Address
26	33	TIM1_TRG_COM	TIM1 Trigger and Commutation interrupts	0x0000_00A8
27	34	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00AC
28	35	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	I2C1_EV	I2C1 error interrupt event interrupt	0x0000_00BC
32	39	I2C1_ER	I2C1 error interrupt	0x0000_00C0
33	40	I2C2_EV	I2C2 event interrupt	0x0000_00C4
34	41	I2C2_ER	I2C2 error interrupt	0x0000_00C8
35	42	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	USART1	USART1 global interrupt	0x0000_00D4
38	45	USART2	USART2 global interrupt	0x0000_00D8
39	46	USART3	USART3 global interrupt	0x0000_00DC
40	47	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
42	49	OTG_FS_WKUP	USB On-The-Go FS Wakeup through EXTI line interrupt	0x0000_00E8

Interrupções - Diagrama de blocos de interrupção



- NVIC - *Nested Vector Interrupt Controller*;
- EXTI - *External Interrupt/Event Controller*.

- Antes da execução do código de interrupção, existe uma fase que armazena os registros na pilha (PUSH);
- Após a execução do código de interrupção, os registros armazenados anteriormente na pilha são recuperados (POP);
- Em ambas as fases, PUSH e POP, o microcontrolador leva 12 ciclos de *clock*;
- As fases PUSH e POP são feitas por hardware (não há necessidade de fazer isso no código C ou ASM);
- Os registradores que são salvos na pilha durante a fase PUSH são:



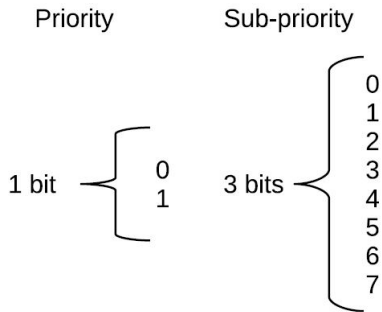
- Todas as interrupções têm uma prioridade associada atribuída por um número:
 - ▶ Um número menor significa uma prioridade mais alta;
 - ▶ Um número mais alto significa uma prioridade mais baixa.
- Se não houver configuração, significa que todas as prioridades têm o número 0;
- Existem interrupções nas quais não é possível alterar as prioridades porque elas são fixas (Reset, Hardfault, NMI).

- Há sempre dois campos na configuração de prioridades:
 - ▶ Campo que define as prioridades do grupo;
 - ▶ Campo que define as sub-prioridades dentro do grupo.
- A preempção de interrupção é determinada apenas pelo grupo prioritário;
- Quando há várias interrupções com a mesma prioridade, a sequência de execução é feita seguindo o grupo de sub-prioridade;
- Quando as interrupções têm a mesma prioridade e sub-prioridade, executa o IRQ mais baixo.

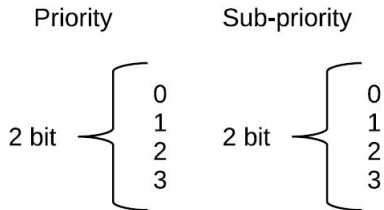
- Existem 5 grupos prioritários diferentes;
- Cada grupo de prioridades é representado por um conjunto de bits e cada grupo de sub-prioridades também é representado por um conjunto de bits. O total de ambos os conjuntos de bits é 4;

Priorities group	Priorities bits	Sub-priorities bits
NVIC_PriorityGroup_0	0	4
NVIC_PriorityGroup_1	1	3
NVIC_PriorityGroup_2	2	2
NVIC_PriorityGroup_3	3	1
NVIC_PriorityGroup_4	4	0

- Valores possíveis com 1 bit de prioridade de preempção e 3 bits de sub-prioridade ($1 + 3 = 4$):

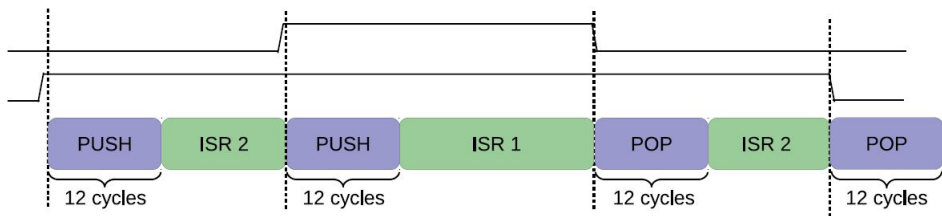


- Valores possíveis com 2 bits de prioridade de preempção e 2 bits de prioridade secundária ($2 + 2 = 4$):

