



Universidade Tecnológica Federal do Paraná – UTFPR
Bacharelado em Ciência da Computação

BCC34C – Sistemas Microcontrolados

Prof. Frank Helbert Borsato

Aula Hoje

- **Capítulo 9. TEMPORIZADORES/CONTADORES**
 - **9.1 Temporizando e Contando**
 - **9.2 Modulação por Largura de Pulso – PWM**
 - **9.3 Temporizador/Contador 0**
 - 9.3.1 Registradores do TC0

Temporizadores

- **Temporizadores e Contadores:**
- **No trabalho com microcontrolador é necessário a geração de sinais periódicos e eventos**
 - Temporizadores/Contadores (TCs) realizam este trabalho
- **No ATmega328**
 - Dois temporizadores/contadores de 8 bits (TC0 e TC2)
 - Um de 16 bits (TC1)
 - Todos independentes e com características próprias

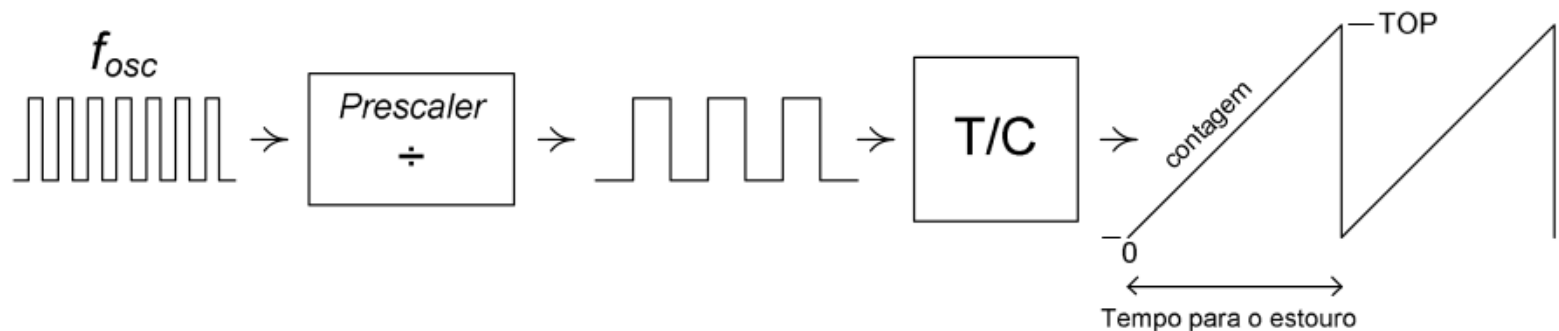


Fig. 9.1 – Funcionamento de um temporizador/contador do ATmega.

Temporizadores

- **Frequência**
- **Para contagens por unidade de tempo**
 - A unidade no SI para a frequência é o hertz (Hz), homenagem ao físico alemão Heinrich Hertz
 - 1 Hz significa que o evento se repete uma vez por segundo
 - Um nome anterior para esta unidade foi ciclos por segundo
- **Período**
- **O período, normalmente indicado por T, é o período de tempo correspondente a um ciclo**
 - É o recíproco da frequência f
 - A unidade no SI para o período é o segundo

$$f = \frac{1}{T}$$

Temporizadores

- Temporizando e Contando
- Uma das principais função dos TCs é a contagem de pulsos de clock
 - Os quais podem ser externos ou internos ao microcontrolador
 - Permitem a geração de eventos periódicos:
 - » Para uso do programador ou
 - » Para a determinação de um número de contagens

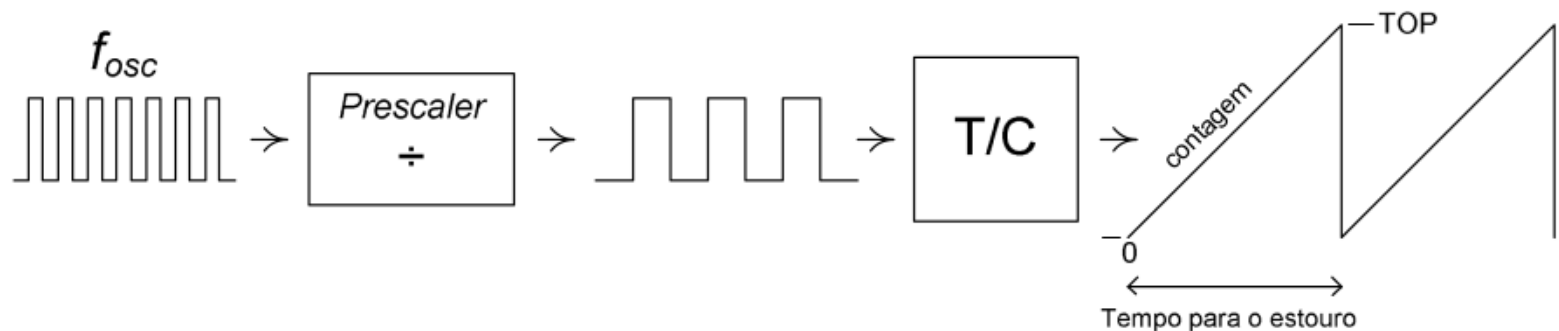


Fig. 9.1 – Funcionamento de um temporizador/contador do ATmega.

Temporizadores

- Um estouro do contador ocorre quando ele passa do valor máximo permitido para a contagem para o valor zero
- O Temporizador/Contador (TC) de 8 bits:
 - Contará de 0 até 255 resultando em 256 contagens
- O Temporizador/Contador (TC) de 16 bits:
 - Contará de 0 até 65535, resultando em 65536 contagens
- Assim, o tempo que um TC leva para estourar é dado por:

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

Temporizadores

- **Exemplo:**
- **Supondo um TC de 8 bits:**
 - Conta até o valor superior (TOP value) de 255
 - Trabalhando com uma frequência de 1 MHz
 - Sem divisor de frequência
 - O tempo para o seu estouro é de:

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

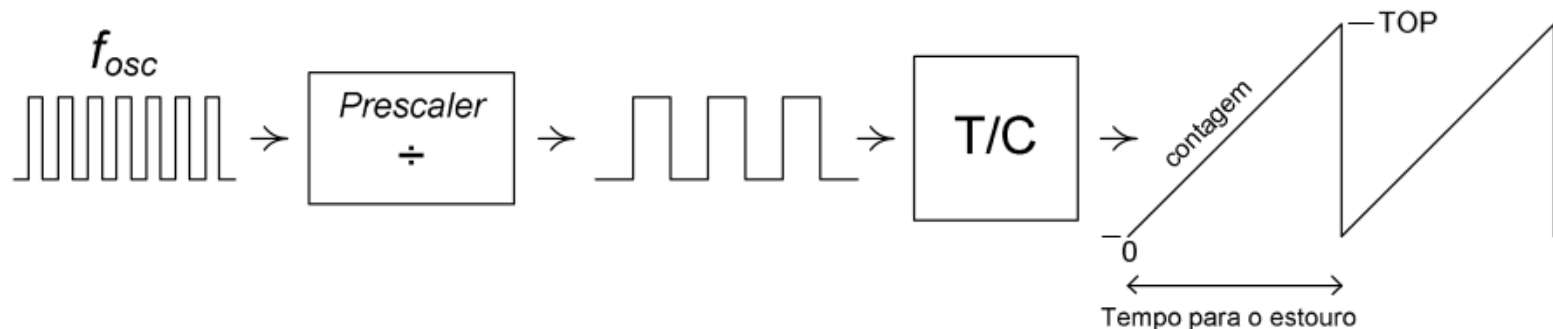


Fig. 9.1 – Funcionamento de um temporizador/contador do ATmega.

Temporizadores

- **Exemplo:**
- **Supondo um TC de 8 bits:**
 - Conta até o valor superior (TOP value) de 255
 - Trabalhando com uma frequência de 16 MHz
 - Com divisor de frequência de 1024
 - O tempo para o seu estouro é de:

$$t_{\text{estouro}} = 256 \times 1/16 \text{ MHz} \times 1024 = 16,384 \text{ ms}$$

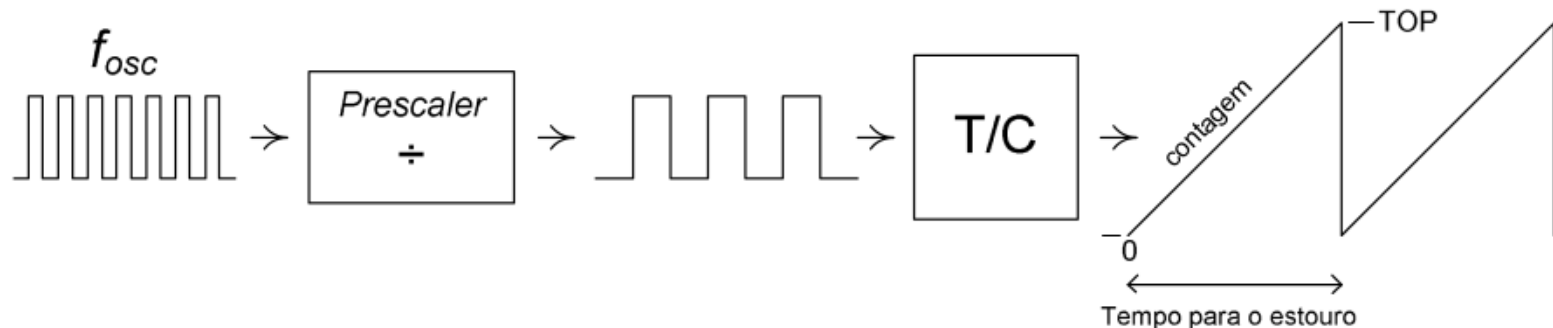


Fig. 9.1 – Funcionamento de um temporizador/contador do ATmega.

Temporizadores

```
//----- //
//   AVR e Arduino: Técnicas de Projeto, 2a ed. - 2012.   //
//----- //
//===== //
//   HABILITANDO A INTERRUPÇÃO POR ESTOURO DO TC0           //
//===== //
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

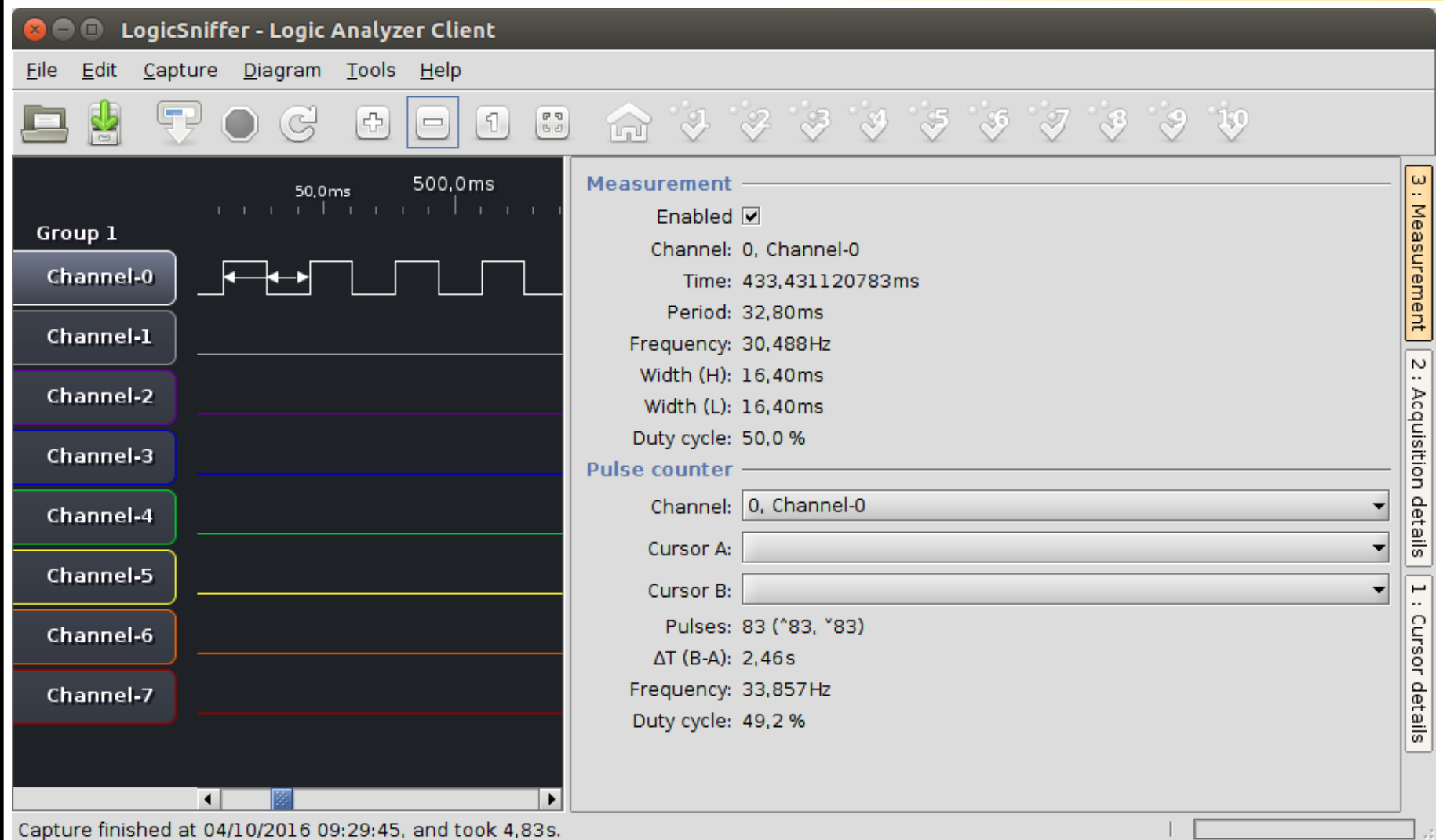
#define cpl_bit(y,bit) (y^=(1<<bit)) //troca o estado lógico do bit x da variável Y
#define LED    PB5

//-----
ISR(TIMERO_OVF_vect)           //interrupção do TC0
{
    cpl_bit(PORTB,LED);
}
//-----
int main()
{
    DDRB  = 0b00100000;    //somente pino do LED como saída
    PORTB = 0b11011111;    //apaga LED e habilita pull-ups nos pinos não utilizados

    TCCR0B = (1<<CS02) | (1<<CS00); //TC0 com prescaler de 1024, a 16 MHz gera uma interrupção a cada 16,384 ms
    TIMSK0 = 1<<TOIE0;           //habilita a interrupção do TC0
    sei();                       //habilita interrupções globais

    while(1)
    {
        //Aqui vai o código, a cada estouro do TC0 o programa desvia para ISR(TIMERO_OVF_vect)
    }
}
//=====
```

