

Capítulo

7

Interrupções no STM32F407

7.1. Visão geral das interrupções

As interrupções são um recurso comum disponível em praticamente todos os microcontroladores. Interrupções são eventos, tipicamente gerados pelo hardware (periféricos ou pinos de entrada, por exemplo), que causam alterações no fluxo de execução normal do programa (para fornecer serviço a um periférico, por exemplo). Interrupções permitem a um microcontrolador responder rapidamente a eventos sensíveis ao tempo, com baixíssima latência.

Quando um periférico ou hardware precisa ser prontamente atendido pelo processador, normalmente ocorre a seguinte sequência de eventos:

1. O periférico declara uma solicitação de interrupção ao processador (*IRQ – Interrupt Request*)
2. O processador suspende a tarefa atualmente em execução
3. O processador executa uma *ISR (Interrupt Service Routine)* para atender o periférico
4. O processador retoma a tarefa suspensa anteriormente

Interrupções também permitem ao processador executar múltiplas tarefas simultaneamente. Em um dado instante, o microcontrolador está executando apenas uma atividade. Entretanto, as interrupções permitem ao processador executar múltiplas tarefas alternativamente de forma multiplexada no tempo. Dessa forma, múltiplas tarefas podem ser mantidas de uma maneira preemptiva ou não preemptiva.

1. No cenário preemptivo, se uma nova tarefa é mais urgente que a tarefa corrente, esta nova tarefa pode interromper imediatamente a tarefa corrente. A nova tarefa assume o controle do processador. O processador retoma a tarefa antiga depois que a nova tarefa for concluída.
2. No cenário não preemptivo, a nova tarefa não pode parar a tarefa corrente até que essa última, voluntariamente, passe o controle para a nova tarefa.

Uma técnica alternativa de atendimento a um periférico é a “sondagem periódica” (*polling*). Nessa técnica, o processador continuamente consulta o periférico para verificar se um evento específico ocorreu e atende aquele evento, caso tenha ocorrido. A latência da detecção do evento é determinada pela periodicidade da sondagem. Nas interrupções, o processador fornece um mecanismo de hardware que permite a um dispositivo interno ou externo gerar um sinal para imediatamente informar ao processador que um evento ocorreu. Podemos assim dizer que uma interrupção é simplesmente uma chamada de função feita pelo hardware.

Usemos como exemplo uma chamada telefônica para comparar a eficiência das técnicas *pooling* e com interrupções no atendimento de eventos oriundos de periféricos. Suponha que você esteja esperando uma ligação. No *pooling*, você pega o telefone a cada 10 segundos para verificar se tem alguém chamando. Na interrupção, você continua realizando suas tarefas cotidianas enquanto espera o telefone tocar. Quando o telefone finalmente toca, isto é, quando você é interrompido, você pode parar a tarefa que estava executando e atender a ligação. Como você pode perceber, a técnica de *polling* é muito menos eficiente que as interrupções. Com o *polling*, você gasta tempo que poderia ser empregado em outras tarefas verificando repetidamente, na maioria das vezes sem sucesso, se uma chamada está sendo recebida.

7.2. Interrupções no STM32F407

O STM32F407 dispõe de um controlador de interrupção vetorado e aninhado (NVIC - *Nested Vectored Interrupt Controller*) para configuração e manipulação de diversas fontes de interrupções. Além das solicitações de interrupção, existem outros eventos que precisam ser atendidos pelo processador, os quais chamamos de "exceções". Na terminologia da arquitetura ARM, uma interrupção é um tipo específico de exceção. As partes do código do programa que atendem as exceções e interrupções, isto é, as sub-rotinas de software que são executadas quando um evento de interrupção acontece, geralmente são chamadas de manipuladores (*Handler*) de exceção.

No STM32F407, o NVIC recebe solicitações de interrupção de várias fontes, como mostrado na Figura 1.