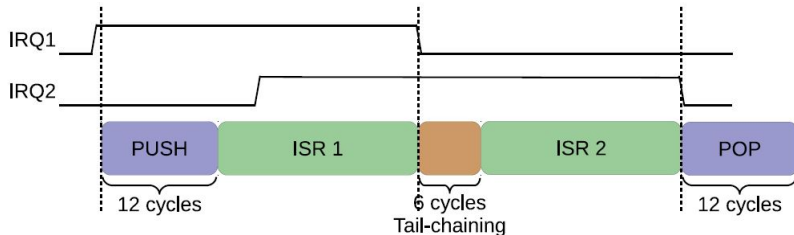


■ Na seguinte sequência:

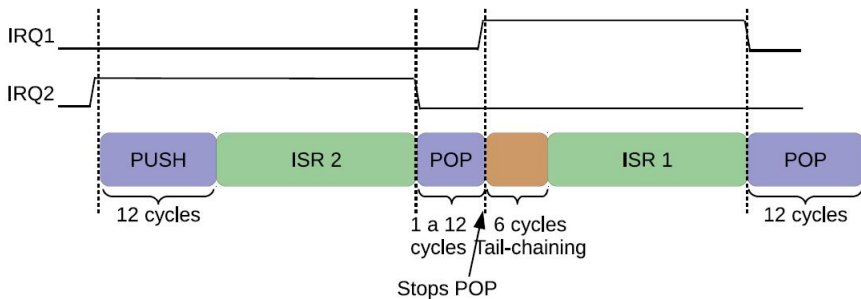
- ▶ O IRQ2 tem a menor prioridade e acontece primeiro;
- ▶ O IRQ1 tem a prioridade mais alta e acontece quando o IRQ2 está sendo executado.



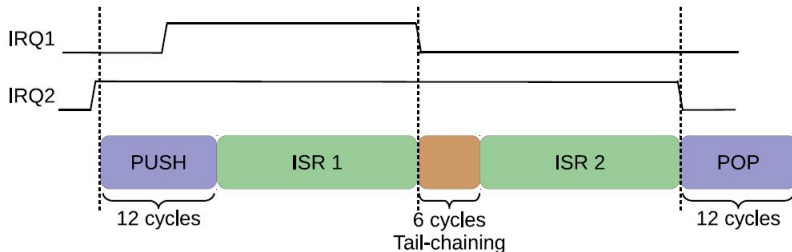
- Se durante uma execução de interrupção ocorrer outra com prioridade menor, o Cortex-M3, após terminar a execução do primeiro, executa a segunda interrupção sem ter uma sequência POP, PUSH, POP:
- Durante esta fase, a *tail chains* do NVIC ataca ambas as interrupções, levando apenas 6 ciclos de *clock* entre elas.



- Mesmo se a segunda interrupção ocorrer apenas quando o NVIC estiver na fase POP, esta fase é suspensa, o ponteiro da pilha é colocado na posição anterior e a interrupção é encadeada na cauda;
- Neste caso, há também os 6 ciclos da sobrecarga da cauda.

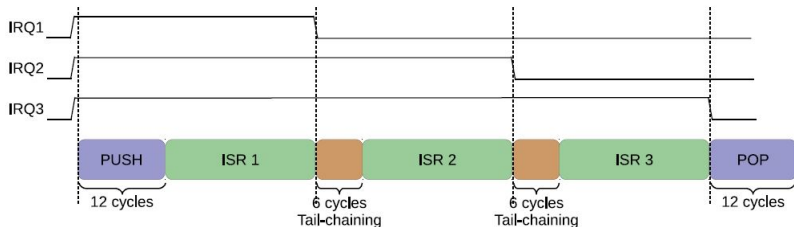


- Depois que uma interrupção acontece, o córtex imediatamente começa a empilhar os registros.
- Se durante esta fase uma prioridade mais alta acontece (tardia), a primeira a ser executada é a prioridade mais alta.



Interrupções - Interrupções com a mesma prioridade

- Quando várias interrupções ocorrem ao mesmo tempo, a sequência de execução será a previamente definida no vetor de interrupção.