Temporizadores



Temporizadores

- **Exemplo:**
- **Supondo um TC de 16 bits:**
 - Conta até o valor superior (TOP value) de 65535
 - Trabalhando com uma frequência de 16 MHz
 - Com divisor de frequência de 64
 - O tempo para o seu estouro é de:

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

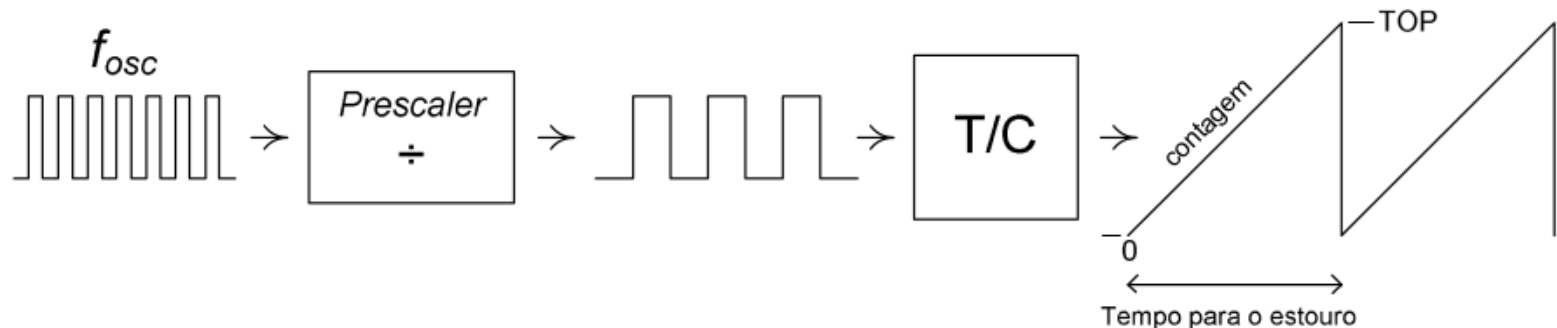


Fig. 9.1 – Funcionamento de um temporizador/contador do ATmega.

Temporizadores

```
//----- //
//   AVR e Arduino: Técnicas de Projeto, 2a ed. - 2012. //
//----- //
//===== //
//   HABILITANDO A INTERRUPÇÃO POR ESTOURO DO TC1 //
//===== //

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

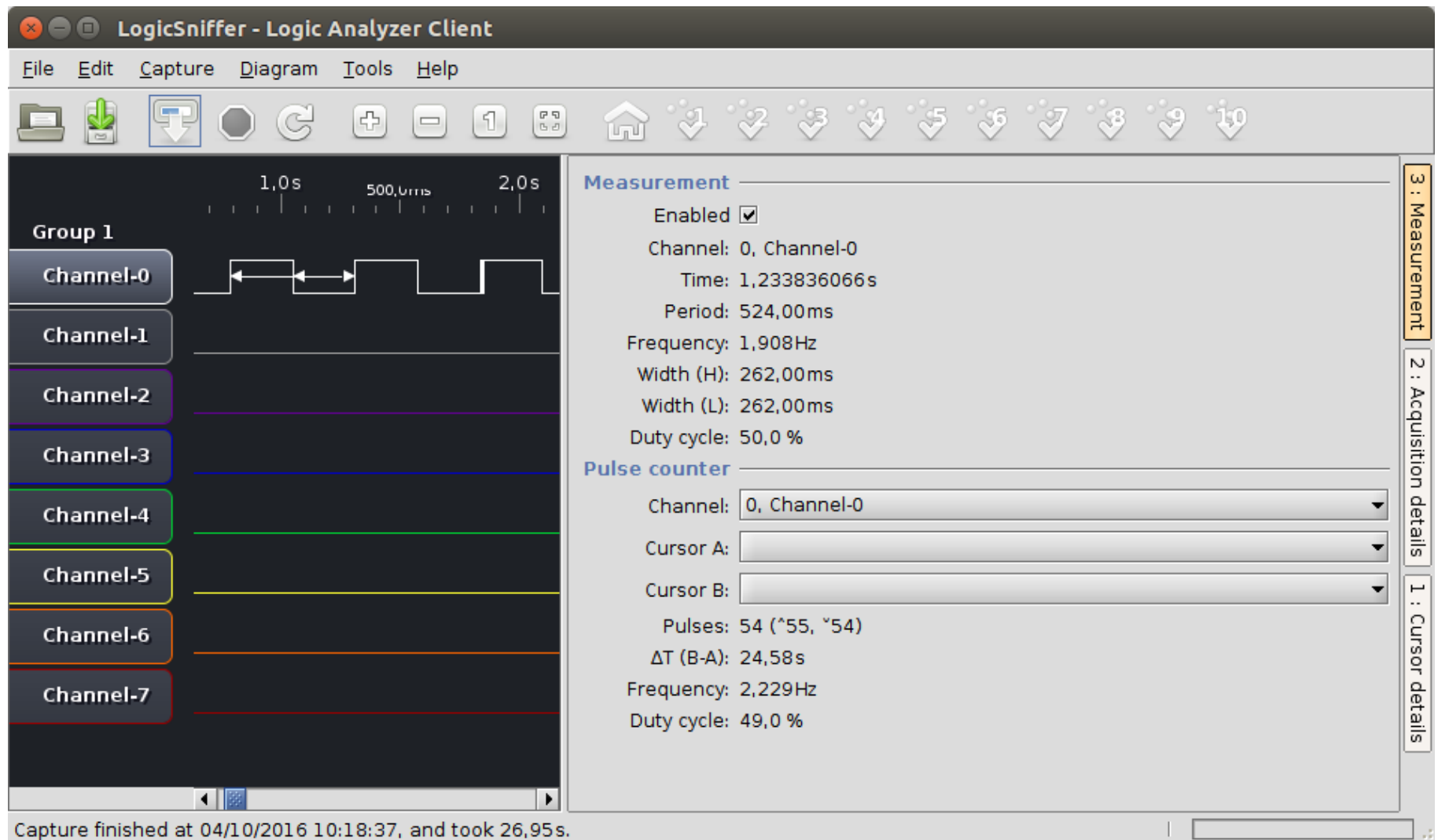
#define cpl_bit(y,bit) (y^=(1<<bit)) //troca o estado lógico do bit x da variável Y
#define LED    PB5

//-----
ISR(TIMER1_OVF_vect)           //interrupção do TC1
{
    cpl_bit(PORTB,LED);
}
//-----
int main()
{
    DDRB  = 0b00100000;           //somente pino do LED como saída
    PORTB = 0b11011111;           //apaga LED e habilita pull-ups nos pinos não utilizados

    TCCR1B = (1<<CS11) | (1<<CS10); //TC1 com prescaler de 64, a 16 MHz gera uma interrupção a cada 0,262 s
    TIMSK1 = 1<<TOIE1;             //habilita a interrupção do TC1
    sei();                          //habilita interrupções globais

    while(1)
    {
        //Aqui vai o código, a cada estouro do TC1 o programa desvia para ISR(TIMER1_OVF_vect)
    }
}
//=====
```

Temporizadores



Temporizadores

- **Modulação por Largura de Pulso – PWM**
 - Pulse-width modulation (PWM)
- **O uso desses sinais é baseado no conceito de valor médio de uma forma de onda periódica**

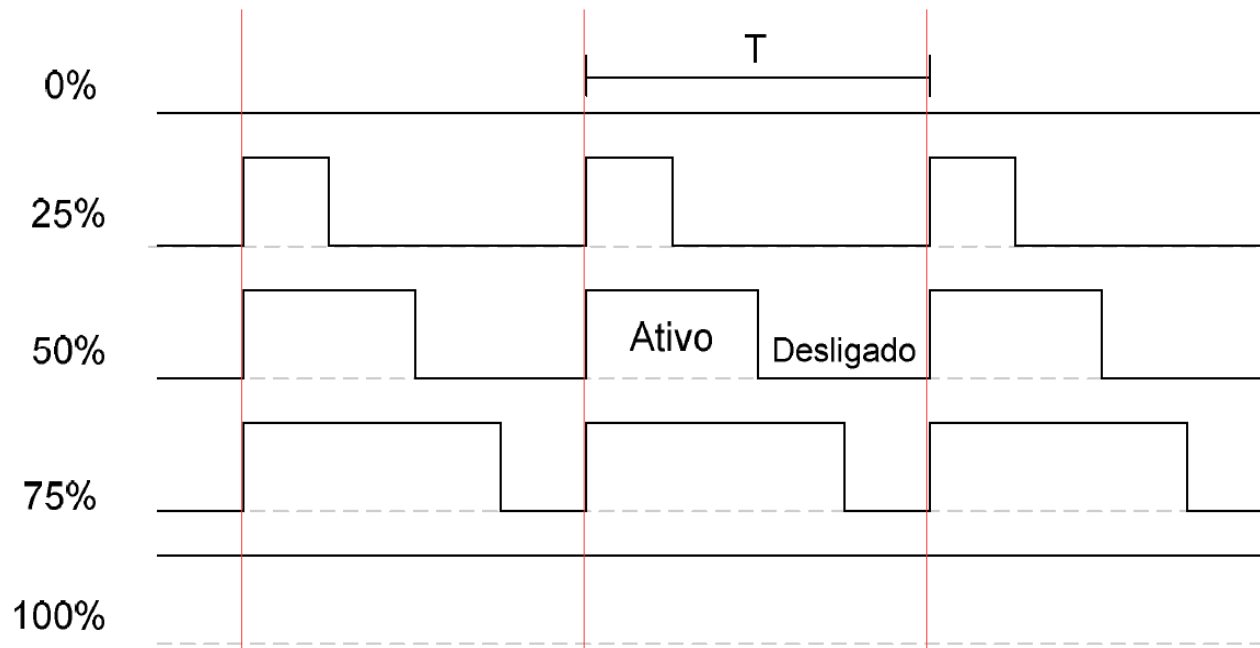


Fig. 9.2 – PWM com período T e ciclo ativo de 0%, 25%, 50%, 75% e 100%.

Temporizadores

- **Baseado no valor médio de um sinal PWM**
 - Dispositivos eletrônicos podem ser controlados, como por exemplo:
 - » **Motores, lâmpadas, LEDs, fontes chaveadas e circuitos inversores**

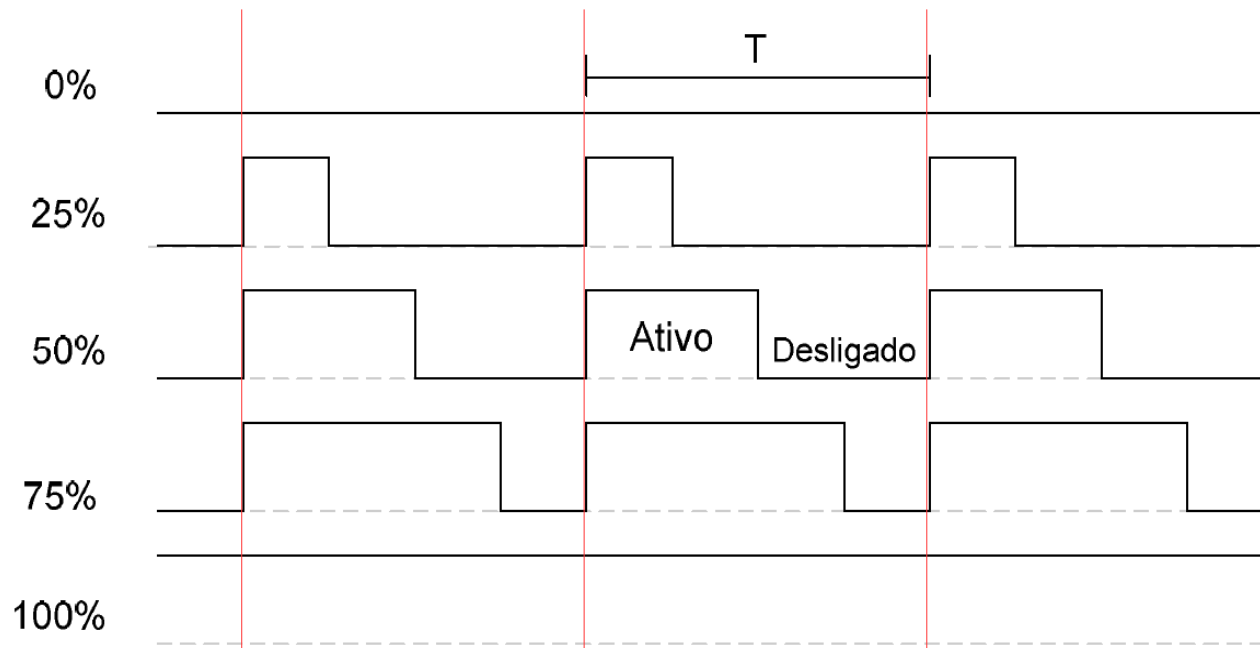


Fig. 9.2 – PWM com período T e ciclo ativo de 0%, 25%, 50%, 75% e 100%.