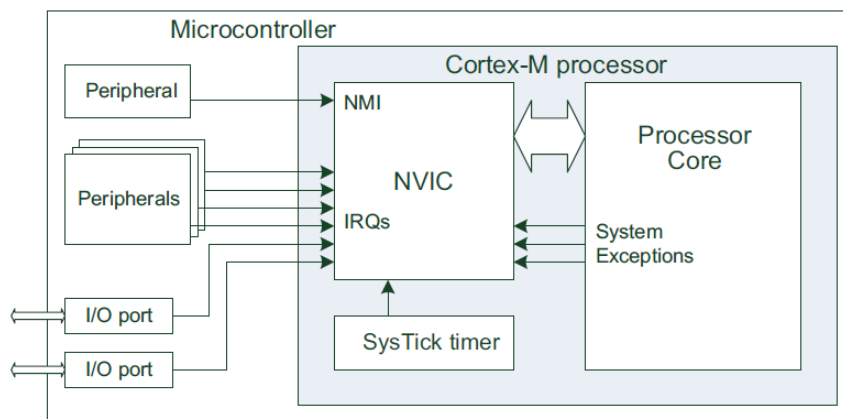


Figura 1 – Várias fontes de interrupção do STM32F407.

O controlador NVIC do Cortex-M4, embutido no STM32F407, pode suportar até 256 fontes de interrupções, apesar de conter apenas 92 fontes implementadas em hardware. Cada interrupção, exceto o *Reset*, que é um tipo especial de exceção, é identificada por um único número, que varia de -15 a 240. Os números identificadores de cada interrupção são definidos pela ARM e pelos fabricantes de chips, coletivamente. Estes números são fixados pelo hardware e não podem ser redefinidos.

As interrupções são caracterizadas em dois grupos:

1. As primeiras 16 (numeradas de -15 a 0) são interrupções do sistema ou do núcleo do processador, também chamadas de exceções do sistema. Estes números são pré-definidos pela ARM.
2. As outras 240 interrupções são reservadas para os periféricos internos e/ou externos. O número e a quantidade de interrupções dos periféricos de um determinado microcontrolador são definidos pelo fabricante do chip. O STM32F407, por exemplo, implementa 82 dessas interrupções.

Diferentes microcontroladores podem ter diferentes quantidades de fontes de interrupção implementadas fisicamente no chip e diferentes níveis de prioridade. Isso ocorre porque os fabricantes de chips podem configurar o design dos seus chips para diferentes requisitos de aplicações.

Dessa forma, na Figura 1 temos 240 fontes de solicitação de interrupção (*Interrupt Requests – IRQs*), uma interrupção não mascarável (*Non-Maskable Interrupt – NMI*) e uma interrupção do temporizador SysTick, ambas exceções do sistema, além de várias outras exceções (*System Exceptions*). A maioria das *IRQs* é gerada por periféricos como timers, conversores, portas de I/O e interfaces de comunicação. A NMI é geralmente gerada a partir de periféricos como um temporizador *watchdog* ou um circuito do tipo *Brown-Out Detector* (BOD). As demais exceções são do próprio núcleo do processador e, também, podem ser geradas usando o software.

Para retomar um programa interrompido, a sequência de atendimento à exceção precisa armazenar o status do programa interrompido, para que isso possa ser restaurado após a conclusão da sub-rotina da exceção. Em geral, isso pode ser feito por um mecanismo de hardware ou uma mistura de operações de hardware e software. Nos processadores Cortex-M, alguns dos registradores são salvos na pilha automaticamente quando uma exceção é aceita e são restaurados automaticamente em uma sequência de retorno de exceção. Esse mecanismo permite que os manipuladores de exceção sejam gravados como funções C normais, sem qualquer sobrecarga de software adicional.

As exceções também podem ser enumeradas de 1 a 15, para exceções do sistema, e de 16 a 256 para entradas de interrupção. Na maioria das exceções, incluindo todas as interrupções, é possível programar o nível de prioridade de atendimento enquanto algumas exceções do sistema têm prioridade fixa.

Os tipos de exceção de 1 a 15 são exceções do sistema, conforme descrito na Tabela 1. Exceções do tipo 16 ou superior são entradas externas de interrupção (consulte a Tabela 2).

Tabela 1. Lista de exceções do sistema

Número da exceção	Tipo de exceção	Prioridade	Descrição
1	Reset	-3 (mais alta)	Reset
2	NMI	-2	Interrupção não mascarável. Pode ser gerada por periféricos internos ou fontes externas
3	Hard Fault	-1	Todas as condições de faltas internas
4	MemManage Fault	Programável	Gerenciamento de memória, violação de MPU ou acesso a endereços proibidos
5	Bus Fault	Programável	Erros de barramento
6	Usage Fault	Programável	Exceções devido a erros de programa ou tentativas de acessar coprocessadores não existentes no chip
7-10	Reservado	-	-
11	SVC	Programável	(<i>SuperVisor Call</i>) Chamada supervisória. Usualmente usada em sistemas operacionais para permitir que <i>tasks</i> de aplicativos acessem serviços do sistema
12	Debug Monitor	Programável	Exceções para eventos de <i>Debug</i> como <i>breakpoints</i> e <i>watchpoints</i> .
13	Reservado	-	-
14	PendSV	Programável	Chamada de serviço pendente. Usada por sistemas operacionais para realizar troca de contexto
15	SysTick	Programável	Exceção gerada pelo temporizador do sistema operacional ou como timer de uso geral em aplicações sem SO.

