Inatel Instituto Nacional de Telecomunicações	**RELATÓRIO 4 – Conversor AD**	**Data: / /**
	Disciplina: E209	
	Prof: João Pedro Magalhães de Paula Paiva	
Conteúdo: Microcontroladores AVR		
Tema: Conversor AD		
Nome:	**Matrícula:**	**Curso:**

OBJETIVOS:

- Utilizar ferramentas de simulação para desenvolver programas para o ATmega328p.
- Desenvolver programas que utilizam os conceitos de conversão analógico digital para leitura de sensores
- Utilizar as entradas e saídas do ATmega328p com circuitos de aplicação.

Parte Teórica

Conversor AD

O conversor A/D realiza o processo de amostragem, quantização e codificação de uma tensão aplicada a um pino de entrada analógica do MCU. Quando uma conversão é iniciada, o valor instantâneo da tensão no pino é retido pelo bloco conversor. Esse valor de tensão é associado a uma palavra binária através do método de aproximação sucessiva (SAR). No caso do ATMega328p, a palavra binária tem 10 bits, sendo armazenada uma parte no registro **ADCL**, e outra no **ADCH**. Dessa forma, a tensão de entrada é convertida utilizando as tensões de referência em valores digitais de 0 a 1023.

Para controlar o processo de conversão A/D, os registros **ADMUX** e **ADCSRA** (em anexo) devem ser configurados para que o canal A/D do Atmega328p opere corretamente. O Atmega328p permite que sejam utilizadas diversas formas de conversão em função da aplicação. Nós utilizaremos a função mais simples que é a conversão única. Nesse método uma conversão é disparada quando o bit **ADEN** (ADC ENABLE) e o **ADSC** (ADC Start Conversion) são setados (ligados). Quando a conversão chega ao fim, o **ADSC** vai para 0, indicando que a leitura do valor convertido pode ser realizada.

Os pinos utilizados como entradas analógicas são aqueles identificados de ADC0 a ADC5 (PC0 à PC5).

Leitura e conversão das leituras ADC

Como a conversão é em 10 bits, a tensão de entrada é convertida seguindo a expressão:

- **Palavra_Digital (ADCH + ADCL) = (Vin * 1023) / $Vref$**
- **Vin = (Palavra_Digital * $Vref$) / 1023**

A relação $Vref$ / 1023 é conhecida como resolução do conversor AD, ou seja, a menor variação da tensão de entrada que provoca a alteração de 1 bit na palavra digital de saída.

Vamos considerar a tensão de alimentação VDD = 5V. Sabendo que a resolução do Atmega328 é de 5V/1023 = 4,88mV. Assim, caso o conversor retorne um valor digital lido de 430, a tensão existente na entrada analógica do microcontrolador seria de 2,0984 V.

A linha de programa que é capaz de calcular a tensão aplicada na entrada analógica do microcontrolador seria (considere que a variável que armazena o valor digital é valorLido):

- **tensao = (valorLido * 5000) / 1023;**

Aqui vale ressaltar que se a linha tiver escrita exatamente como acima, pode-se gerar um resultado incorreto devido a limitação de armazenamento da variável "**tensão**". Vejamos, se **tensao** for do tipo **unsigned int**, ela não poderá armazenar o resultado de **valorLido*5000**, caso valorLido tenha seu valor máximo de **1023**, uma vez que **1023*5000=5115000**, o que ultrapassa o limite de armazenamento de uma variável do tipo **unsigned int**, que é **65535**.

É preferível fazer o seguinte:

```
unsigned long int aux;
unsigned int valorLido;
unsigned int tensao;

aux = valorLido * 5000;       // Variável auxiliar para o cálculo
aux = aux / 1023;             // Divide-se por 1023
tensao = (unsigned int) aux; // Retorna o resultado do tipo unsigned int
```

A última linha utiliza o recurso denominado ***casting*** no qual um tipo de variável é convertido para o outro. No caso, de **unsigned long int** para **unsigned int**.

Configuração do bloco ADC

Para fazermos uso do conversor AD, devemos configurá-lo. Primeiramente vamos configurar a tensão de referência ($Vref$), o prescaler utilizado, e habilitar o ADC. Para definir o $Vref$ devemos setar os BITs escolhidos no registro ADMUX, por exemplo, configurando $Vref$ = 5V:

```
ADMUX = BIT6; // 0b01000000
```

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, internal V_{REF} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V voltage reference with external capacitor at AREF pin

Figura 1. Bits de configuração da tensão de referência

Para configurar o prescaler se faz uso do registro **ADCSRA**: (analisar User Guide (em anexo) para escolha).

```
ADCSRA |= (BIT2 + BIT1 + BIT0); // divisor 128
```

Table 23-5. 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 2. Bits de configuração do Prescaler

Por fim habilitamos o ADC:

```
ADCSRA |= BIT7; // habilita o ADC
```

Lembre-se, as tensões podem ser modificadas ao longo do programa de acordo com o necessário. Após as configurações principais devemos iniciar a parte de conversão do nosso AD. Para isso faremos uso dos registros **ADCSRA** e **ADMUX**.