Temporizadores

- O valor médio de uma forma de onda é controlado digitalmente:
 - Pelo tempo em que o sinal fica em nível lógico alto durante um determinado intervalo de tempo
 - » No PWM, esse tempo é chamado de ciclo ativo (Duty Cycle)

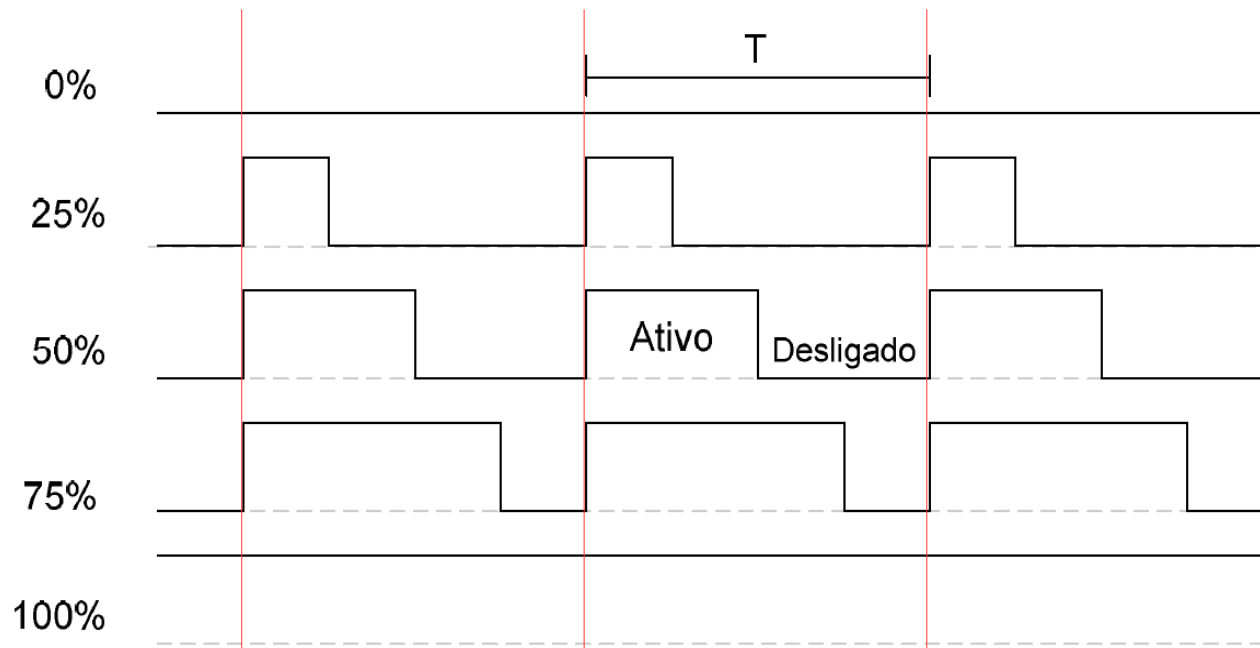


Fig. 9.2 – PWM com período T e ciclo ativo de 0%, 25%, 50%, 75% e 100%.

Temporizadores

- O cálculo valor médio de um sinal digital é dado por:

$$V_{\text{médio}} = \frac{\text{Amplitude Máxima}}{\text{Período}} \times (\text{Tempo Ativo no Período}) \quad (9.2)$$

- **Se o sinal digital tem variação de 0 a 5 V**
 - Um ciclo ativo de 50% corresponde a um valor médio de 2,5 V
 - Um ciclo ativo de 75% corresponderia a 3,75 V.
- **Se alterar o ciclo útil do sinal PWM, altera-se o seu valor médio**

Temporizadores

- Na fig. 9.3, é ilustrada a variação do ciclo ativo de um sinal PWM de 0 a 100% do seu ciclo útil (período) com o passar do tempo

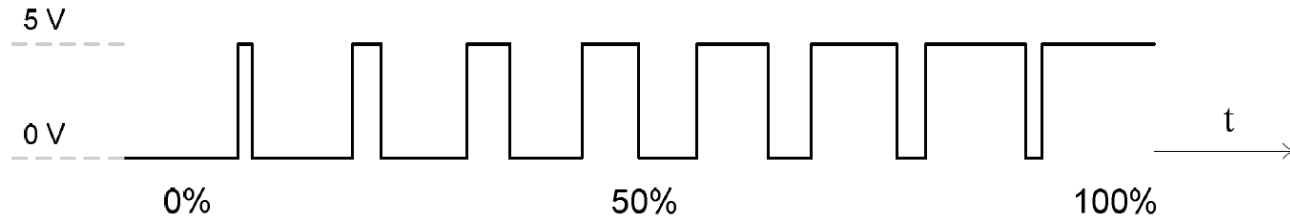


Fig. 9.3 – Variação do ciclo ativo de um sinal PWM em um intervalo de tempo.

Temporizadores

- **É importante notar que o período do sinal PWM não se altera, e sim sua largura de ciclo ativo**
 - Quando se gera um sinal PWM, deve-se em primeiro lugar determinar sua frequência de trabalho

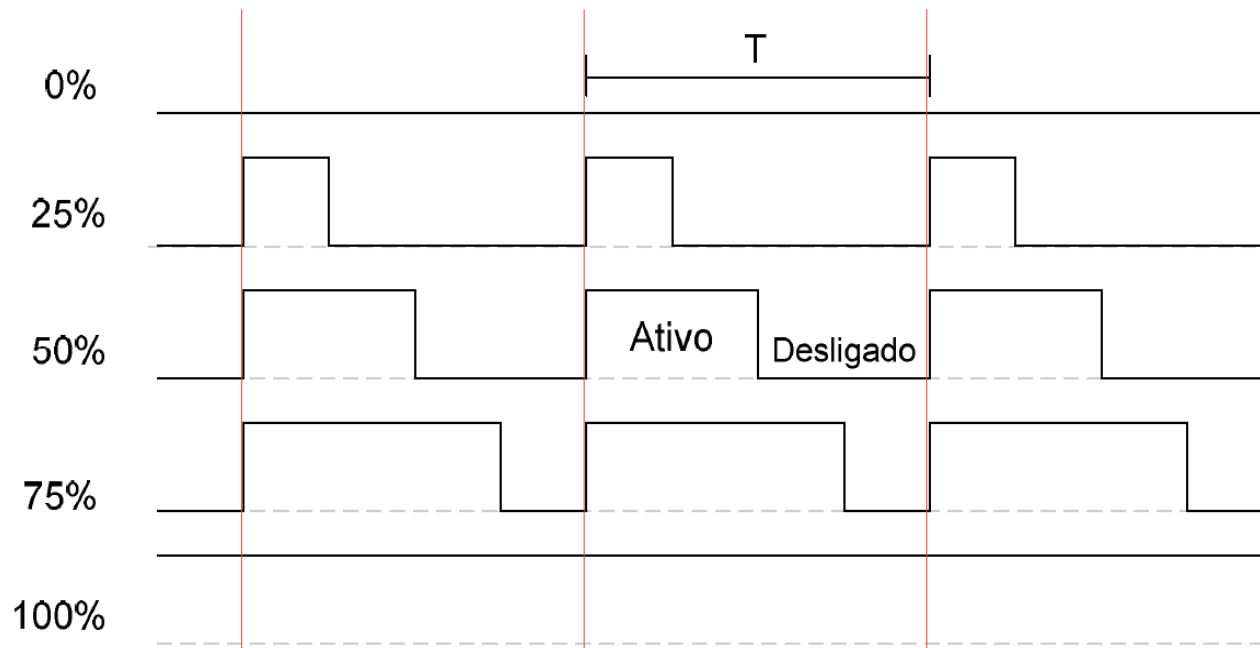


Fig. 9.2 – PWM com período T e ciclo ativo de 0%, 25%, 50%, 75% e 100%.

Temporizadores

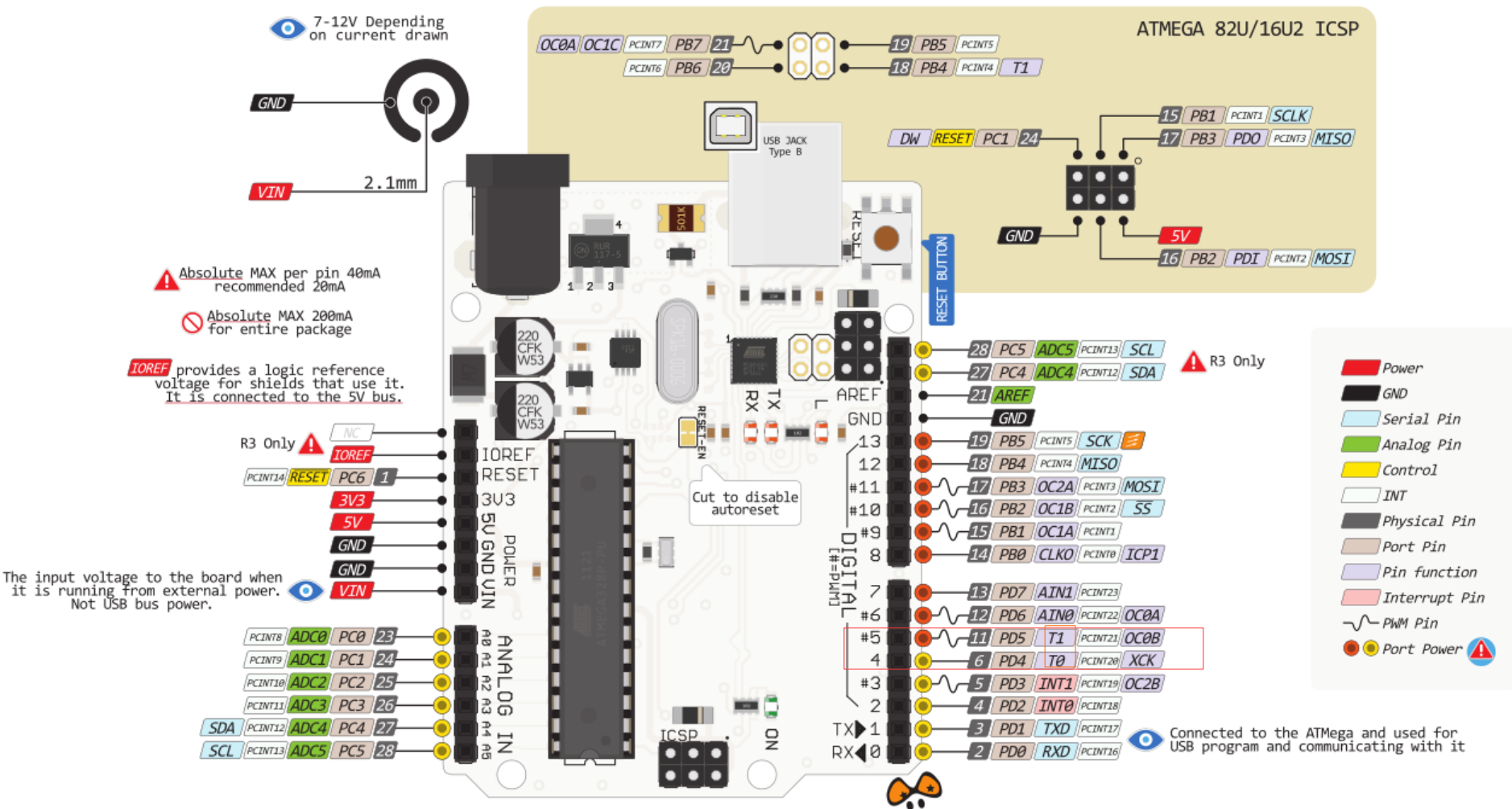
- **A resolução do PWM em um microcontrolador**
- **É dada pelo seu número de bits e indica quantos diferentes ciclos ativos podem ser gerados**
- **Um PWM de 10 bits:**
 - Pode gerar 1024 diferentes níveis (0...1023)
 - » **Começando com o nível de tensão 0 e terminando com 100 % do ciclo ativo**
 - No ATmega o ciclo ativo de um sinal PWM é controlado pela escrita em registradores específicos e sua resolução vai depender do TC empregado

Temporizadores

- **Temporizador/Contador 0**
- **O TC0 é um contador de 8 bits que é incrementado com pulsos de clock obtido:**
 - Da fonte interna de clock do microcontrolador (prescaler)
 - De uma fonte externa (T0 ou T1)



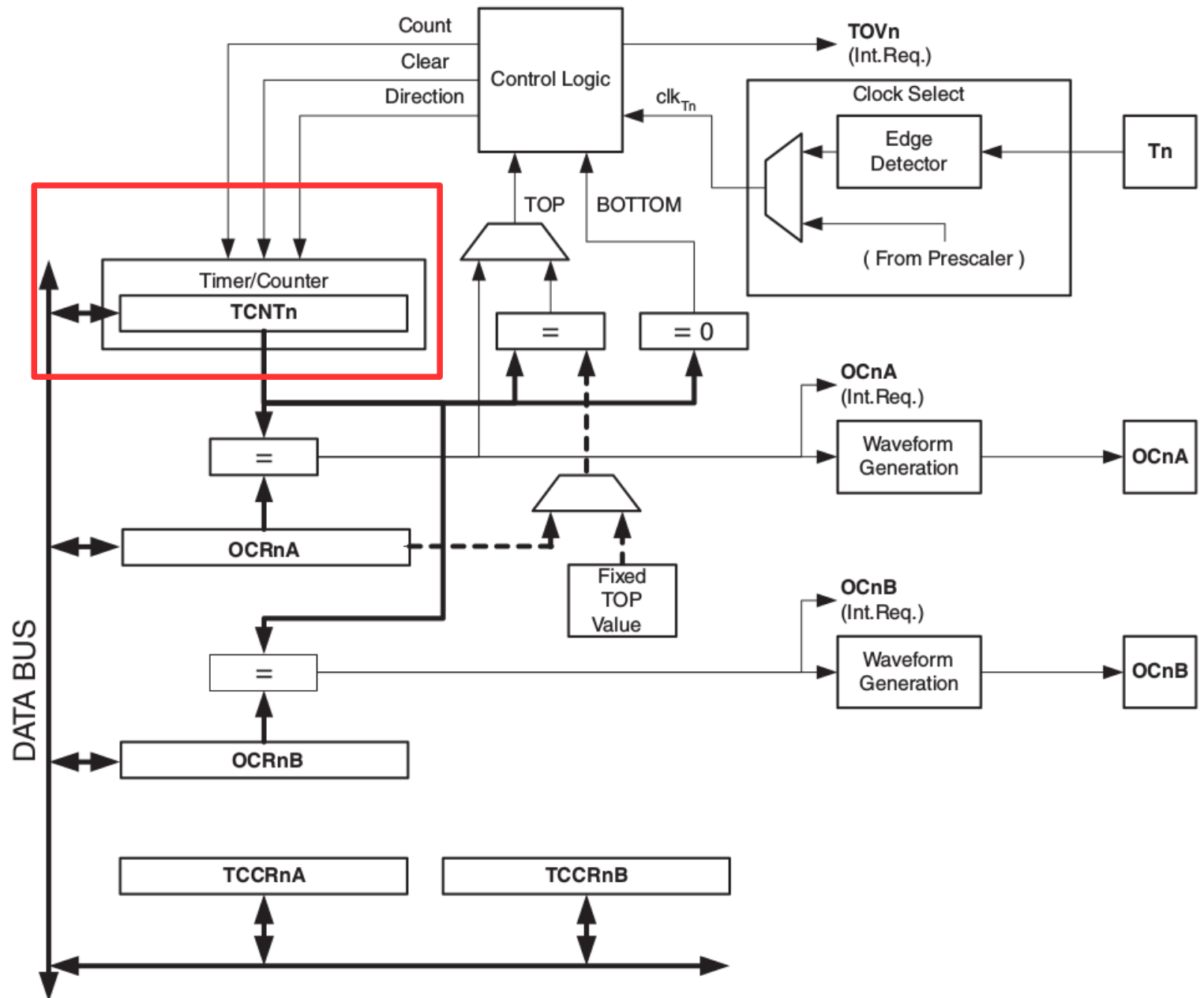
UNO PINOUT



Temporizadores

- **Temporizador/Contador 0**
- **Modo Normal:**
 - **É o modo mais simples de operação, onde o TC0 conta continuamente de forma crescente:**
 - A contagem se dá de 0 até 255 voltando a 0, num ciclo contínuo.
 - Quando a contagem passa de 255 para 0, ocorre o estouro e então, o bit sinalizador de estouro (TOV0) é colocado em 1.
 - **Se habilitada, uma interrupção é gerada.**
 - **A contagem é feita no registrador TCNT0 e um novo valor de contagem pode ser escrito a qualquer momento**
 - Permitindo-se alterar o número de contagens via programação.

