Laboratório de Microprocessadores e Microcontroladores

4º Experimento - Filtro Digital

Luiz Oliveira, Felipe Carvalhedo
10/46969, 09/0037791

Resumo— Desenvolver um programa que simule um filtro. O filtro é representado pela média das quatro ultimas amostras e assemelha-se a um filtro passa baixa. O resultado do filtro deverá ser expresso em uma barra de LED's, em displays de 7 segmentos ou em um LCD gráfico.

Index Terms—MSP430, Microprocessadores, Microcontroladores, Filtro Digital

1 Procedimentos Experimentais

1.1 Objetivos

Este experimento tem como objetivo a apresentação de um sinal de onda desconhecido. A apresentação poderá ser efetuada através de uma barra de LED's, displays de 7 segmentos ou com um LCD gráfico. É importante que a apresentação do sinal poderá ser feita de duas formas distintas. A primeira é o sinal exatamente da mesma forma como ele é colhido pelo MSP, a outra opção é utilizando um filtro, que pode ser representado pela equação 1.

1.1.1 Material

- MSP430 LaunchPad
- Code Composer v4 ou MSPGCC
- MSP430G2553
- – 10x LED', ou
  – 4 Displays de 7 segmentos, ou
  – LCD Gráfico
- Protoboard

1.2 Introdução Teórica

Para a realização deste experimento é importante a compreensão de um conversor AD (ADC). O que o conversor analógico/digital faz é capturar amostras do sinal analógico ao longo do tempo. Cada amostra será convertida em um número, levando em consideração seu nível de tensão.

1.2.1 Taxa de Amostragem

A frequência com que a amostragem irá ocorrer é chamada de taxa de amostragem. Se uma taxa de amostragem de $22.050Hz$ for usada, por exemplo, isto significa que em um segundo 22.050 pontos serão capturados (ou sampleados). A distância de cada ponto capturado será de $\frac{1}{22.050}$ segundo (45,35μs, neste caso). Se a taxa de amostragem for de $44.100Hz$, isto significa que 44.100 pontos serão capturados por segundo. Neste caso a distância de cada ponto será de $\frac{1}{44.100}$ segundo ou 22,675μs, e assim por diante.[1]

1.2.2 Quantização

Os valores instantâneos da tensão do sinal de entrada, que são obtidos na saída do circuito de amostragem e retenção precisam ser convertidos para a forma digital. Este processo recebe o nome de "quantização". Os DSPs (Processadores Digitais de Sinais) processam os sinais analógicos convertidos para a forma digital e fazem uso deste processo. O que um DSP pode fazer com o sinal vai depender justamente da precisão com que a quantização é feita. A representação dos valores instantâneos amostrados pelos circuitos anteriores depende do nível de quantização realizado, ou seja, quantos bits são

Reviewed in 13 de fevereiro de 2012.

usados para representar cada valor amostrado. Assim, se usamos 2 bits teremos uma precisão menor do que se usarmos 4 bits para fazer a quantização.

1.2.3 Resolução

A resolução da conversão indica o número discreto de valores que podem ser produzidos sobre o intervalo de valores analógicos. Os valores são normalmente armazenados eletronicamente de forma binaria, assim a resolução é usualmente expressa em bits. Em consequência, o número de valores discretos disponíveis - níveis - são dados em potência de dois.

A resolução pode também ser definida eletricamente, e expressada em volts. A tensão mínima reconhecida pelo circuito é chamada de bit menos significativo - LSB (*least significant bit*). A resolução (**Q**) do ADC é igual ao LSB. A voltagem de resolução é dada pela amplitude da tensão dividida pelo número de intervalos de voltagem discretas:

$$Q = \frac{E_{FSR}}{N}$$

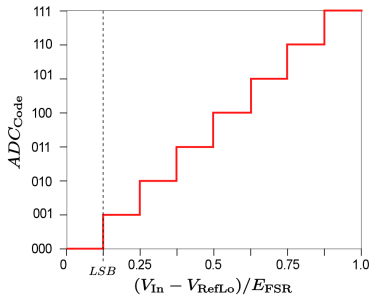

Figura 1: Esquema de codificação ADC de 3 bits. Imagem de Wikipedia[2]

Para o experimento, será aplicado um filtro que tem como função cortar as frequências mais altas, mostrando como saída um valor próximo da mediana. Ele pode ser representado pela equação 1, que é a média das quatro ultimas amostras:

$$Y = \frac{X_0 + X_1 + X_2 + X_3}{4} \quad (1)$$

2 Descrição do Hardware & Software

2.1 Descrição do Hardware

Pelo grupo ter escolhido apresentar o experimento utilizando apenas o LCD gráfico, modelo LS020, o único circuito eletrônico acrescentado ao LaunchPad foi o LCD e uma bateria de 9V para alimentação do backlight do display, como pode ser observado na figura 2.

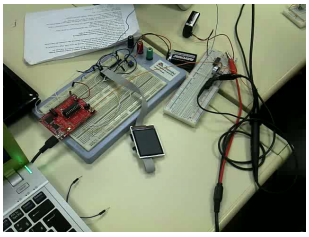

Figura 2: Vista da protoboard na apresentação final.

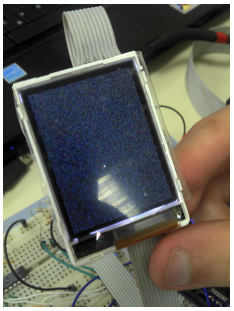

Figura 3: Lcd Inicializado.