- Função de configuração do grupo prioritário:
`void NVIC_PriorityGroupConfig(u32 NVIC_PriorityGroup);`
- Exemplo:
`NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);`

NVIC_PriorityGroup	Descrição
NVIC_PriorityGroup_0	0 bits for pre-emption priority, 4 bits for subpriority
NVIC_PriorityGroup_1	1 bits for pre-emption priority, 3 bits for subpriority
NVIC_PriorityGroup_2	2 bits for pre-emption priority, 2 bits for subpriority
NVIC_PriorityGroup_3	3 bits for pre-emption priority, 1 bits for subpriority
NVIC_PriorityGroup_4	4 bits for pre-emption priority, 0 bits for subpriority

Interrupções - Configuração das prioridades

■ Declaração variável:

NVIC_InitTypeDef NVIC_InitStructure;

■ Preencha a estrutura com os valores desejados:

```
typedef struct
{
    u8 NVIC_IRQChannel;
    u8 NVIC_IRQChannelPreemptionPriority;
    u8 NVIC_IRQChannelSubPriority;
    FunctionalState NVIC_IRQChannelCmd;
} NVIC_InitTypeDef;
```

NVIC_IRQChannel	Descrição
WWDG_IRQChannel	Window WatchDog Interrupt
PVD_IRQChannel	1 PVD through EXTI Line detection Interrupt
TAMPER_IRQChannel	Tamper Interrupt
RTC_IRQChannel	RTC global Interrupt
FlashItf_IRQChannel	FLASH global Interrupt
RCC_IRQChannel	RCC global Interrupt
EXTI0_IRQChannel	EXTI Line0 Interrupt
...	...
TIM2_IRQChannel	TIM2 global Interrupt
...	...

■ Inicialização da estrutura:

NVIC_InitStructure(NVIC_InitStructure);

NVIC_PriorityGroup	NVIC_IRQChannel PreemptionPriority	NVIC_IRQChannel SubPriority	Descrição
NVIC_PriorityGroup_0	0	0-15	0 bits for pre-emption priority 4 bits for subpriority
NVIC_PriorityGroup_1	1	0-7	1 bits for pre-emption priority 3 bits for subpriority
NVIC_PriorityGroup_2	2	0-3	2 bits for pre-emption priority 2 bits for subpriority
NVIC_PriorityGroup_3	3	0-1	3 bits for pre-emption priority 1 bits for subpriority
NVIC_PriorityGroup_4	4	0	4 bits for pre-emption priority 0 bits for subpriority

- Depois de configurada, a ativação da interrupção é realizada de acordo com o periférico;
- Alguns exemplos de interrupções na configuração de periféricos diferentes:

```
TIM_ITConfig(TIM5, TIM_IT_CC1, ENABLE);  
RCC_ITConfig(RCC_IT_PLLRDY, ENABLE);  
RTC_ITConfig(RTC_IT_ALR, ENABLE);
```

- Rotinas de interrupção são criadas declarando a função com um nome simbólico que está no vetor, mas adicionando o “IRQHandler”;
- Exemplos de declaração de rotinas de interrupção de alguns periféricos:

```
void EXTI0_IRQHandler(void);  
void TIM3_IRQHandler(void);  
void USART1_IRQHandler(void);
```

- Os manipuladores de função já estão escritos em `stm32f10x_it.c` (com seu protótipo no arquivo `stm32f10x_it.h`);
- Após a configuração da interrupção, é necessário adicionar o respectivo código neste arquivo.

```
void FLASH_IRQHandler(void)
{
}

/*****
 * Function Name : RCC_IRQHandler
 * Description   : This function handles RCC interrupt request.
 * Input        : None
 * Output       : None
 * Return       : None
 *****/
void RCC_IRQHandler(void)
{
}

/*****
 * Function Name : EXTI0_IRQHandler
 * Description   : This function handles External interrupt Line 0 request.
 * Input        : None
 * Output       : None
 * Return       : None
 *****/
void EXTI0_IRQHandler(void)
{
}

/*****
 * Function Name : EXTI1_IRQHandler
 * Description   : This function handles External interrupt Line 1 request.
 * Input        : None
 * Output       : None
 * Return       : None
 *****/
void EXTI1_IRQHandler(void)
{
}
```

- No Cortex-M3 é sempre necessário limpar o sinalizador de interrupção antes de retornar da rotina de interrupção.