Figure 15-1. 8-bit Timer/Counter Block Diagram



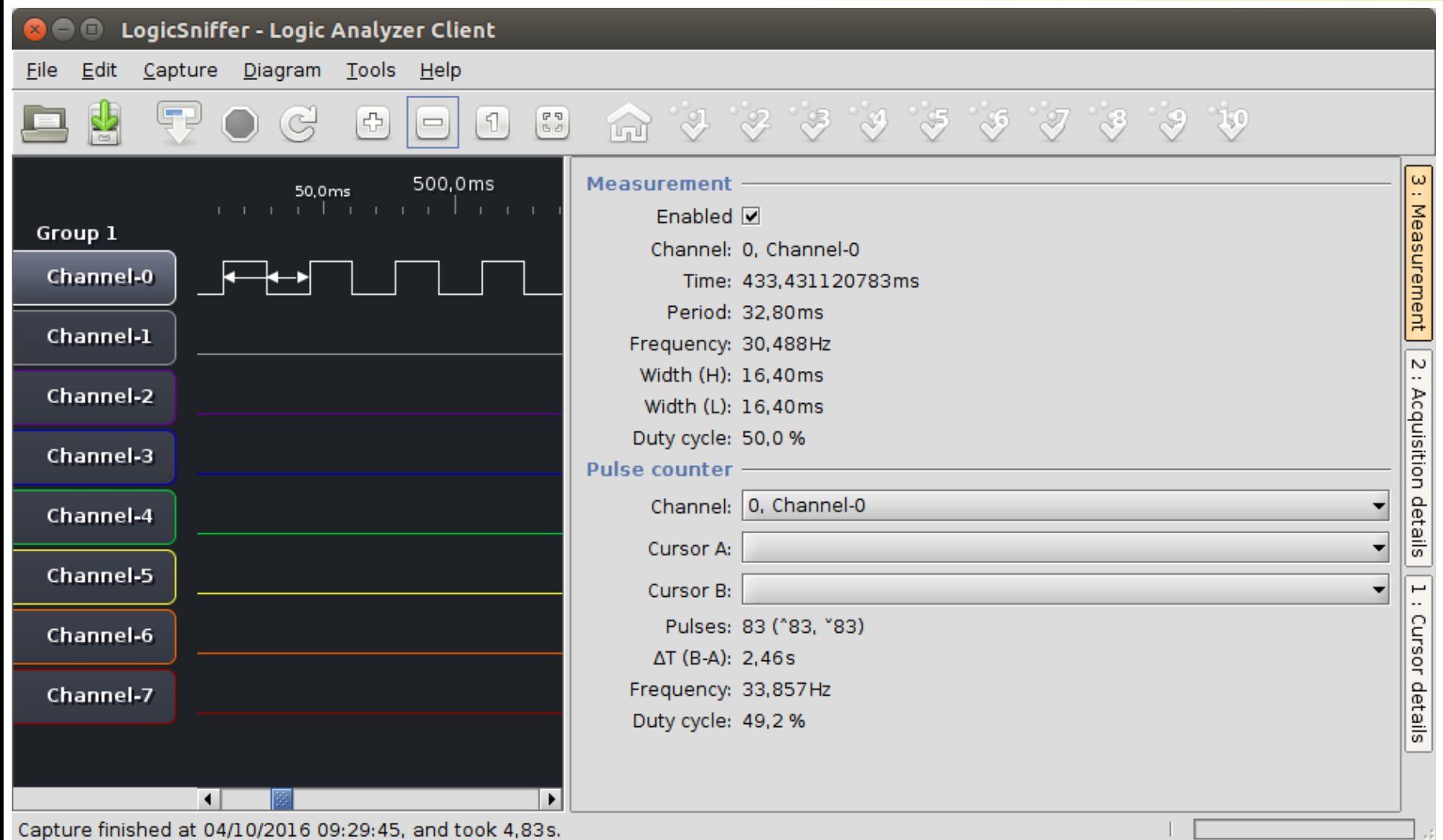
Temporizadores

- Temporizador/Contador 0
- Modo Normal:

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

$$t_{estouro} = 256 \times 1/16 \text{ MHz} \times 1024 = 16,384 \text{ ms}$$

Temporizadores



TC0_estouro.c

```
//-----  
//      AVR e Arduino: Técnicas de Projeto, 2a ed. - 2012.  
//-----  
//=====  
//      HABILITANDO A INTERRUPÇÃO POR ESTOURO DO TC0  
//=====  
#define F_CPU 16000000UL  
#include <avr/io.h>  
#include <avr/interrupt.h>  
  
#define cpl_bit(y,bit)  (y^=(1<<bit))  //troca o estado lógico do bit x da variável Y  
#define LED            PB5  
  
//-----  
ISR(TIMERO_OVF_vect)                      //interrupção do TC0  
{  
    cpl_bit(PORTB,LED);  
}  
//-----  
int main()  
{  
    DDRB  = 0b00100000;                //somente pino do LED como saída  
    PORTB = 0b11011111;                //apaga LED e habilita pull-ups nos pinos não utilizados  
  
    TCCR0B = (1<<CS02) | (1<<CS00);    //TC0 com prescaler de 1024, a 16 MHz gera uma interrupção a cada 16,384 ms  
    TIMSK0 = 1<<TOIE0;                 //habilita a interrupção do TC0  
    sei();                              //habilita interrupções globais  
  
    while(1)  
    {  
        //Aqui vai o código, a cada estouro do TC0 o programa desvia para ISR(TIMERO_OVF_vect)  
    }  
}  
//=====
```

Tab. 6.1 – Interrupções do ATmega328 e seus endereços na memória de programa.


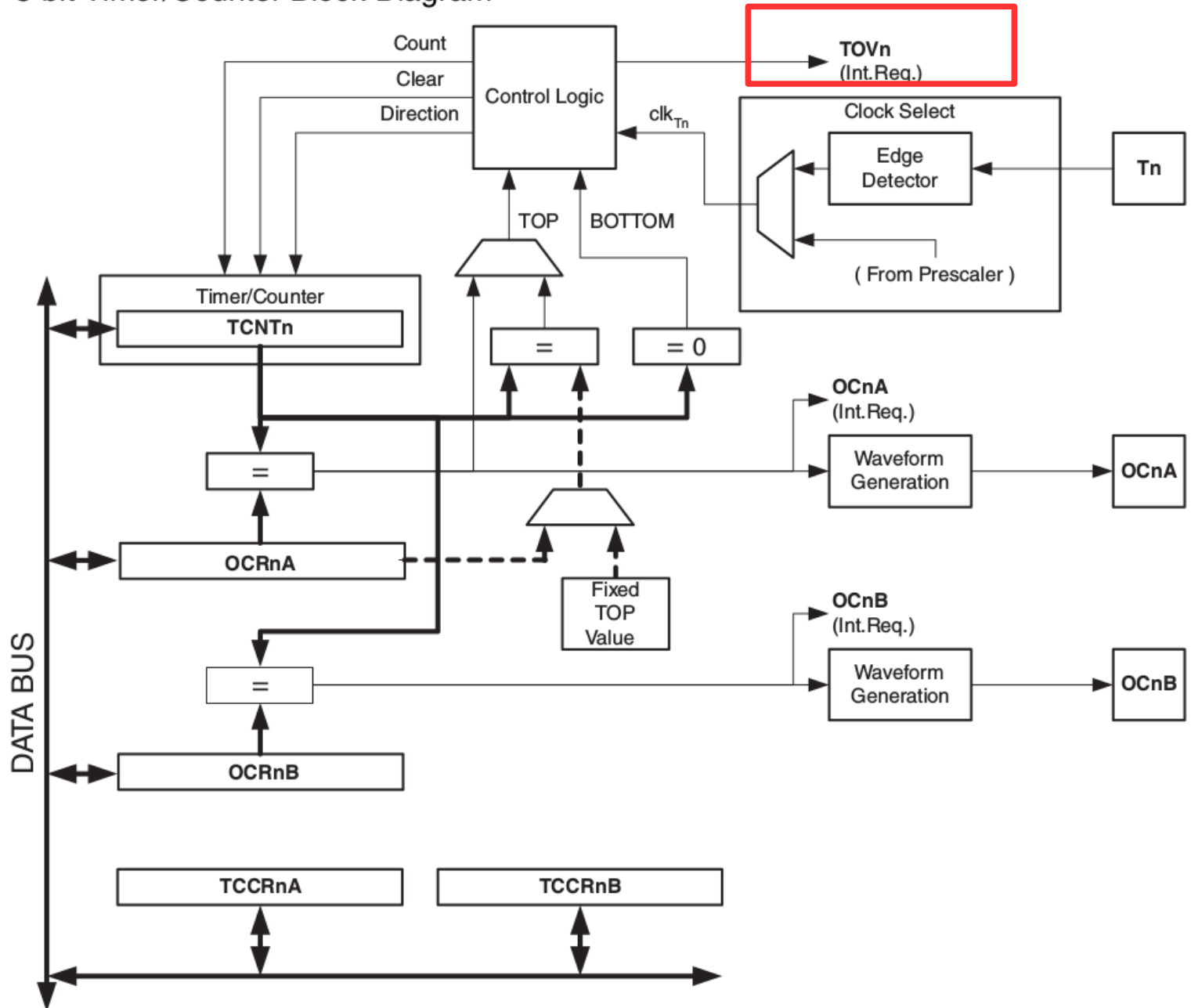
Vetor	End.	Fonte	Definição da Interrupção	Prioridade
1	0x00	RESET	Pino externo, Power-on Reset, Brown-out Reset e Watchdog Reset	
2	0x01	INT0	interrupção externa 0	
3	0x02	INT1	interrupção externa 1	
4	0x03	PCINT0	interrupção 0 por mudança de pino	
5	0x04	PCINT1	interrupção 1 por mudança de pino	
6	0x05	PCINT2	interrupção 2 por mudança de pino	
7	0x06	WDT	estouro do temporizador <i>Watchdog</i>	
8	0x07	TIMER2 COMPA	igualdade de comparação A do TC2	
9	0x08	TIMER2 COMPB	igualdade de comparação B do TC2	
10	0x09	TIMER2 OVF	estouro do TC2	
11	0x0A	TIMER1 CAPT	evento de captura do TC1	
12	0x0B	TIMER1 COMPA	igualdade de comparação A do TC1	
13	0x0C	TIMER1 COMPB	igualdade de comparação B do TC1	
14	0x0D	TIMER1 OVF	estouro do TC1	
15	0x0E	TIMER0 COMPA	igualdade de comparação A do TC0	
16	0x0F	TIMER0 COMPB	igualdade de comparação B do TC0	
17	0x10	TIMER0 OVF	estouro do TC0	
18	0x11	SPI, STC	transferência serial completa - SPI	
19	0x12	USART, RX	USART, recepção completa	
20	0x13	USART, UDRE	USART, limpeza do registrador de dados	
21	0x14	USART, TX	USART, transmissão completa	
22	0x15	ADC	conversão do ADC completa	
23	0x16	EE_RDY	EEPROM pronta	
24	0x17	ANA_COMP	comparador analógico	
25	0x18	TWI	interface serial TWI – I2C	
26	0x19	SPM_RDY	armazenagem na memória de programa pronta	

