MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CAMPUS SÃO JOSÉ
ENGENHARIA DE TELECOMUNICAÇÕES

RELATÓRIO
TRABALHO 1 STE

Aluno: Mário André Lehmkuhl de Abreu
Matéria: Sistemas Embarcados (STE)
Professor: Roberto de Matos

Setembro
2018

1 Introdução

O Trabalho consiste em desenvolver 5 laboratorios definidos pelo professor utilizando a plataforma arduino, para entender como é programado o chip AVR da plataforma atraves da configuração de seus registradores. Nesse trabalho o Arduino usado foi o modelo Mega com o AVR ATmega2560. Para cada laboratorio se desenvolveu primeiro códigos utilizando a IDE do arduino, e agora deve-se re-implementar esses códigos usando C++ e AVR-Libc. O procedimento de primeiro programar com a IDE do arduino e depois utilizando C++ e AVR-Libc, aplica-se para demonstrar como a IDE do arduino facilita o processo na hora de configurar o chip AVR, pois na sua programação não é necessario saber qual registrador vai ser usar, isso porque a IDE ja faz isso ocultamente atraves das funções da bibliotecas. Já utilizando o C++ e AVR-Libc deve-se saber qual registrador vai se usar. A seguir é mostrado a descrição de cada laboratório proposto.

1.1 Lab 01: GPIO - Hello LED

Nesse laboratório deve-se construir um experimento com dois botões (botão 1 e 2) e dois leds (led 1 e 2) para usar o GPIO. O GPIO é chamado General Purpose Input/ Output, e é basicamente um grupo de pinos responsáveis pela comunicação de sinais digitais de entrada e saída em uma placa. Portanto, a proposta é ligar e desligar um led quando o botão é pressionado ou não. Desse modo ao apertar o botão 1 o led 1 ascende, e ao apertar o botão 2 o led 2 ascende. A figura 1 mostra o esquematico desse laboratório.

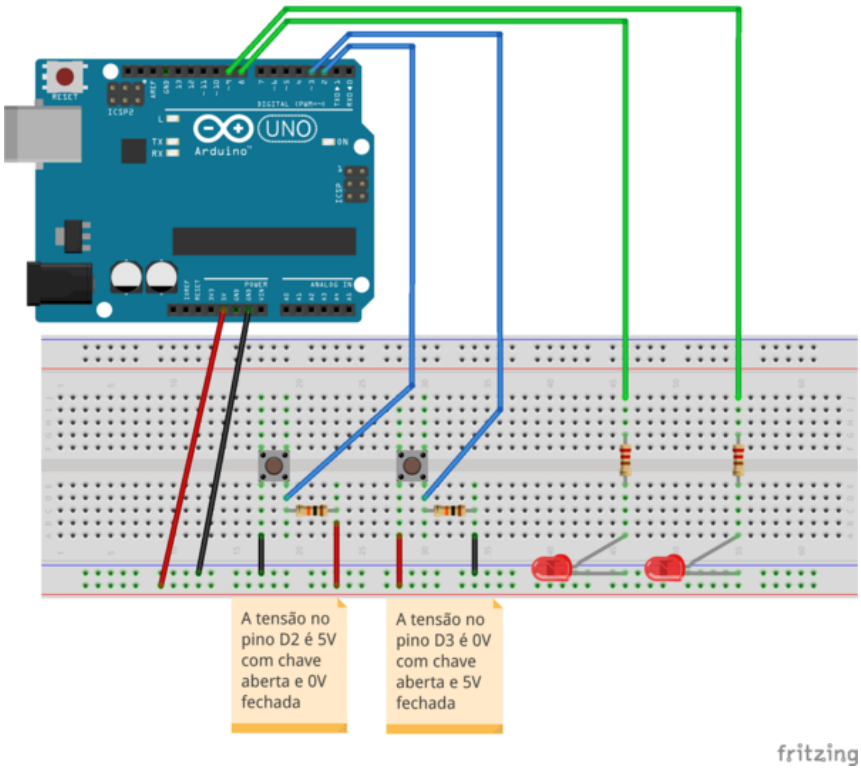

Figura 1: esquema Lab 01

1.2 Lab 02: UART - Serial Communication

Nesse laboratório deve-se fazer um experimento para usar o UART. O UART (transmissor e receptor assíncrono universal) é um tipo de comunicação serial, e representa que apenas um bit de informação é enviado por vez. A UART é, portanto, uma porta serial em uma placa usada para comunicação entre a placa e um computador ou outros dispositivos. Desse modo a proposta é fazer a comunicação serial da UART com um computador e usando monitores seriais visualizar os resultados dos processos.

