

Microcontroladores

Prof. Marcos Chaves

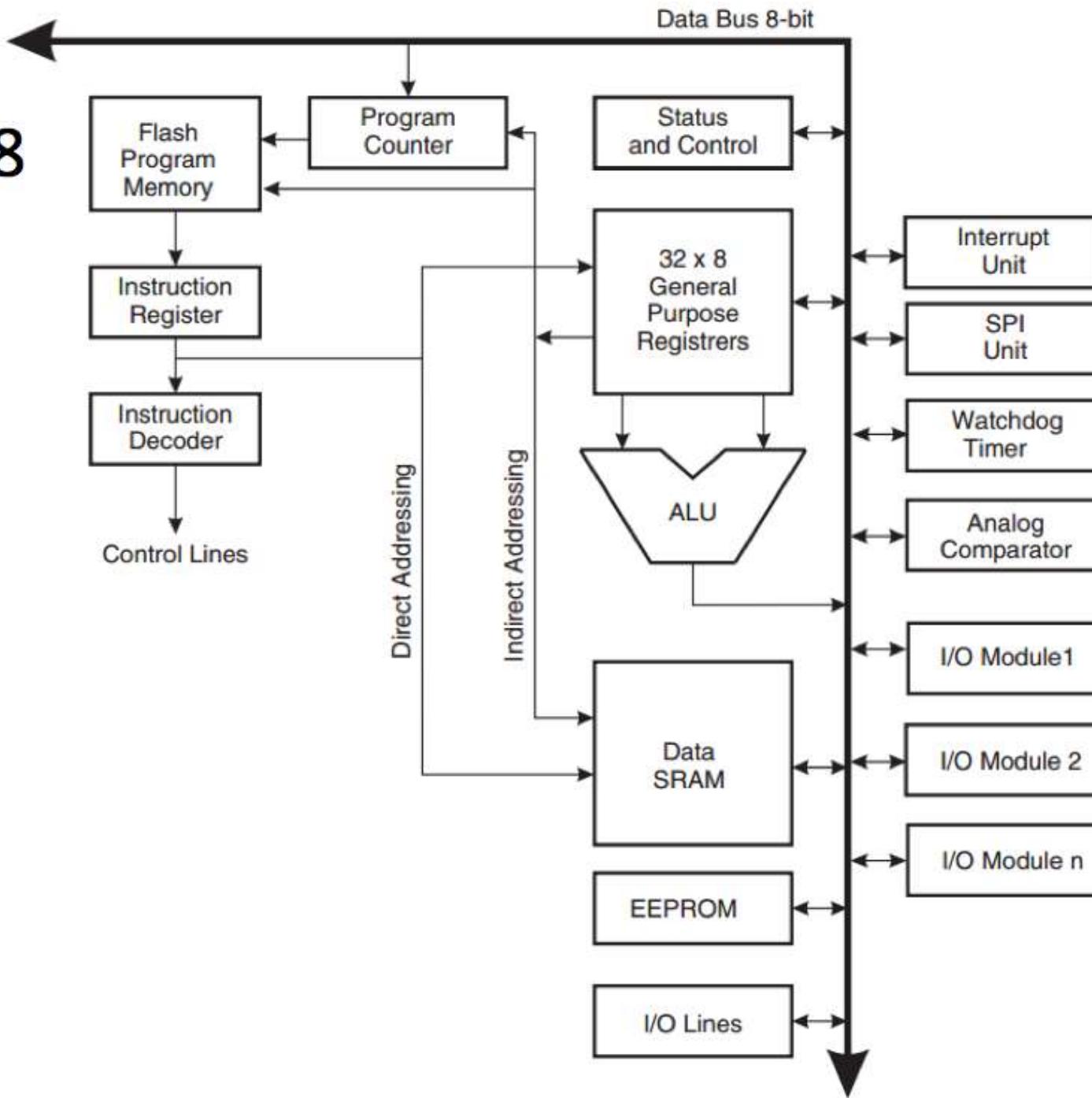
Introdução de interrupções - Externa, TIMERO e PWM

Fonte:

PICmicro Microcontroladores - José Carlos Fonseca
Desbravando o PIC Ampliado e Atualizado – David José de Souza

ATmega328

- CPU



Interrupções

- Até agora, o fluxo do programa é determinado apenas pelo programa principal
- Problema
 - Alguns periféricos necessitam de um tempo para que possam executar suas tarefas
 - Este tempo era gasto fazendo nada - loop vazio - enquanto esperamos uma variável ou flag mudar de estado
 - Esta técnica utilizada até agora é denominada pooling

Interrupções

As interrupções são causadas através de eventos assíncronos (podem ocorrer a qualquer momento) causando um desvio no processamento. Este desvio tem como destino um endereço para tratamento da interrupção. Uma boa analogia para melhor entendermos o conceito de interrupção é a seguinte: você está trabalhando digitando uma carta no computador quando o seu telefone toca. Neste momento você, interrompe o que está fazendo, para atender ao telefone e verificar o que a pessoa do outro lado da linha está precisando. Terminada a conversa, você coloca o telefone no gancho novamente e retoma o seu trabalho do ponto onde havia parado.

Observe que não precisamos verificar a todo instante, se existe ou não alguém na linha, pois somente quando o ramal é chamado, o telefone toca avisando que existe alguém querendo falar com você.



Comportamento de uma interrupção

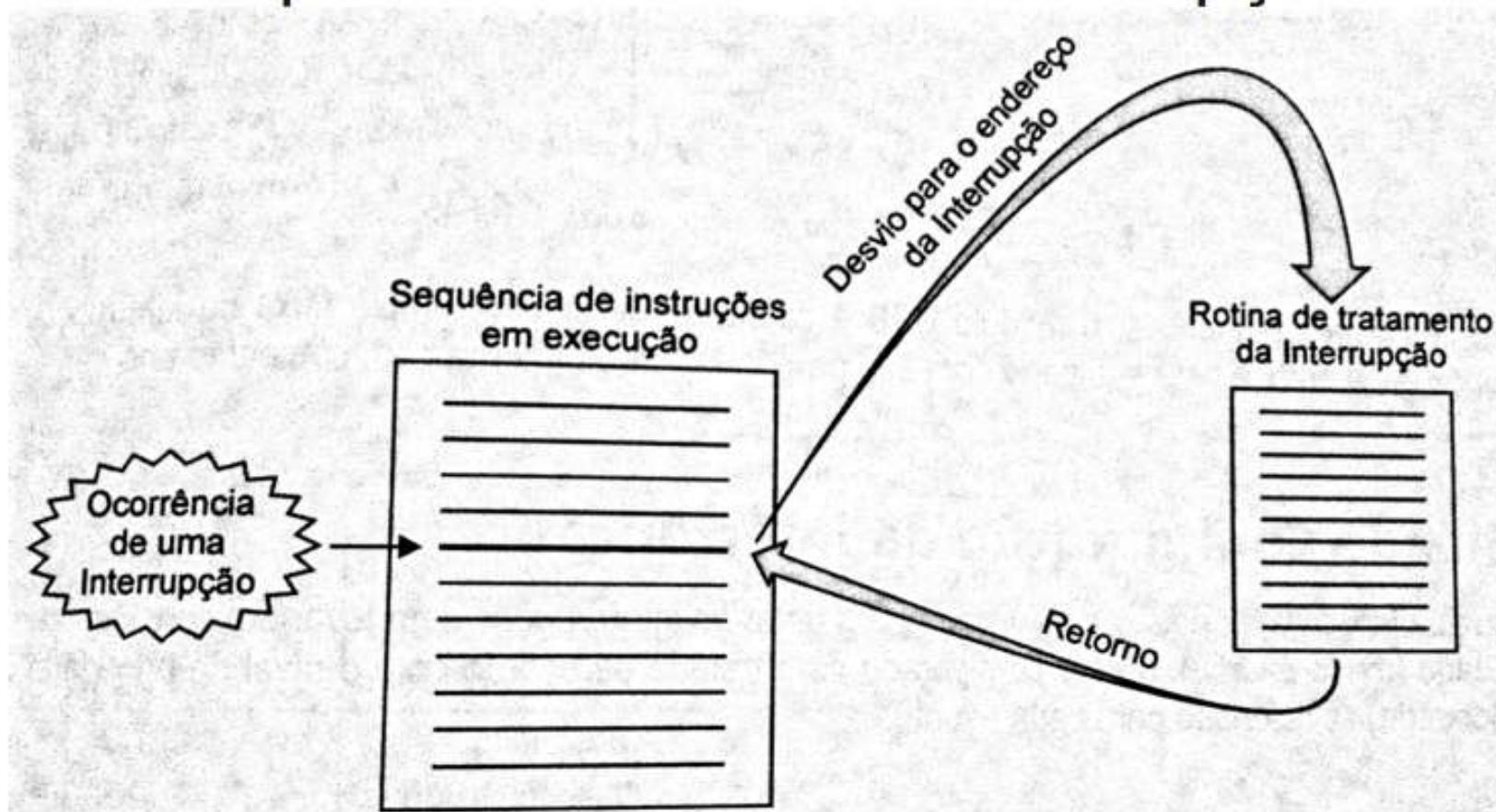


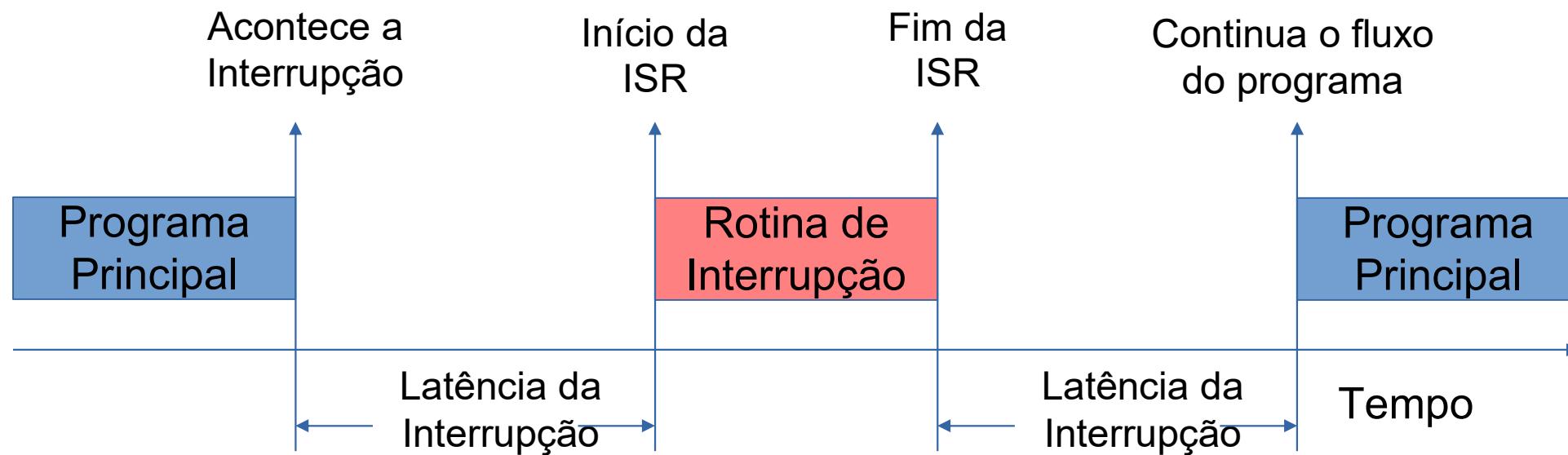
Figura 9.1: Interrupção.

Uma interrupção suspende a execução de uma aplicação, salva as informações de contexto e desvia o controle para uma rotina de serviço de interrupção (ISR) para que o evento possa ser processado. Ao finalizar as informações são restauradas e a execução normal é retomada.

Interrupções

- Solução: Interrupção
 - Desenvolver um sistema que nos avise quando uma determinada tarefa acabou
 - Este sistema deve ser acionado automaticamente
 - O fluxo do programa não deve ser alterado
- O dispositivo/periférico em questão deve estar preparado para gerar uma interrupção

Interrupções



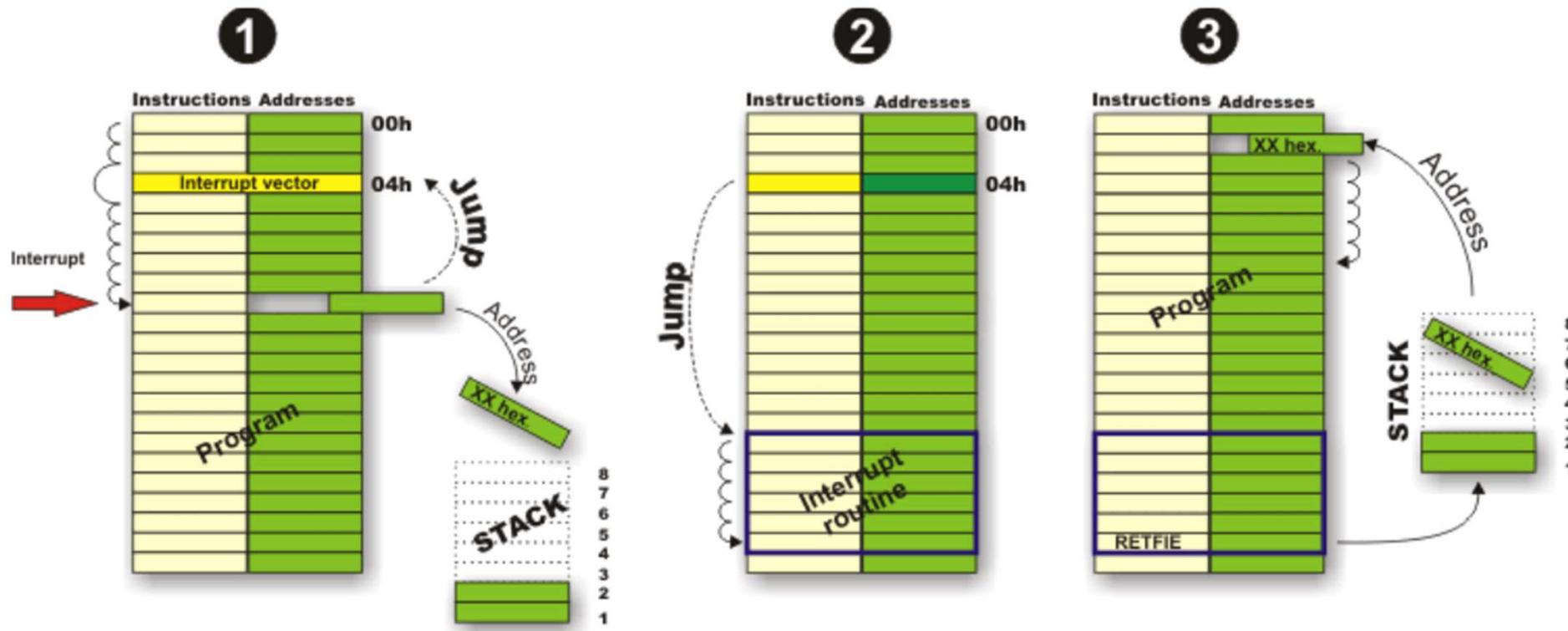
Interrupções

- Quando gerada uma interrupção o programa é paralisado e uma função de interrupção é executada
- Exemplos:
 - Conversor AD: quando o resultado da conversão estiver disponível para leitura.
 - Porta B: quando algum dos bits configurados como entrada altera seu valor.
 - Timer 0: quando acontece overflow} em seu contador

Interrupções

- A função que irá tratar da interrupção não retorna nem recebe nenhum valor
- Cada compilador/microcontrolador possui um modo diferente de implementar a interrupção
 - Utilização de palavras chaves reservadas
 - Utilização de diretivas de pré compilação
 - Utilização de nomes de funções pré definidos
- O framework Wiring, do Arduino, faz uso de estruturas de programação para esconder essa complexidade do programador
 - Isso é feito via função attachInterrupt()

Fases de atendimento de uma interrupção



- 1 – Quando a interrupção ocorre, o programa principal é interrompido; O endereço atual é salvo na pilha (STACK); O contador de programa(PC) é carregado com o endereço do vetor de interrupção específica ativada.
- 2 – No endereço do vetor interrupção deve ocorrer a chamada da rotina da rotina de tratamento da interrupção (Interrupt routine).
- 3 – Ao final da rotina de interrupção, o PC é carregado com o último endereço salvo na pilha, retornando ao ponto original do programa principal.