Tabela 2. Lista de interrupções

Número da exceção	Tipo de exceção	Prioridade	Descrição
16	Interrupção #0	Programável	Pode ser gerada por periféricos internos ou outras fontes externas
17	Interrupção #1	Programável	
...	
256	Interrupção #239	Programável	

O CMSIS (*Cortex Microcontroller Software Interface Standard*) é uma camada de software que propicia a abstração do hardware dos microcontroladores baseados em processadores ARM. Sua interface de software simplifica a reutilização do software e reduz a curva de aprendizado para desenvolvedores. O CMSIS fornece interfaces para processador, periféricos e sistemas operacionais em tempo real. No CMSIS, as exceções do sistema usam valores negativos para enumeração, conforme mostrado na Tabela 3.

O CMSIS também define os nomes dos manipuladores de exceção do sistema. As interrupções são específicas do dispositivo e são definidas nos arquivos de cabeçalho do dispositivo, fornecidos pelos fabricantes de microcontroladores. Observe que o número da interrupção (por exemplo, Interrupção #0) refere-se às entradas de interrupção no NVIC do processador.

Tabela 3. Definições de interrupções no CMSIS

Número da exceção	Tipo da exceção	Enumeração CMSIS (IRQn)	Enumeração CMSIS (valor)	Handler da exceção
1	Reset	-	-	Reset_Handler
2	NMI	NonMaskableInt_IRQn	-14	NMI_Handler
3	Hard Fault	HardFault_IRQn	-13	HardFault_Handler
4	MemManage Fault	MemoryManagement_IRQn	-12	MemManage_Handler
5	Bus Fault	BusFault_IRQn	-11	BusFault_Handler
6	Usage Fault	UsageFault_IRQn	-10	UsageFault_Handler
11	SVC	SVC_IRQn	-5	SVC_Handler
12	Debug Monitor	DebugMonitor_IRQn	-4	DebugMon_Handler
14	PendSV	PendSV_IRQn	-2	PendSV_Handler
15	SYSTICK	SysTick_IRQn	-1	SysTick_Handler
16	Interrupção #0	Específico do dispositivo	0	Específico do dispositivo
17 a 256	Interrupção #1 - #239	Específico do dispositivo	1 a 239	Específico do dispositivo

Nas tabelas 1 e 2, a prioridade determina a ordem na qual as interrupções serão atendidas quando mais de um tipo ocorrer simultaneamente. O projeto dos processadores Cortex-M3 e Cortex-M4, por exemplo, suporta até 256 níveis de prioridade programável. O número real de níveis de prioridade programáveis disponíveis em um determinado microcontrolador é decidido pelo fabricante e depende da quantidade de periféricos disponíveis.

A maioria dos chips Cortex-M3 ou Cortex-M4 têm uma quantidade menor de níveis de prioridade, por exemplo, 8, 16, 32 e assim por diante. Isso ocorre porque ter muitos níveis de prioridade pode aumentar a complexidade do NVIC e pode aumentar o consumo de energia. Na maioria dos casos, as aplicações requerem apenas um pequeno número de níveis de prioridade programáveis. Portanto, os projetistas de chips de silício precisam personalizar o projeto do processador com base no número de níveis de prioridade nos aplicativos-alvo. Esta redução nos níveis é implementada cortando-se a parte menos significativa (LSB) dos registradores de configuração de prioridade.

O nível de prioridade de cada exceção pode ser programado por meio dos registradores SHP (*System Handlers Priority*) que ficam no bloco de controle do sistema (SCB – *System Control Block*), na última região do mapa de memória. Já o nível de prioridade de cada interrupção pode ser programado por meio dos registradores IP (*Interrupt Priority*). Existem 12 registradores SHP e 240 registradores IP, onde cada um é utilizado para especificar a prioridade de uma determinada exceção ou interrupção, conforme a enumeração CMSIS da Tabela 3.

