Sistemas Embebidos

Trabalho Prático 6

Carlos Abreu[1] e João Faria[2]

[1]**cabreu@estg.ipvc.pt**

[2]**joao.pedro.faria@estg.ipvc.pt**

Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Viana do Castelo

Instituto Politécnico de Viana do Castelo
Escola Superior de Tecnologia e Gestão
2024

Carlos Abreu
www.estg.ipvc.pt/~cabreu

Curso:

Licenciatura em Engenharia de Redes e Sistemas de Computadores

Objetivo Pedagógico

ADCs - Analogue-to-Digital Converters (Conversão analógico-digital).

Sumário:

Duração: 3 horas

1. Entender o modo de funcionamento de um módulo A/D que utiliza o método das aproximações sucessivas para conversão analógico-digital.
2. Configurar o módulo A/D do MCU ATmega328 para aquisição de sinais analógicos.
3. Configurar e utilizar a interrupção interna do Timer 1 para efetuar amostragem periódica.

1. *ADCs* - Introdução

Os módulos A/D (também frequentemente chamados ADCs - Analogue-to-Digital Converters) são usados para converter uma tensão analógica (que varia continuamente dentro de um intervalo conhecido) num valor digital. São habitualmente usados para adquirir de forma periódica a tensão de saída de sensores (temperatura, pressão, etc) permitindo por exemplo a implementação de sistemas de controlo.

A conversão de analógico para digital necessita de uma tensão de referencia com valor conhecido e a tensão analógica na entrada (Vin) deve situar-se entre a tensão de referência negativa ($Vref_-$) e a tensão de referência positiva ($Vref_+$). As tensões de referência podem ser obtidas interna ou externamente, sendo frequente utilizar $Vref_- = 0\,V$ e $Vref_+ = Vref$.

Sendo $nbits$ o número de bits com que o valor digital é apresentado, o código digital obtido após conversão pode ser obtido através da fórmula:

$$ValorDigital = round((Vin \div Vref) \times (2^{nbits} - 1)).$$

Na figura 1 está representado o diagrama simplificado de operação do módulo A/D no ATmega328. A entrada do módulo A/D é representada por um *multiplexer* analógico usado para selecionar uma entre oito entradas analógicas. Isso significa que é possível converter até oito sinais analógicos (não em simultâneo, mas em instantes diferentes). Após a conversão, o valor digital é transferido para os registos

ADCH e ADCL.

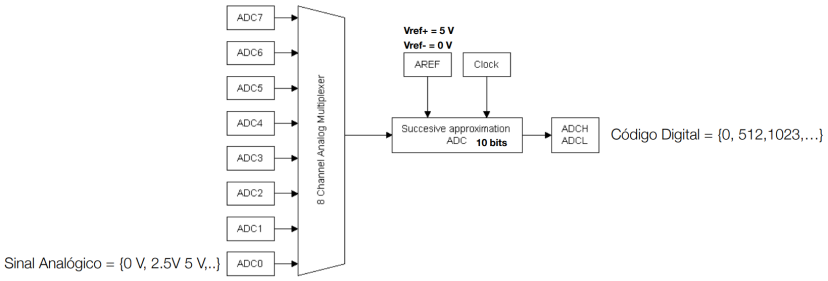

Figura 1

Diagrama simplificado de operação do módulo A/D no ATmega328

O modo de operação da *ADC* do Atmega328 encontra-se descrito na secção 24 (*Analog-to-Digital Converter*) do *datasheet* que se encontra no moodle. Abra o datasheet e procure a resposta ás seguintes questões:

- Qual o tipo de ADC existente no MCU ?
- O resultado da conversão para digital é apresentado com quantos bits?
- Qual a função do registo ADMUX ?
- Qual a função do registo ADCSRA ?
- Qual a função dos registos ADCH/ADCL ? Porque são necessários 2 registos ?

De acordo com a formula apresentada anteriormente e com a secção 24.7 (ADC Conversion Result) do datasheet disponível no Moodle, considerando $Vref = 5V$:

- $ValorDigital = round((Vin \div 5) \times 1023)$
  - se Vin for igual a 0 V, o resultado da conversão será o código digital 0.
  - se Vin for igual a 2.5 V o resultado da conversão será o código digital 512.
  - se Vin for igual a 5 V, o resultado da conversão será o código digital 1023.

Esta conversão de analógico para digital produz um erro conhecido como *"erro de quantização"*. Quanto maior o número de bit da ADC menor será o *"erro de quantização"*.