Interrupções no Atmega328

Vetor	Endereço	Fonte	Definição
1	0x0000	RESET	Reset Ext., Power-On Reset,
2	0x0002	INT0	Int. Externa no pino INT0
3	0x0004	INT1	Int. Externa no pino INT1
4	0x0006	PCINT0	Mudança estado pino Req. 0
5	0x0008	PCINT1	Mudança estado pino Req. 1
6	0x000A	PCINT2	Mudança estado pino Req. 2
7	0x000C	WDT	Watchdog Time-out
8	0x000E	TIMER2 COMPA	T/C2 Compare Match A
9	0x0010	TIMER2 COMPB	T/C2 Compare Match B
10	0x0012	TIMER2 OVF	T/C2 Overflow
11	0x0014	TIMER1 CAPT	T/C1 Evento de Captura
12	0x0016	TIMER1 COMPA	T/C1 Compare Match A
13	0X0018	TIMER1 COMPB	T/C1 Compare Match B
14	0X001A	TIMER1 OVF	T/C1 Overflow
15	0x001C	TIMER0 COMPA	T/C0 Compare Match A
16	0x001E	TIMER0 COMPB	T/C0 Compare Match B
17	0x0020	TIMER0 OVF	T/C0 Overflow
18	0x0022	SPI, STC	SPI Transfer. completa
19	0x0024	USART, RX	USART Recep. completa
20	0x0026	USART, UDRE	USART Reg. dados vazio
21	0x0028	USART, TX	USART Transm. completa
22	0x002A	ADC	Conv. A/D completada
23	0x002C	EE_READY	EEPROM Pronta
24	0X002E	ANALOG COMP	Comparador Analógico
25	0x0030	TWI	2-Wire serial interface
26	0x0032	SPM READY	Memória Flash Pronta

Vetores de Interrupções no Atmega328 em Assembly

Address	Labels	Code	Comments
0x0000		jmp RESET	; Reset Handler
0x0002		jmp EXT_INT0	; IRQ0 Handler
0x0004		jmp EXT_INT1	; IRQ1 Handler
0x0006		jmp PCINT0	; PCINT0 Handler
0x0008		jmp PCINT1	; PCINT1 Handler
0x000A		jmp PCINT2	; PCINT2 Handler
0x000C		jmp WDT	; Watchdog Timer Handler
0x000E		jmp TIM2_COMPA	; Timer2 Compare A Handler
0x0010		jmp TIM2_COMPB	; Timer2 Compare B Handler
0x0012		jmp TIM2_OVF	; Timer2 Overflow Handler
0x0014		jmp TIM1_CAPT	; Timer1 Capture Handler
0x0016		jmp TIM1_COMPA	; Timer1 Compare A Handler
0x0018		jmp TIM1_COMPB	; Timer1 Compare B Handler
0x001A		jmp TIM1_OVF	; Timer1 Overflow Handler
0x001C		jmp TIM0_COMPA	; Timer0 Compare A Handler
0x001E		jmp TIM0_COMPB	; Timer0 Compare B Handler
0x0020		jmp TIM0_OVF	; Timer0 Overflow Handler

Vetores de Interrupções no Atmega328 em Assembly

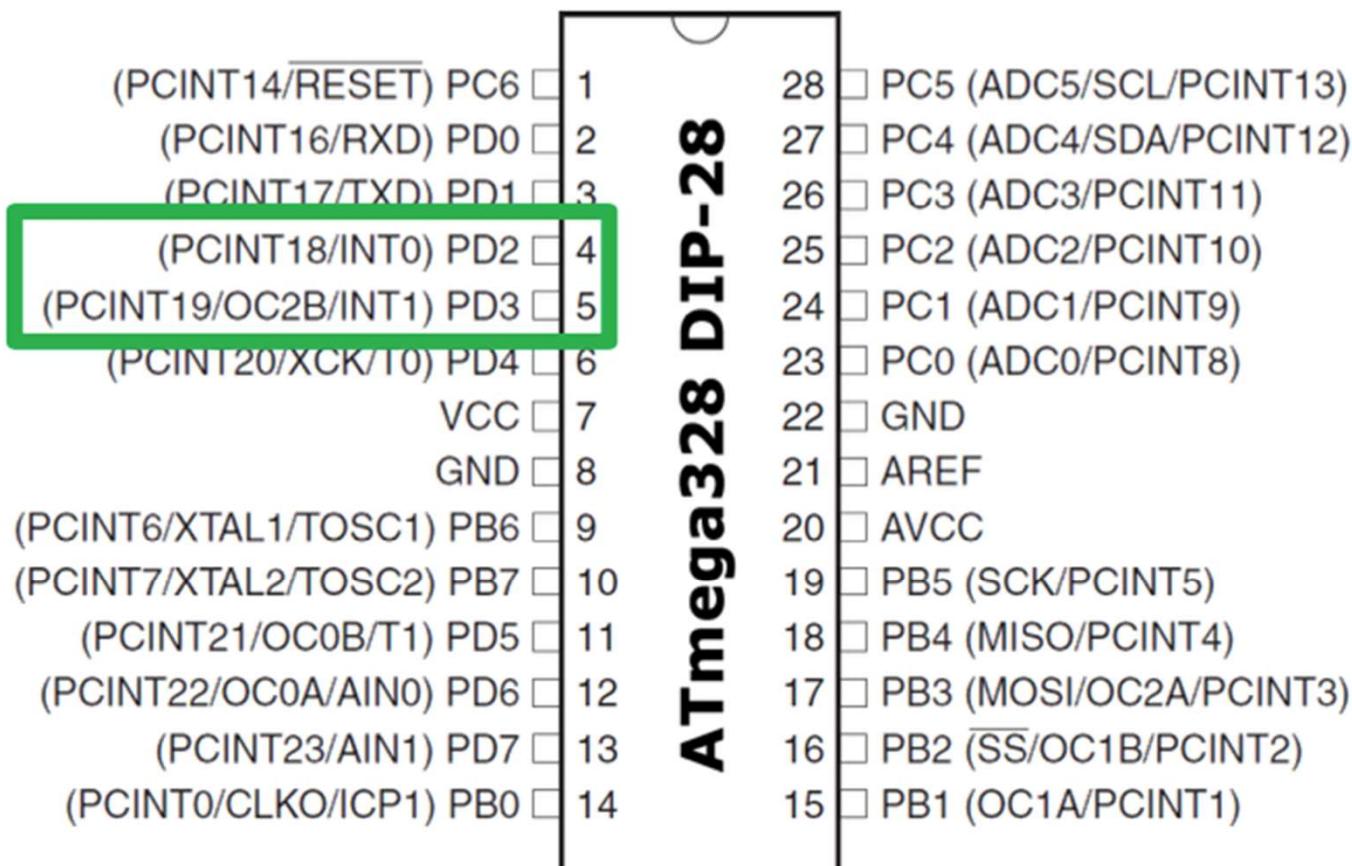
```
0x0022      jmp    SPI_STC      ; SPI Transfer Complete Handler
0x0024      jmp    USART_RXC    ; USART, RX Complete Handler
0x0026      jmp    USART_UDRE   ; USART, UDR Empty Handler
0x0028      jmp    USART_TXC    ; USART, TX Complete Handler
0x002A      jmp    ADC          ; ADC Conversion Complete Handler
0x002C      jmp    EE_RDY       ; EEPROM Ready Handler
0x002E      jmp    ANA_COMP     ; Analog Comparator Handler
0x0030      jmp    TWI          ; 2-wire Serial Interface Handler
0x0032      jmp    SPM_RDY     ; Store Program Memory Ready Handler
;
0x0033RESET: ldi    r16, high(RAMEND); Main program start
0x0034      out    SPH,r16      ; Set Stack Pointer to top of RAM
0x0035      ldi    r16, low(RAMEND)
0x0036      out    SPL,r16
0x0037      sei               ; Enable interrupts
0x0038      <instr>  xxx
...
...
...
...
```

Vetores de Interrupções no Atmega328 em C

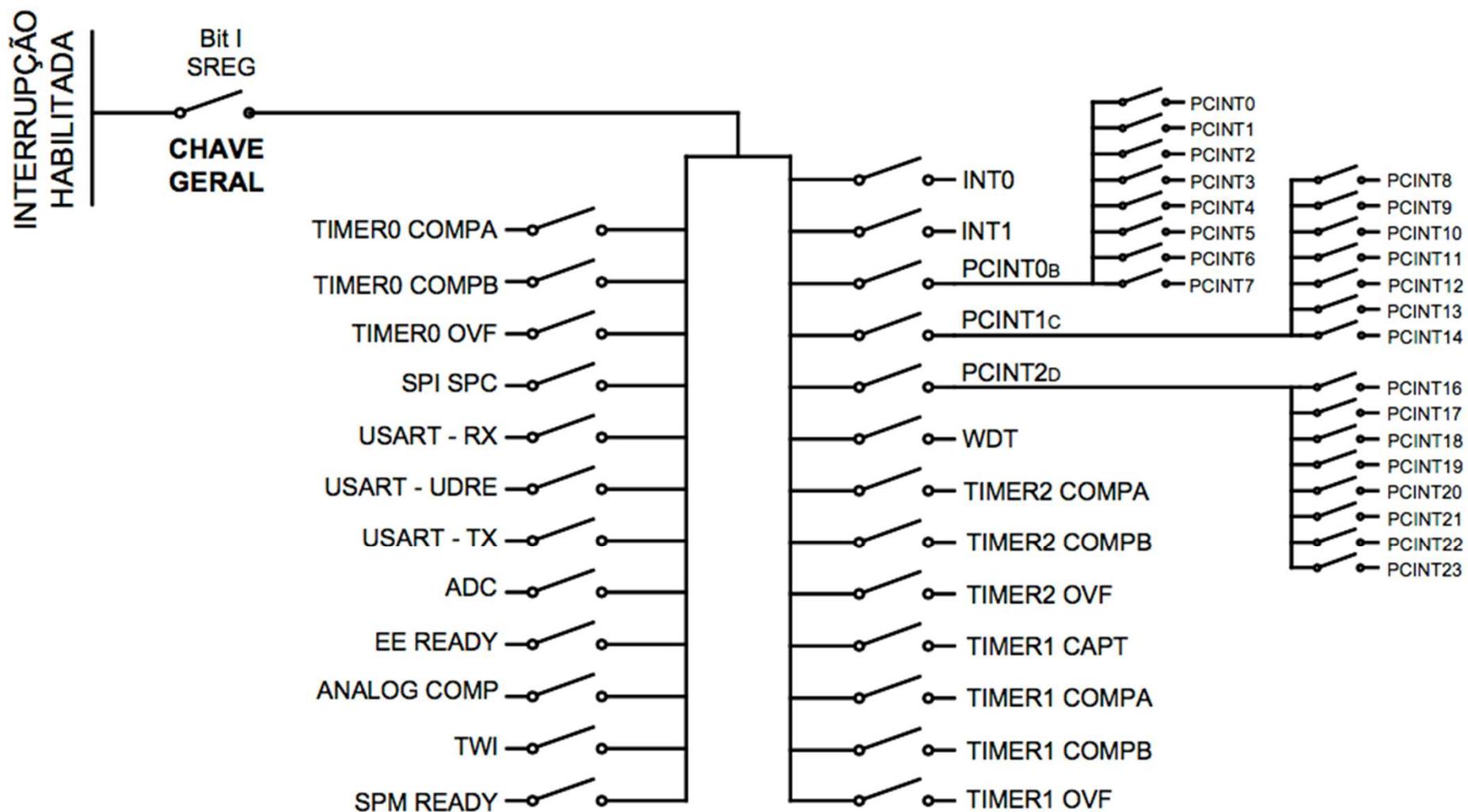
Código (função)

```
int main()          //aqui vai o programa principal
ISR(INT0_vect)    //interrupção externa 0
ISR(INT1_vect)    //interrupção externa 1
ISR(PCINT0_vect)  //interrupção 0 por mudança de pino
ISR(PCINT1_vect)  //interrupção 1 por mudança de pino
ISR(PCINT2_vect)  //interrupção 2 por mudança de pino
ISR(WDT_vect)     //estouro do temporizador Watchdog
ISR(TIMER2_COMPA_vect) //igualdade de comparação A do TC2
ISR(TIMER2_COMPB_vect) //igualdade de comparação B do TC2
ISR(TIMER2_OVF_vect) //estouro do TC2
ISR(TIMER1_CAPT_vect) //evento de captura do TC1
ISR(TIMER1_COMPA_vect) //igualdade de comparação A do TC1
ISR(TIMER1_COMPB_vect) //igualdade de comparação B do TC1
ISR(TIMER1_OVF_vect) //estouro do TC1
ISR(TIMER0_COMPA_vect) //igualdade de comparação A do TC0
ISR(TIMER0_COMPB_vect) //igualdade de comparação B do TC0
ISR(TIMER0_OVF_vect) //estouro do TC0
ISR(SPI_STC_vect)   //transferência serial completa - SPI
ISR(USART_RX_vect) //USART, recepção completa
ISR(USART_UDRE_vect) //USART, limpeza do registrador de dados
ISR(USART_TX_vect) //USART, transmissão completa
ISR(ADC_vect)      //conversão do ADC completa
ISR(EE_READY_vect) //EEPROM pronta
ISR(ANALOG_COMP_vect) //comparador analógico
ISR(TWI_vect)      //interface serial TWI
ISR(SPM_READY_vect) //armazenagem na memória de programa pronta}
```

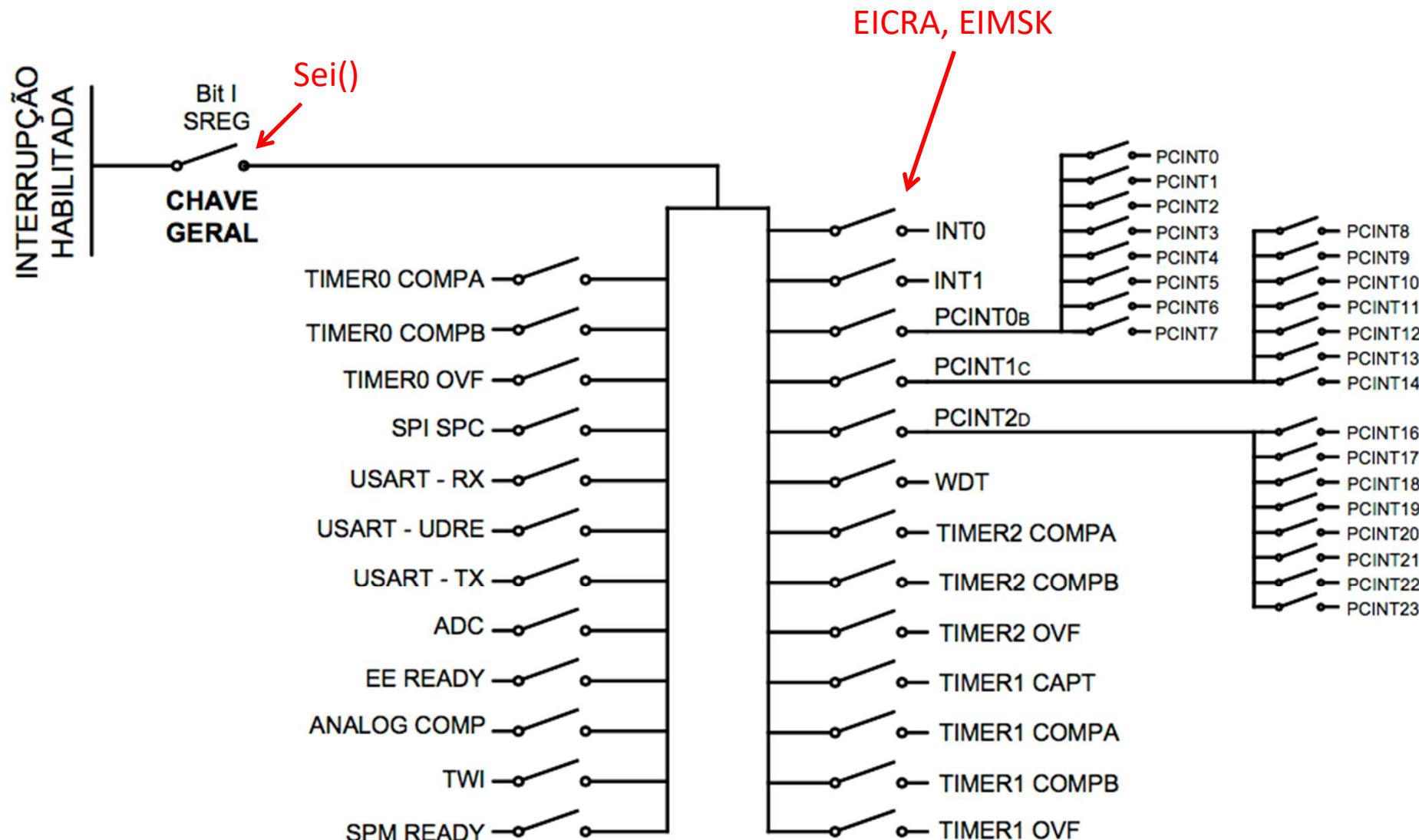
- O ATmega328 possui dois pinos dedicados para interrupções externas:
 - INT0 e INT1