Especificamente, no STM32F407, é possível programar até 16 níveis (0-15) distintos de prioridade. Quanto menor o valor do nível, maior é a prioridade da interrupção. O valor zero tem a prioridade mais alta. Quando duas interrupções tem o mesmo valor no nível de prioridade, aquela que tiver o menor número de exceção na Tabela 3 será atendida primeiro.

Uma exceção de prioridade mais alta (número menor no nível de prioridade) pode antecipar uma exceção de prioridade mais baixa (número maior no nível de prioridade); esse é o cenário de exceção/interrupção aninhada. Algumas das exceções (*reset*, NMI e HardFault) têm níveis fixos de prioridade. Seus níveis de prioridade são representados com números negativos na enumeração CMSIS para indicar que elas são de maior prioridade do que outras exceções. Outras exceções têm níveis de prioridade programáveis, que variam de 0 a 15. Após o reset, todas as interrupções são desativadas e recebem um valor de nível de prioridade igual a 0.

Antes de usar qualquer interrupção, é necessário:

- Configurar o nível de prioridade da interrupção (esta etapa é opcional)
- Configurar e ativar o controle de requisição de interrupção no periférico que aciona a interrupção
- Habilitar a interrupção no NVIC ou no SCB (no caso de exceções)

Cada entrada de interrupção no NVIC tem seis bits de controle que estão alocados em registradores distintos:

- *Enable* bit, localizado no registrador *Interrupt Set Enable Register* (ISER);
- *Disable* bit, localizado no *Interrupt Clear Enable Register* (ICER);
- *Pending* bit, localizado no *Interrupt Set Pending Register* (ISPR);
- *Un-pending* bit, localizado no *Interrupt Clear Pending Register* (ICPR);
- *Active* bit, localizado no *Interrupt Active Bit Register* (IABR);
- *Software trigger* bit, localizado no *Software Trigger Interrupt Register* (STIR).

Cada um desses registradores é, na verdade, um array com 8 registradores, indexados a partir do zero. Por exemplo, o registrador ISER é formado pelos registradores ISER0, ISER1, ..., ISER7, que podem habilitar qualquer uma das interrupções, já o registrador ICER é formado pelos registradores ICER0, ICER1, ..., ICER7, que podem desabilitar qualquer uma das interrupções. O registrador ISER0 possui os 32 bits de habilitação das 32 primeiras interrupções, o registrador ISER1 possui os 32 bits de habilitação das 32 interrupções seguintes e assim, sucessivamente até o registrador ISER7.

Para habilitar uma interrupção no NVIC, basta setar o bit correspondente no registrador ISER correspondente. Para desabilitar uma interrupção no NVIC, basta setar o bit correspondente no registrador ICER correspondente. Quando uma interrupção ocorre, o bit correspondente no registrador ISPR correspondente é setado e permanece assim, indicando uma pendência, até que sua ISR seja executada. Para limpar uma pendência, basta setar o bit correspondente no registrador ICPR correspondente. Quando um bit é setado por software em algum registrador IABR ou STIR, o processador executa a ISR correspondente. Algumas exceções são ativadas apenas por hardware. Quando a interrupção é acionada, a rotina de serviço de interrupção (ISR) correspondente será executada (pode ser necessário limpar a solicitação de interrupção do periférico dentro da ISR).

Para facilitar o gerenciamento de interrupções e exceções, o CMSIS oferece várias funções de acesso aos registradores acima descritos. Para a programação geral de aplicações, a prática recomendada é usar as funções de acesso do CMSIS. Por exemplo, as funções de controle de interrupção mais comumente usadas são mostradas na Tabela 4.

Você também pode acessar diretamente os registros no NVIC ou no SCB, se necessário. Mas tal prática pode limitar a portabilidade do software ao portar o código de um microcontrolador Cortex-M para outro com um tipo de processador diferente.

Tabela 4. Funções CMSIS comumente usadas no gerenciamento de interrupções no NVIC

Função CMSIS	Descrição
void NVIC_EnableIRQ (IRQn_Type IRQn)	Habilita uma interrupção
void NVIC_DisableIRQ (IRQn_Type IRQn)	Desabilita uma interrupção
void NVIC_SetPriority (IRQn_Type IRQn, uint32_t priority)	Especifica a prioridade de uma interrupção
void __enable_irq(void)	Habilitação global de todas as interrupções
void __disable_irq(void)	Desabilita todas as interrupções

7.3. Tabela de vetores de interrupção

Existe uma rotina de serviço de interrupção (ISR) associada a cada tipo de interrupção ou exceção. O STM32F407 armazena o endereço inicial de cada ISR em um array especial chamado de tabela de vetores de interrupção, que está localizada na memória de programa a partir do endereço 0x00000004. Como cada entrada na tabela representa um endereço de memória, cada entrada ocupa 4 bytes, como mostrado na Figura 2.