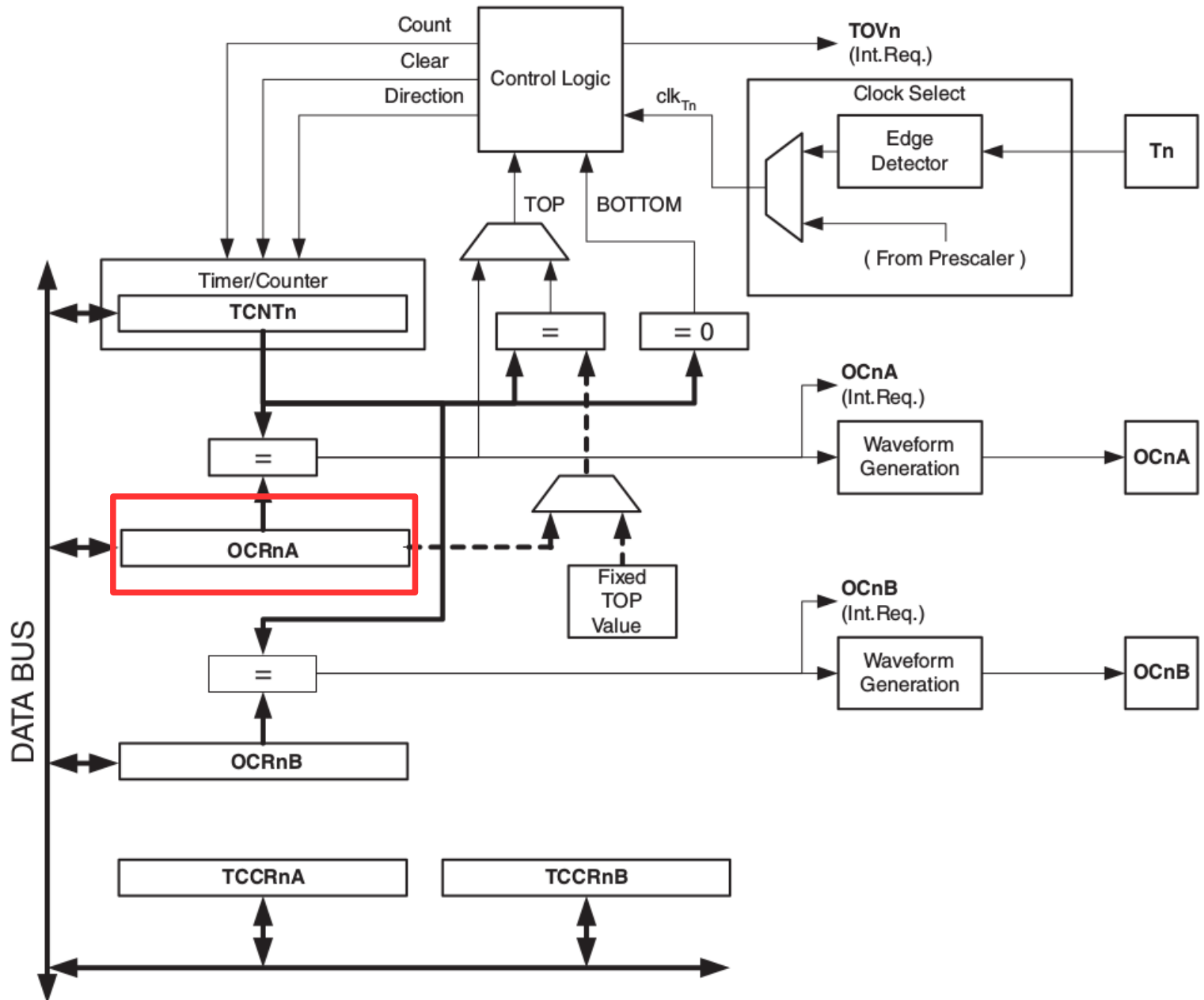
Figure 15-1. 8-bit Timer/Counter Block Diagram



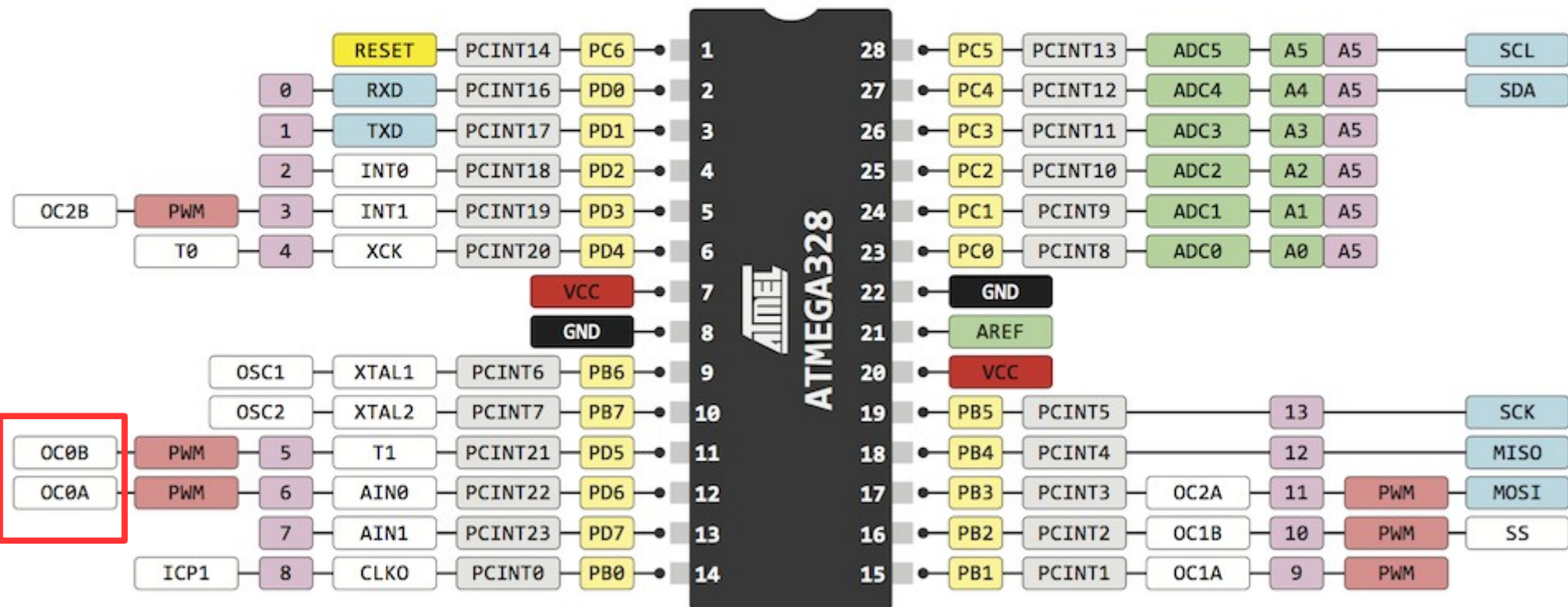
Temporizadores

- **Temporizador/Contador 0**
- **Modo CTC**
- **CTC (Clear Timer on Compare) – limpeza do contador na igualdade de comparação**
 - Utilizado para manipular a resolução do TC0
 - O contador é zerado quando o valor do contador (TCNT0) é igual ao valor de OCRA0 (valor TOP de contagem)
 - » **O contador conta de 0 até o valor de OCRA0**
 - O modo CTC permite configurar os pinos OC0A e OC0B para gerar ondas quadradas
 - Uma interrupção pode ser gerada cada vez que o contador atinge o valor de comparação (OCR0A ou OCR0B)

Figure 15-1. 8-bit Timer/Counter Block Diagram



THE
DEFINITIVE
ATMEGA328
&Arduino
PINOUT DIAGRAM



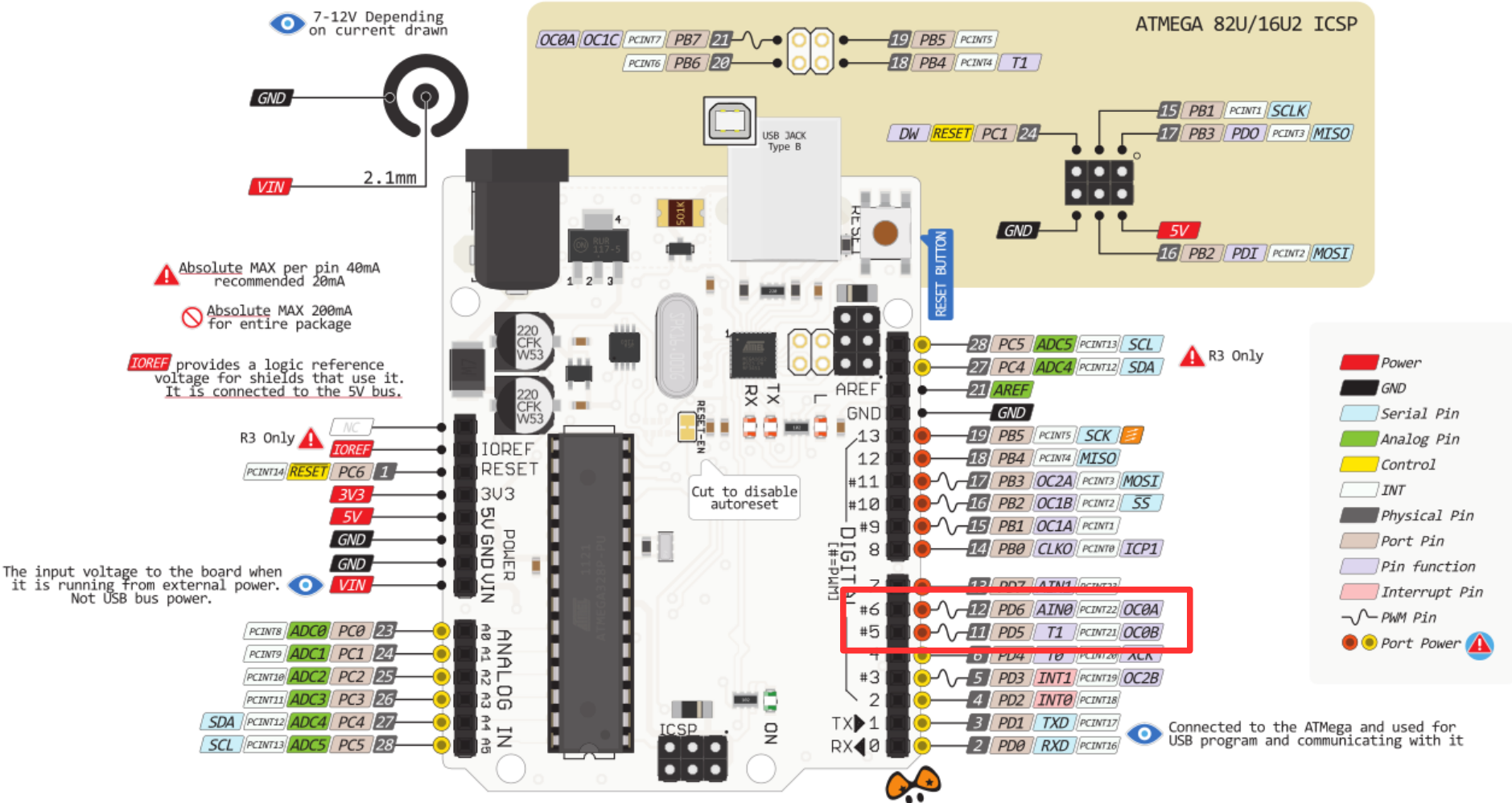
www.piggyback.com




18 FEB 2013

ver 2 rev 0 - 19.02.2013

UNO PINOUT



Tab. 6.1 – Interrupções do ATmega328 e seus endereços na memória de programa.

Vetor	End.	Fonte	Definição da Interrupção	Prioridade
1	0x00	RESET	Pino externo, Power-on Reset, Brown-out Reset e Watchdog Reset	
2	0x01	INT0	interrupção externa 0	
3	0x02	INT1	interrupção externa 1	
4	0x03	PCINT0	interrupção 0 por mudança de pino	
5	0x04	PCINT1	interrupção 1 por mudança de pino	
6	0x05	PCINT2	interrupção 2 por mudança de pino	
7	0x06	WDT	estouro do temporizador <i>Watchdog</i>	
8	0x07	TIMER2 COMPA	igualdade de comparação A do TC2	
9	0x08	TIMER2 COMPB	igualdade de comparação B do TC2	
10	0x09	TIMER2 OVF	estouro do TC2	
11	0x0A	TIMER1 CAPT	evento de captura do TC1	
12	0x0B	TIMER1 COMPA	igualdade de comparação A do TC1	
13	0x0C	TIMER1 COMPB	igualdade de comparação B do TC1	
14	0x0D	TIMER1 OVF	estouro do TC1	
15	0x0E	TIMER0 COMPA	igualdade de comparação A do TC0	
16	0x0F	TIMER0 COMPB	igualdade de comparação B do TC0	
17	0x10	TIMER0 OVF	estouro do TC0	
18	0x11	SPI, STC	transferência serial completa - SPI	
19	0x12	USART, RX	USART, recepção completa	
20	0x13	USART, UDRE	USART, limpeza do registrador de dados	
21	0x14	USART, TX	USART, transmissão completa	
22	0x15	ADC	conversão do ADC completa	
23	0x16	EE_RDY	EEPROM pronta	
24	0x17	ANA_COMP	comparador analógico	
25	0x18	TWI	interface serial TWI – I2C	
26	0x19	SPM_RDY	armazenagem na memória de programa pronta	

Temporizadores

- **Modo CTC**

- Os pinos OC0A e OC0B foram configurados para trocar de estado quando ocorre a igualdade entre o valor do registrador TCNT0 com OCR0A e OCR0B

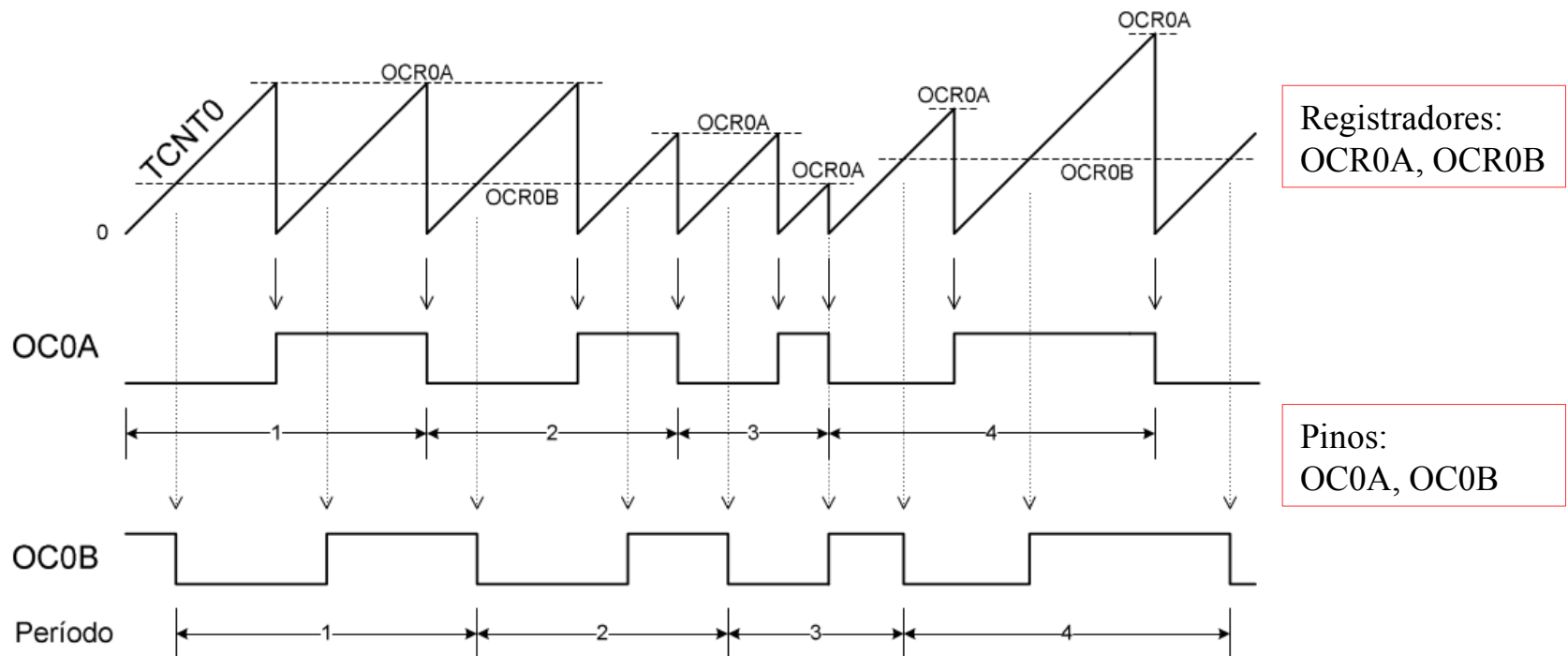


Fig. 9.4 – Modo CTC para a geração de ondas quadradas.