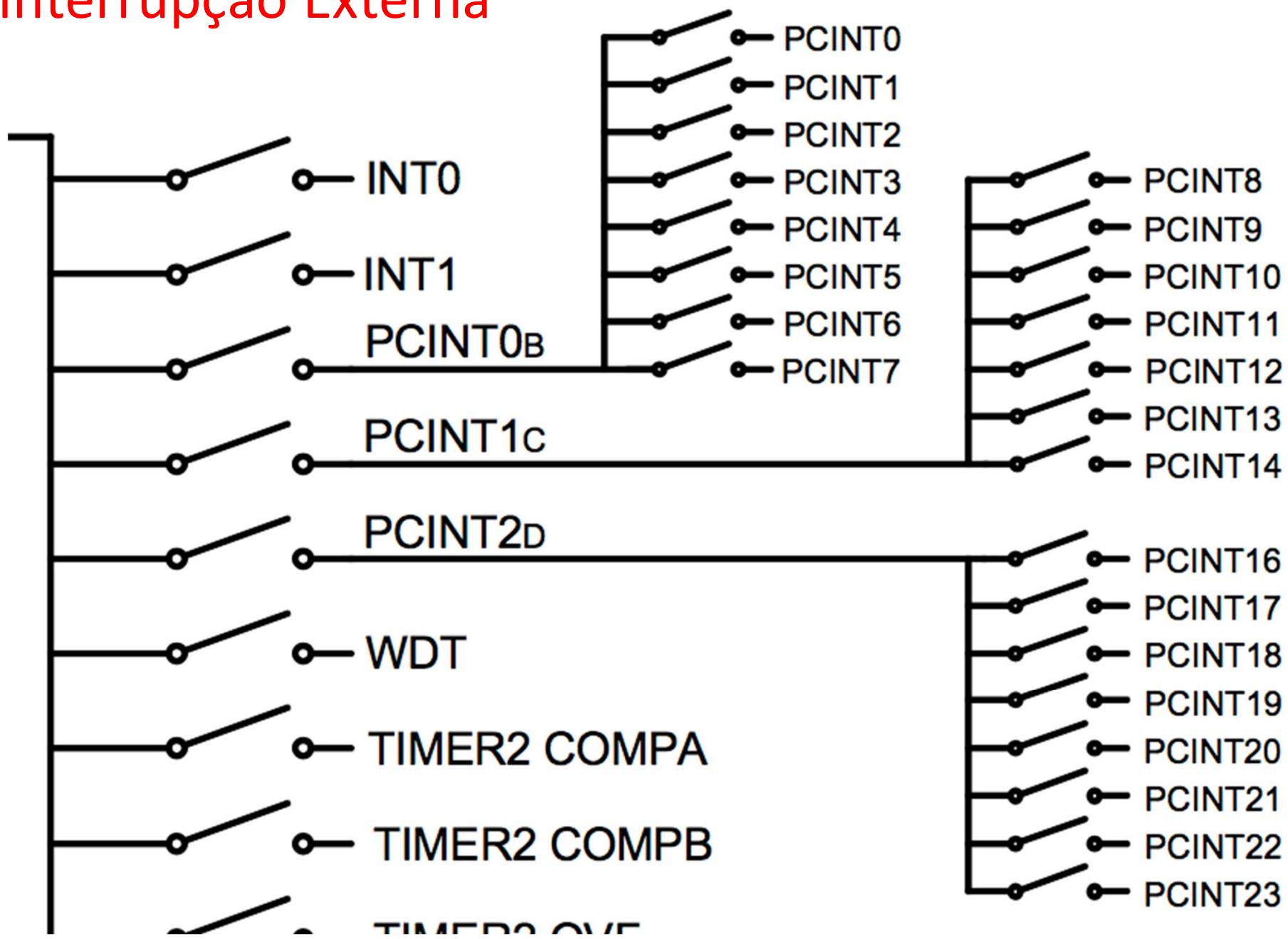
Interrupção



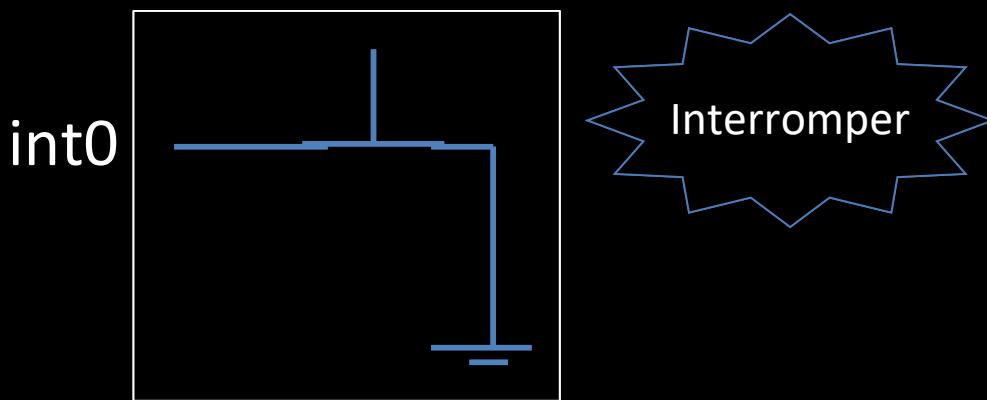
Interrupção Externa



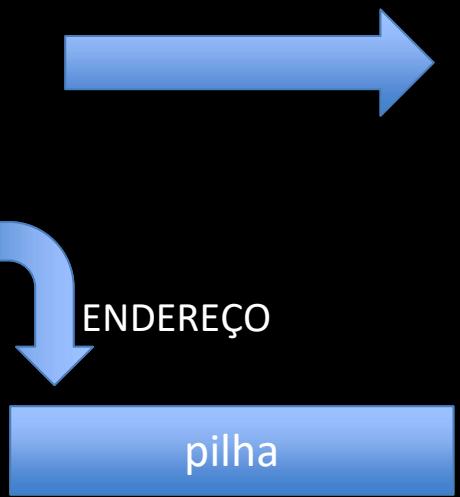
Interrupção Externa



```
void main ()  
{  
    ...  
    while (1)  
    {  
        ...  
        ...  
    }  
}
```



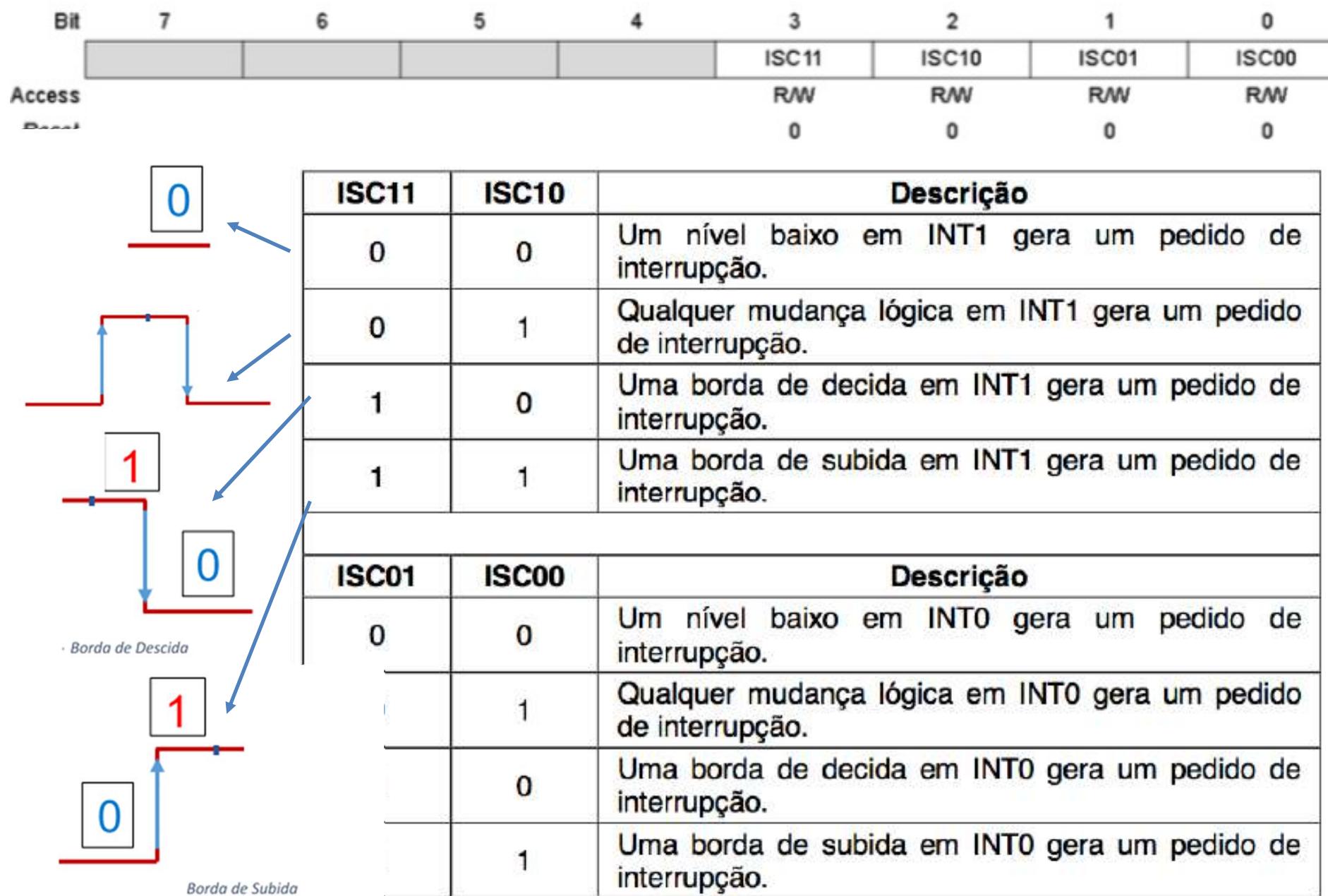
```
ISR(INT0_vect)  
{  
    ...  
    ...  
    cpl_bit(PORTB,0);  
    ...  
}
```



Registradores Interrupção Externa

- ✓ Há 5 interrupções externas:
 - INT1 e INTO;
 - PCI2, PCI1 e PCIO.
- ✓ As interrupções externas são gatilhadas a partir dos pinos INT[1:0] (INT1 e INTO), PCINT[23:16] (PCI2), PCINT[14:8] (PCI1) e PCINT[7:0] (PCIO).

Interrupção INTO E INT1 - Registrador EICRA



Registradores Interrupção Externa EIMSK

Name: EIMSK

Offset: 0x3D

Reset: 0x00

Property: When addressing as I/O Register: address offset is 0x1D

Bit	7	6	5	4	3	2	1	0
Access							INT1	INT0
Reset							0	0

- ✓ Bits 1 e 0 - INT1 e INT0 → Habilita, respectivamente, as interrupções INT1 e INT0. Se I = 1, uma interrupção ocorrerá quando um determinado evento ocorrer no respectivo pino.
- ✓ Nota: mesmo se o pino for configurado como saída, uma requisição de interrupção poderá ocorrer se uma atividade (evento configurado em EICRA) ocorrer.

Registradores Interrupção Externa PCICR

Name: PCICR
Offset: 0x68
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
Access						PCIE2	PCIE1	PCIE0
Reset						R/W	R/W	R/W
						0	0	0

- ✓ Bit 2 - PCIE2 → Habilita interrupção de mudança no pino. Qualquer mudança em qualquer um dos pinos (PCINT[23:16]) habilitados causará uma interrupção (PCI2).
- ✓ Bit 1 - PCIE1 → idem para os pinos PCINT[14:8] e interrupção PCI1.
- ✓ Bit 0 - PCIE0 → idem para os pinos PCINT[7:0] e interrupção PCI0.

Registradores Interrupção Externa PCIFR

Name: PCIFR

Offset: 0x3B

Reset: 0x00

Property: When addressing as I/O Register: address offset is 0x1B

Bit	7	6	5	4	3	2	1	0
Access						PCIF2	PCIF1	PCIF0
Reset						R/W	R/W	R/W

- ✓ Bit 2 - PCIF2 → Indica a ocorrência da mudança de sinal em um dos pinos PCINT[23:16].
- ✓ Bit 1 - PCIF1 → Indica a ocorrência da mudança de sinal em um dos pinos PCINT[14:8].
- ✓ Bit 0 - PCIFO → Indica a ocorrência da mudança de sinal em um dos pinos PCINT[7:0].
 - Esses flags são zerados automaticamente quando a ISR for executada ou, alternativamente, via software escrevendo '1'.

Registradores Interrupção Externa PCMSK2

Name: PCMSK2

Offset: 0x6D

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
Access	R/W							

Reset 0 0 0 0 0 0 0 0

- ✓ Seleciona se a interrupção de mudança no pino está habilitada no correspondente pino de E/S - PCI2.
- ✓ Ex.: se PCINT16 = 1, qualquer mudança nesse pino ($1 \rightarrow 0$ ou $0 \rightarrow 1$), gerará uma interrupção se PCIE2 de PCICR for '1' e I=1. O bit PCIF2 de PCIFR será setado.

Registradores Interrupção Externa PCMSK1

Name: PCMSK1
Offset: 0x6C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
Access		PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
Reset	0	0	0	0	0	0	0	0

- ✓ Seleciona se a interrupção de mudança no pino está habilitada no correspondente pino de E/S - PCI1.
- ✓ Ex.: se PCINT14 = 1, qualquer mudança nesse pino ($1 \rightarrow 0$ ou $0 \rightarrow 1$), gerará uma interrupção se PCIE1 de PCICR for '1' e I=1. O bit PCIF1 de PCIFR será setado.

Registradores Interrupção Externa PCMSK0

Name: PCMSK0

Offset: 0x6B

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
Access	R/W							
Reset	0	0	0	0	0	0	0	0

- ✓ Seleciona se a interrupção de mudança no pino está habilitada no correspondente pino de E/S - PCIO.
- ✓ Ex.: se PCINT3 = 1, qualquer mudança nesse pino ($1 \rightarrow 0$ ou $0 \rightarrow 1$), gerará uma interrupção se PCIE0 de PCICR for '1' e I=1. O bit PCIFO de PCIFR será setado.

Exemplo de uma operação das duas interrupções externas, INT0 e INT1

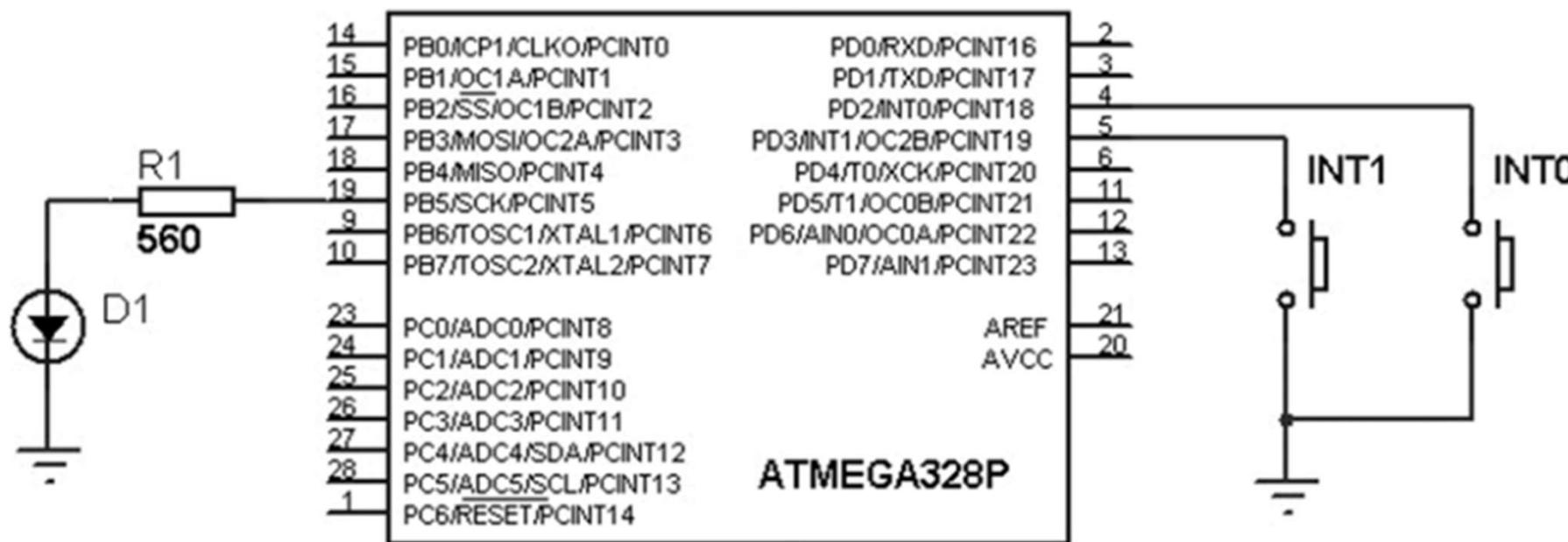


Fig. 6.2 – Circuito paraemploi das interrupções externas INT0 e INT1.

INT0_1.c

```
//===== //  
// HABILITANDO AS INTERRUPÇÕES INT0 e INT1 POR TRANSIÇÃO E NÍVEL, RESPECTIVAMENTE //  
//===== //  
#define F_CPU 16000000UL  
#include <avr/io.h>  
#include <util/delay.h>  
#include <avr/interrupt.h>  
  
//Definições de macros - empregadas para o trabalho com bits  
#define set_bit(Y,bit_x) (Y|=(1<<bit_x)) //ativa o bit x da variável  
#define clr_bit(Y,bit_x) (Y&=~(1<<bit_x)) //limpa o bit x da variável  
#define tst_bit(Y,bit_x) (Y&(1<<bit_x)) //testa o bit x da variável  
#define cpl_bit(Y,bit_x) (Y^=(1<<bit_x)) //troca o estado do bit x  
  
#define LED PB5 //LED está no pino PB5
```

```
ISR(INT0_vect);
ISR(INT1_vect);

//-----
int main()
{
    DDRD = 0x00;          //PORTD entrada
    PORTD = 0xFF;          //pull-ups habilitados
    DDRB = 0b00100000;    //somente pino do LED como saída
    PORTB = 0b11011111;   //desliga LED e habilita pull-ups

    UCSR0B = 0x00;        /*necessário desabilitar RX e TX para trabalho com os pinos
                           do PORTD no Arduino*/
    EICRA = 1<<ISC01;//interrupções externas: INT0 na borda de descida, INT1 no nível zero.
    EIMSK = (1<<INT1) | (1<<INT0); //habilita as duas interrupções
    sei();                //habilita interrupções globais, ativando o bit I do SREG

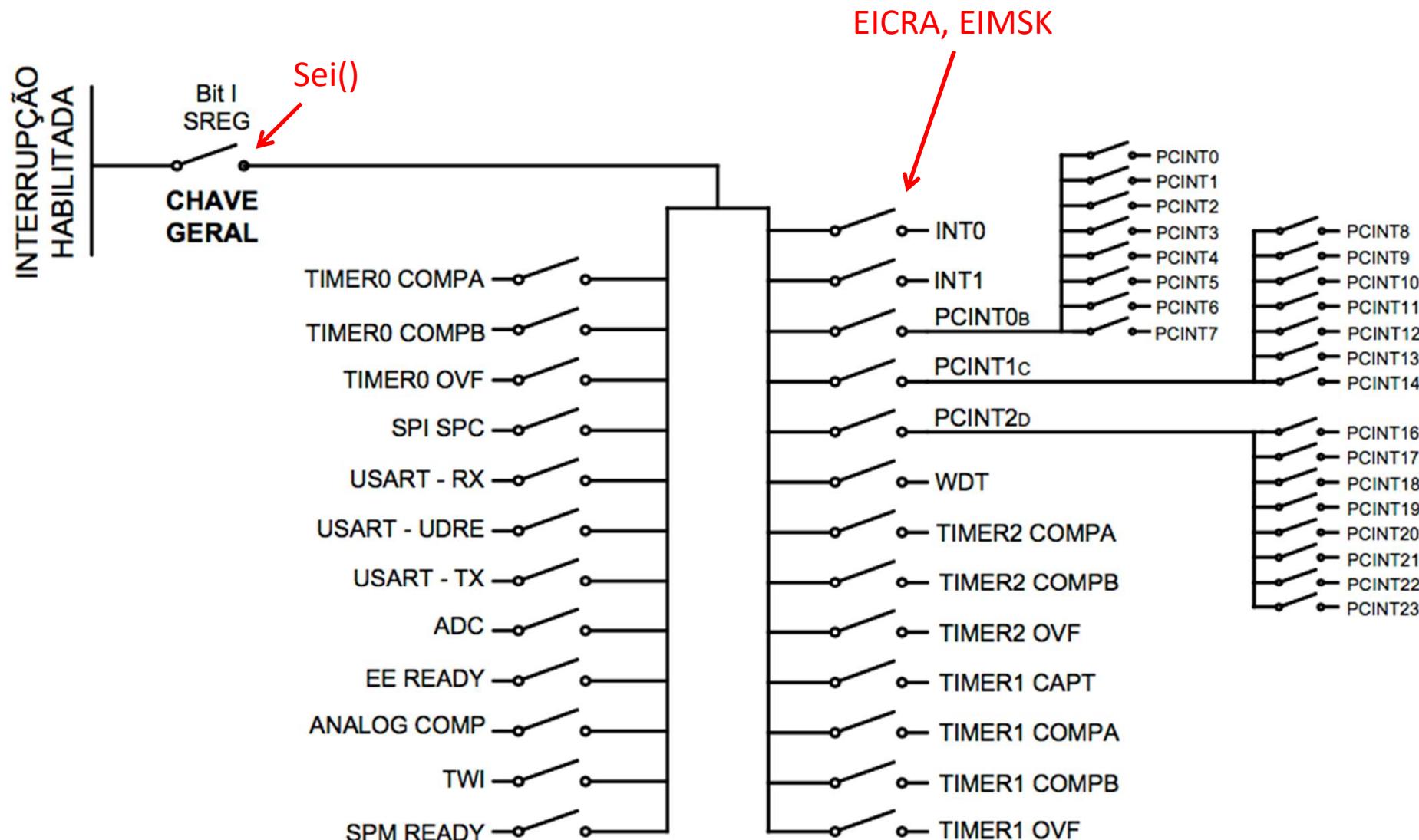
    while(1){}
}

//-----
ISR(INT0_vect) //interrupção externa 0, quando o botão é pressionado o LED troca de estado
{
    cpl_bit(PORTB,LED);
}

//-----
ISR(INT1_vect) //interrupção externa 1, mantendo o botão pressionado o LED pisca
{
    cpl_bit(PORTB,LED);
    _delay_ms(200);      //tempo para piscar o LED
}

//=====
```