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	Debug Monitor vector	12
0x0000002C	SVC vector	11
0x00000028	Not used	10
0x00000024	Not used	9
0x00000020	Not used	8
0x0000001C	Not used	7
0x00000018	Usage Fault vector	6
0x00000014	Bus Fault vector	5
0x00000010	MemManage vector	4
0x0000000C	HardFault vector	3
0x00000008	NMI vector	2
0x00000004	Reset vector	1
0x00000000	MSP initial value	0

Figura 2 – Tabela de vetores de interrupção.

7.4. Controlador de interrupções/eventos externos no STM32F407

Interrupções externas são geradas por periféricos ou dispositivos externos, como botões e chaves conectados aos pinos do microcontrolador. Interrupções externas são muito úteis porque elas permitem que o microcontrolador monitore eficientemente sinais fornecidos aos pinos e prontamente responda a eventos externos.

O STM32F407 possui um controlador de interrupções/eventos externos conectado ao barramento APB2 que consiste em 23 circuitos de detecção de borda usados para gerar solicitações de interrupção e enviá-las para o NVIC. Cada linha de interrupção pode ser configurada independentemente para selecionar o tipo de interrupção e o evento de acionamento correspondente (uma borda de subida, uma borda de descida ou ambas). Cada linha também pode ser habilitada/desabilitada de forma independente. Um registrador de pendências mantém o status das solicitações das linhas. Os principais recursos do controlador de interrupções externas (EXTI) são os seguintes:

- gatilho e mascaramento independentes em cada linha de interrupção
- bit de status dedicado para cada linha de interrupção
- geração de até 23 solicitações de interrupção de software

Na Figura 3 é mostrado o diagrama de blocos do controlador de interrupção/evento externo.

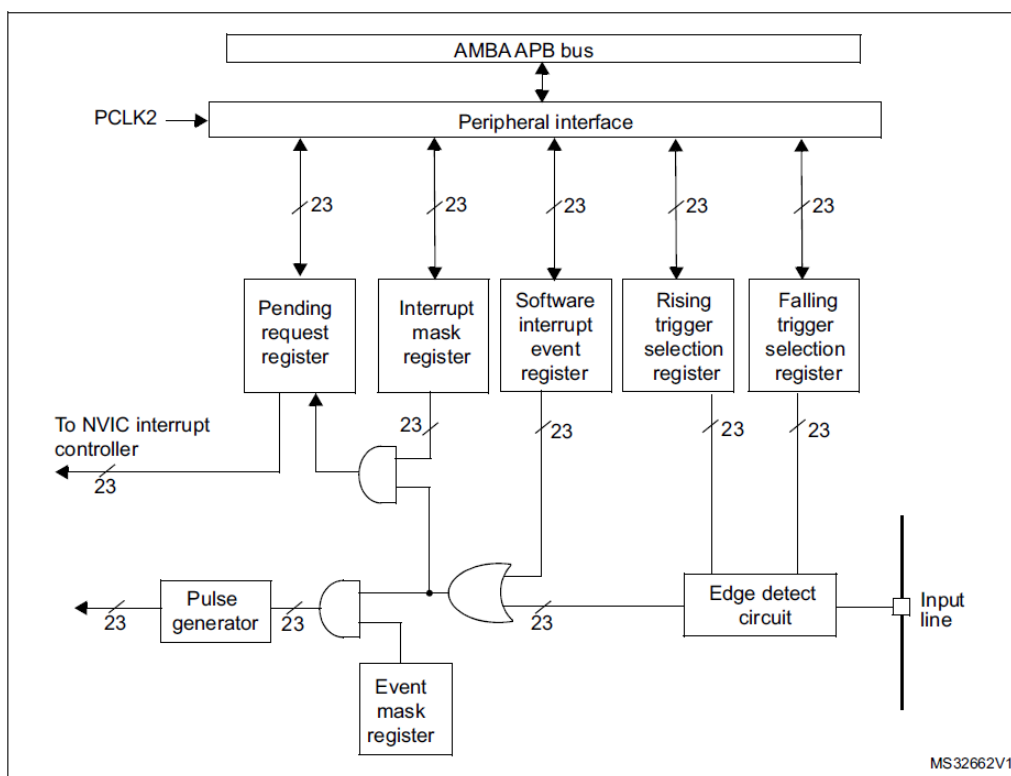


Figura 3 – Diagrama de blocos do controlador de interrupção/evento externo (EXTI).