Temporizadores

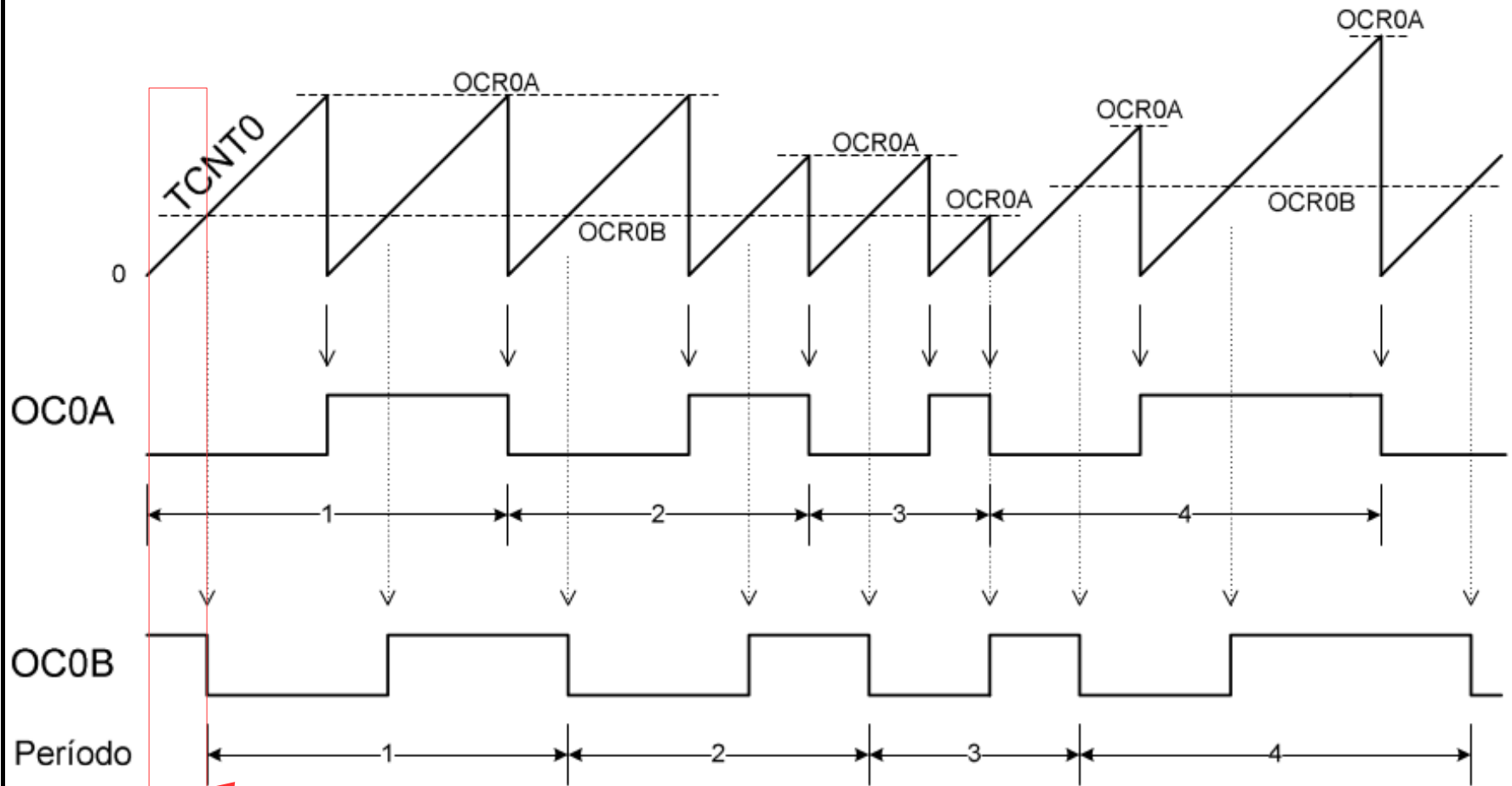
- **Modo CTC**

- Para o pino OC0B, **o registrador OCR0B é empregado para deslocar a forma de onda em relação a OC0A**
 - » **Deve ter valor menor ou igual a OCR0A**
- Os pinos OC0A e OC0B devem ser configurados como saída e ajustados para trocar de estado em cada igualdade de comparação através dos bits do modo de comparação
- A frequência no modo CTC é definida por:

$$f_{OC0A} = \frac{f_{osc}}{2N(1+OCR0A)} \quad [\text{Hz}] \quad f_{OC0A} = 16000000/2 \times 1(1+200) = 39800\text{Hz}$$

- » **f_{osc} é a frequência de operação do microcontrolador, OCR0A assume valores entre 0 e 255 e N é o fator do prescaler (1, 8, 64, 256 ou 1024)**

Temporizadores



Onda do pino OC0B deslocada em relação ao pino OC0A

Fig. 9.4 – Modo CTC para a geração de ondas quadradas.

TC0_PWMs.c

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

int main(void)
{
    DDRD = 0b01100000;           //pinos OC0B e OC0A (PD5 e PD6) como saída
    PORTD = 0b10011111;          //zera saídas e habilita pull-ups nos pinos não utilizados

    //MODO CTC
    //TCCR0A = 0b01010010;        //habilita OC0A e OC0B para trocar de estado na igualdade de comparação
    TCCR0A = (1<<COM0A0)|(1<<COM0B0)|(1<<WGM01); //habilita OC0A e OC0B para trocar de estado na igualdade de comparação
    //TCCR0B = 0b00000001;        //liga TC0 com prescaler = 1.
    TCCR0B = (1<<CS00);           //liga TC0 com prescaler = 1.
    OCR0A = 200;                  //maximo valor de contagem
    OCR0B = 100;                  //deslocamento de OC0B em relação a OC0A