1.3 Lab 04: UART - ADC e Sensor Reading

Nesse laboratório deve-se fazer o uso do ADC que é um conversor analógico-digital. Ele é um dispositivo eletrônico capaz de gerar uma representação digital a partir de uma grandeza analógica (geralmente um sinal representado por um nível de tensão ou intensidade de corrente elétrica). Os ADCs são úteis e normalmente necessários na interface entre dispositivos digitais (microprocessadores e microcontroladores) e dispositivos analógicos e são usados em aplicações como sensores de leitura, varredura de áudio e vídeo. Desse modo a proposta é ler uma porta analogica e com o valor lido fazer a conversão de analogico para digital e o calculo RMS do valor lido. Os resultados devem ser mostrados na tela de um computador utilizando comunicação serial(UART) entre o arduino e o computador.

1.4 Lab 07: GPIO and External Interrupts - part 1

Nesse laboratório deve-se usar o GPIO e interrupções. Uma interrupção é uma ação executada pelo microcontrolador que permite parar de fazer o que está sendo executado no momento e responder imediatamente ao dispositivo solicitando a interrupção. Existem dois tipos de solicitações de Interrupções: chamadas do hardware (sinal externo) e do software (código de programação). Neste laboratório deve-se usar a interrupção externa. Assim a proposta é construir um experimento com um led e um botão e usando a interrupção externa deve-se ligar e desligar um led a cada vez que o botão é precionado. A figura 2 mostra o esquematico desse laboratório.

1.5 Lab 08: GPIO and External Interrupts - part 2

Nesse laboratório deve-se usar o GPIO e interrupções. Desse modo a proposta é construir um experimento com um led e um botão e usando a interrupção externa deve-se ligar um led sempre que o botão for pressionado e desliga-lo quando o botão não estiver mais pressionado. A figura 2 mostra o esquematico desse laboratório.

2 Desenvolvimento

Aqui é descrito o procedimentos realizados para o desenvolvimento de cada laboratório.

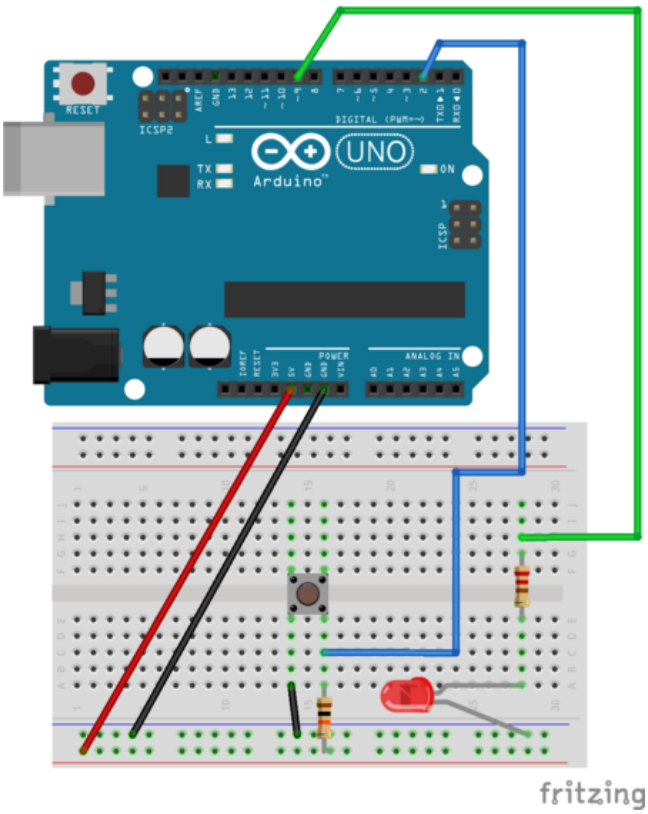

Figura 2: Esquema Lab 07 e 08

2.1 Lab 01

Nessse experimento se definiu os pinos 2 e 3 como entrada para ler os estados dos botões e os pinos 8 e 9 como saída para alimentar os leds.

Os pinos de entrada e saída no AVR são divididos em PORTs. Cada PORT possui 3 registradores que são:

DDR(Data Direction Register): Responsável por determinar se os pinos de um determinado PORT se comportarão como entrada ou saída. Cada bit do registrador DDR controla o estado do respectivo pino. Se o bit estiver em 0 o pino é uma entrada e se for 1 é uma saída.