Interrupção Externa



Interrupções no Atmega328

Vetor	Endereço	Fonte	Definição
1	0x0000	RESET	Reset Ext., Power-On Reset,
2	0x0002	INT0	Int. Externa no pino INT0
3	0x0004	INT1	Int. Externa no pino INT1
4	0x0006	PCINT0	Mudança estado pino Req. 0
5	0x0008	PCINT1	Mudança estado pino Req. 1
6	0x000A	PCINT2	Mudança estado pino Req. 2
7	0x000C	WDT	Watchdog Time-out
8	0x000E	TIMER2 COMPA	T/C2 Compare Match A
9	0x0010	TIMER2 COMPB	T/C2 Compare Match B
10	0x0012	TIMER2 OVF	T/C2 Overflow
11	0x0014	TIMER1 CAPT	T/C1 Evento de Captura
12	0x0016	TIMER1 COMPA	T/C1 Compare Match A
13	0X0018	TIMER1 COMPB	T/C1 Compare Match B
14	0X001A	TIMER1 OVF	T/C1 Overflow
15	0x001C	TIMER0 COMPA	T/C0 Compare Match A
16	0x001E	TIMER0 COMPB	T/C0 Compare Match B
17	0x0020	TIMER0 OVF	T/C0 Overflow
18	0x0022	SPI, STC	SPI Transfer. completa
19	0x0024	USART, RX	USART Recep. completa
20	0x0026	USART, UDRE	USART Reg. dados vazio
21	0x0028	USART, TX	USART Transm. completa
22	0x002A	ADC	Conv. A/D completada
23	0x002C	EE_READY	EEPROM Pronta
24	0X002E	ANALOG COMP	Comparador Analógico
25	0x0030	TWI	2-Wire serial interface
26	0x0032	SPM READY	Memória Flash Pronta

Vetores de Interrupções no Atmega328 em C

Código (função)

```
int main()          //aqui vai o programa principal
ISR(INT0_vect)    //interrupção externa 0
ISR(INT1_vect)    //interrupção externa 1
ISR(PCINT0_vect)  //interrupção 0 por mudança de pino
ISR(PCINT1_vect)  //interrupção 1 por mudança de pino
ISR(PCINT2_vect)  //interrupção 2 por mudança de pino
ISR(WDT_vect)     //estouro do temporizador Watchdog
ISR(TIMER2_COMPA_vect) //igualdade de comparação A do TC2
ISR(TIMER2_COMPB_vect) //igualdade de comparação B do TC2
ISR(TIMER2_OVF_vect) //estouro do TC2
ISR(TIMER1_CAPT_vect) //evento de captura do TC1
ISR(TIMER1_COMPA_vect) //igualdade de comparação A do TC1
ISR(TIMER1_COMPB_vect) //igualdade de comparação B do TC1
ISR(TIMER1_OVF_vect) //estouro do TC1
ISR(TIMER0_COMPA_vect) //igualdade de comparação A do TC0
ISR(TIMER0_COMPB_vect) //igualdade de comparação B do TC0
ISR(TIMER0_OVF_vect) //estouro do TC0
ISR(SPI_STC_vect)   //transferência serial completa - SPI
ISR(USART_RX_vect)  //USART, recepção completa
ISR(USART_UDRE_vect) //USART, limpeza do registrador de dados
ISR(USART_TX_vect)  //USART, transmissão completa
ISR(ADC_vect)       //conversão do ADC completa
ISR(EE_READY_vect)  //EEPROM pronta
ISR(ANALOG_COMP_vect) //comparador analógico
ISR(TWI_vect)        //interface serial TWI
ISR(SPM_READY_vect) //armazenagem na memória de programa pronta}
```

Interrupções Temporizadores ou Contadores

Muitos programas usam cronómetros miniatura, que são SFR de 8 ou 16 bits, e o seu conteúdo é automaticamente incrementado a cada pulso recebido. Quando um registo chegar ao fim da contagem (255 ou 65535), uma Interrupção é gerada.



```
void main ()
```

```
{
```

```
    TCCR0B = 0b00000011;
```

```
    sei();
```

```
    while (1)
```

```
{
```

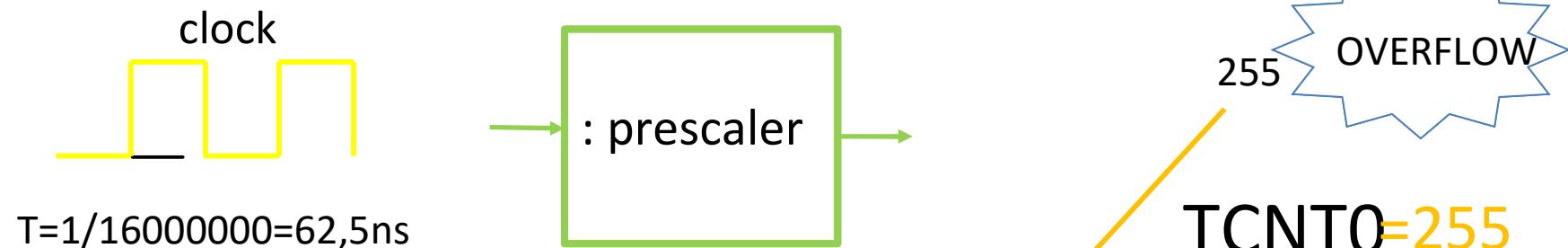
```
    ...
```

```
    ...
```

```
    ...
```

```
}
```

```
}
```



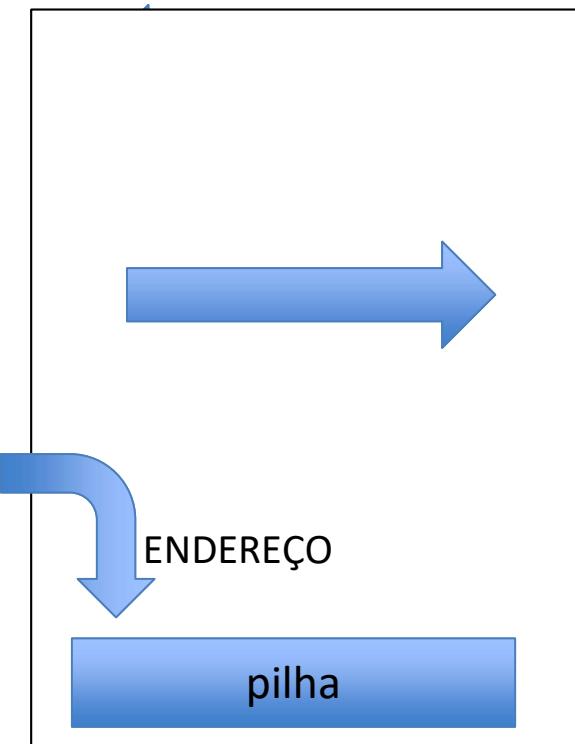
$$\text{TEMPO_OVERFLOW}=62,5\text{ns}*(255+1)*1024=16,384\text{ms}$$

```
ISR(TIMERO_OVF_vect)
```

```
{
```

```
    cpl_bit(PORTB,0);
```

```
}
```



FUNCIONAMENTO TEMPORIZADOR COM PRESCALER

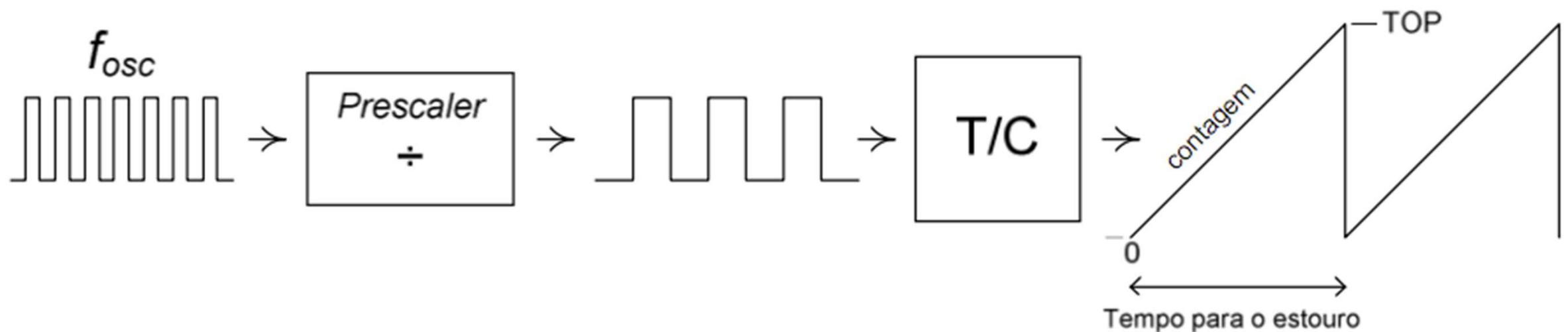


Fig. 9.1 – Funcionamento de um temporizador/contador do ATmega.

TEMPO DE ESTOURO:

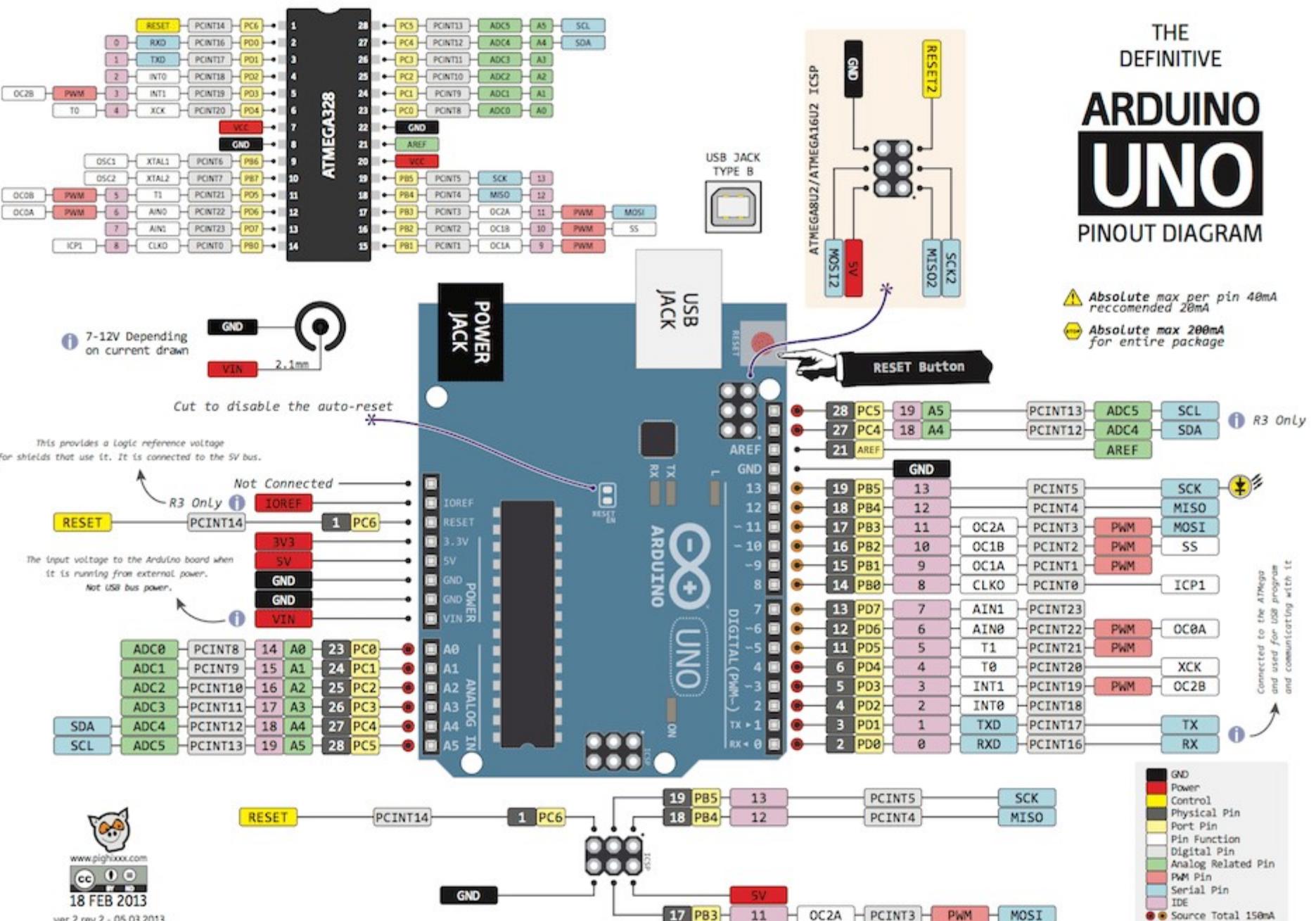
$$t_{estouro} = \frac{(TOP+1) \times prescaler}{f_{osc}} \quad (9.1)$$

Exemplos:

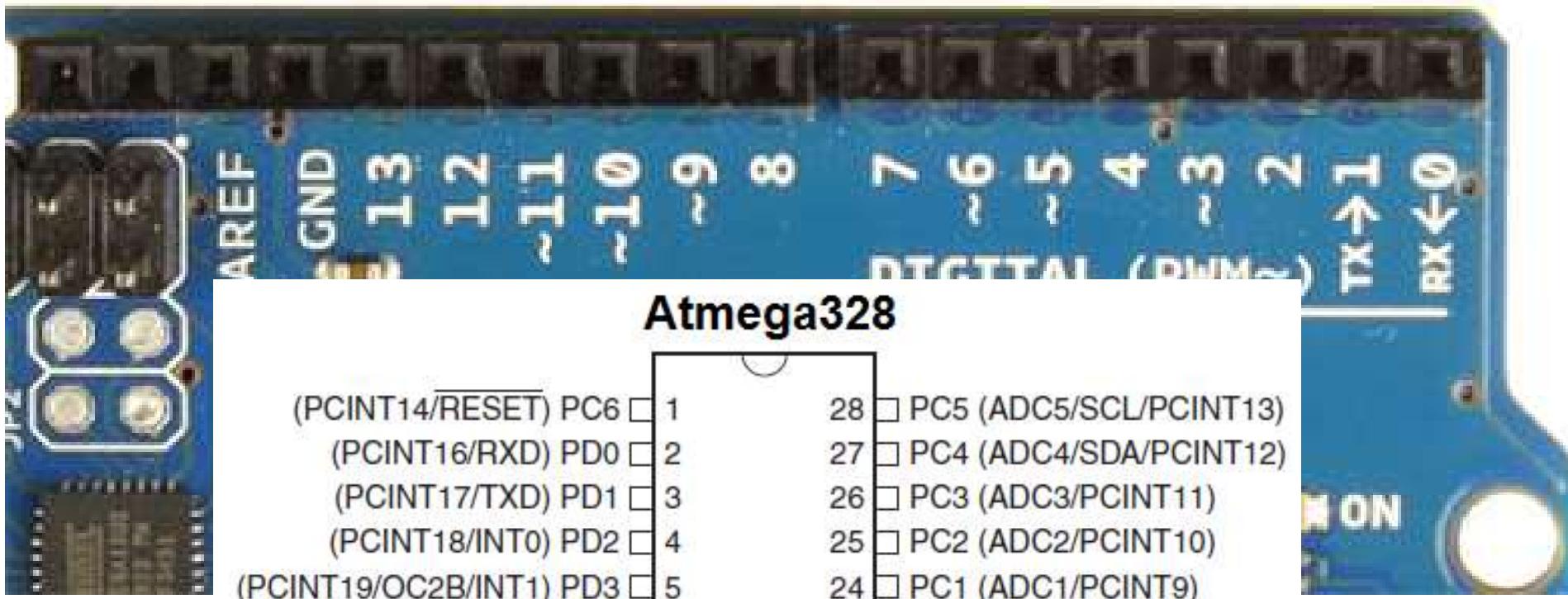
1. Supondo um TC de 8 bits (conta até o valor TOP de 255), trabalhando com uma frequência de 1 MHz, sem divisor de frequência, o tempo para o seu estouro é de:

$$t_{estouro} = 256 \times \frac{1}{1 \text{ MHz}} \times 1 = 256 \text{ } \mu\text{s}$$

THE
DEFINITIVE
ARDUINO
UNO
PINOUT DIAGRAM



Pinos PWM (OCxA e OCxB)