    while(1)
    {
        //O programa principal vai aqui
    }
}
//-----
```


Temporizadores

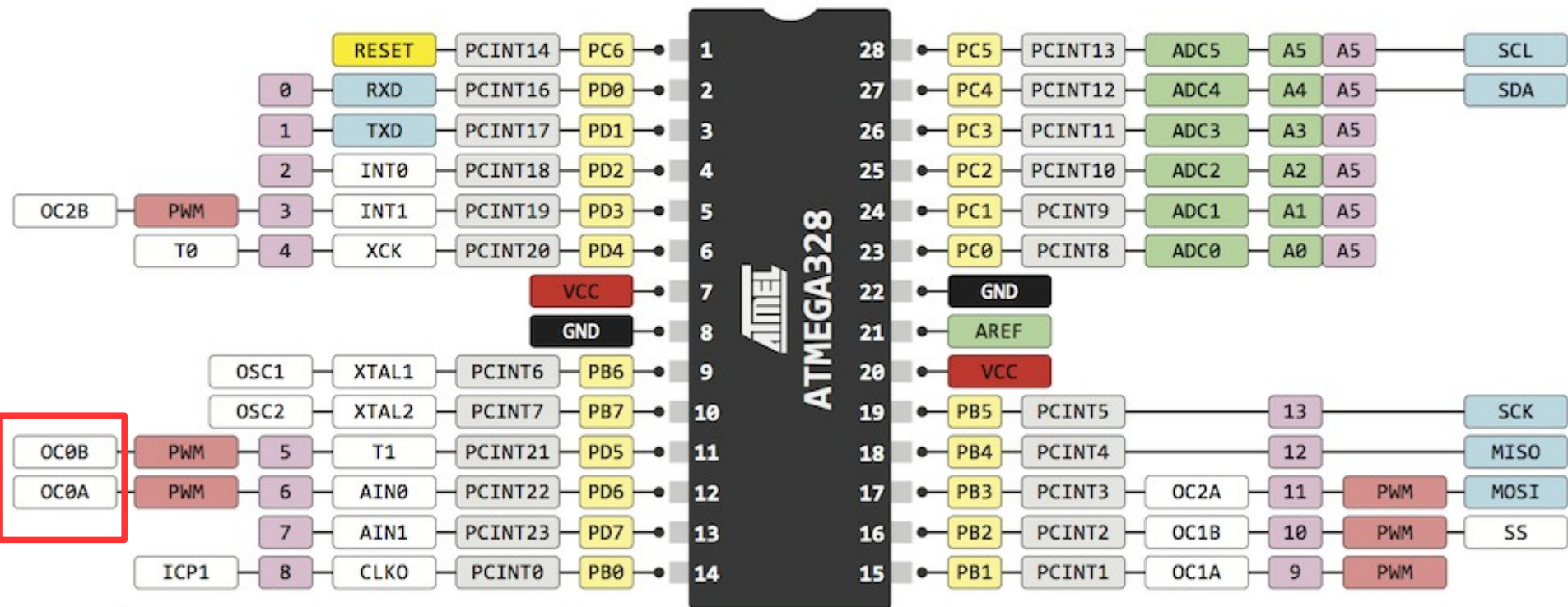
- **Modo CTC**

- Para calcular o valor de OCR0A baseado na frequência desejada, basta isolá-lo na equação acima, o que resulta em:

$$OCR0A = \frac{f_{osc}}{2N \cdot f_{OC0A}} - 1$$

- » O modo CTC não necessita ser empregado para a geração de formas de onda, pode ser utilizado para temporizações e contagens externas
 - » Ele permite flexibilizar o número de contagens do TC0 e gerar interrupções por igualdade de comparação do TCNT0 com os registradores OCR0A e OCR0B

THE
DEFINITIVE
ATMEGA328
&Arduino
PINOUT DIAGRAM



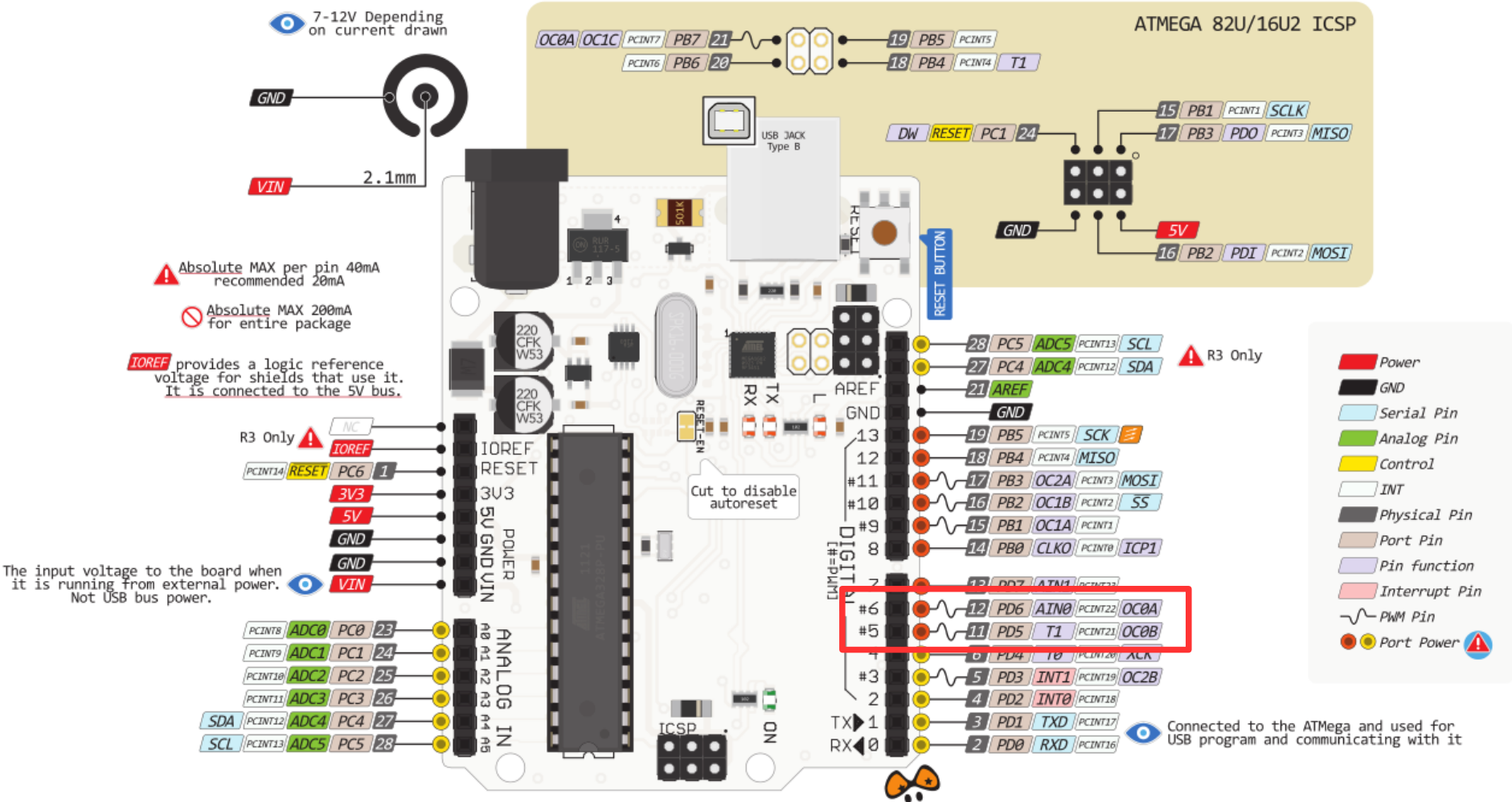
www.piggyback.com



18 FEB 2013

ver 2 rev 0 - 19.02.2013

UNO PINOUT



Temporizadores

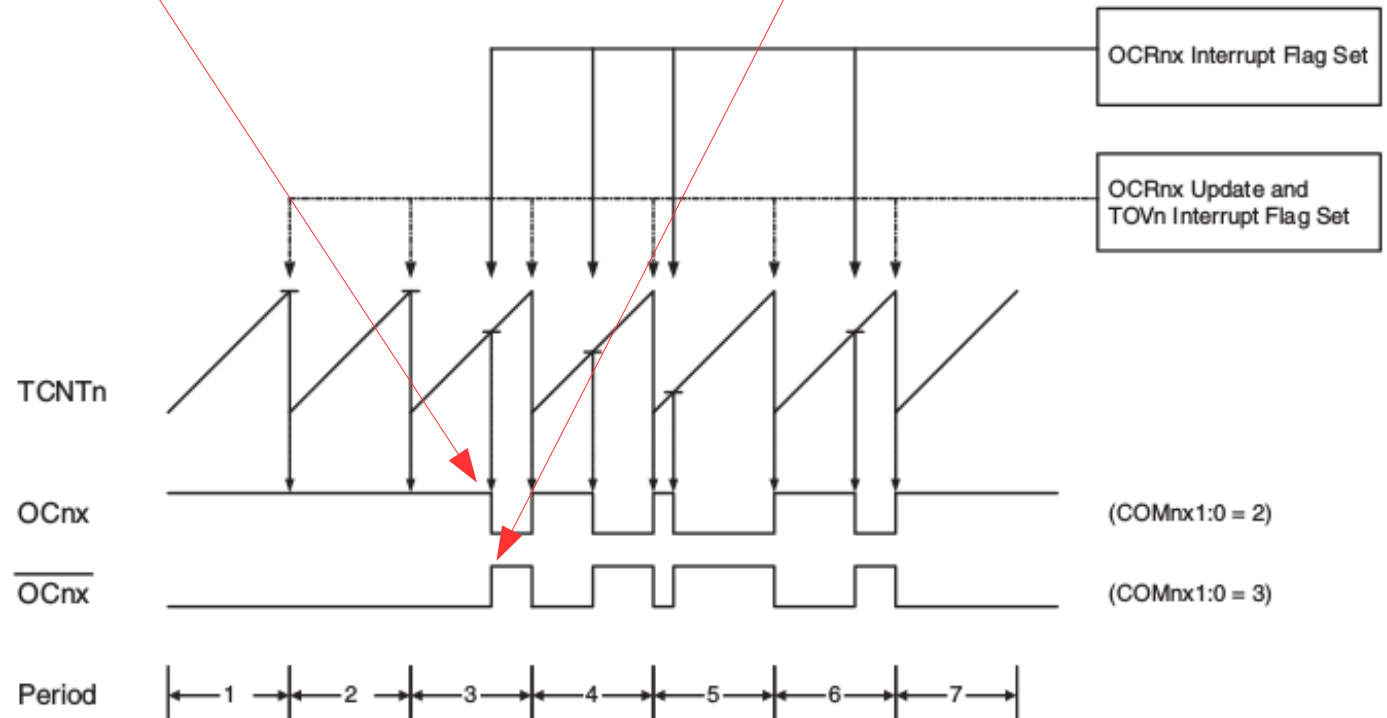
- **Modo PWM RÁPIDO**

- O modo de modulação por largura de pulso rápido permite a geração de um sinal PWM de alta frequência
- O contador conta de zero até o valor máximo e volta a zero
 - » **No modo de comparação com saída não-invertida:**
 - » O pino OC0A é zerado na igualdade entre TCNT0 e OCR0A, e colocado em 1 no valor mínimo do contador
 - » **No modo de comparação com saída invertida:**
 - » O pino OC0A é ativo na igualdade e zerado no valor mínimo
- O valor de comparação OCR0A pode ser atualizado na interrupção por estouro do contador cada vez que o contador chegar no valor máximo de contagem, gerando um sinal PWM variável

Saída não invertida: OCnx é zerado na igualdade entre TCNTn e OCRnx

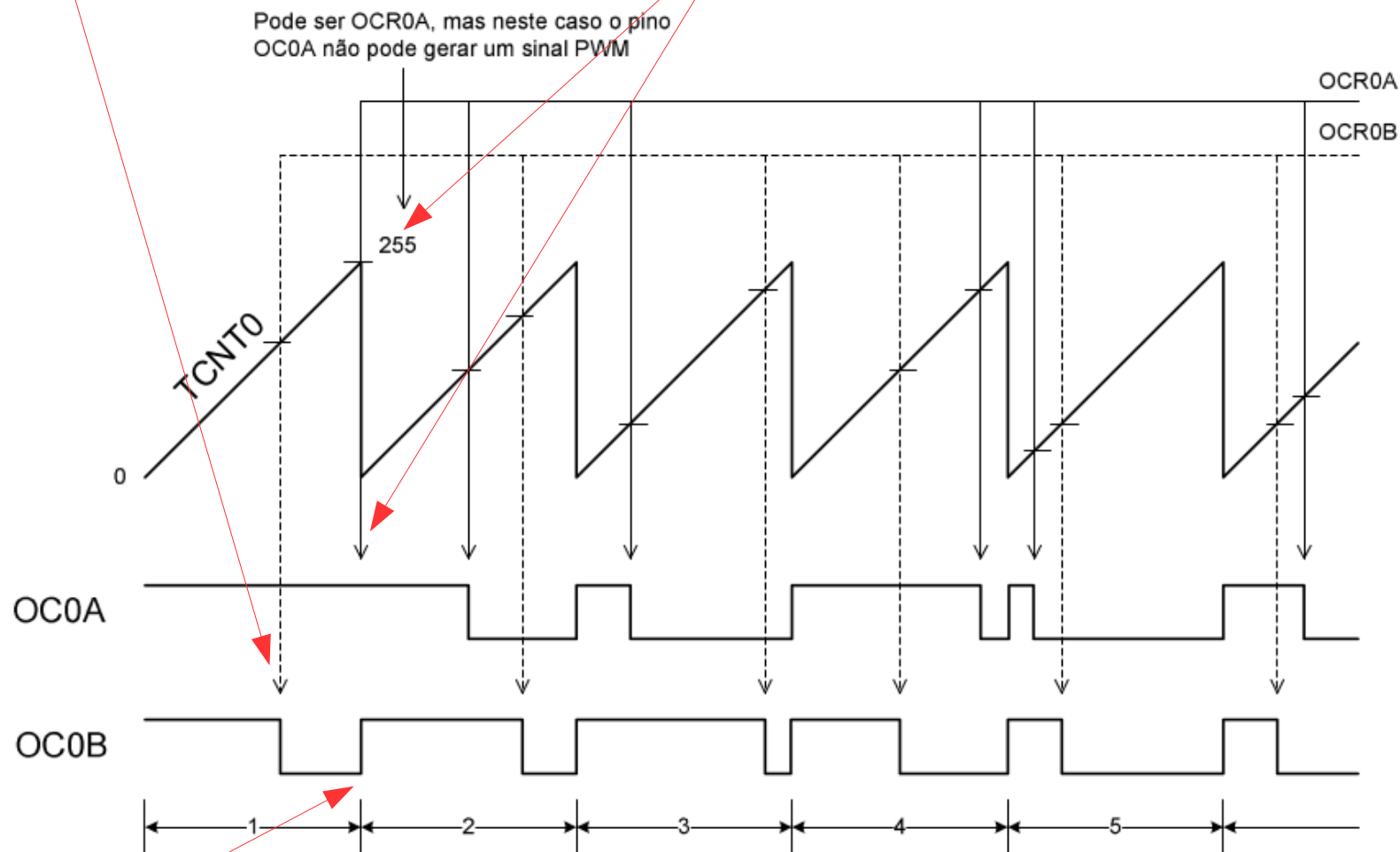
Saída invertida: OCnx é ativado na igualdade entre TCNTn e OCRnx

Figure 15-6. Fast PWM Mode, Timing Diagram



O pino OC0A/OC0B é zerado na igualdade entre TCNT0 e OCR0A/OCR0B

O valor máximo (255) vai deixar o sinal PWM em 1 se foi habilitada a saída não invertida



O pino OC0A/OC0B é colocado em 1 no valor mínimo do contador

Fig. 9.5 – Modo PWM rápido, saídas não invertidas.

Temporizadores

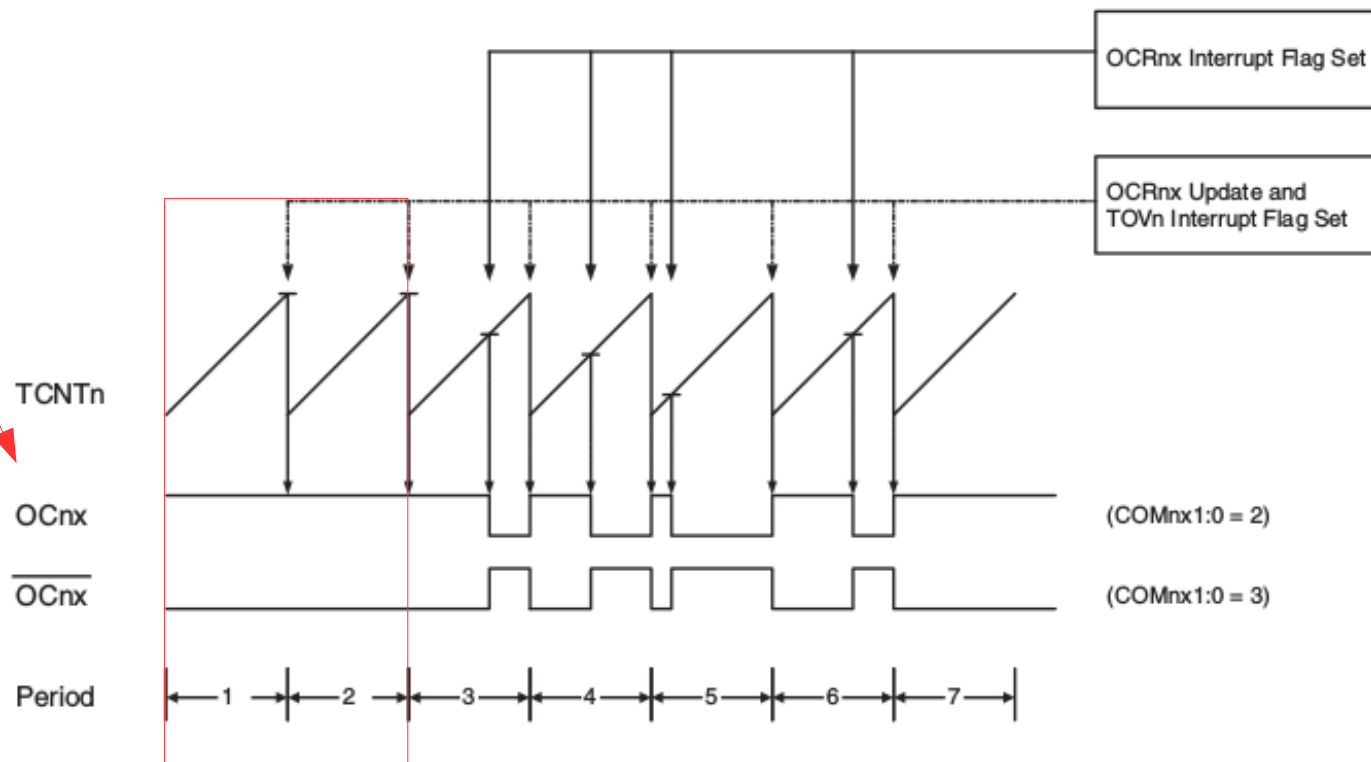
- **Modo PWM RÁPIDO**

- Quando se controla o ciclo ativo do sinal PWM

- » O valor zero para OCR0A/OCR0B produz um pequeno ruído
 - » O valor máximo (255) vai deixar o sinal PWM em 0 se foi habilitada a saída invertida
 - » O valor máximo (255) vai deixar o sinal PWM em 1 se foi habilitada a saída não invertida

O valor máximo (255) vai deixar o sinal PWM em 1 se foi habilitada a saída não invertida

Figure 15-6. Fast PWM Mode, Timing Diagram



O valor máximo (255) vai deixar o sinal PWM em 0 se foi habilitada a saída invertida

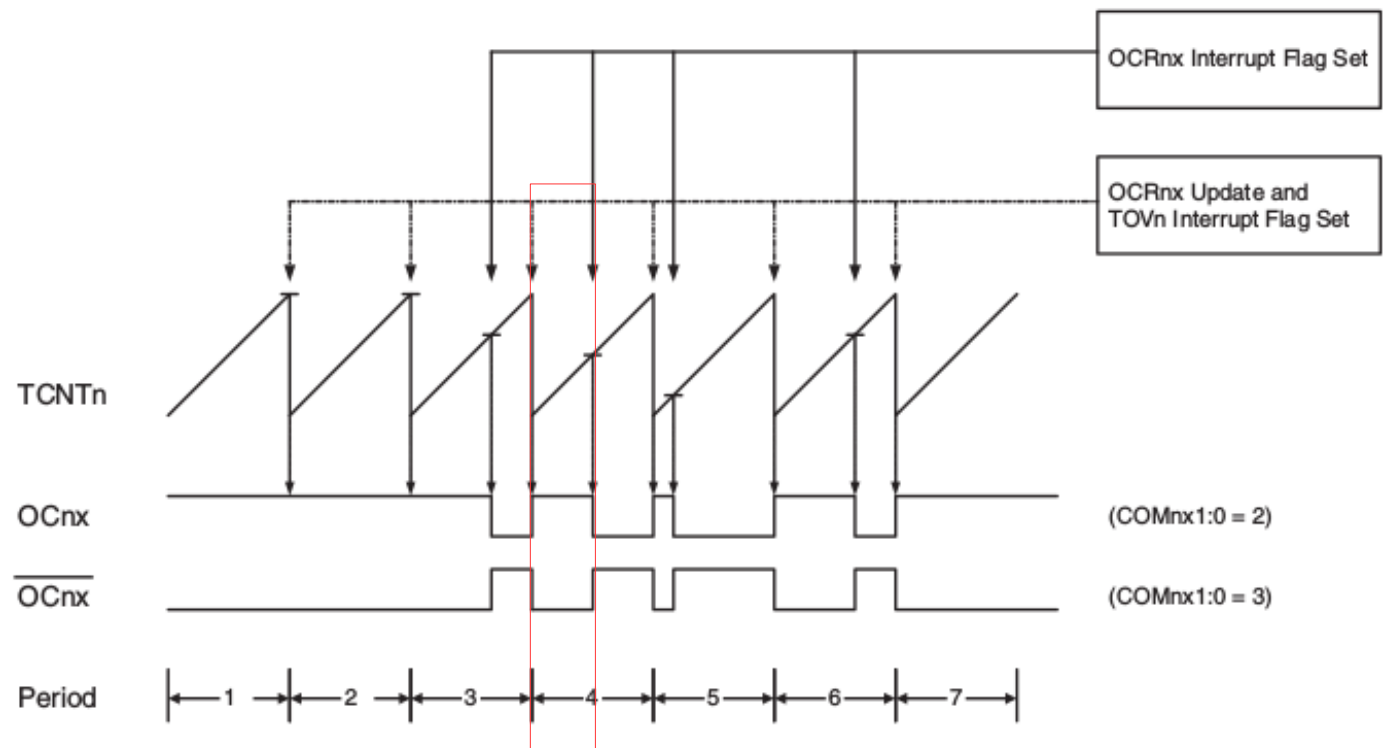
Temporizadores

- **Modo PWM RÁPIDO**

- É o registrador OCR0A/OCR0B que determina

- » **O ciclo ativo do sinal PWM com saída não-invertida**
 - » **O ciclo inativo do sinal PWM com saída invertida**

Figure 15-6. Fast PWM Mode, Timing Diagram



Temporizadores

- **Modo PWM RÁPIDO**

- A frequência de saída do PWM é calculada como:

$$f_{OC0A_PWM} = \frac{f_{osc}}{N(1+TOP)} \quad [\text{Hz}]$$

» Onde f_{osc} é a frequência de operação do microcontrolador, N é o fator do prescaler (1, 8, 64, 256 ou 1024) e TOP assume valores conforme tab. 9.7

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

Temporizadores

- **Modo PWM RÁPIDO**

- A frequência de saída do PWM é calculada como:

$$f_{OC0A_PWM} = \frac{f_{osc}}{N(1+TOP)} \quad [\text{Hz}]$$

- Um sinal com ciclo ativo do PWM de 50% pode ser obtido ajustando-se OC0A para mudar de estado a cada igualdade de comparação
- A máxima frequência é obtida quando OCR0A é zero ($f_{OC2_PWM} = f_{osc}/2$)

Temporizadores

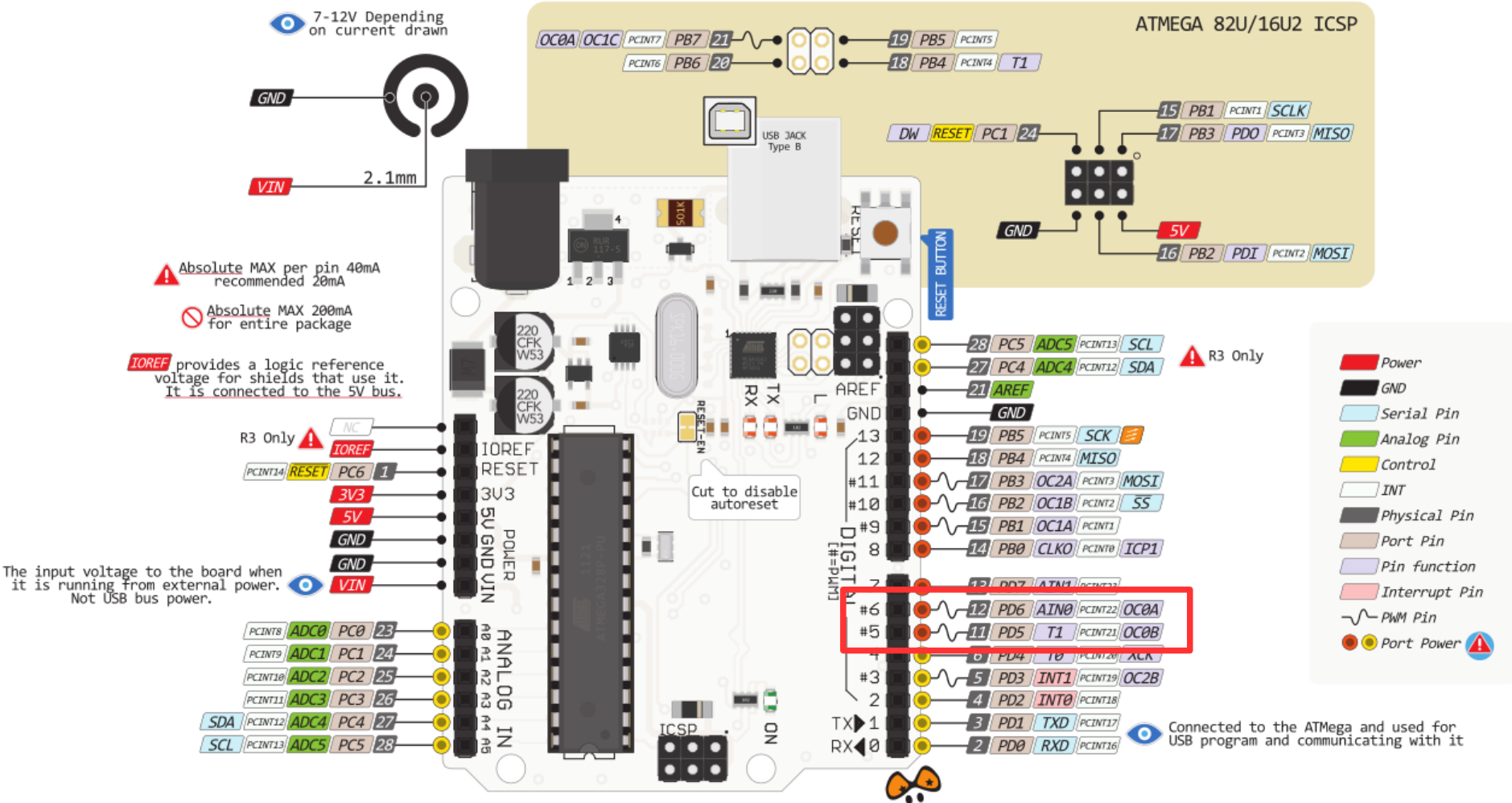
Tab. 9.2 – Modo PWM rápido.

COM0A1	COM0A0	Descrição
0	0	Operação normal do pino, OC0A desconectado.
0	1	WGM02 = 0: operação normal do pino, OC0A desconectado. WGM02 = 1: troca de estado do OC0A na igualdade de comparação.
1	0	OC0A é limpo na igualdade de comparação, OC0A ativo no valor do TC mínimo (modo não invertido).
1	1	OC0A é ativo na igualdade de comparação e limpo no valor do TC mínimo (modo invertido).

PINOUT DIAGRAM



UNO PINOUT



Temporizadores

- **Modo PWM RÁPIDO**

- A frequência de saída do PWM é calculada como:

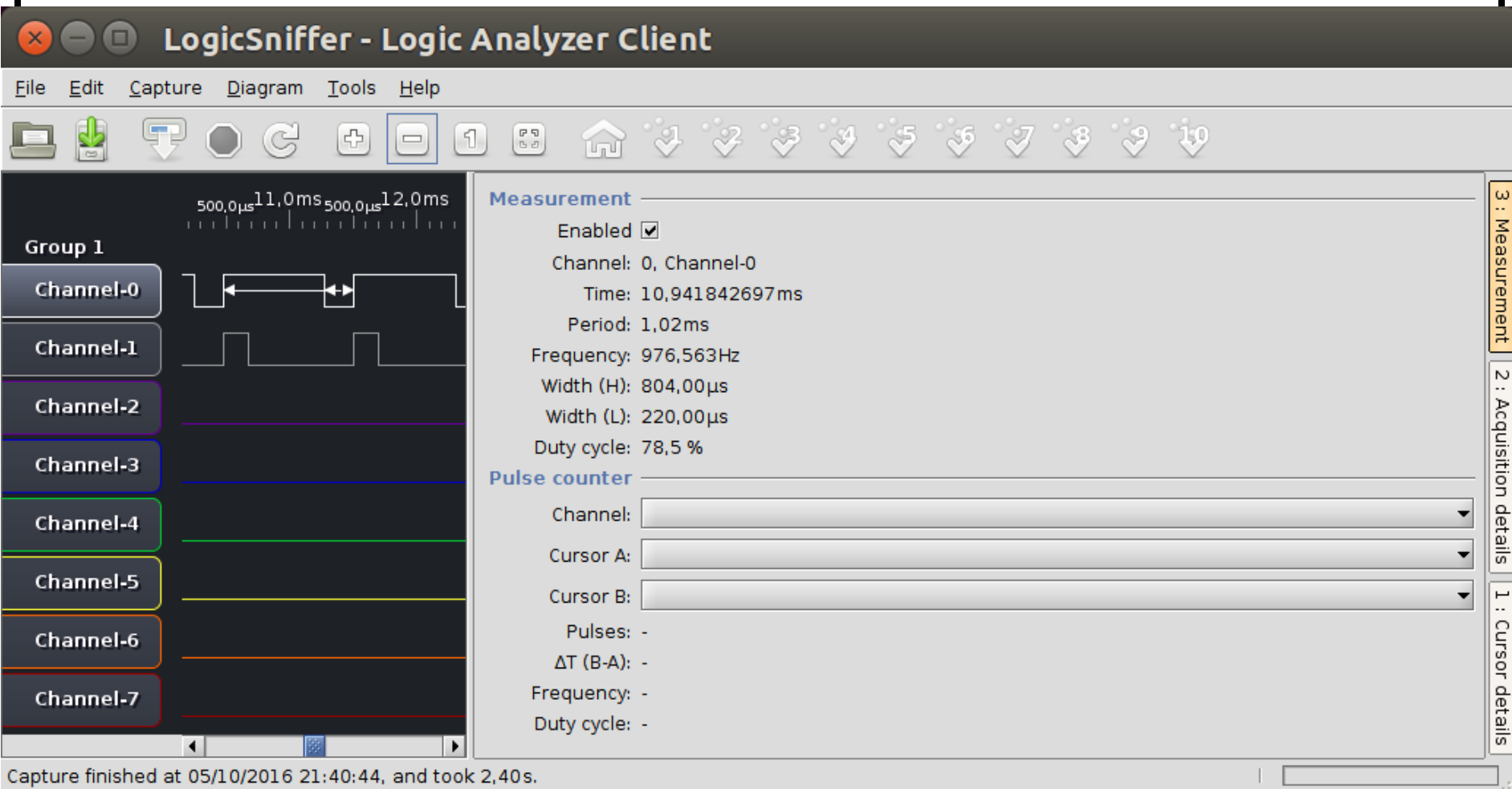
$$f_{OCA_PWM} = \frac{f_{osc}}{N(1+TOP)} \quad [\text{Hz}]$$

$$f_{OCA_PWM} = 16000000 / 64(1+255) = 976,5625 \text{ Hz}$$

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

$$t_{estouro} = 256 \times 1/16 \text{ MHz} \times 64 = 1,024 \text{ ms} = \text{Período}$$

Temporizadores



Temporizadores

- **TOP = 255**
- **Período = 1,02 ms**
- **OCR0A = 200**
 - Para calcular o tempo do sinal ativo (HIGH) utiliza-se regra de três:
 $255 \rightarrow 1,02 \text{ ms}$
 $200 \rightarrow x \text{ ms}$
 $x = (200 \times 1,02)/255$
 $x = 0,8 \text{ ms} = 800 \mu\text{s}$

 $255 \rightarrow 100\%$
 $200 \rightarrow x \%$
 $X = (200 \times 100)/255 = 78,43\%$

TC0_PWMs.c