PORT: Responsável por determinar se um pino está definido como alto(HIGH) bit com valor 1 ou baixo(LOW) bit com valor 0.

PIN: Responsável por guardar o estado lógico de um pino.

Para se utilizar os pinos desejado, deve-se configurar os bits desses registradores. Para saber qual PORT esta associado cada pino, deve-se olhar o datasheet do AVR. Na figura 3 é mostrado o diagrama de pinos do AVR retirado do datasheet, para poder identificar o PORT dos pinos.

Olhando o diagrama se verificou que os pinos 2 e 3 digital estão associados ao

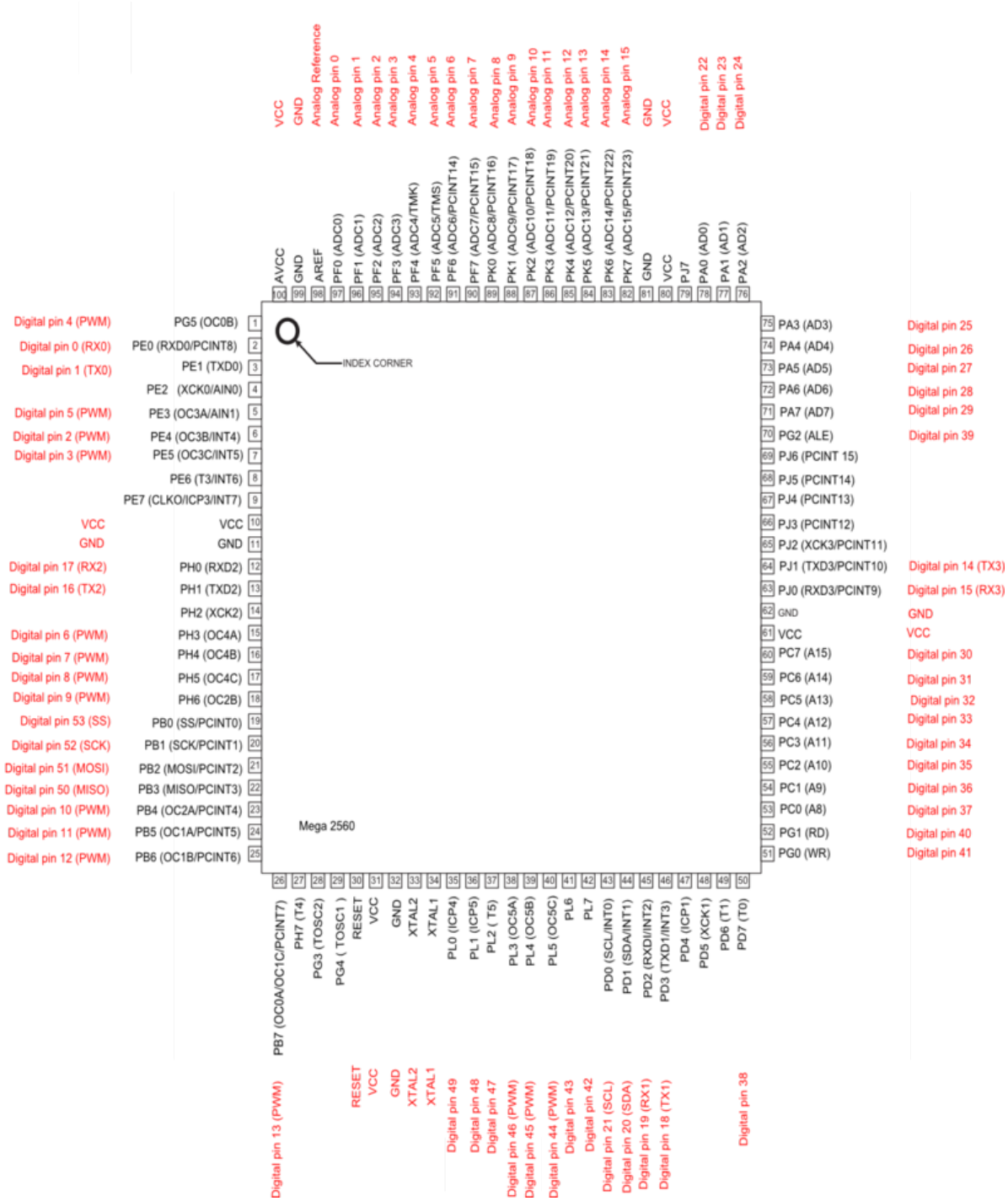

Figura 3: Diagrama de pinos do Atmega2560

PORTE onde o pino 2 representa o bit 4 (**PE4**) e o pino 3 o bit 5(**PE5**). E os pinos 8 e 9 digital ao **PORTH**, onde o pino 8 representa o bit 5(**PH5**) e o pino 9 o bit 6(**PH6**). Verificado os PORTs, foi feito a configuração dos bits.