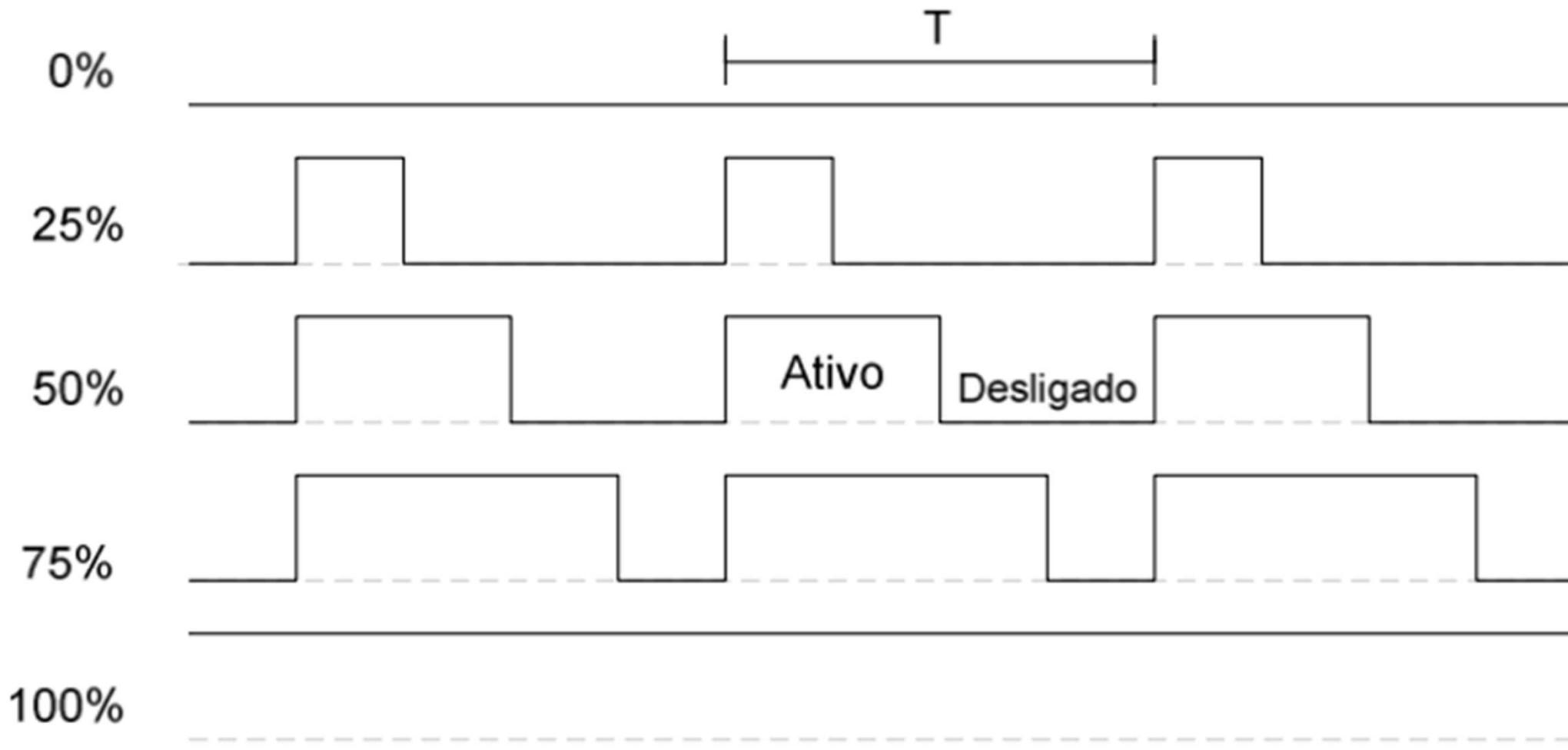


Timer 0:
PD6: OC0A
PD5: OC0B

Timer 1:
PB1: OC1A
PB2: OC1B

Timer 2:
PB3: OC2A
PD3: OC2B

PWM



TEMPO DE ESTOURO:

$$t_{estouro} = \frac{(TOP+1) \times prescaler}{f_{osc}} \quad (9.1)$$

2. Supondo um TC de 16 bits (conta até o valor TOP de 65535), trabalhando com uma frequência de 16 MHz, com divisor de frequência de 64, o tempo para o seu estouro é de:

$$t_{estouro} = 65536 \times \frac{1}{16 \text{ MHz}} \times 64 = 0,262 \text{ s}$$

Timer 0

Temporizador ou Contador

- Se a origem do sinal de clock é o Temporizador de Pre-Divisão, a operação do T/C0 será como **Temporizador**;
- Se a origem deste sinal de clock é o pino T0, a operação será como **Contador** de Eventos;
- A seleção do modo de operação, é feita através dos bits: CS02, CS01, e CS00.

Temporizador/Contador 0 de 8 Bits

Características Principais do T/C0:

- Temporizador/Contador de Eventos Bidirecional de 8 bits;
- Pré-Divisão (Prescaler) do clock da CPU de 10 bits;
- Unidade de Comparação de Valores c/ Recarga Automática;
- Gerador de Frequências;
- Saída PWM c/ Resolução de 8 bits;
- Duas fontes de interrupção: Overflow e Comparação

9.3.1 REGISTRADORES DO TCO

O controle do modo de operação do TCO é feito nos registradores TCCR0A e TCCR0B (Timer/Counter Control 0 Register A e B).

TCCR0B – Timer/Counter 0 Control Register B

TIFR0 – Timer/Counter 0 Interrupt Flag Register

TIMER 0

✓ Exemplo de código - Timer 0 no modo normal.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 16000000UL

ISR(TIMER0_OVF_vect)
{
    PORTB^=(1<<PB5);
}

#define TIMER0_OVF_vect_num 16
#define TIMER0_OVF_vect _VECTOR(16) /* Timer/Counter0 Overflow */

int main(void)
{
    DDRB = 0x20; //
    TIFR0 = 0x07; //
    TIMSK0 = 0x01, // sei(); //

    PORTB = 0x00; /
    TCNT0 = 0x00; /

    TCCR0A = 0x00;
    TCCR0B = 0x05;

