Experimento 4 - Microcontroladores e Microprocessadores

André Mateus R. Dantas
Matrícula: 11/0008359
Universidade de Brasília - Faculdade Gama
ymateus@hotmail.com

Marcella Jeronimo Ferreira Nunes
Matrícula: 11/0016611
Universidade de Brasília - Faculdade Gama
marcellajfn@outlook.com

1. Objetivos

Utilizar as funções de interrupção e de baixo consumo,bem como misturar as duas linguagens de programação (Assembly e C) para desenvolver o experimento.

2. Introdução

O experimento consiste em projetar um dado digital que possua seis valores (1 a 6), de forma que cada valor possua a mesma probabilidade de ser sorteada, ou seja, um dado não viciado. O objetivo do experimento é fazer com que o microcontrolador entre em modo de baixo consumo 4, e saia quando o usuário pressionar o botão, quando isso occorrer algum dos seis LED's deverá acender e permanencer por cerca de um segundo e em seguida apagar.

3. Descrição de Software

Para esse experimento utilizou-se a linguagem Assembly dentro do código em linguagem C, primeiro uma nova biblioteca foi adicionada (legacy), como se pode ver no código em anexo. Na função main, o whatch dog timer foi desligado e em seguida algumas opções foram habilitadas, tais como:

- P1REN = BTN $\rightarrow$ habilita o resistor de pull up/down do botão;
- P1OUT = BTN $\rightarrow$ selecionado o modo pull up;
- P1IES = BTN $\rightarrow$ configurar a interrupção como borda de descida;
- P1IE = BTN $\rightarrow$ habilitar a interrupção por meio do botão;
- P1SEL = 0 $\rightarrow$ utilizar pinos para a porta P1;

Logo após se entra em modo de baixo consumo 4 (o mais econômico de todos) e seta-se os bit que indica a possibilidade de interrupções mascaráveis. Assim o MSP430 ficará aguarda (em modo de baixo consumo 4) que aconteça alguma interrupção, ao acontecer entra-se em um loop infinito contendo seis if's, em cada um destes seta-se um valor diferente para uma variável de controle i, neste ponto utiliza-se do código em Assembly para garantir que na passagem entre esses if's aconteça igualmente, a condição para a execução do if é que o botão estejá solto, assim ao entrar em um if será setado um valor para i e o loo infinito será quebrado.

Em seguida, é setado de forma a mostrar o determinado número associado a i no display que é logo em seguida apagado, a flag que indica a interrupção é desativada e o MSP volta ao estado de baixo consumo.

O código completo se encotra em anexos.

4. Descrição do Hardware

Os componentes utilizados foram:

- A LaunchPad MSP-EXP430G2553, versão de 20 pinos, 8kB de memoória flash, 512B RAM, 16 GPIO, 1x16-bit timer, Watch Dog Timer, BOR, 1xUSI(I2C/SPC/UART), conversor analógico/digital de 8ch 10-bit , comparador de 8ch;
- Protoboard;
- Display de sete segmentos de ânodo comum;
- 1 resistores de 1 kΩ de resistência.

5. Resultados e Discussões

A dupla optou por trocar os LED's e usar um display de 7 segmentos, assim ao invés de sair do modo de consumo e acender um dos LED's, um número seria sorteado e seria mostrado no display, após alguns ajustes, principalmente devido ao tipo do display (anodo comum), o experimento funcionou de acordo com o esperado. Usou-se um trecho de código em Assmbly para que a propabilidade de cada caso fosse igual, pois em C haveria mais instruções o que acarretaria em maiores chances de sair o número 1.

6. Conclusão

Dados de seis faces são utilizados em todo mundo em dezenas de jogos de tabuleiro diferentes, esta proposta de dado pode vir a substituir facílmente o dado tradicional, pois além da possibilidade de variar a quantidade de lados facilmente também tem um facíl manuseio, muito diferente do convencional, não cairá no tabuleiro, derrubando peças ou longe dos jogadores. A importância de verificar o fato do dado não ser viciado e totalmente plaúsivel pois para qualquer aplicão essa característica seria necessária, como vimos é impressindível que saibamos como ocorre o processo de alocão de memória do compilador no microcontrolador, o que nós mostra o quanto é importante conhecer como funciona a compilação de nossos códigos para que possamos impor características bastante específicas a nossos produtos.