Os pinos 8 e 9 foram configurados como saída e iniciados com valor baixo(LOW). E os pinos 2 e 3 foram configurados como entrada.

```
DDRH |= (1 << DDH5); //pino 8 como saida
PORTH &= ~(1 << PH5); //pino 8 definido como LOW
DDRH |= (1 << DDH6); //pino 9 como saida
PORTH &= ~(1 << PH6); //pino 9 definido como LOW
DDRE &= ~(1 << DDE4); //pino 2 como entrada
DDRE &= ~(1 << DDE5); //pino 3 como entrada
```

Depois foi feito o teste para verificar quando o botão é apertado

```
//Loop
while (1) {
  if(PINE & (1 << PINE4)){ //le pino 2
    PORTH |= 1 << PH5; //Define pino 8 como HIGH, liga o led
  }else{
    PORTH &= ~(1 << PH5); //Define pino 8 como LOW, desliga o led
  }

  if(PINE & (1 << PINE5)){ //le pino 3
    PORTH |= 1 << PH6; //Define pino 9 como HIGH, liga o led
  }else{
    PORTH &= ~(1 << PH6); //Define pino 9 como LOW, desliga o led
  }
}
```

2.2 Lab 02

Nessse experimento se utilizou os pinos 0 e 1 digitais que representam respectivamente pino de transmissão(Tx0) e recepeção(Rx0) no AVR. Para a utilização da comunicação serial se configurou os seguites registradores:

UBRR0H e UBRR0L - USART Baud Rate Registers: Estes dois registradores possue 8 bits cada um como mostrado na figura 4 e foram usados pra definir a taxa de transmisão de 9600 bps. Para isso primeiro foi necessario fazer um calculo usando como parametro o valor da taxa de transmissão e a frequencia que trabalha o AVR, como mostrado na figura 5. Depois o resultado desse calculo é representado em binario e passado para os registradores para setar o bits correspondentes que vão configurar a taxa de transmissão para 9600 bps.

UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRHn
	UBRR[7:0]								UBRRLn
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Figura 4: Registradores UBRRnL e UBRRnH

UCSR0C - USART MSPIM Control and Status Register n C: Este registrador possue 8 bits como mostrado na figura 6. Desses 8 bits se configurou os bits:

Table 22-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD Baud rate (in bits per second, bps).

f_{osc} System Oscillator clock frequency.

UBRRn Contents of the UBRRHn and UBRRLn Registers, (0-4095).

Figura 5: Calculo Baud Rate e UBRRn

- **UMSEL01 e UMSELL00** em 00 para definir a comunicação assincrona, como mostrado na figura 7
- **UPM01 e UPM00** em 00 para definir o formato sem paridade, como mostrado na figura 8.
- **USBS0** em 0 para definir 1 stop bit, como mostrado na figura 9.
- **UCSZ00, UCSZ01 e UCSZ02** em 011 para definir o tamanho do caracter em 8 bits, como mostrado na figura 10. O bit **UCSZ02** é acessado pelo registrador **UCSR0B**.

UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

Figura 6: Bits registrador UCSR0C

UCSR0B - USART Control and Status Register n B: Este registrador possue 8 bits como mostrado na figura 11. Desses 8 bits se configurou os bits:

- **RXEN0 e TXEN0** em 11 para ativar o pino transmissor 0 (Tx0) e o receptor 0 (Rx0).

Table 22-4. UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)[1]

Figura 7: Configuração de bits para definir modo

Table 22-5. UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Figura 8: Configuração de bits para definir moo de paridade

Table 22-6. USBS Bit Settings

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

Figura 9: Configuração de bits para definir Stop Bit(s)

Table 22-7. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

Figura 10: Configuração de bits para definir tamanho da caracter

UCSR0A - USART Control and Status Register A: Este registrador possue 8 bits como mostrado na figura 12. Desses 8 bits se configurou os bits:

- **UDRE0** para verificar se ainda havia bytes no buffer antes de enviar a mensagem. Neste caso quando o bit estivesse em 1 o buffer esta vazio e entao a mensagem é colocada em **UDR0** e enviada
- **RXC0** para verificar se há bytes sendo recebido no buffer antes de receber

UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 11: Registrador UCSRnB

UCSRnA – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Figura 12: Registrador UCSRnA

completamente a mensagem em **UDR0**. Para saber se o buffer esta vazio o bit deve estar em 1.