■ Limpar o sinalizador de interrupção depende do periférico. Exemplos:

▶ **Limpando a *flag* do TIM1 Capture / Compare1:**

```
TIM_ClearITPendingBit(TIM3, TIM_IT_CC1);
```

▶ **Limpando a *flag* do TIM2 Update Event:**

```
TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
```

▶ **Limpando a *flag* da intrrupção externa 2:**

```
EXTI_ClearITPendingBit(EXTI_Line2);
```

- Quando são configurados mais de uma interrupção para o mesmo periférico, é necessário testar qual *flag* origina a interrupção;
- A função `GetITStatus ()` permite saber se determinado sinalizador está ou não ativado.

Exemplo:

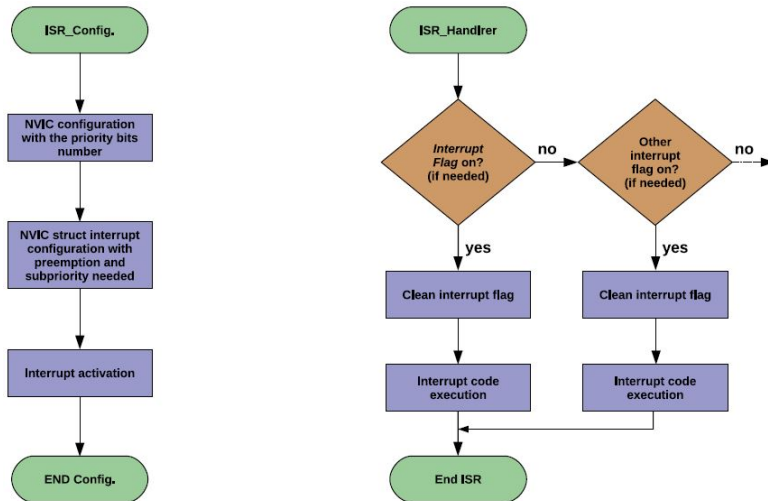
```
EXTIStatus = EXTI_GetITStatus(EXTI_Line8);
```

```
TIMStatus = TIM_GetITStatus(TIM2,TIM_IT_Update);
```

- O teste das *flags* é feito desta maneira:

```
if(TIM_GetITStatus(TIM1, TIM_IT_CC1) == SET)
{
    /*code to be executed if the flag is active */
}
```

Interrupções - Fluxograma de Implementação



- Comece com a configuração do temporizador para escolher a operação necessária;
- A tabela mostra todas as possíveis fontes de interrupção para os temporizadores (alguns deles são apenas para temporizadores avançados):

TIM_IT	Descrição
TIM_IT_Update	TIM Update Interrupt source
TIM_IT_CC1	TIM Capture/Compare 1 Interrupt source
TIM_IT_CC2	TIM Capture/Compare 2 Interrupt source
TIM_IT_CC3	TIM Capture/Compare 3 Interrupt source
TIM_IT_CC4	TIM Capture/Compare 4 Interrupt source
TIM_IT_COM	TIM COM Interrupt source
TIM_IT_Trigger	TIM Trigger Interrupt source
TIM_IT_Break	TIM Break Interrupt source

- A ativação da interrupção é feita da seguinte forma (exemplo):
TIM_ITConfig(TIM4, TIM_IT_CC1, ENABLE);
- Neste momento é necessário ter o manipulador configurado corretamente:

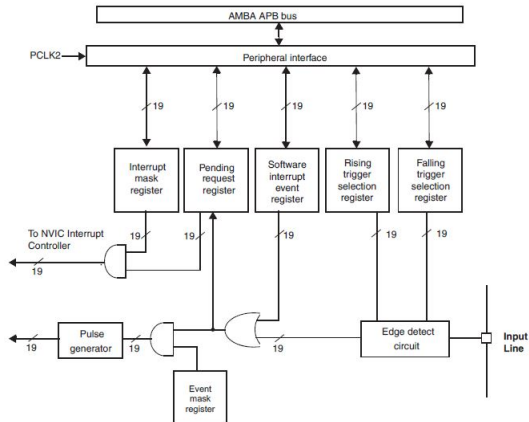
```
void TIM4_IRQHandler(void)
{
}
```

```
/* TIM3 up counting configuration */
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
TIM_TimeBaseStructure.TIM_Period = 1000;
TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_Prescaler = 1000;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
TIM_Cmd(TIM3, ENABLE);

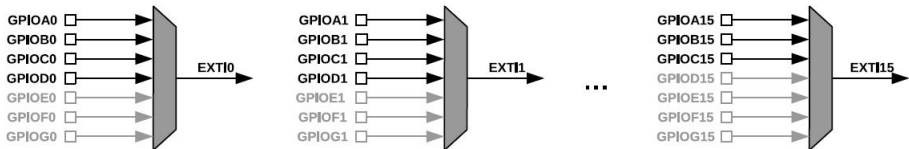
NVIC_InitTypeDef NVIC_InitStructure;
/* Priority Group with 1 bit configuration */
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
/* TIM3 Global interrupt with 0 priority and sub-priority 2 */
NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQChannel;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 2;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
/*TIM3 Update Event Interrupt enable*/
TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE );
```

```
void TIM3_IRQHandler(void)
{
    TIM_ClearITPendingBit(TIM3, TIM_IT_Update);
    /* Interrupt Code */
}
```