Referências

[1] J. Davies. *MSP430 Microcontroller Basics*. Elsevier, Erewhon, NC, 2008.

[2] https://docs.google.com/document/d/12Qgy0671VeMx3Gg1WJlWWaB3jtXm2iYfeuqfxOgqwwA/edit?pli=1. Roteiro do experimento 4. *Acessado em : 23/05/2013*.

7. Anexos

Código do experimento em Linguagem C# padrão Ansi

```
#include <msp430g2553.h>
#include <legacymsp430.h>

#define APAGAR (BIT0|BIT1|BIT2|BIT4|BIT5|BIT6|BIT7)
#define BTN BIT3
#define BTN_DLY 6000
#define SEGUNDO 60000

int main(){
        volatile unsigned long int i;

        WDTCTL= WDTPW+WDTHOLD;
        P1OUT = P1REN  = BTN;
        P1DIR = (BIT0|BIT1|BIT2|BIT4|BIT5|BIT6|BIT7);
        P1OUT |= APAGAR;
        P1SEL2 = P1SEL  = 0;
        P1IE    = BTN;
        _BIS_SR(LPM4_bits + GIE);
}
interrupt(PORT1_VECTOR) P1_ISR(void)
{
        volatile unsigned char num[6]={0x49,0xee,0xeb,0xd9,0xbb,0xbf};
        volatile unsigned int i, j;
        __asm__(    ".1:  \n"
                "       .loc 1 34 0 \n"
                "       mov.b   &__P1IN, r15\n"
                "       and     #8, r15\n"
                "       jeq     .2\n"
                "       .loc 1 35 0\n"
                "       mov     #0, 2(r1)\n"
                "       .loc 1 36 0\n"
                "       jmp     .FIM\n"
                ".2:\n"
                "       .loc 1 38 0\n"
                "       mov.b   &__P1IN, r15\n"
                "       and     #8, r15\n"
                "       jeq     .3\n"
                "       .loc 1 39 0\n"
                "       mov     #1, 2(r1)\n"
                "       .loc 1 40 0\n"
                "       jmp     .FIM\n"
                ".3:\n"
                "       .loc 1 42 0\n"
                "       mov.b   &__P1IN, r15\n"
                "       and     #8, r15\n"
                "       jeq     .4\n"
                "       .loc 1 43 0\n"
                "       mov     #2, 2(r1)\n"
                "       .loc 1 44 0\n"
```

```
            "        jmp      .FIM\n"

            ".4:\n"
            "        .loc 1 46 0\n"
            "        mov.b    &__P1IN, r15\n"
            "        and      #8, r15\n"
            "        jeq      .5\n"
            "        .loc 1 47 0\n"
            "        mov      #3, 2(r1)\n"
            "        .loc 1 48 0\n"
            "        jmp      .FIM\n"
            ".5:\n"
            "        .loc 1 50 0\n"
            "        mov.b    &__P1IN, r15\n"
            "        and      #8, r15\n"
            "        jeq      .6\n"
            "        .loc 1 51 0\n"
            "        mov      #4, 2(r1)\n"
                     ".loc 1 52 0\n"
                     "jmp     .FIM\n"
            ".6:\n"
            "        .loc 1 54 0\n"
            "        mov.b    &__P1IN, r15\n"
            "        and      #8, r15\n"
            "        jeq      .1\n"
            "        .loc 1 55 0\n"
            "        mov      #5, 2(r1)\n"

            ".FIM:\n"
            "        .loc 1 59 0\n"
            "        mov      #0, @r1\n");


/*
        equivalente do Assembly em C
        for(;;){
                if((P1IN & BTN) != 0){
                        i=0;
                        break;
                }
                if((P1IN & BTN) != 0){
                        i=1;
                        break;
                }
                if((P1IN & BTN) != 0){
                        i=2;
                        break;
                }
                if((P1IN & BTN) != 0){
                        i=3;
                        break;
                }
                if((P1IN & BTN) != 0){
```

```
                        i=4;
                        break;
                }
                if((P1IN & BTN) != 0){
                        i=5;
                        break;
                }
        }
*/

        P1OUT = ~num[i] |BTN;
        for(j=0; j<SEGUNDO; j++);
        P1OUT = BTN|APAGAR;
        P1IFG  =0;
}
```