Para gerar uma interrupção, a linha específica de interrupção deve ser configurada e ativada. Isso é feito programando os dois registradores de gatilho com a detecção da(s) borda(s) desejada(s), borda de subida (*Rising Trigger Selection Register* – RTSR), borda de descida (*Falling Trigger Selection Register* - FTSR), e habilitando a solicitação de interrupção escrevendo um '1' no bit correspondente no registrador da máscara de interrupção (*Interrupt Mask Register* – IMR). Quando a borda selecionada ocorre na linha de interrupção externa, uma solicitação de interrupção é gerada, o bit de pendência correspondente à linha de interrupção é setado no registrador de pendência (*Pending Request Register* – PR) e a solicitação é enviada ao NVIC. É necessário limpar essa solicitação de interrupção dentro da ISR escrevendo um '1' no registrador de pendência.

Além de requisição de interrupção, o controlador EXTI pode ser usado para gerar eventos. O objetivo principal da utilização de eventos, ao invés de interrupções externas, é fazer com que o processador possa ser acordado de estados de espera (*wakeup*). As interrupções normalmente são usadas para executar ISRs enquanto os eventos podem sinalizar outro periférico para fazer algo sem a intervenção do processador. Por exemplo, um evento em EXTI pode inicializar uma amostragem no conversor ADC e, em seguida, gravar o valor medido na memória usando DMA. Portanto, as interrupções envolvem a execução de código. Os eventos não requerem necessariamente a execução do código.

Para gerar um evento, a linha de eventos deve ser configurada e ativada. Isso é feito programando os dois registradores de gatilho com a detecção de borda desejada e ativando a solicitação de evento, escrevendo um '1' no bit correspondente no registrador de máscara de evento (*Event Mask Register* – EMR). Quando a borda selecionada ocorre na linha de evento, um pulso de evento é gerado. O bit pendente correspondente à linha do evento não é resetado.

Uma solicitação de interrupção/evento também pode ser gerada pelo software, escrevendo um '1' no bit correspondente à linha específica no registrador de interrupção/evento de software (*Software Interrupt Event Register* - SWIER).

Até 80 pinos de GPIO são multiplexados e conectados a 16 das 23 linhas de interrupção externa, como mostrado na Figura 4. As outras sete linhas de interrupção externa são conectadas a periféricos como RTC, USB e Ethernet.

O controlador de interrupção externa tem 16 linhas de interrupções externas associadas com pinos de I/O. Essas interrupções são nomeadas como EXTI0 (*External Interrupt 0*) até EXTI15 (*External Interrupt 15*). Cada uma dessas interrupções é associada com um pino de I/O específico. Porém, como o microcontrolador possui mais de 16 pinos de I/O, existem 16 multiplexadores que selecionam um, dentre 9 pinos de entrada. Cada multiplexador recebe os pinos de diferentes portas, mas que têm a mesma numeração, como mostrado na Figura 4.

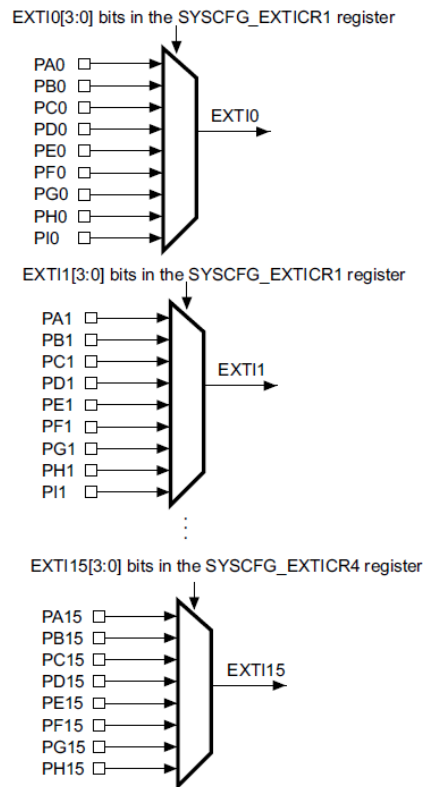


Figura 4 – Mapeamento dos pinos de I/O nas linhas do controlador de interrupção/evento externo.