2. *ADCs* - Exercícios

Exercício 1 *Pretende-se implementar um sistema para amostrar um sinal analógico convertendo uma tensão de entrada Vin (entre $0\ V$ e $5\ V$) para um código digital. A variação da tensão pode ser obtida com recurso a um divisor de tensão implementado*

Tensão de entrada	Código Cor
$0 \leq Vin < 1$	0 - LED_RGB(255, 0, 0)
$1 \leq Vin < 2$	1 - LED_RGB(255, 127, 0)
$2 \leq Vin < 3$	2 - LED_RGB(255, 255, 0)
$3 \leq Vin < 4$	3 - LED_RGB(0, 255, 0)
$4 \leq Vin \leq 5$	4 - LED_RGB(0, 0, 255)

Tabela 1 Exercício 1 - representação cromática da tensão de entrada.

através de um potenciómetro. Adicionalmente pretende-se utilizar um LED RGB a funcionar como dispositivo de saída. Do ponto de vista funcional o programa apresenta os seguintes requisitos de aplicação:

- Na etapa de inicialização deverá configurar o módulo A/D para amostrar a entrada analógica $AN0$ do módulo A/D do ATmega328 com tensões de referência $Vref_+ = 5\ V$ e $Vref_- = 0\ V$.
- A amostragem de Vin deverá ser efectuada na função loop() via pooling com um período de $100\ ms$;
- O led RGB deverá alterar a sua cor mediante o valor de Vin medido, tendo em conta a informação na tabela 1.
- Sempre que amostrar a tensão de entrada Vin, deverá imprimir na consola o respetivo valor em Volt, numa escala de $0.0\ V$ a $5.0\ V$, com resolução à décima de Volt;

1.1 Desenhe o fluxograma de operação genérico da aplicação que inclua device drivers para inicialização do módulo A/D, inicialização do LED RGB, controlo do processo amostragem e controlo do LED RGB. Considere os seguintes protótipos das funções:

```
void ADC_init(void) { // Inicialização da ADC
    pinMode(POT, INPUT);          // Analog pin: INPUT
    bitWrite(ADCSRA,ADEN, _ );    // ADC Enable Bit
    bitWrite(ADCSRA,ADIF, _ );    // ADC Interrupt Flag
    bitWrite(ADCSRA,ADIE, _ );    // ADC Interrupt Enable
    bitWrite(ADMUX,REFS1,0);      //
    bitWrite(ADMUX,REFS0,1);      // AVCC with external capacitor at AREF pin
    bitWrite(ADMUX,ADLAR,0);      // Result Justified
}

void RGB_Led_init(void) { // Inicialização do Led RGB
    ...
}
```

```
float ADC_read_Volt(void) { // Leitura do resultado da conversão A/D
    bitWrite(ADMUX,MUX3, _ );  //--------
    bitWrite(ADMUX,MUX2, _ ;   // Select
    bitWrite(ADMUX,MUX1, _ );  //  ADC 0
    bitWrite(ADMUX,MUX0, _ );  //--------

    bitWrite(ADCSRA,ADSC, _ );  // Start Conversion
    while (bitRead(______,____) == _ );// Wait for end of convertion

    // Compute Digital Code
    int digital_code =  ADCH<<8;
    digital_code = digital_code + ADCL;

    // Convert Digital Code to Volt
    float vin = digital_code * 5.0f / 1023;
    return vin;
}

void SET_RGB_COLOR(unsigned int color) { // Controo do Led RGB
    ...
}
```

1.2 *Importe para a sua área de trabalho o circuito disponibilizado neste link e implemente em Arduino C a solução apresentada no ponto anterior.*

1.3 *Teste o código implementado e não se esqueça de o comentar convenientemente.*

Exercício 2 *A partir da implementação do exercício anterior altere o código de forma a efetuar a amostragem do sinal analógico recorrendo agora a um evento periódico gerado a partir do Timer 1 que deverá ser configurado para gerar uma interrupção a cada* $100\ ms$*. Considere que a atuação no LED RGB é feita em* background *e que, do ponto de vista funcional, o LED RGB deverá piscar com a cor e período (*T_{on}*/*T_{off}*) definidos na tabela 2:*

2.1 *Desenhe o fluxograma de operação genérico da aplicação análogo ao do exercício anterior mas que inclua o paradigma de operação orientado a eventos no processo de amostragem do sinal analógico. Na rotina de atendimento à interrupção do timer 1 deve ser despoletada a conversão analógico-digital e na rotina de atendimento à interrupção de conversão analógico-digital terminada deve ser processado o resultado da conversão.*

Tensão de entrada	Código Cor	Periodo
$0 \leq Vin < 1$	0 - LED_RGB(255, 0, 0)	1.0 seg
$1 \leq Vin < 2$	1 - LED_RGB(255, 127, 0)	0.8 seg
$2 \leq Vin < 3$	2 - LED_RGB(255, 255, 0)	0.6 seg
$3 \leq Vin < 4$	3 - LED_RGB(0, 255, 0)	0.4 seg
$4 \leq Vin \leq 5$	4 - LED_RGB(0, 0, 255)	0.2 seg

Tabela 2 Exercício 2 - Comportamento do LED em função da tensão de entrada.

2.2 *Importe para a sua área de trabalho o circuito disponibilizado neste link e implemente em Arduino C a solução apresentada no ponto anterior. Considere o seguinte protótipo da rotina de atendimento à interrupção de conversão analógico-digital terminada:*

```
// ADC INTERRUPT
ISR(ADC_vect) {// new ADC value is ready
    ...
}
```

2.3 *Teste o código implementado e não se esqueça de o comentar convenientemente.*