Vamos informar qual a entrada queremos converter:

```
ADMUX = (ADMUX & 0xF8) | input; // seleciona o canal de entrada (0 a 5)
```

Agora devemos começar a conversão, para isso é necessário setar o bit **ADSC**.

```
ADCSRA |= BIT6; // inicia a conversão
```

Considerações importantes

Um detalhe importante é que **só podemos fazer a conversão de 1 canal por vez**, ou seja, se tivermos 2 canais, devemos esperar um pouco para iniciar a conversão do outro.

Agora considere a seguinte ocasião. Você irá fazer uma conversão simples de um sensor que estava com tensão de 1,2V e no momento de converter um ruído 5 volts entra no seu sistema por um breve período de tempo. Esse ruído irá gerar uma resposta no ADC result (ADCL + ADCH) de 1023 (5V) e não 245 (1,2V) que seria o correto. Para evitarmos esse tipo de erro devemos criar uma amostragem de, por exemplo, 100

leituras e tirar a média de todas elas, isso irá garantir que o valor lido está aproximado ao que realmente deveria ter sido lido.

Para uma boa conversão, podemos realizar a seguinte operação:

```
// Espera o ADC terminar a conversão e salva
for (i = 0; i < NUMERO_DE_AMOSTRAS; i++) {
    ADCSRA |= BIT6; // inicia a conversão
    while ((ADCSRA & BIT6) == BIT6); // Espera o fim da conversão
    var_temp = ADCL; // Leitura do ADCL
    var_temp += (ADCH << 8); // Leitura do ADCH
    var_armazenagem += var_temp; // Soma dos valores lidos
}
var_media = var_armazenagem / NUMERO_DE_AMOSTRAS; // Calcula a média
```

O código acima espera a conversão AD terminar (linha **WHILE**). Internamente o **ADSC** é resetado quando uma conversão foi finalizada. Também é possível tratar a leitura utilizando interrupções, da seguinte maneira:

```
ADCSRA |= (BIT6 + BIT3); // inicia a conversão e habilita interrupção


ISR(ADC_vect){
    Value = (ADCL | (ADCH << 8)); // ou Value = ADC;
}
```

Parte prática:

1. (Fácil) Elaborar um firmware para ler a tensão recebida de uma fonte (0 a 5V) no pino PC4 e apresentar o valor lido na Serial.

2. (Fácil) Elaborar um firmware para ler e apresentar na Serial um valor de temperatura que varia de 0 à 200°C de acordo com um transmissor de tensão padronizado de 0 à 5V colocado no PC2.

3. (Médio) Elaborar um firmware para ler e apresentar na Serial a temperatura de um PT100 (sensor de temperatura) operando em escala cheia transmitido por um transmissor de corrente padronizado de 4 à 20 mA, cuja corrente de saída alimenta um resistor de 250 Ohms colocado em paralelo ao pino PC1.

4. (Médio) Elaborar um firmware para apresentar na Serial a média das tensões lidas nos pinos PC0 a PC5 (0 a 5V).

5. (Difícil) Repita o exercício anterior, porém acrescentando a leitura de 50 amostras de cada pino. Você deve apresentar na tensão a média da leitura de cada pino, além da média global.

6. (Difícil - Extra) Em um sistema de controle de temperatura, a medida é feita utilizando um sensor PT100, que varia sua resistência em função da temperatura. O cartão de entrada analógica do CLP não está preparado para realizar a leitura direta do PT100, sendo necessário utilizar um transmissor de temperatura.

    a. Ajustar o transmissor de temperatura para a faixa de trabalho de 0 a 100 °C
    b. Elaborar um programa em Texto estruturado para controlar a frequência o acendimento de uma lâmpada, como apresentado abaixo:
        i. Para temperaturas menores que 30°C a lâmpada deve piscar com frequência de 2 Hz
        ii. Para temperaturas entre 30°C e 45°C, a lâmpada deve piscar com frequência de 0.5 Hz.
        iii. Para temperaturas maiores que 45°C, a lâmpada deve piscar com frequência de 1 Hz.