Por exemplo, a fonte da interrupção EXTI0 pode ser selecionada como um dos pinos PA0, PB0, PC0, etc. O pino específico para uma determinada linha de interrupção é selecionado por meio de quatro bits EXTI0[3:0] no registrador EXTICR1 do módulo SYSCFG. Então, o microcontrolador responde ao pedido de interrupção externa e executa a rotina de serviço associada àquela linha de interrupção, no caso, *EXTI0_IRQHandler()*.

Os nomes das ISRs podem ser encontrados na tabela de vetores dentro do código de inicialização, que também é fornecido pelo fornecedor do microcontrolador. O nome da ISR precisa corresponder ao nome usado na tabela de vetores para que o linker possa colocar corretamente o endereço inicial da ISR na tabela de vetores.

Abaixo é mostrada a configuração de software para selecionar um pino k de uma porta GPIO como a fonte de gatilho de uma interrupção externa EXTI k :

1. Habilite o clock do controlador de configuração do sistema (SYSCFG) e da porta GPIO correspondente;
2. Configure o pino k do GPIO como entrada digital;
3. Configure o registrador correspondente EXTICR do módulo SYSCFG para rotear o pino k para a linha de entrada de interrupção k , EXTI k ;
4. Selecione a(s) borda(s) ativa(s) para a interrupção EXTI k nos registradores RTSR e FTSR no controlador EXTI;
5. Habilite a interrupção da linha EXTI k no registrador IMR do controlador EXTI;
6. Selecione o nível de prioridade da interrupção EXTI k no registrador IP (*Interrupt Priority*) correspondente no controlador NVIC;
7. Habilite a interrupção da linha EXTI k no registrador ISER correspondente no controlador NVIC;
8. Escreva a rotina de atendimento à interrupção de EXTI k (os nomes das rotinas de interrupção são pré-definidos no arquivo *startup_stm32.s*. Por exemplo, a rotina de atendimento à interrupção de EXTI0 é chamada de *EXTI0_IRQHandler()*);
9. Na rotina de interrupção, o software precisa limpar o bit correspondente no registrador de pendência (PR) do controlador EXTI para indicar que o atendimento à interrupção foi concluído. Isso é feito escrevendo um '1' no registrador PR.

Abaixo, é mostrado um exemplo de código que configura o pino PA0 como fonte de interrupção da linha de interrupção externa EXTI0. A função *Configure_EXTI0()* configura o pino PA0 como entrada digital com resistor de *pull-down*, seleciona a borda de subida do sinal em PA0 como gatilho e habilita a interrupção no controlador EXTI e no NVIC, configurando nível 0 de prioridade no atendimento da ISR. A função de atendimento à interrupção de EXTI0, *EXTI0_IRQHandler()*, alterna o estado da saída PA6 toda vez que ocorrer uma interrupção. O pino PA6 deve ter sido previamente configurado como saída. ■

```

void Configure_EXTI0()
{
    RCC->AHB1ENR |= 1;                //habilita o clock do GPIOA

    //Configuração do pino PA0 como entrada digital com resistor de pull-down
    GPIOA->MODER &= ~(3);             //seleciona modo de entrada digital no pino PA0
    GPIOA->PUPDR |= 2;                //habilita o resistor de pull-down no pino PA0

    //Configuração da interrupção em EXTI0
    RCC->APB2ENR |= 1 << 14;          //habilita o clock de SYSCFG
    SYSCFG->EXTICR[0] &= ~(0xF);      //seleciona PA0 como gatilho de EXTI0
    EXTI->RTSR |= 1;                  //seleciona borda de subida
    EXTI->IMR |= 1;                   //habilita a interrupção EXTI0 em EXTI
    NVIC_EnableIRQ(EXTI0_IRQn);       //habilita a interrupção EXTI0
    NVIC_SetPriority(EXTI0_IRQn, 0);   //nível de prioridade 0 para EXTI0_IRQn
}

void EXTI0_IRQHandler()
{
    GPIOA->ODR ^= 1 << 6;             //alterna o estado da saída PA6
    EXTI->PR |= 1;                    //limpa o flag de pendência em EXTI0
}

```