UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO

07/04/2016

EA871 - LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

EXPERIMENTO 4 – Representação e Armazenamento de Dados

Profª Wu Shin-Ting
Aluno: Tiago Eidi Hatta RA: 160388

Proposta do Experimento

Controlar a frequência de piscadas dos leds do kit FRDM-KL25Z, aplicando conceitos de armazenamento de dados na memória, através da linguagem C, e utilizando dos pushbuttons da placa auxiliar [5].

Metodologia:

O primeiro passo para resolver o problema é configurar o pino relativo ao pushbutton 5. Isso permite com que o utilizemos como provedor de sinais de entrada para o programa a ser executado. Através do registrador PTDD (Port Data Direction Register), pode-se fazer isso (Seção 41.2.6 de [2]).
Além disso, o desenvolvimento do programa é crucial para o bom desempenho do microcontrolador. Para este caso, estarão sendo utilizados arranjos. Essas estruturas de dados podem ser alocadas estáticamente ou dinamicamente. A segunda opção é necessária quando a quantidade de memória a alocar só se torna conhecida durante a execução do programa. Tratando-se de microcontroladores, o programador deve ter o cuidado de utilizar o espaço disponível de forma eficiente, sempre atentando ao fato, no caso do MCU estudado, de ser uma arquitetura 32 bits.

Proposta de Solução

Abaixo está o pseudo-código do experimento:

Algoritmo (Pseudo-Código)

```
INÍCIO: delay10us
        ENTRADA: número total de iterações
        ENQUANTO número total de iterações é maior que 0
                x←21
                ENQUANTO x é diferente de 0
                        Passa um ciclo do núcleo
                        DECREMENTA x
                DECREMENTA número total de iterações
        FIM ENQUANTO
FIM: delay10us

INICIO: configuraGpio
        CONFIGURE registrador de controle SIM_SCGC5[10] de forma a habilitar clock do módulo GPIO
        CONFIGURE registrador de controle PORTA_PCR5[8:10] de forma que o pino 5 da porta A sirva
        GPIO
        CONFIGURE registrador de controle PORTA_PCR12[8:10] de forma que o pino 12 da porta A
        sirva GPIO
        CONFIGURE registrador de controle PORTB_PCR18[8:10] de forma que o pino 18 da porta B
        sirva GPIO
```

```
        CONFIGURE registrador de controle PORTB_PCR19[8:10] de forma que o pino 19 da porta B sirva GPIO
        CONFIGURE registrador de controle PORTD_PCR1[8:10] de forma que o pino 1 da porta D sirva GPIO
        CONFIGURE registrador de controle GPIOB_PDDR[18, 19] de forma que os pinos 18 e 19 da porta B sejam de saída
        CONFIGURE registrador de controle GPIOD_PDDR[1] de forma que o pino 1 da porta D seja de saída
        CONFIGURE registrador de controle GPIOA_PDDR[5, 12] de forma que os pinos 5 e 12 da porta A sejam de entrada
FIM

INÍCIO: tempo_espera
        ENTRADA: identificador de botao
        Indice = 0
        SE ENTRADA FOR BOTAO 12
                ENQUANTO BOTAO 12 ESTÁ PRESSIONADO
                        Indice += 1
        SE ENTRADA FOR BOTAO 5
                ENQUANTO BOTAO 5 ESTA PRESSIONADO
                        Indice += 1
        RETORNA: Indice

INÍCIO: main
        INICIALIZA PINOS (chama configuraGpio)

        ENQUANTO VERDADEIRO
                SE BOTAO PTA12 ESTA APERTADO
                        VERIFICA QUANTO TEMPO FICA PRESSIONADO (chama tempo_espera)
                        Indice ← tempo_espera(12)

                ACENDE led de acordo com o retornado em tempo_espera

                SE BOTAO 5 ESTÁ APERTADO
                        VERIFICA QUANTO TEMPO FICA PRESSIONADO (chama tempo_espera)
                        nIteracoes ← tempo_espera(5)

                ESPERA(chama delay10us com número de iterações)

        FIM ENQUANTO
FIM
```

Testes

Para os testes, foram realizadas as seguintes etapas:

1) Primeiro, foram feitos os testes com a função delay10us implementada. O resultado foi satisfatório, com os leds alterando suas cores de acordo com o tempo de pressionamento do pushbutton pta12.
2) Depois de consultar os manuais do MCU e configurar a porta A do pino 5 corretamente, relativo ao pushbutton. Fez-se Build e Debug, e assim o programa foi executado.
3) Houve problemas com o tempo de resposta dos botões ao fazer os testes tanto de mudança de cor quanto de frequência
4) Alterou-se a função delay10us, pois o tempo estava longe do esperado
5) Depois de tentativas de ajuste, houve uma boa resposta dos leds

Conclusão

Trabalhar com microcontroladores requer um bom conhecimento não somente na parte de hardware, como os sinais elétricos e digitais do equipamento, mas também de um outro principal meio de comunicação entre humano e computador, a programação. A dificuldade em encontrar um tempo de resposta satisfatório, bem como o cuidado em alocar a memória, são dois exemplos que ilustram bem isso.

Referências

[1] FRDM-KL25Z User's Manual.

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/FRDMKL25Z.pdf

[2] KL25 Sub-Family Reference Manual

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf

[3] Extended Asm - Assembler Instructions with C Expression Operands

https://gcc.gnu.org/onlinedocs/gcc/Extended-Asm.html

[4] ARMv6-M Architecture Reference Manual – ARM Limited.

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/ARMv6-M.pdf

[5] EA871 – Descrição do Hardware da Placa Auxiliar

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/DescricaoDoHardware.pdf

[6] Alocação dinâmica de memória

http://www.ime.usp.br/~pf/algoritmos/aulas/aloca.html