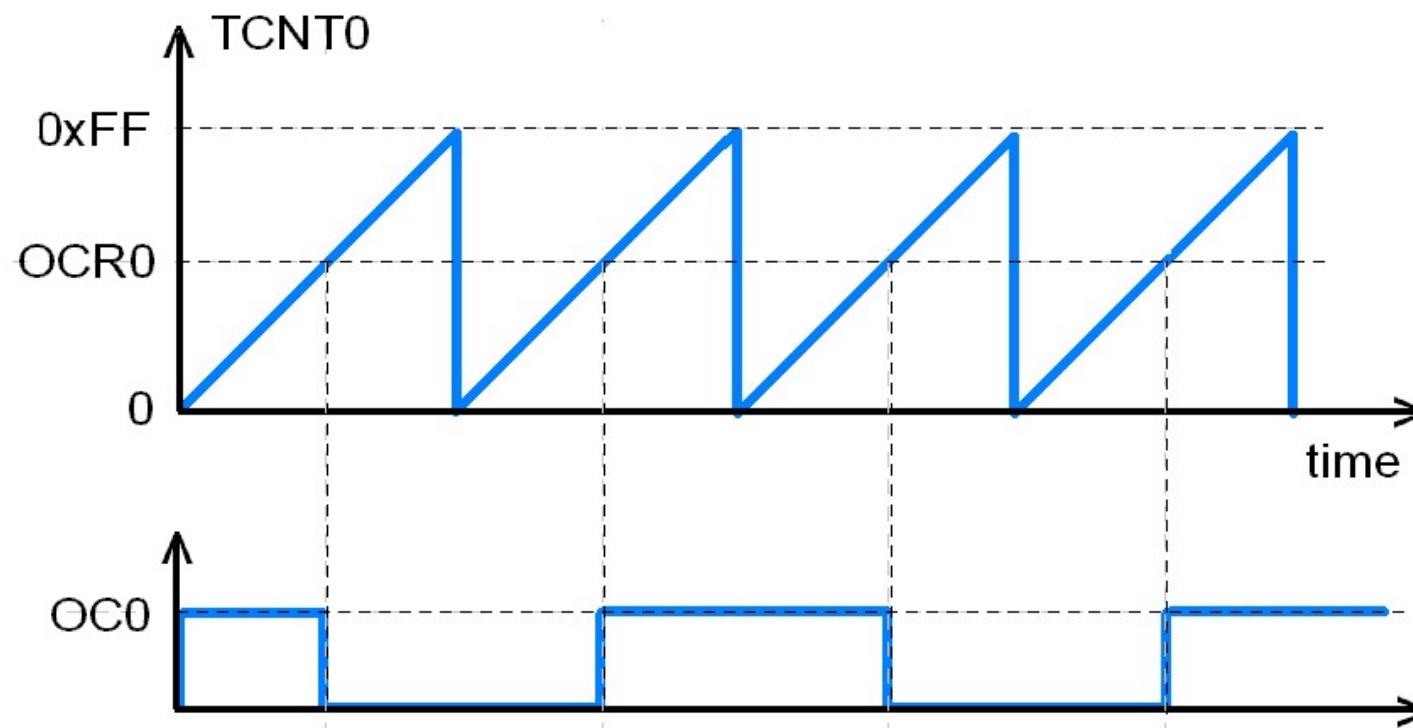
    while (1)
    {
    }
}
```

The code snippet shows the setup for Timer 0 in normal mode. It includes header files for I/O and interrupts, defines the CPU frequency, and sets up the ISR for Timer 0 overflow. The ISR toggles port B pin 5. It also defines constants for the vector number and overflow vector. The main function initializes pins, enables the timer overflow interrupt, and enters an infinite loop. Arrows point from the highlighted code sections to their corresponding definitions in the included header files: `interrupt.h`, `inttypes.h`, `io.h`, and `iom328p.h`.

Timer 0 - Modo Normal:

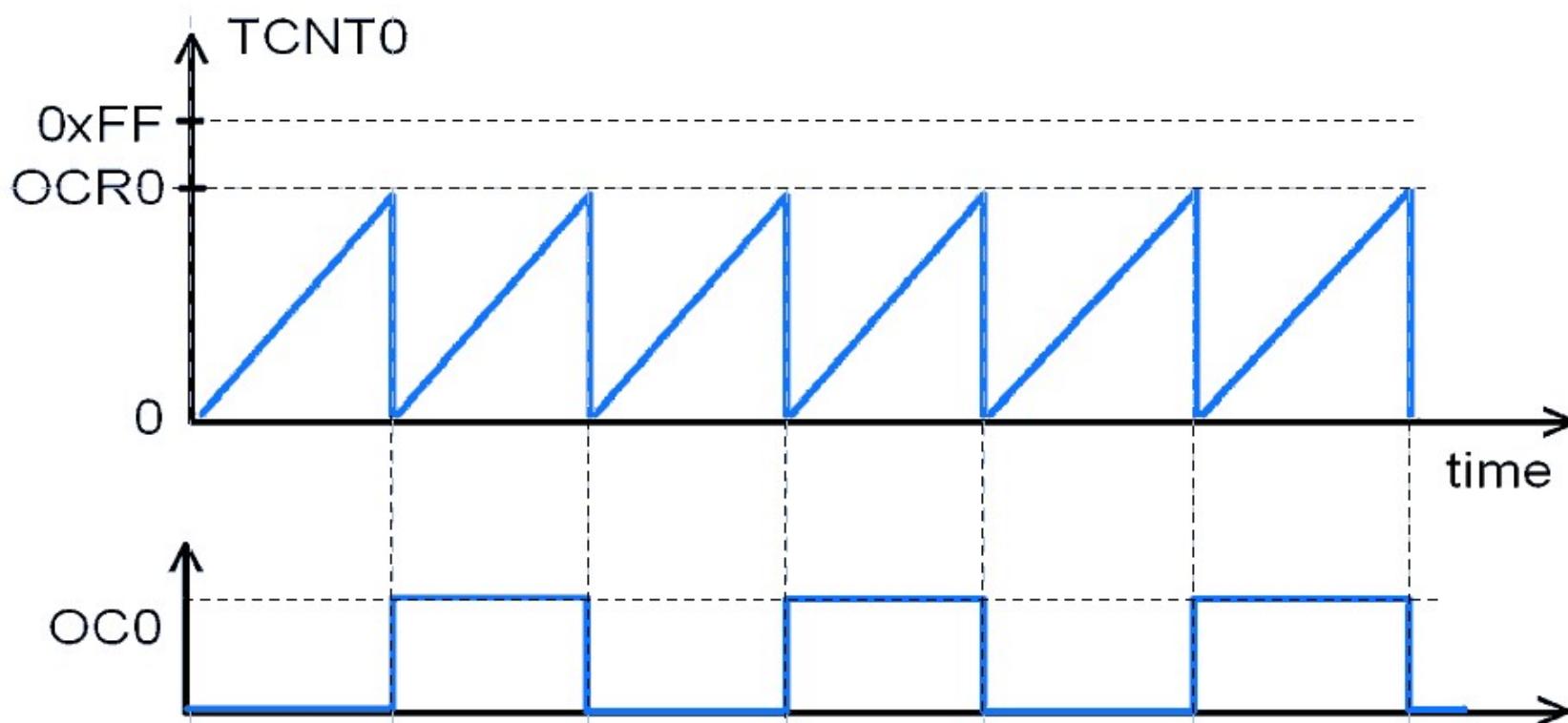
Contagem crescente até valor máximo (255),
sem recarga automática;

- O contador sempre incrementará;
- Interrupção quando atingir o topo;
- Interrupção quando $\text{TCNT} == \text{OCR}$;
- Pode setar/resetar/toggle o pino OC quando $\text{TCNT} == \text{OCR}$;



Timer 0 - Modo CTC

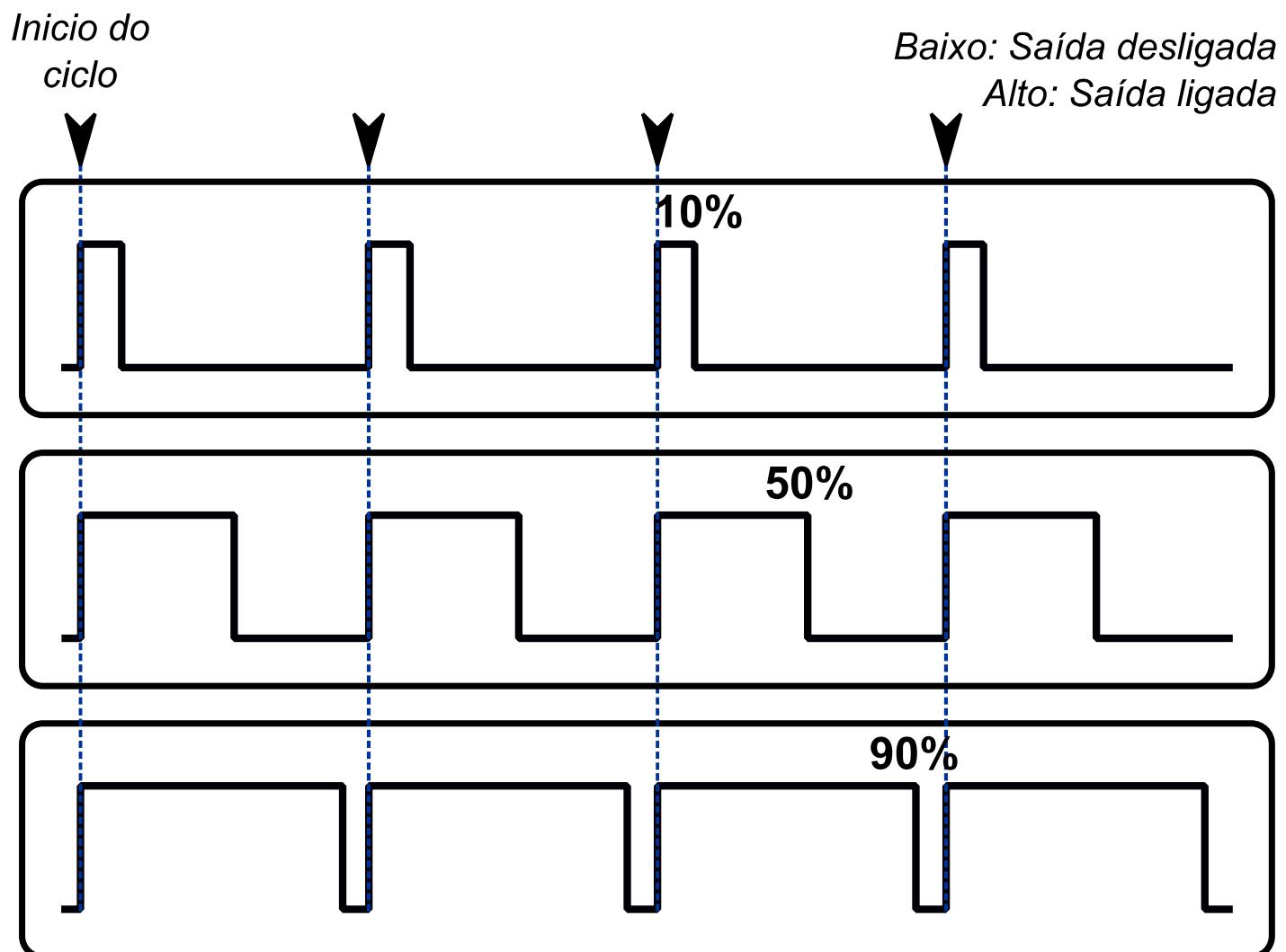
- O contador sempre incrementará;
 - Recomeça a contagem quando TCNT == OCR;
 - Usado para geração de onda quadrada no pino OC (quando habilitado como pino de saída);
 - Adequado para aplicações regulação de potência, retificação e outras usando DACs;



Sinal de PWM

- Um sinal PWM é uma onda quadrada com período definido
- Onda quadrada é um sinal que permanece num nível baixo durante um tempo e num nível alto durante um tempo.
 - Um sinal de clock é um tipo de onda quadrada
- No PWM os tempos em nível baixo/nível alto são complementares.
- O controle do PWM visa, principalmente, controlar esses tempos
- A razão entre o tempo ligado e o tempo do ciclo é conhecida como duty cycle

Sinal de PWM (exemplo)

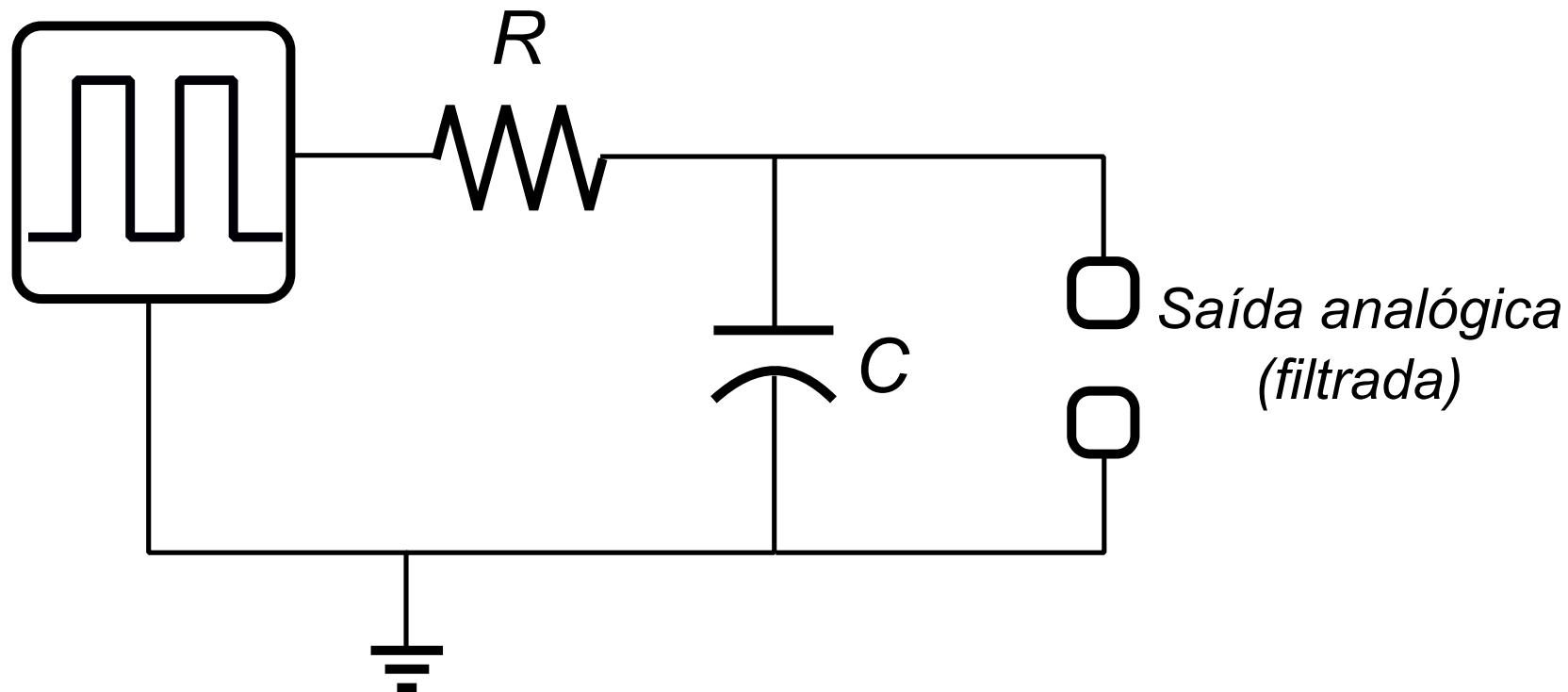


Sinal de PWM x Saída Analógica

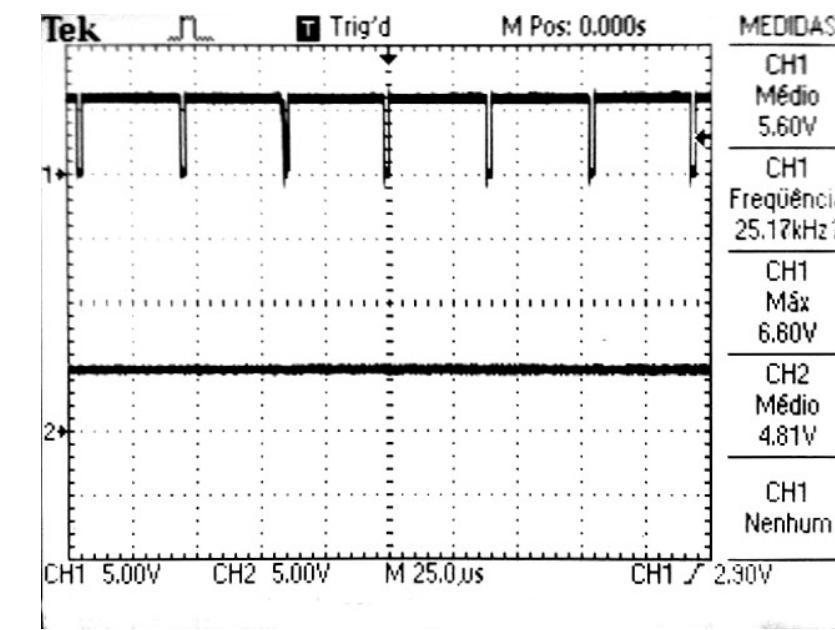
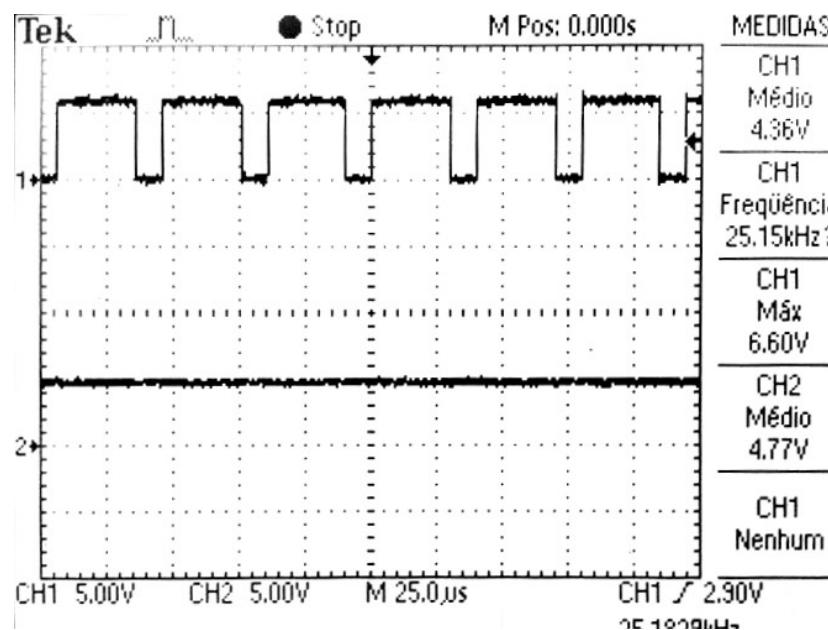
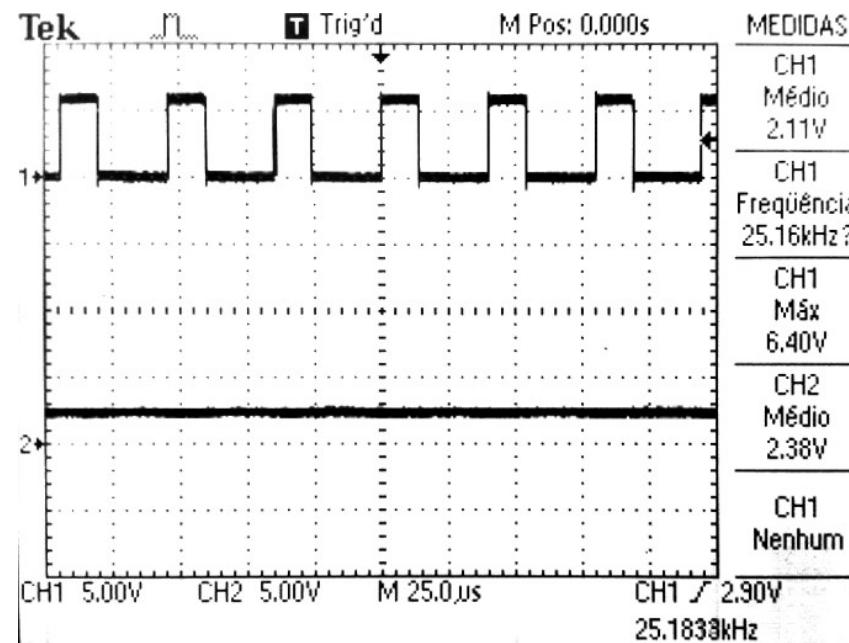
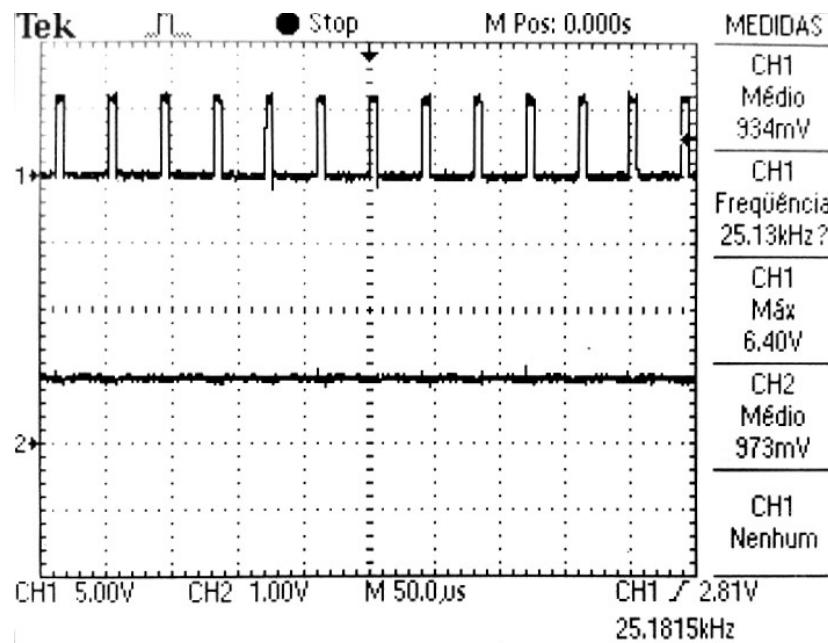
- Vantagens
 - Mais barato que uma saída analógica
 - Pode ser transformado numa saída analógica através de um filtro passa baixa
 - Para algumas situações o filtro não é necessário
 - Quando o dispositivo a ser controlado já possui um filtro passa baixa, normalmente por causa da sua construção física. Ex: aquecimento por resistências
- Desvantagens
 - Não é uma saída analógica

Sinal de PWM x Saída Analógica

Saída PWM



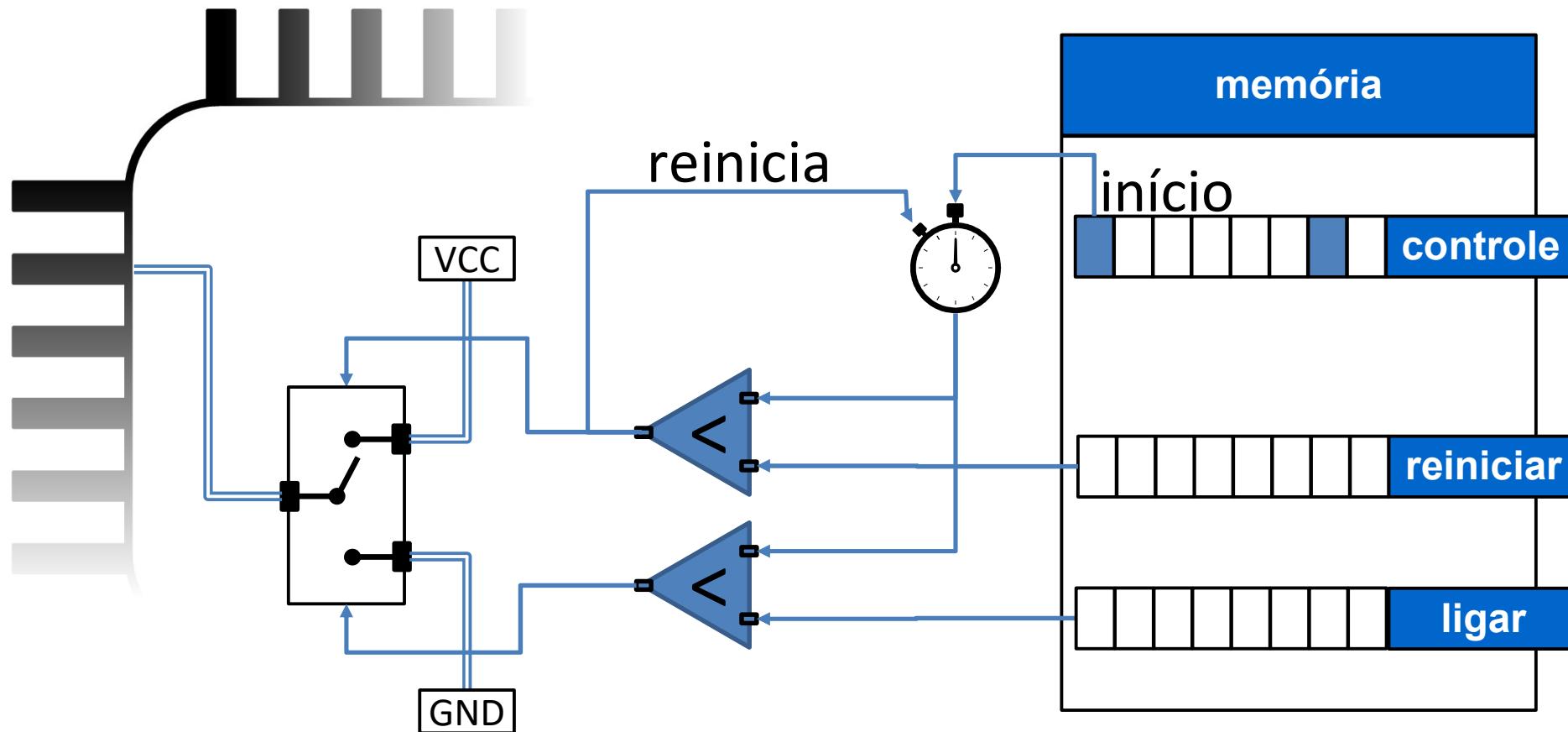
Conversão PWM em analógico



Soft PWM

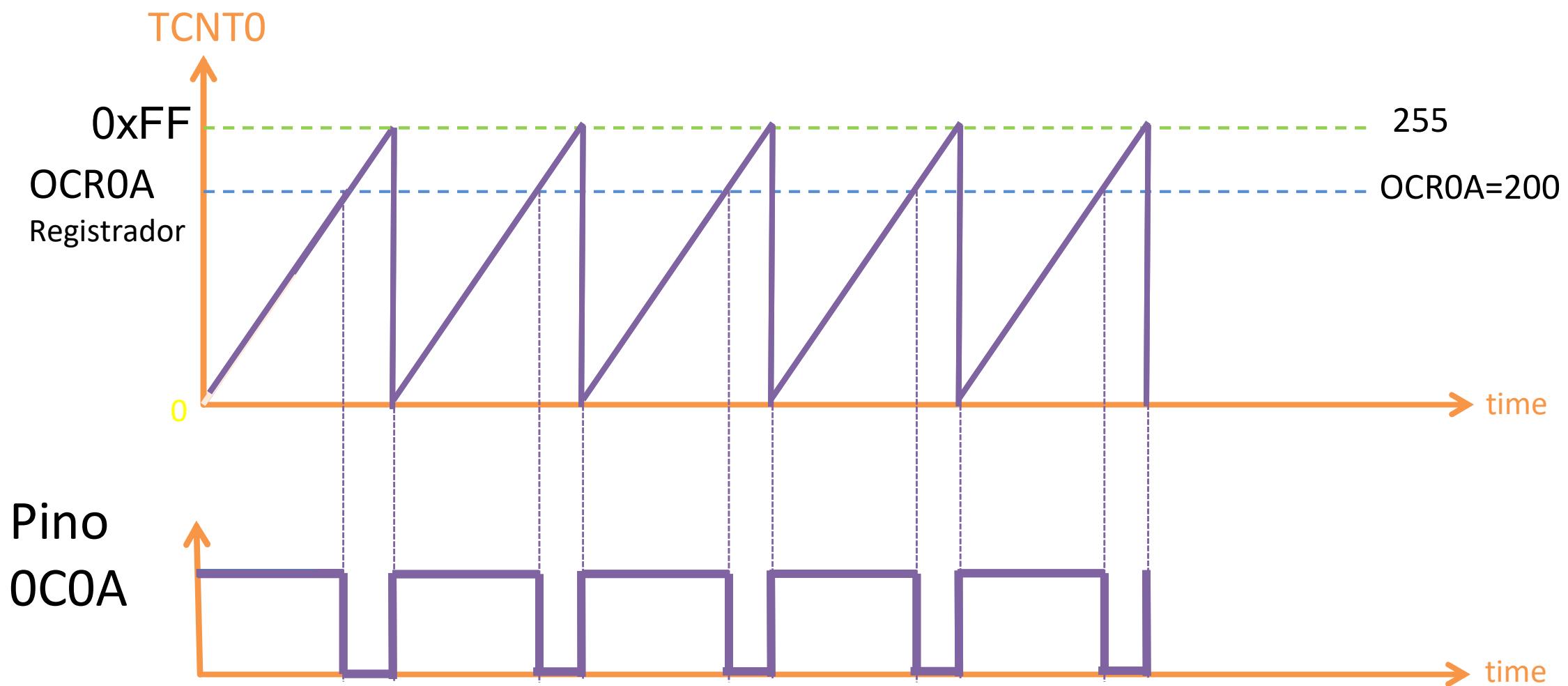
1. Inicialização do terminal de saída
2. Escolha do tempo de ciclo
3. Escolha do duty cycle
4. Liga-se a saída e inicia-se contagem de tempo
5. Quando o tempo for maior que o duty cycle desliga a saída
6. Quando o tempo acabar retorna ao passo 4

PWM por hardware



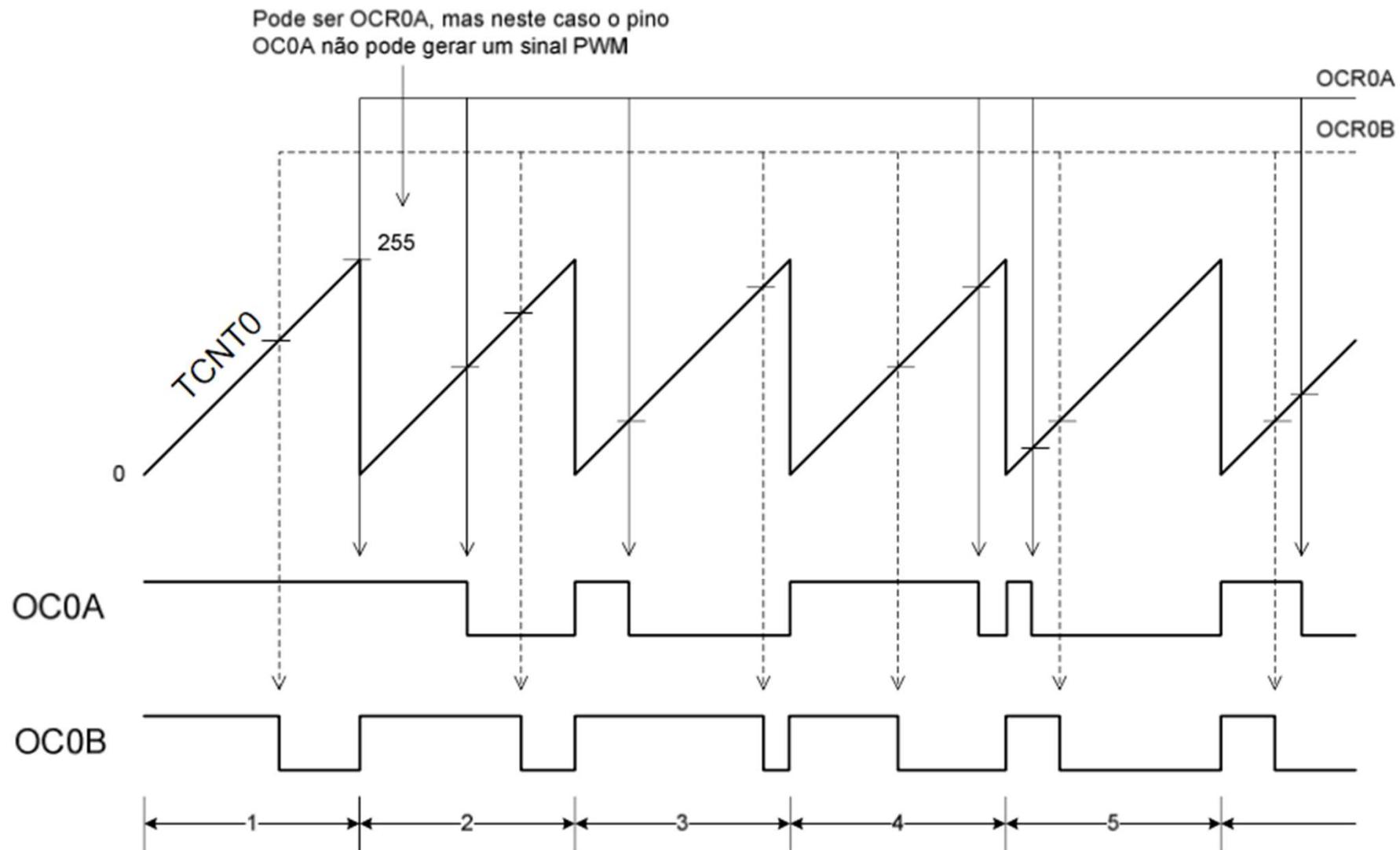
Timer 0 - Modo PWM Rápido

Permite a geração de um sinal PWM de alta frequência. O contador conta de zero até o valor máximo e volta a zero. Na comparação (invertida ou não invertida) Os pinos OCROA e OCBR0B geram sinal PWM invertendo na igualdade da comparação e no reinício da contagem.



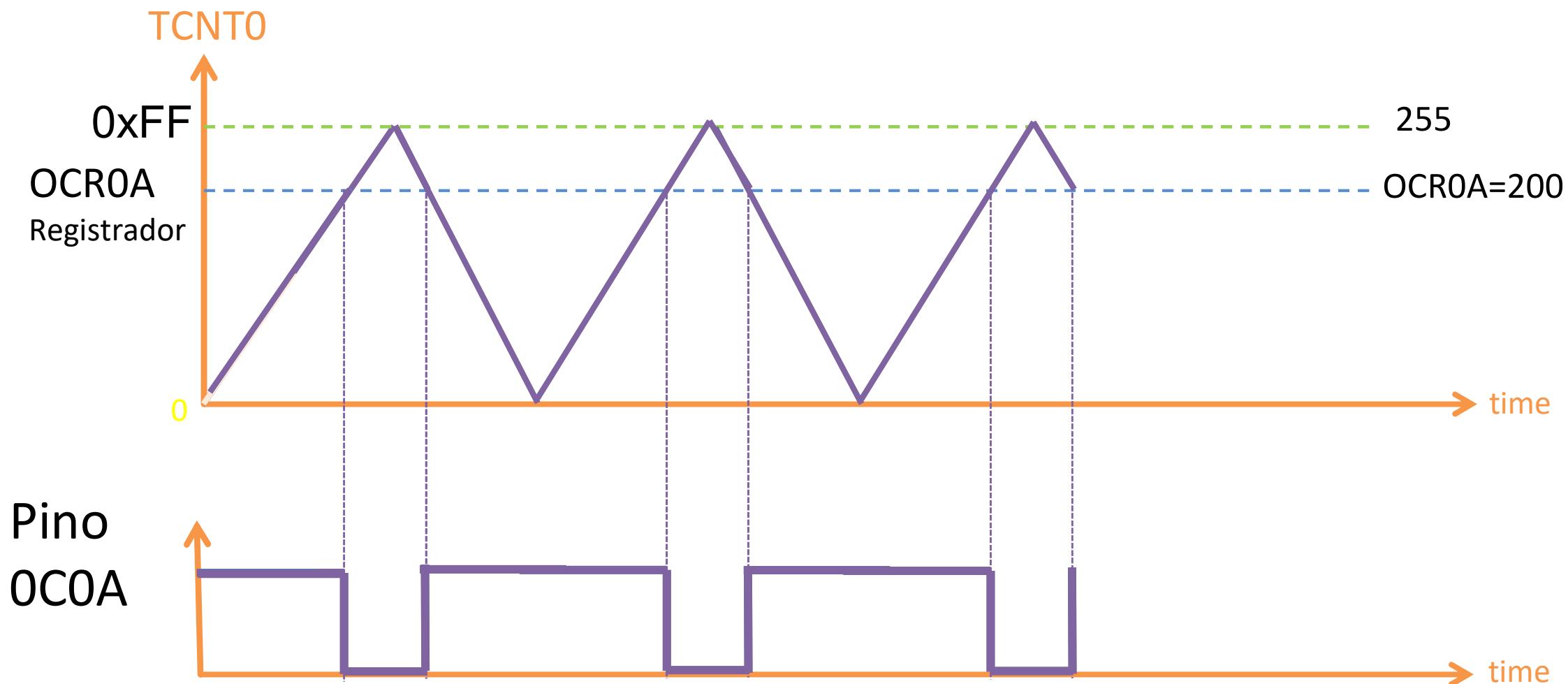
Timer 0 - Modo PWM Rápido

: Geração de sinal PWM no pino OC0, c/ frequências de PWM cerca de duas vezes mais altas, que no PWM Corrigido em Fase;



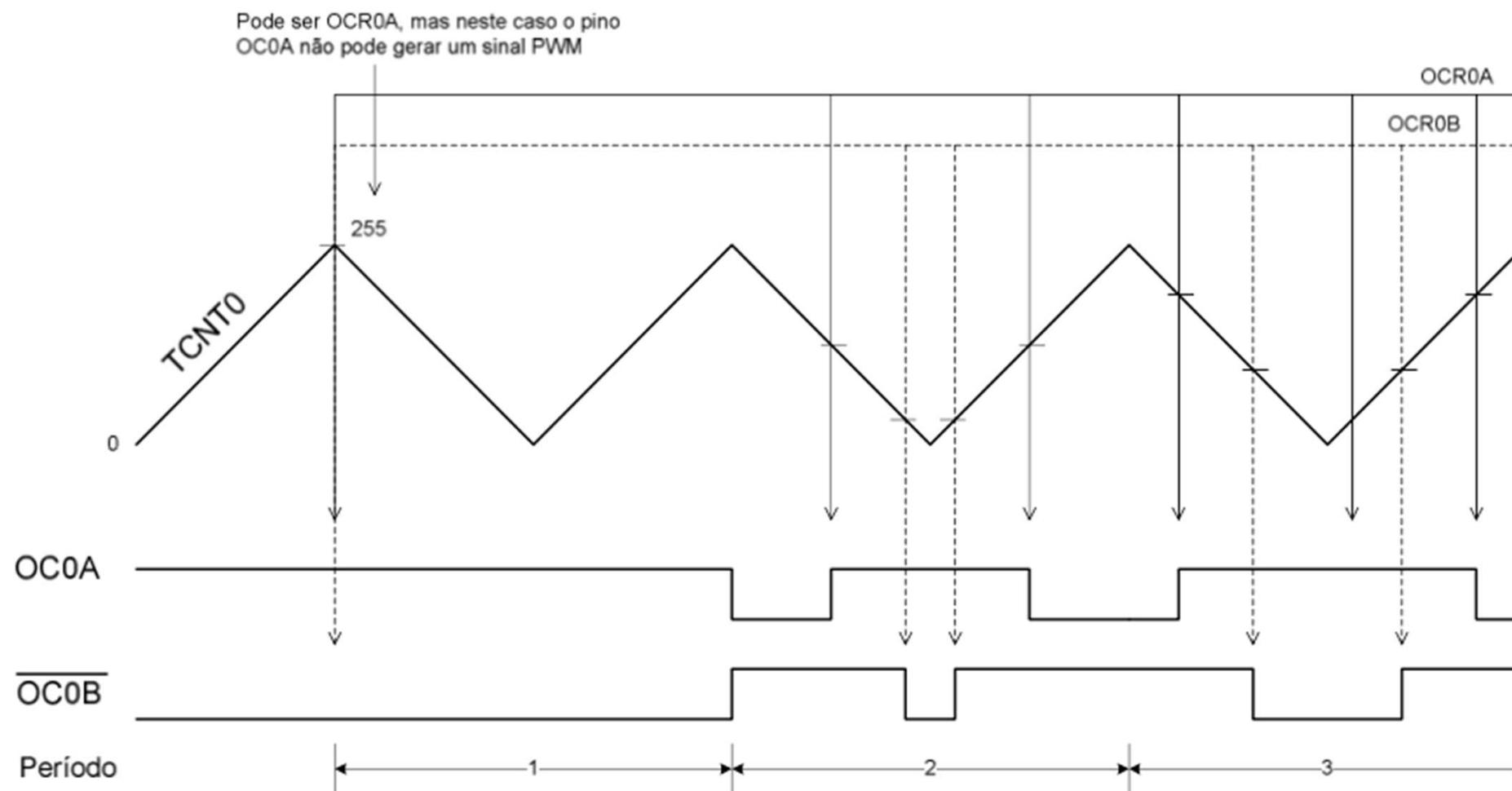
Timer 0 - Modo PWM Corrigido em fase

Permite a geração de um sinal PWM. O contador conta de zero até o valor máximo e descrece com mesmo período. Na comparação (invertida ou não invertida) os pinos OCR0A e OCBOB invertem na igualdade gerando o sinal PWM.

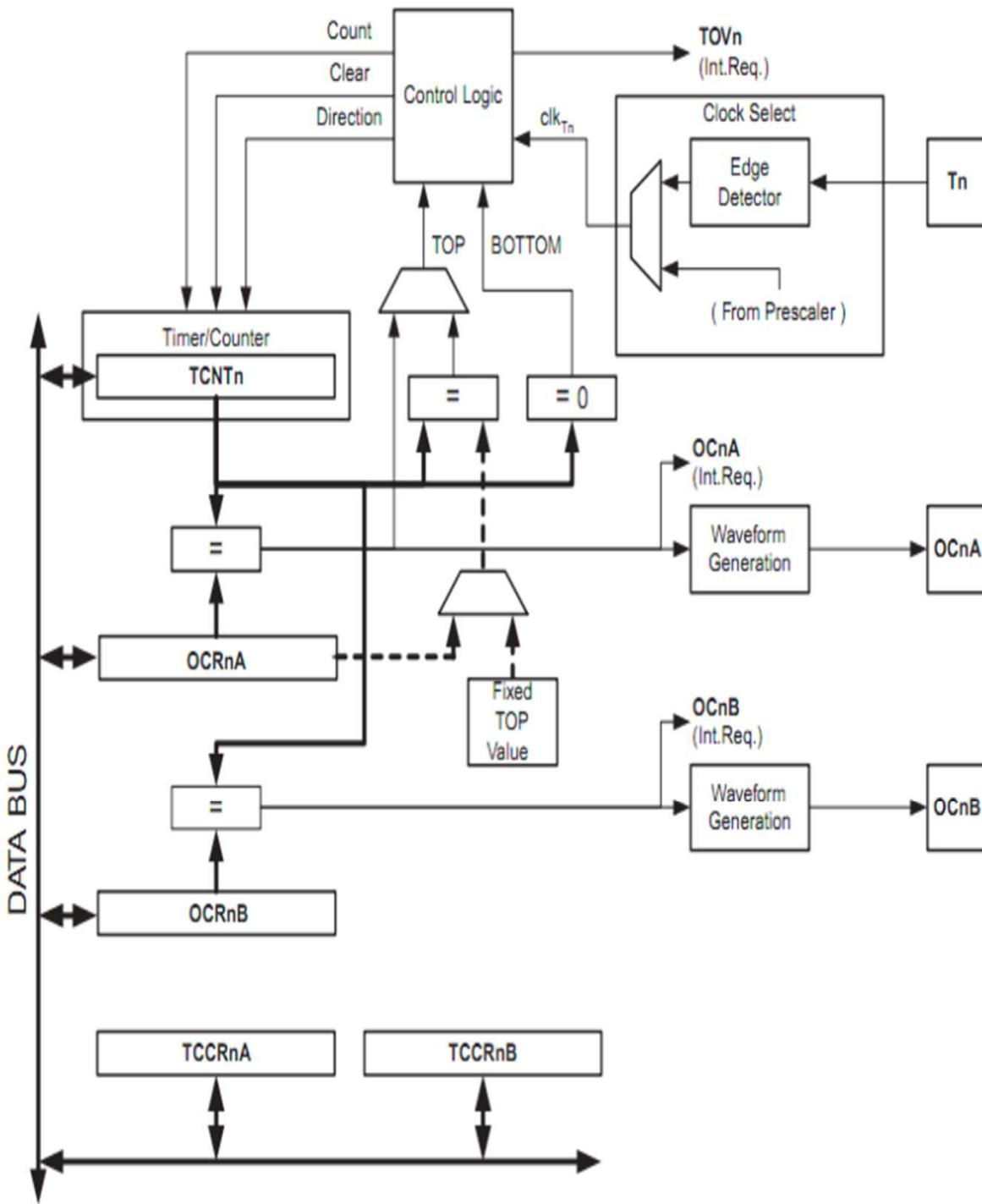


Timer 0 - PWM Corrigido em Fase

: Geração de sinal PWM no pino OC0, onde se obtém um sinal PWM com fase corrigida. É baseado na contagem crescente e decrescente do TCNT0



Arquitetura Timer 0



Registro de Contagem - TCNT0

- É onde ocorre a contagem em 8 bits, podendo contar de forma crescente ou decrescente;
- Quando o valor de contagem chega em 255 (FFH), na contagem crescente, ou 0 (zero), na contagem decrescente, ocorre o Overflow na contagem;
- Neste caso, bit de flag *TOV0* é setado, e uma interrupção por Overflow será requisitada, se previamente habilitada.

Modo de Ativação da Saída OC0A

COM0A1	COM0A0	Modo Normal (WGM02...0=000)
0	0	OC0A Desconectado.
0	1	OC0A Muda de Estado na Comparaçāo .
1	0	Reseta OC0A na Comparaçāo.
1	1	Seta OC0A na Comparaçāo.
COM0A1	COM0A0	Modo PWM Rápido (WGM02...0=011)
0	0	OC0A Desconectado.
0	1	Reservado.
1	0	OC0A=0 na Comparaçāo, 1 no valor MĀX.
1	1	OC0A=1 na Comparaçāo, 0 no valor MĀX.
COM0A1	COM0A0	PWM Fase Corrig. (WGM02...0=001)
0	0	OC0A Desconectado.
0	1	Reservado.
1	0	OC0A=0 na Comparaçāo em contagem crescente, OC0A=1 na contagem decres.
1	1	OC0A=1 na Comparaçāo em contagem crescente, OC0A=0 na contagem decres.

Modos de Operação do T/C0