```
//-----  
//      AVR e Arduino: Técnicas de Projeto, 2a ed. - 2012.      //  
//-----  
  
#define F_CPU 16000000UL  
#include <avr/io.h>  
  
int main(void)  
{  
    DDRD = 0b01100000;          //pinos OC0B e OC0A (PD5 e PD6) como saída  
    PORTD = 0b10011111;        //zera saídas e habilita pull-ups nos pinos não utilizados  
  
    //fast PWM, TOP = 0xFF, OC0A e OC0B habilitados  
    //TCCR0A = 0b10100011;      //PWM não invertido nos pinos OC0A e OC0B  
    TCCR0A = (1<<COM0A1)|(1<<COM0B1)|(1<<WGM01)|(1<<WGM00); //PWM não invertido nos pinos OC0A e OC0B  
    //TCCR0B = 0b00000011;      //liga TC0, prescaler = 64  
    TCCR0B = (1<<CS01)|(1<<CS00); //liga TC0, prescaler = 64  
    OCR0A = 200;                //controle do ciclo ativo do PWM OC0A  
    OCR0B = 50;                //controle do ciclo ativo do PWM OC0B  
    while(1)  
    {  
        //O programa principal vai aqui  
    }  
}
```

Temporizadores

- **PWM e Arduino**

- <https://www.arduino.cc/en/Reference/AnalogWrite>
- <https://www.arduino.cc/en/Tutorial/PWM>
- http://garretlab.web.fc2.com/en/arduino/inside/arduino/wiring_analog.c/analogWrite.html
- <http://www.righto.com/2009/07/secrets-of-arduino-pwm.html>
- <https://123d.circuits.io/circuits/2124936>

Referências

- AVR e Arduino – Técnicas de Projeto
- Datasheet - Atmel 8-bit Microcontroller with 4/8/16/32KBytes
 - » **15. 8-bit Timer/Counter0 with PWM**