2.2 Descrição do Software

Este laboratório teve sem duvidas o código principal mais simples de todos, se desconsiderar todo o código para o funcionamento do LCD. Abaixo encontra-se o arquivo *main.c*, contendo a rotina principal e a interrupção do ADC. Das linhas 11 à 14 o ADC é configurado, habilitado, porém a linha 13 garante que nenhuma amostra será colhida. As linhas 17 e 18 chamas as rotinas para inicializar o LCD (semelhante à figura 3) e preenche-lo com a cor branca. Em seguida o LED no bit 6 é aceso, representado que o chip esta pronto para inicializar a conversão.

```
1  /*****************************************************************************
2  *                                  main.c                                   *
3  *****************************************************************************/
4  #include "disp.h"
5  #include "ploc.h"
6
7  int main (int argc, char *argv[])
8  {
9      WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
10     ADC10CTL1 = CONSEQ_2 + INCH_1;            // Repeat single channel, A1
11     ADC10CTL0 = ADC10SHT_2 + MSC + ADC10ON + ADC10IE; // ADC10ON, interrupt enable
12     ADC10DTC1 = 0;                            // no reads yet
13     ADC10AE0 |= 0x02;                         // P1.1 ADC option select
14
15     P1DIR |= BIT6;                            // Set P1.0 to output direction
16     lcd_init();
17     lcd_clear(0xffff);                        // fill with white
18     P1OUT |= BIT6;                            // ready
19     while(1)
20     {
21         ploc(ADC10MEM);                       // print adc value on display
22         ADC10DTC1 = 1;                        // one read
23         ADC10CTL0 &= ~ENC;
24         while (ADC10CTL1 & BUSY);             // Wait if ADC10 core is active
25         ADC10SA = 0x200;                      // Data buffer address
26         ADC10CTL0 |= ENC + ADC10SC;           // Sampling and conversion start
27         __bis_SR_register(CPUOFF + GIE);      // LPM0, ADC10_ISR will force exit
28     }
29     return 0;
30 }
31
32 // ADC10 interrupt service routine
33 interrupt(ADC10_VECTOR) ADC10_ISR (void)
34 {
35     P1OUT |= BIT6;                            // to show that the chip has waked
36     __bic_SR_register_on_exit(CPUOFF);        // Clear CPUOFF bit from 0(SR)
37     P1OUT &= ~BIT6;
38 }
```

No laço principal (*while(1)*), a função ploc() é chamada utilizando como parâmetro a variável *ADC10MEM*, que contém o valor atual do ADC. O registrador *ADC10DTC1* é atualizado com 1, para a próxima conversão. Desabilita-se então a interrupção do ADC e espera enquanto o conversor esta ocupado com a conversão. Por fim reconfigura-se o ADC e o coloca em baixo consumo, esperando a próxima colheita. Na interrupção o chip é acordado e é feito um pulso no LED do bit 6. O código teve como base os exemplos disponibilizados pela Texas Instruments[3].

```
1  /*****************************************************************************
2  *                                  ploc.c                                   *
3  *****************************************************************************/
4  #define COR     0
5  #define BACK    0xffff
6  #include "disp.h"
```

```
7
8  int ploc(unsigned int value)
9  {
10     static int x=0;
11     static int x0,x1,x2,x3,antiga;
12
13     x3=x2;
14     x2=x1;
15     x1=x0;
16     x0=value/10;                    //to not overflow the display space
17
18 #ifdef __GRAFICO_BARRAS__
19         lcd_retangulo(BACK,x,0,0,DISP_W);       // clean llast value here
20         lcd_retangulo(COR,x++,0,0,(x0+x1+x2+x3)/4); // put new value and inc
21     //  lcd_retangulo(COR,x++,0,0,(x0+x1+x2+x3)/4); // put the value (without filter)
22 #else
23         lcd_retangulo(BACK,x,0,0,DISP_W);       // clean llast value here
24         lcd_set_pixel(COR,x++,(x0+x1+x2+x3)/4); // with filter
25     //  lcd_set_pixel(COR,x++,x0);              // without filter
26 #endif
27
28     antiga = (x0+x1+x2+x3)/4;
29     if(x==DISP_H)
30     //  end of the display
31     {
32         x=0;
33     }
34
35     return 0;
36 }
```

A função ploc é bem simples e apenas plota o valor passado para ela no LCD. Nela é feita a aplicação do filtro, retratado pela equação 1.

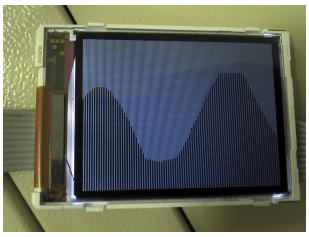

Figura 4: Saída em forma de barras

Para um maior controle do código, é possivel alterar a saída via código, definindo ou não a macro *__GRAFICO_BARRAS__* para alterar a saída do display de linhas (figura 5) ou barras (figura 4).

Figura 5: Saída em forma de linhas

3 Descrição dos resultados

O grupo conseguiu com este experimento uma grande carga de conhecimento. Não em decorrer da conversão analógico-digital, mas por que fomos responsáveis por levantar a biblioteca do LCD que foi usado especificamente para este experimento. Como observado na função principal, o programa é bem simples, porém o peso da biblioteca do LCD é visível no arquivo final, visto que o executável grava 7378 bytes no MSP, o que é um tamanho bem elevado para apenas uma conversão analógica-digital.

Outro ponto de grande aprendizado com este experimento foi a reação de filtros e como implementa-los em microcontroladores, visto que foi aplicado no experimento um filtro que tem um comportamento próximo à um filtro passa baixa.

Referências

[1] G. Torres, *Clube do Hardware*.
[2] Wikipedia, *Analog-to-digital converter*.
[3] T. Instruments, *Texas Instruments-MSP430G2x53, MSP430G2x33, MSP430G2x13, MSP430G2x03 Code Examples (Rev. A)*.