Inatel Instituto Nacional de Telecomunicações	**RELATÓRIO 8**	**Data: / /**
	Disciplina: E209	
	Prof: Yvo Marcelo Chiaradia Masselli **Monitores: João Lucas/Luan Siqueira/Maria Luiza/ Lucas Lares/Rafaela Papale**	
Conteúdo: Microcontrolador ATMega328p		
Tema: Temporizador		
Nome:	**Matrícula:**	**Curso:**

OBJETIVOS:

- Utilizar a ferramentas de simulação para desenvolver programas para o ATMega328p.
- Desenvolver um programa de controle que faça uso do temporizador interno.
- Utilizar as entradas e saídas do ATMega328p com circuitos de aplicação.

Parte Teórica

Temporizador (Timer)

O timer é um **bloco interno do microcontrolador capaz de contar tempo**. Qualquer temporizador programável é um contador no qual os pulsos de clock possuem um período fixo. Ou seja, se o clock está ajustado para um período de 1ms e o contador apresenta em um determinado momento o número 100 significa que se passaram 100ms (admitindo o início em 0).
O ATMega328p possui 3 temporizadores internos, sendo o timer 0 e 2 de 8 bits e o timer 1 de 16 bits, com fonte de clock ajustável. A base de tempo dos TIMERS podem ser ajustadas entre algumas opções. No caso que vamos abordar, a fonte de clock do timer é o clock do cristal **(XTAL = 16MHz)**.O valor padrão é 16 MHz, ou seja, o período base é de 62,5ns. Atente-se que esse valor de frequência está ajustado na inicialização do programa pela calibração do oscilador interno, que também possui outros valores calibrados. Além disso, é possível ajustar diferentes fatores de divisão para o sinal de clock dos TIMERS em 8, 64, 256, 1024, ou seja:

```
Se fator de divisão = 8    => Fclk = 16MHz/8    = 2MHz    => Tclk = 500ns.
Se fator de divisão = 64   => Fclk = 16MHz/64   = 250kHz  => Tclk = 4µs.
Se fator de divisão = 256  => Fclk = 16MHz/256  = 62,5kHz => Tclk = 16µs.
Se fator de divisão = 1024 => Fclk = 16MHz/1024 = 15,6kHz => Tclk = 64µs.
```

O timer do ATMega238p tem vários modos: normal, CTC (Clear timer on compare match), PWM, Fast PWM. O nosso interesse é no modo CTC, ou seja, modo de comparação, que é feito pela configuração dos campos **WGM00 = 0**, **WGM01 = 1** no registrador **TCCR0A**.
A contagem é realizada pelo registro **TCNT0** e ele é comparado com o registro **OCR0A** ou **OCR0B**. Essa contagem está representada nas Figuras 1 e 2. Quando o valor do **TCNT0** atinge o valor de **OCR0A ou OCR0B**, o fim da contagem é indicado por um bit de sinalização chamado de flag de interrupção, nesse caso a flag do canal **OCR0A** que é **OCF0A**. Nesse momento, o valor do **TCNT0** é automaticamente zerado e a contagem é reiniciada e o processo se repete.

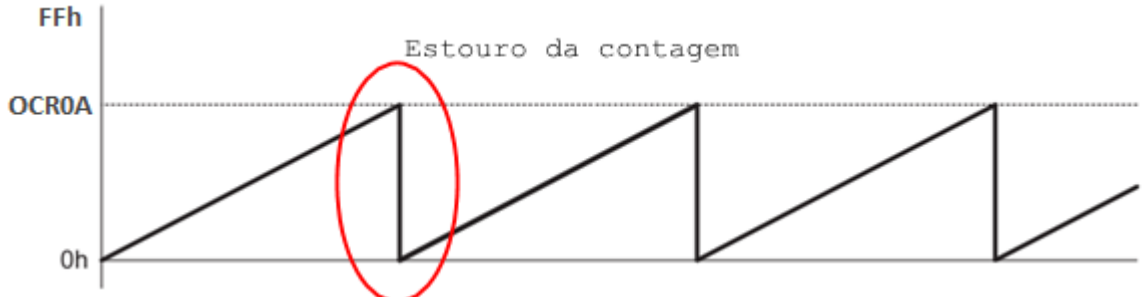

Figura 1 - Contagem crescente do Timer0 até o valor do OCR0A

Os 3 timers internos do ATMega328p são independentes: **Timer0, Timer1 e Timer2.** Como iremos utilizar o **Timer0**, os registros **TCCR0A, TCCR0B, OCR0A** e **TIMSK0** devem ser configurados para gerar uma interrupção a cada intervalo desejado.
De forma geral, sempre utilizamos a seguinte sequência de configurações para um timer qualquer:

1) Configurar o modo de operação do timer e o divisor de clock: **TCCR0A e TCCR0B.**
2) Configurar o valor máximo de contagem: **OCR0A**
3) Habilitar a interrupção do comparador desejado: **TIMSK0**
4) **Habilitar a interrupção global do microcontrolador: sei();**

Exemplos de como configurar o Timer (linhas de comando no programa):

Para essa etapa você primeiro deve saber exatamente como quer preparar seu timer, ou seja, qual a base de tempo e como vai gerar a interrupção de tratamento. A primeira coisa é configurar o modo do contador para CTC.

```
TCCR0A |= (1 << WGM01);
```

Após configurar o modo é necessário decidir o divisor do clock base **(F_CPU = 16MHZ)**, os valores oferecidos são: 8, 64, 256, 1024. Se optar por usar a divisão lembre-se que o tempo base de contagem irá mudar, por exemplo: O clock de 16MHZ dividido por 256, será igual à 62,5kHz (16MHZ/256 = 16,5kHz) assim você terá um período base de 16µs.

```
TCCR0B |= (1 << CS02);
```

Após escolher o clock base e o divisor, você estará apto a escolher o tempo que quer que seu timer conte. Sabendo que o timer conta quantos ciclos de clock ocorreram, se ele contar 10 clocks de 16µs cada, ele terá contado por 160us. Dessa forma se quer contar 100ms com um período de clock de 16µs é necessário contar 6250 pulsos de clock. Lembre-se que o TIMER tem 8 bits, logo pode contar de 0 a 255. O valor a ser contado deve ser:

```
OCR0A = INTERVALO;
```

Agora temos um timer configurado e contando um tempo específico, entretanto, ele ainda não é capaz de avisar ao ATMega328p que o tempo definido foi alcançado. Para isso é necessário ligar a Interrupção do comparador e criar a função de tratamento. Lembre-se de acionar a interrupção global caso ela ainda não esteja acionada.

Linha para ligar interrupção do comparador A do timer0:

```
TIMSK0 |= (1 << OCIE0A);
```

Linha para ligar interrupção GLOBAL:

```
sei();
```

Função de tratamento da interrupção:

```
ISR(TIMER0_COMPA_vect)
```

```
{
      // aqui você colocara o que será realizado toda vez que o timer estourar
}
```

Algo muito comum de acontecer é quando a capacidade de contagem do timer não é o suficiente para alcançar o tempo desejado, para isso podemos dividir o valor total em pequenos valores que cabem entre 0 e 255. Dentro da interrupção do timer colocamos uma variável que contará quantas vezes o timer estourou **(Número de interrupções geradas)**. Usando essa variável podemos saber quanto tempo já se passou desde que o timer iniciou e se o tempo que queremos já foi alcançado.
Por exemplo: Para contar 1 segundo usando um clock de 16MHz e o divisor 1024 (Tclock = 64us) com limite de 200 contagens no OCR0A, ou seja, cada vez que o **TCNT0** zerar terá se passado(64us x 200 = 12,8ms), sendo assim conseguiremos contar 1 segundo. Então dividimos esse 1 segundo por 12,8ms, ou seja, teremos que entrar na função de interrupção 78 vezes (78 * 12,8ms ≈ 1s).

```
ISR(TIMER0_COMPA_vect) // Vetor de interrupção compa
{
  cont++;
  if(cont >= 78) {
      //aqui você colocará sua lógica
      //toda vez que o timer estourar 78 vezes
      //totalizando 1 segundo
       cont = 0;
  }
}
```

Dica: Ao invés de digitar todas essas linhas em sua função MAIN podemos criar funções extras para diferentes propósitos, como:

```
void ConfigTimer0(void)
{
      TCCR0B|= (1<<CS02)|(1<<CS00);
      TCCR0A|= (1<<WGM01);
      OCR0A = INTERVALO;
      TIMSK0|= (1<<OCIE0A);
}
```

E também:

```
void DisableTimer0(void)
{
      TCCR0B = 0;
}
```

Essas funções extras que criamos tornam o código mais organizado e fácil de identificar erros.

Parte Prática

Programa 1 - Pisca LED)
Crie um projeto que após a inicialização do ATMega328p, um LED Vermelho fique piscando com um tempo de 150ms. O projeto deve fazer uso do periférico TIMER e não pode conter linhas de delay.

Programa 2 - Cronômetro)

Desenvolva um programa que faça com que um LED Vermelho indique cada segundo (fique aceso por 100ms e apaga) e um LED Verde indique os minutos (acende por 100ms e apaga). Use o temporizador.

ANEXO) Registros principais (fonte: UserManual ATMega328p)

14.9.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1)2)]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

14.9.2 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

14.9.6 TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATmega328P and will always read as zero.

- **Bit 2 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the status register is set, the Timer/Counter compare match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter interrupt flag register – TIFR0.

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.