Após selecionar a operação do T/C0 como: Temporizador (com um determinado fator de prescaler), ou Contador de Eventos (ativo na borda de descida ou subida), deve-se configurar, em seguida o modo de contagem em TCNT0, através dos bits de Seleção de Modo de Geração de Onda: *WGM02*, *WGM01* e *WGM00*:

- **Modo Normal:** Contagem crescente até valor máximo (255), sem recarga automática;
- **Modo CTC:** Sigla de Clear Timer on Compare Match, onde a contagem crescente em TCNT0 ocorre até a comparação c/ o Reg. de Comparação - OCR0, resultar verdadeiro, quando TCNT0 é zerado, e o pino OC0 será setado, resetado, ou complementado, dependendo da programação efetuada previamente nos bits *COM01/COM00*;

- **Modo PWM Rápido:** Geração de sinal PWM no pino OC0, c/ frequências de PWM cerca de duas vezes mais altas, que no PWM Corrigido em Fase;
- **PWM Corrigido em Fase:** Geração de sinal PWM no pino OC0, onde se obtém um sinal PWM com fase corrigida, apesar de resultar em frequências menores que o modo anterior.

Registro de Controle do T/C0 - TCCR0A

TCCR0A							
<i>COM0A1</i>	<i>COM0A0</i>	<i>COM0B1</i>	<i>COM0B0</i>	-	-	<i>WGM01</i>	<i>WGM00</i>

- Bits *COM0A1*, *COM0A0*: Compare Output Mode, ou Modo de Saída de Comparaçāo, seleciona o modo como o pino de saída OC0A se comportará.

- Bits *COM0B1*, *COM0B0*: Compare Output Mode, ou Modo de Saída de Comparaçāo, seleciona o modo como o pino de saída OC0B se comportará.

Modo de Ativação da Saída OC0B		
COM0B1	COM0B0	Modo Normal (WGM02...0=000)
0	0	OC0B Desconectado.
0	1	OC0B Muda de Estado na Comparaçāo .
1	0	Reseta OC0B na Comparaçāo.
1	1	Seta OC0B na Comparaçāo.
COM0B1	COM0B0	Modo PWM Rápido (WGM02...0=011)
0	0	OC0B Desconectado.
0	1	Reservado.
1	0	OC0B=0 na Comparaçāo, 1 no valor MĀX.
1	1	OC0B=1 na Comparaçāo, 0 no valor MĀX.
COM0B1	COM0B0	PWM Fase Corrig. (WGM02...0=001)
0	0	OC0B Desconectado.
0	1	Reservado.
1	0	OC0B=0 na Comparaçāo em contagem crescente, OC0=1 na contagem decres.
1	1	OC0B=1 na Comparaçāo em contagem crescente, OC0B=0 na contagem decres.

Registro de Controle do T/C0 - TCCR0B

TCCR0B							
FOC0A	FOC0B	-	-	WGM01	CS02	CS01	CS00

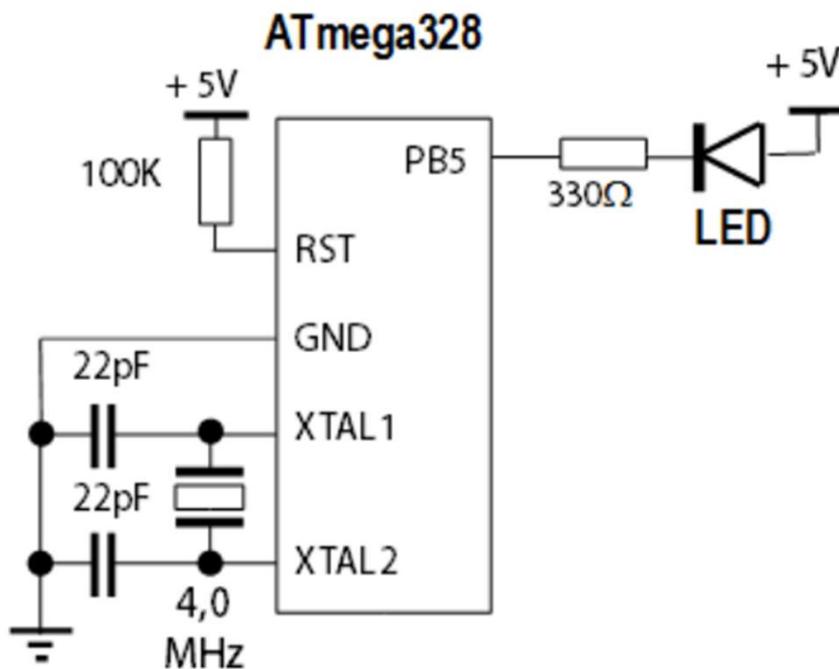
- Bits *CS02*, *CS01*, *CS00*: Bits Clock Selection, selecionam a operação do T/C0, como Temporizador ou Contador de Eventos, o fator de prescaler (se no modo Temporizador), e o modo Contador de Eventos, a ativação do sinal T0 na borda de descida ou subida:

<i>CS02</i>	<i>CS01</i>	<i>CS00</i>	Clock do Contador
0	0	0	Para a contagem do T/C0
0	0	1	Clock proveniente do clock da CPU
0	1	0	Clock = clock da CPU/8
0	1	1	Clock = clock da CPU/64
1	0	0	Clock = clock da CPU/256
1	0	1	Clock = clock da CPU/1024
1	1	0	Clock = T0 na borda de descida
1	1	1	Clock = T0 na borda de subida

Tab. 9.7 – Bits para configurar o modo de operação do TC0.

Modo	WGM02	WGM01	WGM00	Modo de Operação TC	TOP	Atualização de OCR0A no valor:	Sinalização do bit TOV0 no valor:
0	0	0	0	Normal	0xFF	Imediata	0xFF
1	0	0	1	PWM com fase corrigida	0xFF	0xFF	0x00
2	0	1	0	CTC	OCR0A	Imediata	0xFF
3	0	1	1	PWM rápido	0xFF	0x00	0xFF
4	1	0	0	Reservado	-	-	-
5	1	0	1	PWM com fase corrigida	OCR0A	OCR0A	0x00
6	1	1	0	Reservado	-	-	-
7	1	1	1	PWM rápido	OCR0A	0x00	OCR0A

Exemplo: Piscando um Led a cada 1,0s via T/C0



- Usando o T/C0, c/ clock de contagem: $f_{clk}/1024$, p/ um cristal de 16,0MHz, o período de contagem do T/C0 ficaria: $T_{clk} = 0,0625\mu s \cdot 1024 = 64\mu s$

Características dos timers 0, 1 e 2

Todos os temporizadores do Atmega 328 possuem seleção individual, tanto para a fonte do relógio quanto para seu prescaler.

Timer 0 e Timer 1

Os temporizadores 0 e 1 permitem oito configurações de relógio: desligado, relógio principal sem prescaler, relógio principal com prescaler de 8, 64, 256 e 1024 e o pino externo (T0 e T1) na borda de subida e descida.

Timer 2

O temporizador 2 também permite oito configurações de relógio: desligado, relógio principal sem prescaler, relógio principal com prescaler de 8, 64, 256 e 1024. Este temporizador não permite fonte de relógio baseado em pino externo.

Recursos de utilização das interrupções

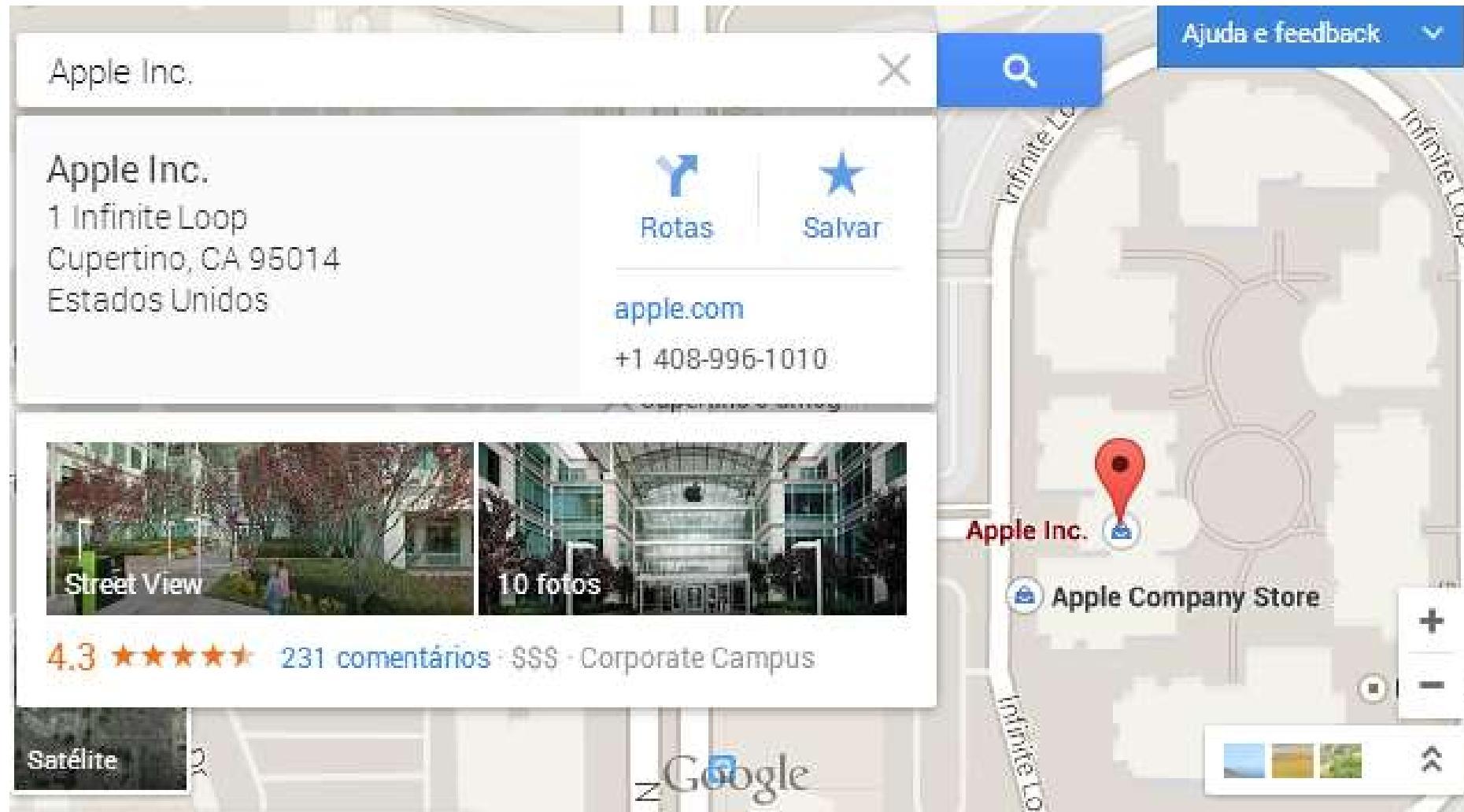
- Monitoramento de periféricos
- Geração de sinais PWM
- Contagem de eventos externos
- Organização de tempos de execução de funções (PROCESSOS)

Arquiteturas de software embarcado

- É sempre interessante definir uma arquitetura para o software
- Dependendo dos requisitos do sistema pode-se usar desde uma programação voltada apenas para a aplicação até um sistema operacional
- Dentre as técnicas mais simples utilizadas em baremetal
 - One single loop
 - Interrupt control system
 - Cooperative multitasking

ONE SINGLE LOOP

One single loop

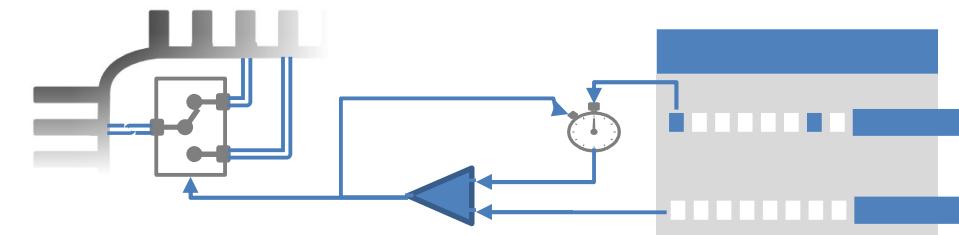


One single loop

- Consiste em um loop infinito dentro do qual todas as tarefas são executadas
- Apenas as rotinas de inicialização acontecem antes dele
- Programação é mais simples
- Dificuldade de garantir requisições temporais
- Escalabilidade, reuso e manutenção prejudicados

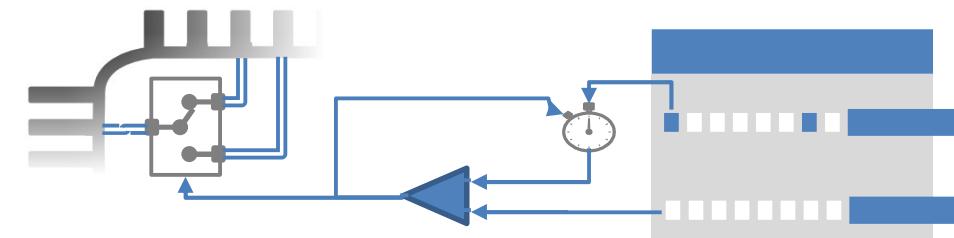
Exemplo