UDR0 - USART I/O Data Register 0: Registrador de 8 bits usado pra receber e enviar as mensagens na comunicação serial, como mostrado na figura 13.

UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 13: Registrador UDR0

2.3 Lab 04

Para esse experimento se deve usar a comunicação serial e a conversão de analogico para digital(ADC). Para a comunicação serial se utilizou os registradores descritos no Lab 02. Para o ADC os registradores usados foram os listados a seguir.

ADMUX - ADC Multiplexer Selection Register: Este registrador possui 8 bits como mostra a figura 14. Desses 8 bits se configurou os bits:

- **REFS1 e REFS0** em 0 e 1 respectivamente para configurar a fonte de tensão de referência para o A/D em 5 v, conforme mostrado na figura 15.
- **MUX4 a MUX0** em 00000 para selecionar o pino ADC0(pino A0 da placa), como mostrado na figura 16. Na figura 16 percebe-se que a seleção de ADC0 é de MUX5 a MUX0, para configura o MUX5 deve-se acessar o registrador ADCSRB, como mostrado a figura 17.

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 14: Bits registrador ADMUX

Table 26-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection[1]
0	0	AREF, Internal V_{REF} turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Internal 1.1V Voltage Reference with external capacitor at AREF pin
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Figura 15: Bits REFS1 e REFS0 para configuração de voltagem no pino ADC

Table 26-4. Input Channel Selections

MUX5:0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
000000	ADC0	N/A		
000001	ADC1			
000010	ADC2			
000011	ADC3			
000100	ADC4			
000101	ADC5			
000110	ADC6			
000111	ADC7			

Figura 16: Bits para selecionar canal de entrada

ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	MUX5	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 17: Bits ADCSRB

ADCSRA - ADC Control and Status Register A: Este registrador possui 8 bits como mostra a figura 18. Desses 8 bits se configurou os bits:

- **ADPS2, ADPS1 e ADPS0** em 111, para definir Prescaler em 128, como mostrado na figura 19. O Prescaler configura o fator de divisão entre o clock do sistema e a entrada de clock do ADC.
- **ADEN** em 1 para habilitar o conversor A/D.
- **ADSC** em 1 para iniciar a conversão do ADC. Depois que a conversão foi

iniciada, para saber quando ela terminou, deve-se verificar o bit ADSC e ver se seu valor esta em 0. Com a conversão terminada, o valor convertido vai estar nos regitradores ADCL e ADCH.

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 18: Bits ADCSRA

Table 26-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Figura 19: ADCSRA - Bits configuração Prescaler

ADCL e ADCH - The ADC Data Register: Estes dois registradores juntos possui 10 bits ADCH(2 bits) e ADCL(8 bits). Quando uma conversão ADC está completa, o valor convertido é colocado nesses dois registradores, e para acessar o valor usa-se o nome chave ADC.

2.4 Lab 07

Para usar a interrupção no pino escolhido, primeiro se verificoiu no diagrama de pino do datasheet(figura 3) qual o vetor associado a ele. Nesse experimento se escolheu o pino 3 como entrada para o botão, que no diagrama apresenta o vetor INT5 associado a ele. Com o vetor determinado se configurou os seguintes registradores relacionado a interrupções:

EICRB - External Interrupt Control Register B: Este registrador tem 8 bits como mostrado na figura 20 e se configurou os bits:

- **ISC50 e ISC51** em 00 para definir que a interrupçao em INT5 ocorra no nivel baixo(LOW) como mostrado na figura 21 .

EIMSK - External Interrupt Mask Register: Este registrador tem 8 bits como mostrado na figura 22 e nele se configurou o bit:

- **INT5** em 1 para habilitar a interrupção no pino 3 digital.

Bit	7	6	5	4	3	2	1	0	
(0x6A)	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 20: Registrador EICRB

Table 15-3. Interrupt Sense Control(1)

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request
0	1	Any logical change on INTn generates an interrupt request
1	0	The falling edge between two samples of INTn generates an interrupt request
1	1	The rising edge between two samples of INTn generates an interrupt request

Figura 21: EICRB - Controle de sensibilidade da interrupção

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 22: Registrador EIMSK

2.5 Lab 08

Nesse experimento se repetiu a mesma configuração de interrupção descrita no Lab 07.