- O EXTI permite configurar a interrupção ou eventos externos;
- Os eventos ou interrupções podem ser gerados pelo detector de borda ou pelo software.



- Todos os pinos GPIO estão conectados às linhas externas de interrupção da seguinte forma:



- Ativando um dos pinos como interrupção externa. Exemplo do GPIOB8 usado como EXTI_Line_8:
`GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource8);`

■ Declaração da variável:

```
EXTI_InitTypeDef EXTI_InitStructure;
```

■ Preencha a estrutura com os valores desejados:

EXTI_Line	Descrição
EXTI_Line0	External interrupt line 0
EXTI_Line1	External interrupt line 1
...	...
EXTI_Line18	External interrupt line 10

```
typedef struct
{
    u32 EXTI_Line;
    EXTI_Mode_TypeDef EXTI_Mode;
    EXTI_Trigger_TypeDef EXTI_Trigger;
    FunctionalState EXTI_LineCmd;
} EXTI_InitTypeDef;
```

EXTI_Mode	Descrição
EXTI_Mode_Event	EXTI lines configured as event request
EXTI_Mode_Interrupt	EXTI lines configured as interrupt request

EXTI_Trigger	Descrição
EXTI_Trigger_Falling	Interrupt request configured on falling edge of the input line
EXTI_Trigger_Rising	Interrupt request configured on rising edge of the input line
EXTI_Trigger_Rising_Falling	Interrupt request configured on rising and falling edge of the input line

■ Inicialização da estrutura configurada como EXTI:

```
EXTI_Init(EXTI_InitStructure);
```

- Os manipuladores de rotinas de interrupção externas são agrupados da seguinte forma:

```
void EXTI0_IRQHandler(void);  
void EXTI1_IRQHandler(void);  
void EXTI2_IRQHandler(void);  
void EXTI3_IRQHandler(void);  
void EXTI4_IRQHandler(void);  
void EXTI9_5_IRQHandler(void); /* from 5 up to 9 */  
void EXTI15_10_IRQHandler(void); /* from 10 up to 15 */
```

- Borda ascendente para GPIO1 e GPIO6 como exemplo de habilitação de interrupção externa:

```
GPIO_EXTILineConfig(GPIO_PortSource_GPIOB, GPIO_PinSource1);  
GPIO_EXTILineConfig(GPIO_PortSource_GPIOA, GPIO_PinSource6);
```

- Ativando interrupções externas Linha EXTI1 e Linha EXTI6:

```
EXTI_InitTypeDef EXTI_InitStructure;  
EXTI_InitStructure.EXTI_Line = EXTI_Line1 | EXTI_Line6;  
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;  
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;  
EXTI_InitStructure.EXTI_LineCmd = ENABLE;  
EXTI_Init(&EXTI_InitStructure);
```

- Depois que a configuração de interrupção for necessária, escreva os manipuladores:

```
void EXTI_Line1_IRQHandler(void){  
    /* EXTI Line1 code */  
    EXTI_ClearITPendingBit(EXTI_Line1);  
}  
void EXTI_Line9_5_IRQHandler(void){  
    /* EXTI Line 6 code */  
    EXTI_ClearITPendingBit(EXTI_Line9_5);  
}
```

- Cortex - M3, Technical Reference Manual, revision r2p1.
- ST Reference Manual (RM008), revision 15. Available in ST web page, June 2014.
- STM32F103RBT6 datasheet, revision 8. Available in ST web page, July 2008.
- Cortex-M3 Programming manual (PM0056), revision 3. Available in ST web page, April 2010.

Seja 100 % Motivado!