```
void main (void){  
    //declaração das variáveis  
    int ia, ib, ic;  
  
    //inicialização dos periféricos  
    InicializaTeclado();  
    InicializaLCD();  
  
    for(;;){  
        //chamada das tarefas  
        ia = LerTeclas();  
  
        MudaDigito(ia,0);  
        //tem que ser executado pelo menos a cada 10(ms)  
        AtualizaDisplay();  
        DebounceTeclas();  
  
    }  
}
```



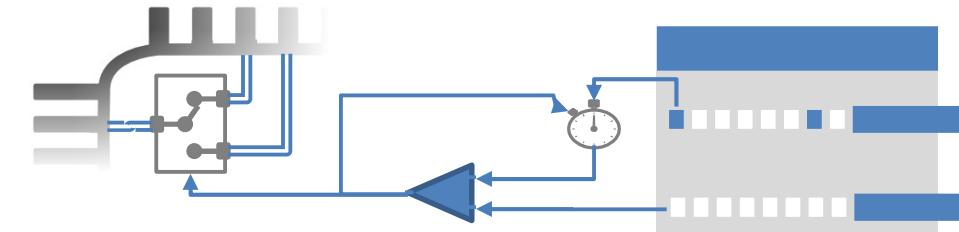
Exemplo com problemas!

```
void main (void){  
    //declaração das variáveis  
    int ia, ib, ic;  
    float fa, fb, fc;  
    //inicialização dos periféricos  
    InicializaTeclado();  
    InicializaLCD();  
  
    for(;;){  
        //chamada das tarefas  
        ia = LerTeclas();  
        EnviaDados(ia);  
        MudaDigito(ia,0);  
        //tem que ser executado pelo menos a cada 10(ms)  
        AtualizaDisplay();  
        DebounceTeclas();  
        //não cumpre o tempo devido ao excesso de atividades  
        ic = RecebeSerial();  
        fa = 2.0 * ic / 3.14;  
        EnviaSerial(fa & 0x00FF);  
        EnviaSerial(fa >> 8);  
    }  
}
```

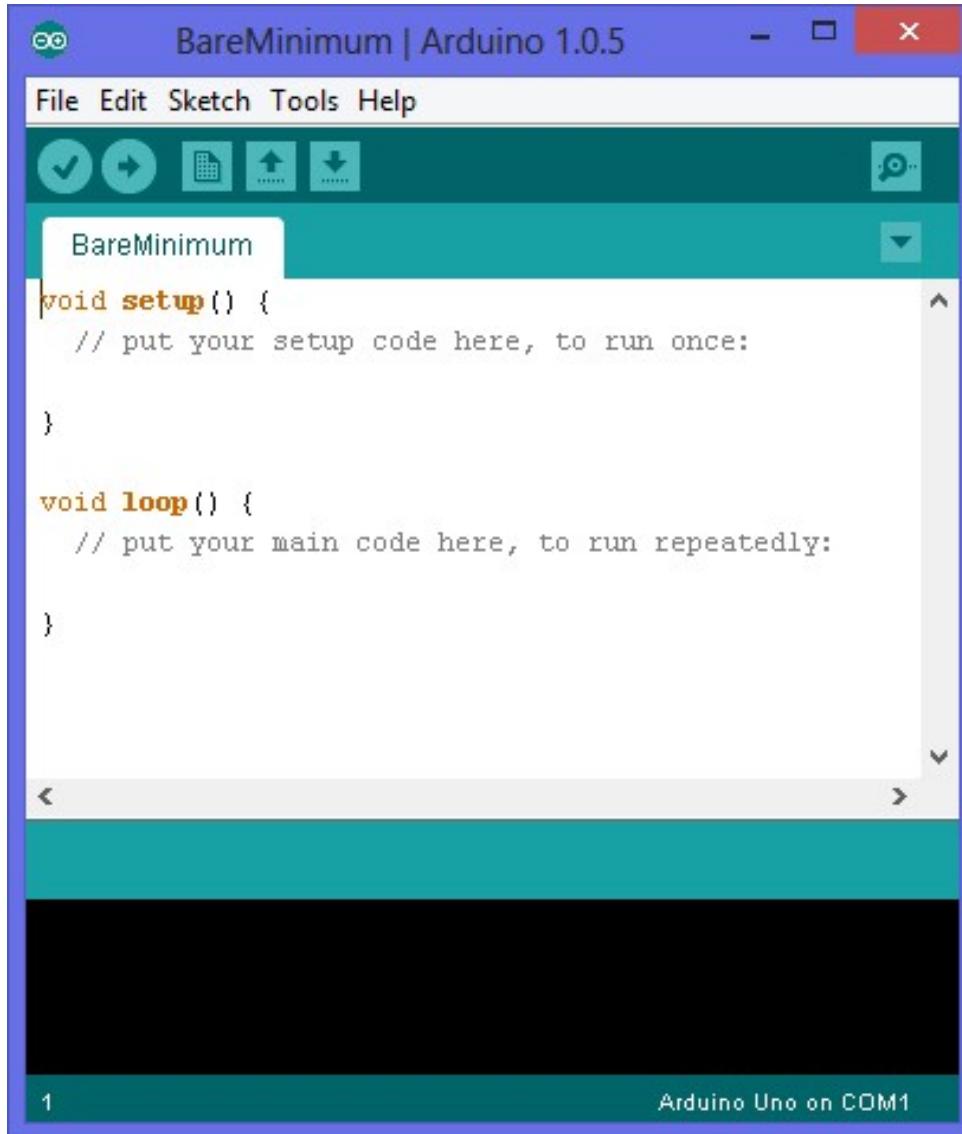


Máquina de estados em C

```
void main(void){
    char slot;
    //Inicializa...
    for(;;){ //início do loop infinito
        //***** top-slot *****
        //***** início da máquina de estado *****
        switch(slot){
            case 0:
                LeTeclado();      slot = 1; break;
            case 1:
                AtualizaDisplay(); slot = 2; break;
            case 2:
                RecebeSerial();   slot = 3; break;
            case 3:
                AtualizaDisplay(); slot = 4; break;
            case 4:
                EnviaSerial();    slot = 5; break;
            case 5:
                AtualizaDisplay(); slot = 0; break;
            default:
                slot = 0; break;
        }
        //***** fim da máquina de estado *****
        //***** bottom-slot *****
    } //fim loop infinito (!?)
}
```



Outro modo de ver o single loop



The screenshot shows the Arduino IDE interface with the title bar "BareMinimum | Arduino 1.0.5". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for checkmark, run, file, upload, and download. A search bar is present above the code editor. The code editor window contains the "BareMinimum" sketch:

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

The status bar at the bottom shows the number "1" and the text "Arduino Uno on COM1".

Cuidados

- Velocidade de execução do loop principal
 - Difícil calculo da duração
 - Previsão de todas as opções
 - Leitura de valores/recepção de dados
- Travamento de uma função do sistema
- Atraso na resposta de eventos
- Pouco escalável

Interrupt control system

- Visa resolver o problema das restrições temporais
- Apenas alguns sub-sistemas podem ser operados via interrupção
- Os demais sistemas operam no loop principal
- Facilita a expansão do sistema, já que as interrupções não dependem do loop principal
- Ótima alternativa para sistemas que tem como requisito o baixo consumo de energia

Interrupt control system

- O programa principal é interrompido e uma função pré definida é executada
 - O programa principal não percebe a pausa
- Redução do tempo de resposta aos eventos
 - O tempo para responder a um determinado evento é o menor possível utilizando um microcontrolador
- As funções devem ser curtas.
 - Deixar o processamento pesado para o loop principal

Interrupt control system

- O desenvolvimento do sistema pode ser simplificado em 3 etapas:
 - Atender as interrupções;
 - Processar o resultado das interrupções no loop principal;
 - Se não houver nada a ser processado entrar em modo de baixo consumo de energia.